UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

TOMAZ ROCHA DA SILVA

Um estudo de interação com displays grandes usando dispositivos iOS

Trabalho de Graduação.

Prof. Dr. Luciana Nedel
Orientador

Porto Alegre, Dezembro de 2011.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus, por ter me dado o dom da vida e por ter me proporcionado todas estas oportunidades maravilhosas.

Agradeço aos meus pais pelo seu carinho incondicional e que sempre me incentivaram e me apoiram nesta caminhada.

Agradeço aos meus irmãos pelo seu apoio e amizade.

Agradeço a minha namorada Joanne, por toda a ajuda, compreensão e amor que sempre dedicou a mim.

Agradeço a minha orientadora Luciana Nedel, por toda a ajuda, liberdade e confiança a mim dedicada.

Agradeço Universidade Federal do Rio Grande do Sul, em especial ao Instituto de Informática, pelo ensiono de qualidade e por toda a sua excelente estrutura.

"As tecnologias mais profundas e duradouras são aquelas que desaparecem. Elas dissipam-se nas coisas do dia a dia até tornarem-se indistingüíveis."

Mark Weiser - 1952 a 1999 - Cientista Chefe do XEROX PARC - "O Computador do Século 21"

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

SDK	Software Development Kit
OPENGL ES	OpenGL for Embebed system
XML	Extensible Markup Language
ARC	Automatic Reference Counting
MVC	Model View Controller
VOIP	Voice Over IP
TCP	Transmission Control protocol
UDP	User Datagram Protocol
IDE	Integrated Development Environment
API	Application Programming Interface

LISTA DE FIGURAS

RESUMO

Neste artigo vê-se o desenvolvimento e o estudo de novas técnicas de interação adaptadas ao paradigma de interação entre os usuários e o ambiente.

O trabalho está organizado em sete capítulos nos quais se vê o desenvolvimento do estudo de interação com displays grandes usando dispositivos iOS.

A crescente melhora no hardware dos dispositivos móveis e a sua popularização estão tornando realidade à computação ubíqua e os dispositivos móveis oferecem um grande número de novas formas de interação, não só com o usuário, mas também com o ambiente ao nosso redor.

Dentre os diversos dispositivos móveis e sistemas operacionais que existem, foi feito um estudo detalhado sobre as principais características do iPod e seu sistema operacional, o iOS. Neste contexto é explorada uma aplicação que permite o estudo com usuários de técnicas não convencionais de interação.

Também foi realizado um estudo sobre a técnica de interação com displays grandes lop-cursor, esta técnica foi adaptada e estendida para permitir a replicação dos dados nos dispositivos móveis.

Foi desenvolvida uma aplicação para criar desenhos em um display grande, utilizando dispositivos móveis para a interação com o display. Nesta aplicação utiliza-se a tela sensível ao toque dos dispositivos móveis para se desenhar no display. O dispositivo móvel possui capacidade de replicar os dados que estão no display grande.

Palavras-Chave: Dispositivos móveis; interação humano-computador; iOS; display_wall.

A study of interaction with tiled displays using mobile devices with iOS

ABSTRACT

This paper present the development and study of new interaction techniques adapted to the paradigm of interaction between users and the environment.

The paper is organized into seven chapters in which we see the development of the study of interaction with large displays using iOS devices.

The increasing improvement in the hardware of mobile devices and their popularity is becoming ubiquitous computing a reality and mobile devices offer a large number of new forms of interaction not only with the user, but also with the environment around us.

Among the various mobile devices and operating systems that exist, a detailed study was made on the main features of the iPod and its operating system, IOS. In this context is explored an application to study of users non-conventional techniques of interaction.

Was also carried out a study on the technique of interaction with large displays lop-cursor, this technique was adapted and extended to allow replication of data on mobile devices.

An application was developed to create drawings on a large display, using mobile devices to interact with the display. In this application it uses the touch screen mobile devices to draw on the display. The mobile device has the ability to replicate data that is on the big display.

Keywords: mobile devices; human-computer interation; iOS; tiled display.

1 INTRODUÇÃO

No Brasil existem aproximadamente 19 milhões de smartphones, sendo que cerca de 30,3% dos internautas brasileiros possuem um destes(W/MCCANN 2011). Com o significativo crescimento da tecnologia móvel os dispositivos estão cada vez mais populares, trazendo para os usuários acesso a uma grande quantidade de conteúdos, das mais diversas utilidades, que variam desde entretenimento até facilidades para a vida em sociedade.

Dispositivos móveis são pequenos computadores, geralmente de uso pessoal, que podem possuir tela sensível ao toque e/ou teclado em miniatura, podendo ser levados para todos os lugares.

Além disso, estes dispositivos possuem um paradigma de interação humano-computador diferente do que estamos acostumados.

Estes dispositivos encerram grandes e instigantes desafios, pois ainda possuem limitações técnicas comparadas aos computadores convencionais, remontando em parte as dificuldades enfrentadas nos primórdios da computação, devido a pouca disponibilidade de memória e processamento.

O presente trabalho tem como foco pesquisar e estudar técnicas de interação em displays grandes utilizando dispositivos móveis.

Displays grandes são constituídos de múltiplos monitores, projetores ou televisões dispostas contiguamente ou sobrepostos de forma a criar uma única tela grande.

Estes displays são especialmente úteis em aplicações que necessitam visualizar grande quantidade de dados, conforme demonstrado em Shupp et a.(2006) o uso de áreas grandes para visualização de dados melhoram a performance do usuário e diminuem a sua frustação.

Entretanto como citado em Peck et al.(2009), o aumento no tamanho do display melhora a performance do usuário, mas em contrapartida cria novas dificuldades de interação com ele.

Por isto o estudo de interação com displays grandes usando dispositivos iOS é importante visto que as técnicas convencionais não são eficientes.

Os smartphones, junto com os tablets, são melhores representantes dos dispositivos móveis. Eles se diferenciam dos telefones celulares convencionais por possuírem hardware aprimorado, com maior capacidade de armazenamento e poder de processamento. Atualmente encontramos diversos sistemas operacionais para estes dispositivos, sendo que, os principais são:

- O Android do Google, sistema operacional baseado no Linux, que é open source; os aplicativos para este sistema são escritos na linguagem Java. Atualmente é o sistema presente na maior parte dos smartphones;

- O Windows Phone da Microsoft, sistema mais recente da nova geração tem como característica uma interface gráfica inovadora, o Metro. Os aplicativos para Windows Phone são desenvolvidos utilizando o framework XNA e Silverlight;

- O iOS é um dos mais populares e robustos, possuindo sistema de interface com o usuário, simples e intuitivo, com capacidade de reconhecimento de gestos. Recentemente ganhou a habilidade de reconhecer sentenças em linguagem natural. Ele foi um dos grandes responsáveis pelo sucesso dos dispositivos produzidos pela Apple.

Este trabalho tem por objetivo o estudo da plataforma e arquitetura do iOS, identificando seus pontos positivos e negativos, especialmente referente ao escopo deste projeto, ou seja, seu uso como dispositivo de interação com outros sistemas.

Na intenção de avaliar a viabilidade de uso de dispositivos com iOS para interação, foram desenvolvidos dois sub-projetos: o estudo e a implementação de uma aplicação desktop que permita o uso de múltiplos mouses, bem como o desenvolvimento de uma aplicação prática visando utilizar uma forma não convencional de interação humano-computador através do uso de dispositivos móveis com iOS para interagir com displays grandes, criando assim uma ferramenta útil tanto de forma educativa quanto corporativa.

Para atingirmos este objetivo será feito um estudo da técnica de interação com displays grandes utilizando dispositivos móveis, chamada lop-cursor, visando sua extensão e adaptação em uma nova aplicação. Esta técnica apresenta uma metáfora para cursores em displays grandes que permite precisão de apontamento e o controle simultâneo do cursor que está sendo desenvolvida pelo grupo de computação gráfica da UFRGS.

1.1 Organização do Texto

A seguir encontra-se a especificação de cada assunto para facilitar a compreensão do trabalho, fazendo com que seja possível visualizá-lo de forma sucinta e sistematizada na intenção de ter uma visão geral do mesmo:

Capítulo 2 – Descreve os trabalhos relacionados com o tema deste estudo.

Capítulo 3 – É um estudo sobre a plataforma iOS e sobre o dispositivo móvel iPod. São apresentadas as principais características do dispositivo e do sistema operacional.

Capítulo 4 – Neste capitulo é apresentado o cenário pensado para este estudo, além do detalhamento da proposta deste trabalho.

Capítulo 5 – Faz uma rápida introdução a técnica lop-cursor, abordando as adaptações e a extensão proposta a esta técnica.

Capítulo 6 – Faz se um detalhamento do desenvolvimento da aplicação descrita na proposta.

Capítulo 7 – É feita uma discussão dos resultados obtidos com este estudo além de apresentar uma séries de possibilidades para trabalhos futuros.

2 TRABALHOS RELACIONADOS

No Brasil, segundo dados da Anatel(2010), existe mais de 202 milhões de telefones celulares, o que segundo a mesma pesquisa significa que o número de celulares é maior que o número de habitantes no Brasil. Ballagas et al.(2006) considera os telefones celulares os primeiros dispositivos verdadeiros de computação ubíqua.

O termo computação ubíqua foi criado por Weiser(1991), e descreve um paradigma de interação humano-computador totalmente diferente do que estamos acostumados na interação com os desktop. Neste paradigma ele propõe que a interação, humano-computador seja invisível para o usuário, pois ela estaria tão profundamente integrada nas nossas atividades e objetos do nosso cotidiano, que não nos daríamos conta que estamos realizando alguma interação com um dispositivo.

Mas apesar de os dispositivos móveis estarem tornando aos poucos este conceito em realidade, Weiser diz que o real poder da computação ubíqua não está em nenhum dispositivo especifico, mas sim da interação entre todos eles.

"Prototype tabs, pads and boards are just the beginning of ubiquitous computing. The real power of the concept comes not from any one of these devices; it emerges from the interaction of all of them."(WEISER,1991)

Existem diversos trabalhos, incluindo este, que tratam da interação entre dispositivos móveis e displays grandes, entre alguns dos mais conhecidos se destacam os baseados em análise ótica.

Jeon et al.(2006) define três técnicas de interação com display utilizando analise óticas. As técnicas definidas por eles são:

- motion flow : utiliza um cursor para mostrar a área de interesse do usuário. O usuário enxerga um cursor e para fazer uma seleção posiciona o cursor sobre o objeto. O movimento do cursor é obtido através dos vetores de movimentos relativos do fluxo ótico.
- marker-object : nesta técnica cada objeto recebe uma marcador, que contém a identificação básica do objeto. Para selecionar um objeto simplesmente é necessário centralizar o objeto câmera do dispositivo.
- marker cursor. Utilizando a câmera do dispositivo, se detecta o cursor na tela, e utilizando ele como uma marca se calcula a posição do dispositivo.

Analisando as técnicas apresentadas por Jeon, foi possível ver que estas não utilizam todo o hardware disponível nos dispositivos móveis hoje em dia, por isso para o presente trabalho foi utilizada outras técnicas.

Além das técnicas baseadas em analise óticas, existem outras abordagem documentadas, como a abordagem dual view, descrita por Finke et a.(2010). A técnica descrita neste artigo se baseia em distribuir os componentes da interface visual ente o display grande e o dispositivo.

Para este trabalho foi utilizada em alguns aspectos esta abordagem, como a replicação dos dados exibidos no display grande na tela do dispositivo móvel.

Foi especialmente importante para este trabalho a técnica lop-cursor, desenvolvida pelo grupo de computação gráfica da UFRGS (DEBARBA,2011,UFRGS). Nesta técnica é apresentada uma metáfora para cursores em displays grandes que permite precisão de apontamento e o controle simultâneo do cursor, que serviu de base para este trabalho.

3 CARACTERÍSTICAS DA PLATAFORMA IOS E DO DISPOSITIVO UTILIZADO

Devido à natureza investigativa deste trabalho, é essencial fazer um estudo completo da plataforma iOS, pois somente assim poderemos explorar de forma eficiente todos os benefícios dela.

Da mesma forma o conhecimento dos aspectos técnicos do dispositivo é fundamental para o bom êxito do estudo.

3.1 Conhecendo o iOS

O iOS é um sistema operacional para dispositivos móveis desenvolvida pela Apple, sendo que a empresa não licencia o seu uso em hardware de terceiros.

Inicialmente a Apple não permitia que aplicativos de terceiros rodassem nos seus dispositivos, somente em março de 2008 quando foi liberado o iPhone SDK(Software Development Kit) se tornou possível a instalação de aplicativos de terceiros.

Entre os principais dispositivos que utilizam este sistema estão os iPhones, iPads e iPods touch.

A primeira versão do iOS foi disponibilizada junto com o primeiro iPhone em junho de 2007, e vem sofrendo diversas melhorias desde então. Atualmente o sistema está na versão 5.0.1.

O iOS é derivado do Mac OS X, sistema desenvolvido pela Apple para rodar nas plataformas convencionais e, é baseado no UNIX.

No iPhone SDK encontramos as ferramentas básicas para podermos desenvolver aplicativos nativos para o iOS.

Junto com o SDK encontramos o Xcode tools, ferramenta na qual podemos editar e compilar os aplicativos gerados, além de prover o iOS Simulator, um simulador da plataforma que executa no computador desktop. Este simulador é um aspecto negativo do iOS em relação aos outros sistemas, como o Android do Google. Nas outras plataformas ao invés de existir um simulador, encontramos um emulador, capaz de emular mais precisamente o hardware dos dispositivos móveis, permitindo que os testes das aplicações neste ambiente virtual sejam mais próximo possível do comportamento nos dispositivos reais.

Um dos destaques do iOS é a Apple App Store, loja virtual que comercializa aplicativos para iOS, que já contava com mais de 500.000 aplicativos em outubro de 2011(WIKIPEDIA,2011).

Outro destaque desta plataforma é a sua interface com o usuário, que provê uma interação fluida, que implementa o conceito de manipulação direta, através da interação com multi-toques e reconhecimento de gestos, como rotações tridimensionais.

3.2 Arquitetura

A arquitetura do iOS atua como um midleware, ou seja como uma camada intermediaria entre a aplicação e o hardware do dispositivo. O acesso ao hardware se dá através de um conjunto de interfaces bem definidas.

Além disso, por razões de segurança, cada aplicação somente pode ler e escrever em uma área especifica, determinada pelo o iOS, chamamos esta área de sandbox.

O iOS possui quatro camadas de abstração exibidas na figura 1.

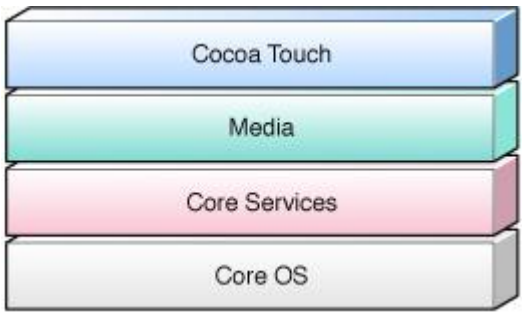

Figura 1Camadas de abstração iOS(iOS Overview, 2011)

As camadas inferiores, Core Services e Core OS, são desenvolvidas em C, e têm nível mais baixo de abstração. Estas camadas contém as interfaces mais fundamentais do iOS entre elas as utilizadas para acessar os serviços nível mais baixo, como os sockets de rede entre outros.

Já as camadas superiores contêm os frameworks com tecnologias mais avançadas, geralmente são escritas em uma mistura de Objective-c e c.

Segue uma breve descrição de cada uma das camadas anteriormente citadas.

3.2.1 Camada Cocoa Touch

A camada Cocoa Touch é uma das mais utilizadas no desenvolvimento de aplicativos para iOS, ela contém os principais frameworks necessários para a criação do programa.

Os principais serviços gerenciados por esta camada são: os Storyboards, utilizados para criar os designs das interfaces com o usuário; multitarefa, impressão, proteção de dados; Serviço de notificações de push da Apple; notificações locais; reconhecimento de gestos; suporte ao compartilhamento de arquivos; serviços peer-to-peer; suporte a displays externos e as view controllers de sistema padrão que possuem interfaces para algumas funções do sistema.

3.2.2 Camada de Mídia

Esta camada é a responsável por gerir os recursos de vídeo, áudio e de computação gráfica.

Os recursos de vídeo e áudio do dispositivo fogem do escopo deste trabalho, por isso descreve-se a seguir com mais de detalhes somente as capacidades gráficas do iOS, pois estas apresentam relevância para o estudo de caso.

Esta camada disponibiliza uma série de tecnologias que podem ser utilizadas de forma combinada ou não para desenhar na tela do dispositivo. As principais tecnologias disponíveis são: Core Graphics que suporta vetores 2D e renderização baseadas em imagens; Core animation que provê suporte avançado a animação das views; Core image que apresenta suporte a manipulação de fotos e vídeos; OpenGL ES e GLKIT que contém suporte a renderização em 2D e 3D utilizando aceleração de hardware; Core Text que suporte a layout e renderização de texto; Image I/O suporte a leitura e escrita em diversos formatos de imagens.

Neste trabalho foi utilizado a tecnologia OpenGL ES (ES significa sistemas embarcados) que é um subconjunto do OpenGL API, devido ao seu suporte para a aceleração de hardware, o que torna mais rápida e eficiente a renderização de imagens no dispositivo.

3.2.3 Camada de Core Services

Esta camada é extremamente importante, ela contém diversos serviços de sistema, e embora ela não seja normalmente utilizada de forma direta, todas as aplicações fazem uso desta, pois diversas partes do sistema são construídas com base nestes serviços.

Os principais serviços desta camada são: armazenamento na iCloud (serviço de armazenamento de documentos nas nuvens desenvolvido pela Apple), suporte a XML, SQLite, In-App Purchase (sistema de compras utilizado dentro de uma aplicação), ARC (Automatic Reference Counting – serviço que simplifica a gerência de memória dos aplicativos), blocos de objetos (referenciado em algumas outras linguagens como closures ou lambda).

3.2.4 Camada de Core OS

A camada Core OS tem mais baixo nível de abstração, e sobre os serviços por ela disponibilizados são construídas todas as outras camadas.

Geralmente esta camada só é utilizada diretamente quando é necessário fazer comunicação com acessórios externos ou com questões referentes à segurança.

3.3 Principais características do iOS

Os aplicativos desenvolvidos no iOS seguem o padrão de arquitetura de software MVC, que tem como principal característica separar a lógica de negócio, da lógica de apresentação.

Na figura abaixo podemos ver a utilização do MVC com os principais elementos do iOS.

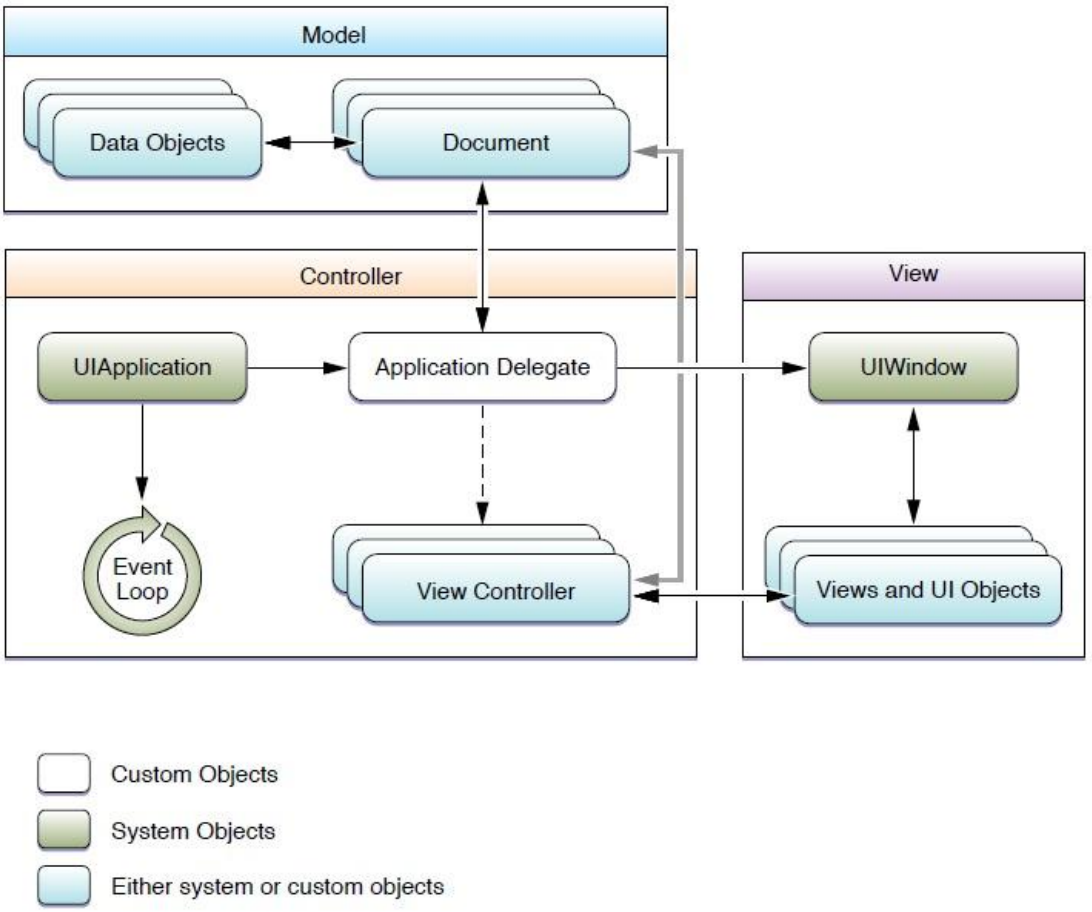

Figura 2: Exemplo do padrão MVC aplicado ao iOS

Nota-se no iOS grande preocupação com a segurança do usuário, que pode ser visto por exemplo na adoção de um sandbox para os aplicativos executarem.

De forma geral o iOS é um sistema operacional, rápido, eficiente e seguro que possui uma interface agradável e fácil de utilizar. Porém há uma série de restrições impostas que podem dificultar o trabalho, tornando tarefas que seriam triviais, em outras plataformas, extremamente difíceis de serem realizadas no iOS.

A interação com o usuário é evidentemente um dos principais pontos positivos no iOS, existem um grande número de componentes visuais para serem utilizados, e estes são intuitivos e apresentam uma excelente resposta. Também há um grande número de gestos multi-toques e movimentos realizados com o dispositivo, que são reconhecidos pelo sistema, estes podem ser utilizados das mais diversas formas.

Recentemente outra importante adição à interação com o usuário foi o Siri, assistente pessoal desenvolvido pela Apple, capaz de reconhecer sentenças complexas em linguagem natural (WIKIPEDIA,2011).

Outra característica marcante do iOS é o fato de ele não ter suporte a multitarefas, somente uma aplicação pode estar rodando por vez no dispositivo, as outras aplicações ficam em modo suspenso quando entram em background. As únicas exceções são as aplicações que fazem uso de Voip, geolocalização ou de áudio. E mesmo nestes casos citados, quando a aplicação está em background só podemos monitorar e responder a eventos disponíveis pelo iOS e específicos para cada um dos casos anteriores para interagir com o aplicativo desenvolvido. A figura a seguir mostra um pouco melhor o que acontece quando uma aplicação passa a rodar em background.

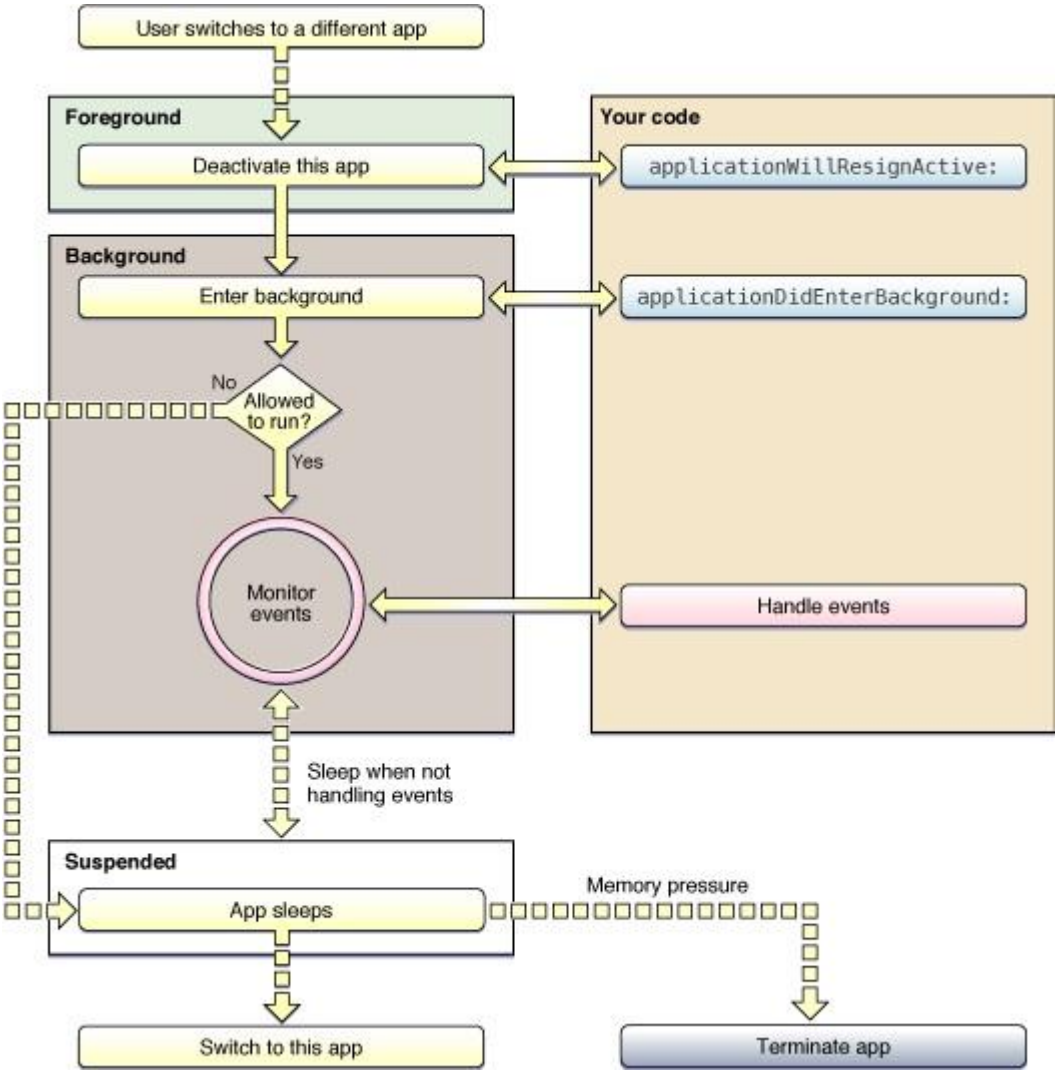

Figura 3 : Exemplo do ciclo do aplicativo quando ele entra em background(iOS App Programming Guide,2011)

Para o usuário final, o fato de as aplicações não poderem executar em background, é um ponto positivo, pois ela ajuda aumentar a duração da bateria, porém para os desenvolvedores este é um fator limitante, visto que não é possível criar aplicativos que fiquem rodando em background como acontece nas plataformas desktop.

Um dos pontos negativos no iOS é a limitação no uso do bluetooth, este não pode ser usado para sincronização do dispositivo com o desktop, nem para troca de arquivos; com exceção da troca de arquivos realizada entre dispositivos móveis da Apple rodando a mesma aplicação. Além disso, só é possível conectar os dispositivos com acessórios bluetooth certificados pela Apple.

3.4 Características dos dispositivos móveis utilizados

Para este estudo de caso foram utilizados como dispositivos móveis dois iPods Touch de quarta geração.

Os dispositivos estavam utilizando a versão 5.0.1 do iOS.

Os principais aspectos técnicos dos dispositivos estão listados a seguir:

- Capacidade de armazenamento: 8GB
- Tela com 3,5 polegadas com suporte a multi-toques (até 10 toques simultâneos)
- Suporte a Wifi e Bluetooth
- Câmera traseira que fotografa com resolução de até 960x720
- Câmera frontal com resolução VGA
- Giroscópio de 3 eixos
- Acelerômetro de 3 eixos
- Sensor de luz ambiente

Dentre estas características foram especialmente importantes para este estudo o giroscópio, sensor que mede o momento angular e o acelerômetro, que mede a aceleração inercial.

Os sensores de giroscópio e acelerômetro presentes neste dispositivo são de alta qualidade e possuem uma grande precisão, sendo muito superiores aos sensores que encontrados no Wiimote por exemplo.

Com a fusão dos dados obtidos nestes sensores conseguimos obter seis degraus de liberdade (6DoF) em relação ao movimento do dispositivo no espaço tridimensional, ou seja, podemos calcular as translações e rotações em torno dos eixos perpendiculares.

A seguir apresenta-se o estudo de caso que mostra interação, humano-computador utilizando dispositivos móveis. Dentro deste contexto a escolha do ipod de quarta geração que utiliza o iOS como sistema operacional parece bastante acertada, visto que este sistema e dispositivo oferecem grande quantidade de ferramentas com foco na interação do usuário, permitindo bastante liberdade na criação de novas formas de interação não convencionais.

4 ESTUDO DE CASO

4.1 Cenário

O cenário proposto para este estudo foi baseado no cenário de interação proposto em Jeon et a.(2006), onde um grupo de pessoas utiliza seus celulares para enviar arquivos de seus telefones e exibirem, de forma simultânea, em um único display público.

O que o autor propõe como cenário para este estudo é um ambiente semelhante, onde temos diversas pessoas interagindo com um único display público, cada uma delas usando seu dispositivo móvel para fazer esta interação.

O objetivo neste cenário é criar um desenho neste display público, que posteriormente pode ser salvo no dispositivo móvel de cada usuário.

Os usuários interagem de forma simultânea, utilizando a tela sensível ao toque dos seus dispositivos para fazerem a interação com o display.

4.2 Proposta

Este trabalho tem como proposta criar um protótipo que permita o estudo da interação colaborativa em displays grandes, utilizando dispositivos móveis.

O protótipo se constitui de duas partes, a primeira é o servidor que executará no display, este será responsável por gerenciar os comandos enviados pelos dispositivos e exibi-los no display. O servidor também fará uma representação visual dos múltiplos cursores, cada um deles comandado por um dispositivo móvel diferente.

A segunda parte do protótipo se constitui no cliente que executará nos dispositivos móveis, a função deste é capturar a interação realizada pelos usuários e enviá-las para o servidor. O cliente também será responsável por replicar na tela do dispositivo a informação exibida em uma área do display público que será selecionada pelo usuário.

Para realizar estes objetivos, será feito um estudo das características de um dispositivo móvel, bem como do seu sistema operacional, com o intuito de verificar possíveis características que seriam proveitosas para este estudo.

Também será necessário fazer um estudo da técnica do lop-cursor, para assim ser possível realizar as adaptações e extensões necessárias a técnica para viabilizar o seu uso no protótipo cliente.

Por fim será desenvolvido o protótipo do servidor, que deverá executar as tarefas citadas anteriormente.

5 EXTENSÃO E ADAPTAÇÃO DO LOP-CURSOR

Durante o trabalho de pesquisa sobre técnicas de interação entre dispositivos móveis e displays grandes foi possível conhecer a técnica lop-cursor, e ao mesmo tempo perceber o potencial desta técnica.

Deste ponto surgiu a ideia de adaptar e estender a técnica para ser utilizada neste estudo, proporcionando assim caso teste concreto para testar a técnica.

5.1 A técnica lop-cursor

O Lop-cursor é baseado em dois níveis de precisão, estes níveis são necessários para contornar o problema de falta de precisão encontrado em muitas técnicas de apontamento direto.

Primeiramente com o lop-cursor, o usuário faz um apontamento direto do dispositivo para o alvo na tela. Nesta primeira etapa é movido através da tela um cursor retangular que contém a seta do cursor normal.

Após esta primeira etapa, o usuário pode movimentar o cursor de forma mais precisa, utilizando para isso a tela sensível ao toque no dispositivo.

Quando o usuário toca na tela, ele fixa a posição do cursor retangular, e pode então deslizar o dedo sobre tela do dispositivo, que resulta no movimento da seta do cursor que se encontra dentro do cursor retangular.

Este funcionamento fica mais bem explicado na figura 4.

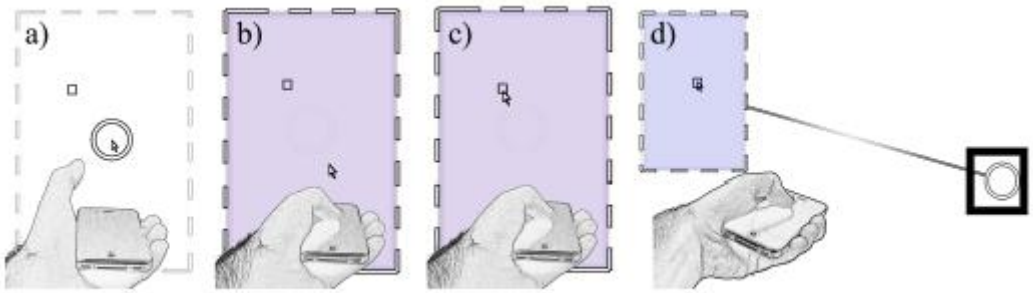

Figura 4 (a)apontamento para uma pequena área alvo (b) quando o usuário toca na tela sensível ao toque, ele fixa a posição do cursor retangular (c) o usuário pode então movimentar o cursor de forma mais precisa deslizando o dedo na tela sensível ao toque

5.2 Principais características

O lop-cursor possui três estados distintos de interação, são ele: free-pointing, hold-control-convas + free-pointing e pin-control-canvas + free-pointing.

No estado de free-pointing só é possível usar a técnica de apontamento direto para mover o cursor, neste estado o usuário aponta o dispositivo para á área do display que deseja apontar.

Quando o usuário toca na tela do dispositivo, e segura o dedo na tela, é ativado o estado hold-control-convas + free-pointing. Neste estado o cursor retangular fica fixo na posição em que ele estava quando o usuário inicia o toque. O usuário pode então mover o dedo sobre a tela para posicionar melhor a seta do cursor. Um diferencial desta técnica é que o mapeamento do toque na tela do dispositivo para o cursor retangular é feito de forma absoluta. Quando o usuário solta o dedo do contato com a tela do dispositivo, o cursor retorna ao estado free-pointing.

Quando o usuário toca a tela do dispositivo, duas vezes seguidas em um curto intervalo de tempo (duplo tap), o cursor entra no estado pin-control-canvas + free-pointing. Neste estado o cursor retangular é fixado na posição que estava sendo apontado quando o usuário faz o primeiro toque na tela. Enquanto o usuário não fizer novamente um duplo tap o cursor não sairá deste estado. Ao deslizar o dedo na tela do dispositivo o usuário move a seta do cursor. Este estado também permite que outras ações de toque sejam feitas, como uma seleção de algum item dando um toque simples na tela, permite que o gesto de pinch (gesto feito utilizando dois dedos, semelhante ao realizado para segurar uma pinça) seja utilizado para reescalar o tamanho da representação do cursor retangular na tela do display.

O cursor retangular é uma representação da tela do dispositivo, e como citado anteriormente é mapeado de forma absoluta para a tela do display. Isso significa que quando o usuário faz um toque no canto superior da tela do dispositivo, a posição deste toque será mapeada para a posição equivalente do cursor retangular, no caso deste exemplo o canto superior da representação.

Para obter a orientação do dispositivo, são utilizados os dados obtidos do giroscópio, acelerômetro e o magnetrômetro, utilizando o algoritmo de fusão dos dados destes sensores descritos por Madgwick et al.(2011), o que melhora a precisão dos dados obtidos. Foi implementada a versão do algoritmo descrito que utiliza os sensores citados anteriormente.

5.3 Modificações realizadas

Para melhor utilizar o lop-cursor neste estudo foram necessárias algumas modificações na implementação original.

No caso do estudo proposto pelo autor, foram utilizados como dispositivos móveis iPods touch de quarta geração, que não possuem magnetrômetro diferentemente do iPhone 4 que foi utilizado no projeto original. Como citado anteriormente o lop-cursor fazia uso do algoritmo descrito por Madgwick et al.(2011), porém o iPod não apresenta o magnetrômetro tornando impossível a aplicação do algoritmo já implementado.

Primeiramente optou-se pelo uso dos algoritmos de fusão disponibilizados pelo iOS, porém o resultados atingido não foi o esperado.

Então implementou-se a versão do algoritmo IMU (Inertial Measurement Units) descrita por Madgwick et al.(2011). que utiliza somente o giroscópio e o aceleromêtro, sensores que estão presente no iPod. Este algoritmo utiliza o acelerômetro para obter as translações que o dispositivo realiza e o giroscópio para obter as rotações realizadas por ele. A precisão no movimento do cursor obtida desta forma foi bem próxima do original, entretanto, o protótipo perde a calibragem com facilidade devido a falta do magnetrômetro, pois sem ele não é possível ir recalibrando os valores obtidos pelo giroscópio.

A versão original se comunicava com o servidor através de sockets de rede UDP, tecnologia que é mais rápida, porém não garante a entrega e a integridade dos dados transmitidos. Na aplicação proposta neste estudo, é feito o desenho na tela do dispositivo que então passa as coordenadas obtidas para o servidor que desenha então no display as mesmas informações desenhadas no dispositivo. Caso haja perda de informações entre o dispositivo e o display, a representação do desenho fica diferente entre os mesmos, para evitarmos este problema optou-se pelo uso de comunicação via sockets TCP que garantem a entrega e integridade dos dados enviados. Não foi possível observar nos testes diferenças de desempenho significativas entre a versão antiga e a atual.

Originalmente os estados do cursor eram controlados pelo servidor, o dispositivo apenas passava as informações para o servidor que verificava quando havia ocorrido um tap, duplo tap ou algum outro tipo de movimento. Este comportamento gerava um problema na detecção do duplo tap, pois para detectar um duplo tap se verificava o tempo entre dois taps seguidos realizados na tela do dispositivo; como o dispositivo apenas enviava as informações para o servidor, eventualmente a informação de um dos taps acabava se perdendo ou devido atrasos de processamento do lado do servidor e o funcionamento ficava comprometido, pois o servidor tinha falsa impressão de que havia ocorrido um tempo maior entre os dois taps do que realmente ocorreu.

Para corrigir o problema citado, o controle dos estados do cursor passou a ser realizado no dispositivo, que além das informações enviadas anteriormente passou a enviar para o servidor também o estado atual do cursor.

Além das modificações citadas acima, fez-se uma extensão do projeto original para adicionar a capacidade de replicação dos dados exibidos dentro da área do cursor retangular do lop-cursor representada no display grande. Para atingir este objetivo, além de enviar informações para o servidor, o lop-cursor passou também a receber informações referentes aos objetos que deveriam ser replicados nele. Utilizou-se para

desenhar na tela do dispositivo o openGL ES. Esta modificação foi uma das mais difíceis e trabalhosas do projeto, nela encontrou-se dificuldade para integrar a view especial que deve ser utilizada para poder renderizar as primitivas de openGL aos componentes de interface visual do iOS. Esta integração é importante pois os menus de

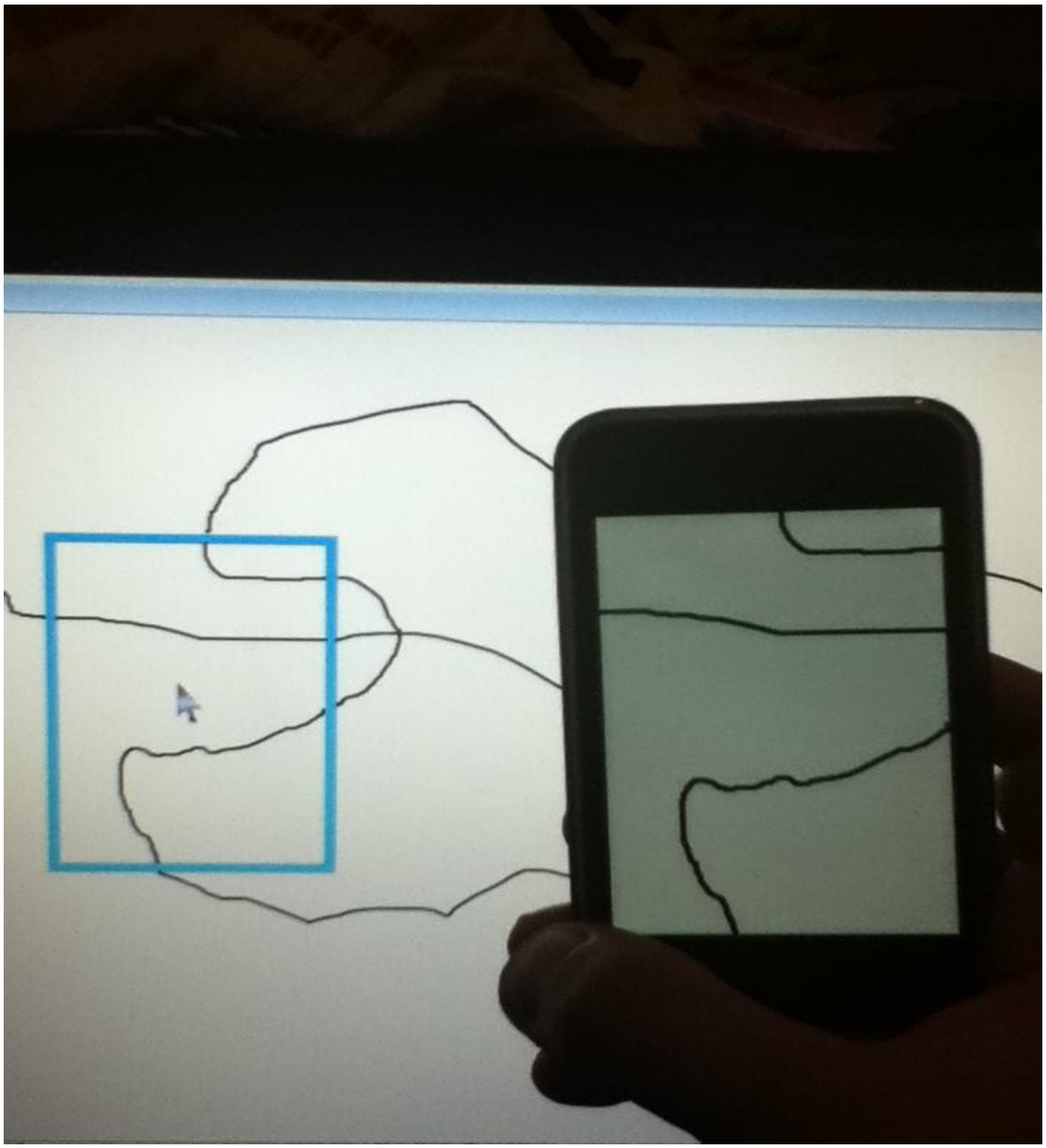

Figura 5: Dispositivo móvel replicando o conteúdo do display que se encontra dentro da área do cursor retangular

seleção de cor e tamanho das linhas desenhadas foi feito utilizando os componentes padrões do iOS.

O fato do openGL ES ser uma versão bem simplificada do openGL tradicional complicou esta modificação. Não existem, por exemplo, funções disponibilizadas pela API para a renderização de primitivas básicas como círculos.

Foi necessária extensa pesquisa para encontrar uma forma de contornar estes obstáculos.

Para auxiliar nesta etapa foi utilizadas a biblioteca Cocos2D, que é uma biblioteca desenvolvida para uso no iOS que simplifica a tarefa de fazer desenho em 2D em openGL ES. A principal vantagem do uso desta biblioteca é que ela permite a criação de subviews do tipo especial necessário para renderizar em openGL ES.

Uma subview, tem características de uma view comum, ela é na verdade um componente de uma view principal, e seu controle está subordinado a esta view principal.

Para resolver o problema de integração da view do openGL ES com os componentes visuais a solução adotada foi a utilização uma view principal do tipo padrão do iOS e de uma subviews do openGL ES um pouco menor que a outra view, que foi sobreposta a view principal.

Na parte da view principal que fica visível para o usuário foi colocada a paleta de cores e um seletor de tamanho para as linhas desenhadas.

A view principal é da cor cinza enquanto a subview é branca, na figura que mostra a interface da aplicação fica clara a divisão entre estas views.

Figura 6: Interface da aplicação rodando no iOS Simulator

Com relação à falta de funções básicas no openGL ES, com a adição da biblioteca Cocos2D quase todas as necessidades foram supridas, faltando somente uma função que que desenha os pontos.

Os pontos foram desenhados utilizando círculos preenchidos e abaixo segue o código utilizado para gerá-los.

```
214  void ccDrawCircleFilled( CGPoint center, float r, float a, NSUInteger segs, BOOL drawLineToCenter)
215  {
216      int additionalSegment = 1;
217      if (drawLineToCenter)
218          additionalSegment++;
219
220      const float coef = 2.0f * (float)M_PI/segs;
221
222      GLfloat *vertices = calloc( sizeof(GLfloat)*2*(segs+2), 1);
223      if( ! vertices )
224          return;
225
226      for(NSUInteger i=0;i<=segs;i++)
227      {
228          float rads = i*coef;
229          GLfloat j = r * cosf(rads + a) + center.x;
230          GLfloat k = r * sinf(rads + a) + center.y;
231
232          vertices[i*2] = j * CC_CONTENT_SCALE_FACTOR();
233          vertices[i*2+1] =k * CC_CONTENT_SCALE_FACTOR();
234      }
235      vertices[(segs+1)*2] = center.x * CC_CONTENT_SCALE_FACTOR();
236      vertices[(segs+1)*2+1] = center.y * CC_CONTENT_SCALE_FACTOR();
237
238      glDisable(GL_TEXTURE_2D);
239      glDisableClientState(GL_TEXTURE_COORD_ARRAY);
240      glDisableClientState(GL_COLOR_ARRAY);
241
242      glVertexPointer(2, GL_FLOAT, 0, vertices);
243      glDrawArrays(GL_TRIANGLE_FAN, 0, (GLsizei) segs+additionalSegment);
244
245      glEnableClientState(GL_COLOR_ARRAY);
246      glEnableClientState(GL_TEXTURE_COORD_ARRAY);
247      glEnable(GL_TEXTURE_2D);
248
249      free( vertices );
250  }
```

Figura 7: Trecho do código utilizado para desenhar os pontos

6 IMPLEMENTAÇÃO DA APLICAÇÃO PARA DISPLAYS GRANDES

Visando estudar a interação em displays grandes utilizando dispositivos móveis, foi desenvolvida uma aplicação para servir como estudo de caso.

A aplicação proposta neste estudo é um aplicativo que permite desenhar em um display grande, utilizando a tela sensível ao toque dos dispositivos móveis.

Este protótipo também permite replicar na tela do dispositivo as informações exibidas na tela do display grande que se encontram dentro da área do cursor que representa o dispositivo móvel.

A aplicação é composta de duas partes: a primeira parte é o aplicativo cliente, que roda nos dispositivos móveis e que foi descrita anteriormente, a segunda parte é o aplicativo servidor, que será descrito em detalhes nos próximos itens.

6.1 Tecnologias utilizadas

Para criar o protótipo do servidor foi utilizada a plataforma dotNET da Microsoft, e a linguagem de programação C#. O servidor foi desenvolvido na IDE Microsoft Visual Studio 2010, rodando em um sistema operacional Windows 7.

6.2 Modelagem do aplicativo

Neste item encontra-se a descrição de algumas classes importantes do aplicativo e também a descrição em detalhes das mensagens trocadas entre os dispositivos e o servidor.

Cada dispositivo se conecta ao servidor atráves de sockets TCP, cada dispositivo envia mensagens com informações para atualização da posição do cursor ou uma lista de primitivas que o servidor que desenhar na tela do display. O servidor envia para cada dispositivo mensagens com as primitivas que devem ser replicadas na tela do dispositivo.

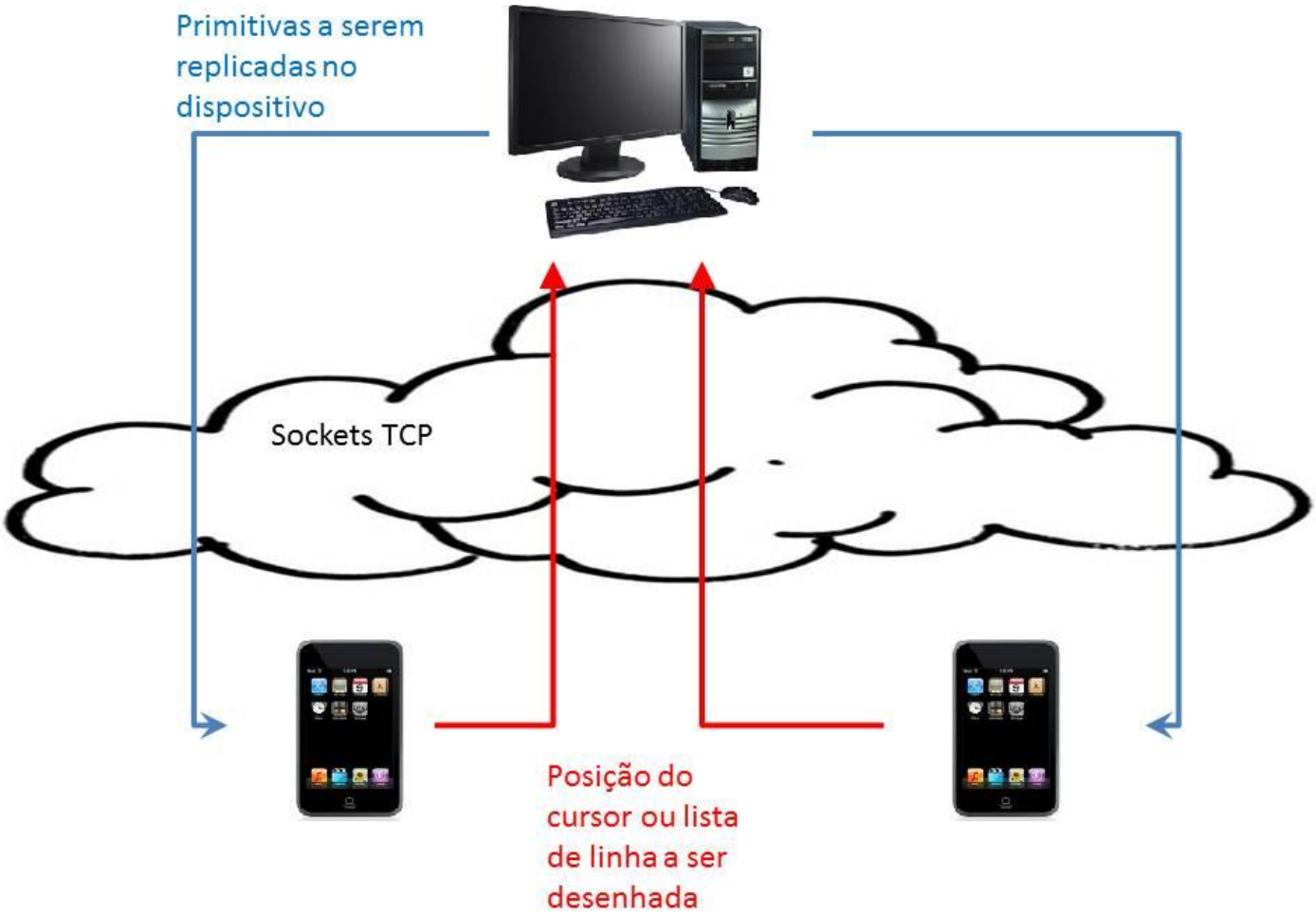

Figura 8 : Visão geral da arquitetura do sistema

Cada dispositivo envia para o servidor mensagens a cada 20 ms. Existem dois tipos diferentes de mensagem que o dispositivo pode enviar, a primeira delas é enviada quando ocorre atualização da posição do cursor ou um clique simples, e a outra quando se desenha na tela.

A mensagem de atualização do cursor é composta pelos seguintes valores: um inteiro que identifica o tipo da mensagem; outro inteiro que define o estado do cursor; um vetor com cinco booleanos, cada um correspondendo ao toque de um dedo na tela do dispositivo; um vetor com dez inteiros que contém local onde ocorre cada um dos toques, sendo que os dois primeiros valores são referentes ao dedo um os dois seguintes ao dedo dois e assim por diante; um vetor que contém três números de ponto flutuante referentes aos valores obtidos pelo acelerômetro em cada um dos seus eixos e um vetor com três números de ponto flutuante contendo os valores obtidos pelo giroscópio em cada um dos seus eixos.

Figura 9: Representação da mensagem de atualização do cursor

A mensagem de desenho na tela possui os seguintes valores: um inteiro que indica qual é o tipo de mensagem; outro inteiro que indica qual o estado do cursor; uma lista com os pontos que devem ser desenhados na tela, um vetor com três inteiros que representam os canais RGB da cor de desenho e um inteiro com o tamanho da linha desenhada.

Figura 10 Representação da mensagem de desenho na tela

A classe principal do servidor possui uma série de variáveis que são utilizadas para controlar o desenho na tela. Um vetor com quatro variáveis booleanas de nome isPaint,é utilizado para informar se o respectivo cursor está desenhando ou não no momento. Utiliza-se um segundo vetor com quatro variáveis do tipo Point (variável que armazena dois valores inteiros), que tem por função armazenar a última posição de cada cursor e recebe o nome lastPosition. Outro vetor com quatro variáveis do tipo color e nome colorLine, guarda a cor que cada dispositivo esta desenhando no momento. Por último usa-se um vetor com quatro variáveis do tipo inteiro, de nome sizeLine, que armazena o tamanho da linha desenhada por cada dispositivo.

A classe principal do servidor possui um vetor de dez objetos do tipo displayRegion, que foi descrita no item anterior. Esta variável possui o nome de displayLogicRegions.

Na figura 11 temos um diagrama de classes com as principais classes e métodos do aplicativo.

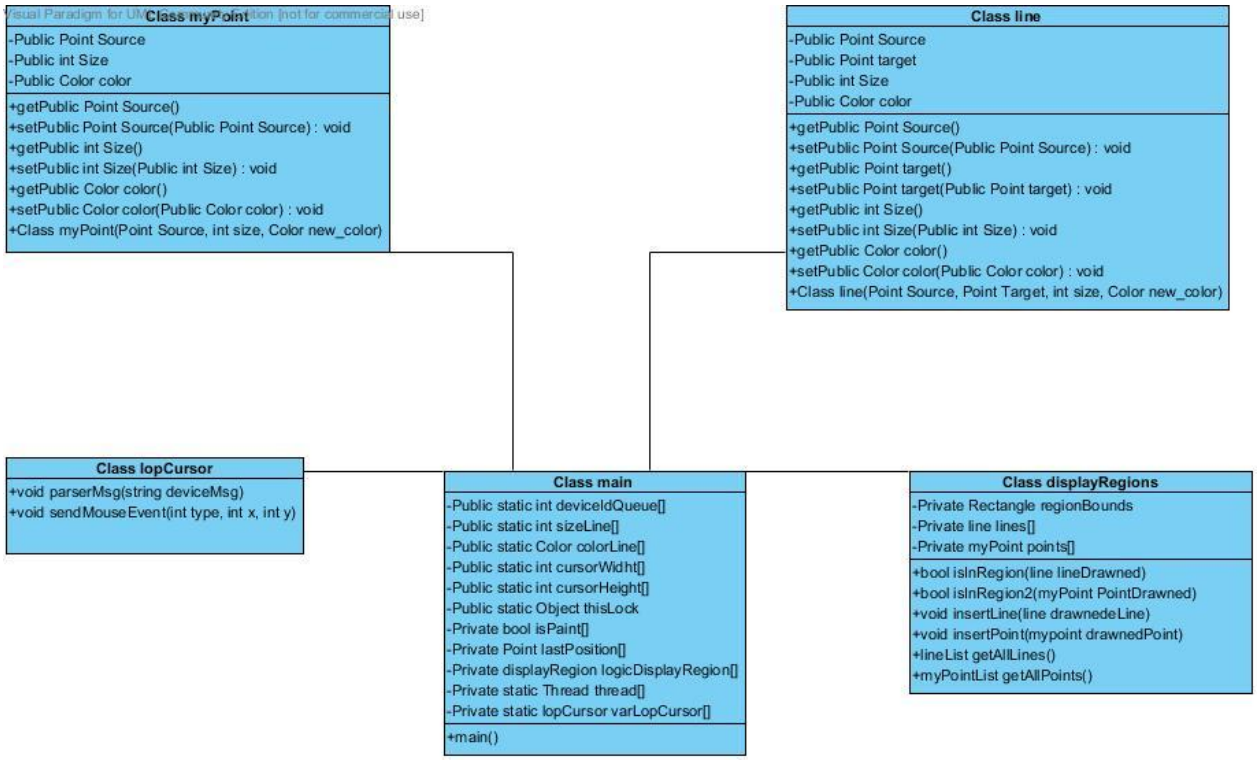

Figura 11: Diagrama de classes, com as principais classes do aplicativo

Para facilitar o entendimento e tornar mais claro o diagrama, algumas classes e métodos foram omitidos.

Os principais detalhes sobre o funcionamento do aplicativo serão descritos no próximo item.

6.3 Desenvolvimento do aplicativo

O aplicativo como citado anteriormente é um programa de desenho, que roda em um display grande onde mais de um usuário pode desenhar simultaneamente.

O primeiro passo, para a criação do protótipo foi o desenvolvimento da parte do servidor que utiliza a conexão de sockets TCP para se comunicar com os dispositivos, sendo que cria quatro threads distintas. Cada thread é responsável por gerenciar uma conexão em portas diferentes. Isso significa que o protótipo consegue lidar com no máximo quatro dispositivos diferentes simultaneamente, conectados a portas distintas.

Foi criada uma variável compartilhada entre todas as threads e a classe principal do servidor, cujo objetivo é guardar o id do dispositivo que está executando a ação no momento. Esta variável é uma lista, que armazena valores do tipo inteiro correspondente ao id do dispositivo que está executando a ação no momento. Esta variável recebeu o nome de deviceIdQueue.

A interface visual do programa foi criada utilizando a ferramenta de Windows Forms, disponibilizada no Microsoft Visual Studio. Para a criação da interface foram utilizados somente os componentes padrões disponibilizados no Visual Studio.

Para se representar graficamente os cursores, também se utilizou componentes padrões da interface visual disponibilizados pelo C#. Para facilitar a identificação cada cursor possui uma cor diferente.

Quando o servidor começa a executar, o cursor do Windows é escondido do usuário, para não causar confusão.

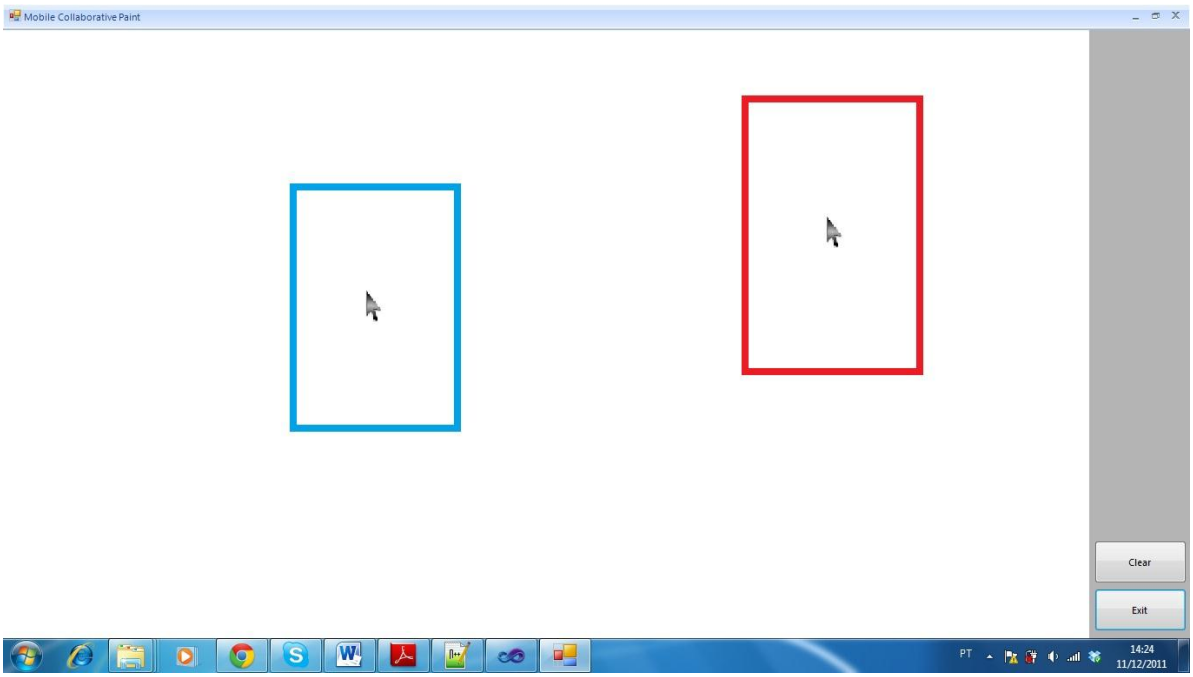

Figura 12: Interface do programa servidor, com dois dispositivos conectados

Para criar a área de desenho no display, foi utilizado o componente Picture box do Windows Form, o qual provê uma região retangular onde podemos manipular uma imagem no formato bmp. Esta imagem é criada e manipulada dinamicamente.

Na primeira implementação os todos os pontos e linhas desenhados ficavam armazenados em uma única lista. Cada vez que era necessário verificar quais linhas e pontos estavam dentro da área do cursor era necessário testar todos os pontos e linhas que já haviam sido desenhados até o momento. Conforme o número de elementos desenhados aumentava, o desempenho do programa caia drasticamente. Surgiu então a ideia de utilizar as regiões lógicas. Com o uso das regiões, ao invés de testar todos os pontos e linhas, primeiramente testa-se em quais regiões lógicas o cursor se encontra, e então verifica-se somente as linhas e pontos que estão armazenadas nas listas das regiões onde o cursor está.

Cada vez que uma linha for desenhada na tela, se testa por quais regiões esta linha passa, e então se adiciona esta linha às listas de linhas das referidas regiões. Se a linha passar por mais de uma região ela aparecerá replicada nas listas das regiões pelas quais ela cruza. De forma análoga os pontos passam por um processo muito semelhante ao descrito para as linhas.

Foram testadas diversas divisões da tela, com valores variando entre uma e vinte regiões. Sendo que nesta aplicação os melhores resultados foram obtidos com dez regiões, porém dependendo da aplicação e do tamanho da tela o número ideal de regiões pode variar.

Outra abordagem possível para solucionar este problema, seria ao invés de passar as primitivas para serem redesenhadas no dispositivos, passar os pixels que compõem a área retangular do cursor.

Todos os desenhos no display público se constituem de linhas ou pontos.

O mecanismo de desenho na tela é bem simples. São utilizados os seguintes eventos do componente Picture box para desenhar na tela: onMouseDown, onMouseUP e onMouseMove.

No evento onMouseDown, que é disparado quando o botão do mouse é pressionado, utiliza-se o id do dispositivo como índice para setar a posição correspondente do vetor isPaint para verdadeiro. Da mesma forma a variável lastPosition recebe a posição atual do cursor do dispositivo que gerou o evento. O id do dispositivo é obtido através da variável deviceIdQueue, ele é o primeiro elemento desta fila. Após a leitura do primeiro elemento remove-se o mesmo da lista.

```
private void pictureBox_MouseDown(object sender, MouseEventArgs e)
{
    int id = deviceIdQueue[0];

    isPaint[id] = true;

    lastPosition[id].X = e.X;
    lastPosition[id].Y = e.Y;

    lock (thisLock)
    {
        deviceIdQueue.RemoveAt(0);
    }
}
```

Figura 13: Código do evento onMouseDown

No evento onMouseMove, verifica-se se a variável isPaint correspondente ao id do dispositivo que gerou o evento possui o valor verdadeiro. Caso isto se confirme é desenhada uma linha entre o ponto armazenado na variável lastPosition correspondente, até a posição atual do cursor. A linha é desenhada com os atributos correspondentes ao configurados no dispositivo, os quais estão atualmente armazenados nas variáveis sizeLine e colorLine. Testa-se então a quais regiões lógicas pertence esta linha e ela é então inserida nas listas correspondentes. O valor da variável lastPositon correspondente é atualizado para a posição atual do cursor. O id do dispositivo é obtido da mesma forma citada no evento anterior. Após a leitura do id, o primeiro elemento da fila é removido.

```
private void panel1_MouseMove(object sender, MouseEventArgs e)
{
    int id = deviceIdQueue[0];

    Graphics gTela = Graphics.FromImage(tela);
    //inicializa a cor da linha
    color = new SolidBrush(colorLine[id]);
    //seta a cor e o tamanho da linha para ser desenhada
    Pen pen = new Pen(color, sizeLine[id]);

    Point position = new Point(e.X, e.Y);

    //verifica se é um ponto
    if (isPaint[id])
    {
        gTela.DrawLine(pen, lastPosition[id], position);
        line newLine = new line(lastPosition[id],e.Location, sizeLine[id],colorLine[id]);
        //verifica em qual região esta a linha e insere ela
        for(int i = 0; i < 10; i ++)
        {
            if (displayLogicRegions[i].isInRegion(newLine) == true)
                displayLogicRegions[i].insertLine(newLine);
        }
    }

    lock (thisLock)
    {
        deviceIdQueue.RemoveAt(0);
    }

    //atualiza a última posição
    lastPosition[id].X = e.X;
    lastPosition[id].Y = e.Y;

    gTela.Dispose();
}
```

Figura 14 Código do evento onMouseMove

No evento onMouseUp verifica-se se a posição armazenada na variável lastPosition correspondente é igual a posição atual, se for é desenhado um ponto, caso contrário é desenhada uma linha da mesma forma que no evento anterior. Testa-se então as regiões as quais ele pertence e o insere nas respectivas listas. A variável correspondente isPaint é então setada para falso. A obtenção do id do dispositivo que gerou o evento, bem como a remoção deste id da fila é igual à citada nos eventos anteriores.

```
private void panel1_MouseUp(object sender, MouseEventArgs e)
{
    int id = deviceIdQueue[0];

    isPaint[id] = false;
    Graphics gTela = Graphics.FromImage(tela);
    //inicializa a cor da linha
    color = new SolidBrush(colorLine[id]);
    //seta a cor e o tamanho da linha para ser desenhada
    Pen pen = new Pen(color, sizeLine[id]);

    Point position = new Point(e.X, e.Y);

    //verifica se é um ponto
    if (lastPosition[id] == e.Location)
    {
        gTela.FillEllipse(color, e.X, e.Y, sizeLine[id], sizeLine[id]);
        myPoint newPoint = new myPoint(e.Location, sizeLine[id], colorLine[id]);
        //verifica em qual região esta a linha e insere ela
        for (int i = 0; i < 10; i++)
        {
            if (displayLogicRegions[i].isInRegion(newPoint) == true)
                displayLogicRegions[i].insertPoint(newPoint);
        }
    }
    //caso contrário é linha
    else
    {
        //desenha a linha
        gTela.DrawLine(pen, lastPosition[id], position);
        line newLine = new line(lastPosition[id], e.Location, sizeLine[id], colorLine[id]);
        //verifica em qual região esta a linha e insere ela
        for (int i = 0; i < 10; i++)
        {
            if (displayLogicRegions[i].isInRegion(newLine) == true)
                displayLogicRegions[i].insertLine(newLine);
        }
    }

    lock (thisLock)
    {
        deviceIdQueue.RemoveAt(0);
    }

    gTela.Dispose();
}
```

Figura 15: Código do evento onMouseUp

Uma das principais dificuldades nesta fase da implementação foi decidir como seria implementado os múltiplos cursores propostos para este estudo.

Existe uma API desenvolvida para C#, o Multipoint SDK, que permite a utilização de mais de um cursor em ambiente Windows. Inicialmente a ideia era utilizar esta API. Entretanto, esta biblioteca só funciona com mouses fisicamente conectados ao computador, e não foi possível simular de forma satisfatória os eventos de mouse especiais com os quais a API funciona.

Após o servidor interpretar a mensagem recebida pelo dispositivo, o servidor seta a variável compartilhada com o id do dispositivo, seta as variáveis colorLine e sizeLine correspondente e utilizando a API do Windows ele cria um evento de mouse que é então disparado.

O tipo de evento de mouse que vai ser criado depende da mensagem recebida do dispositivo, podendo ser um evento de mouse button down, mouse button up ou um evento de movimento do mouse.

```
public void sendMouseEvent(int type, int x, int y)
{
    //inicializa o evento do mouse com as coordenadas x e y recebidas do dispositivo
    Input[] mouseEvent = new Input[1];
    mouseEvent[0].type = INPUT.MOUSE;
    mouseEvent[0].mi.dx = x;
    mouseEvent[0].mi.dy = y;

    //se o tipo for 1, é evento de movimentação do mouse
    if(type == 1)
        mouseEvent[0].mi.dwFlags = MOUSEEVENTF.MOVE | MOUSEEVENTF.ABSOLUTE;
    //se o tipo for 2, é evento mouse button down
    else if (type == 2)
        mouseEvent[0].mi.dwFlags = MOUSEEVENTF.RIGHTDOWN | MOUSEEVENTF.ABSOLUTE;
    //se o tipo for 3, é evento de mouse button up
    else if (type == 3)
        mouseEvent[0].mi.dwFlags = MOUSEEVENTF.RIGHTUP | MOUSEEVENTF.ABSOLUTE;

    //gera o evento de mouse
    UInt32 command;
    command = User32.SendInput((UInt32)mouseEvent.Length, mouseEvent, Marshal.SizeOf(mouseEvent[0]));
}
```

Figura 16: Código de geração de eventos do mouse

Após o servidor executar o comando recebido do dispositivo, ele verifica se o estado do dispositivo é o pin-control-canvas + free-pointing. Em caso afirmativo, significa que o dispositivo móvel está replicando em sua tela o conteúdo que está dentro da área do seu cursor retangular. Para que o dispositivo possa desenhar corretamente em sua tela o que está representado no display, o servidor precisa enviar para o dispositivo, via sockets, uma mensagem contendo todas as linhas e pontos que estão dentro da área do cursor.

O método de verificação para descobrir se a linha se encontra dentro da área do cursor é o algoritmo de corte de linha Cohen–Sutherland.

Abaixo podemos ver uma imagem do protótipo em ação, onde exemplifica-se a replicação das linhas entre o display e o dispositivo.

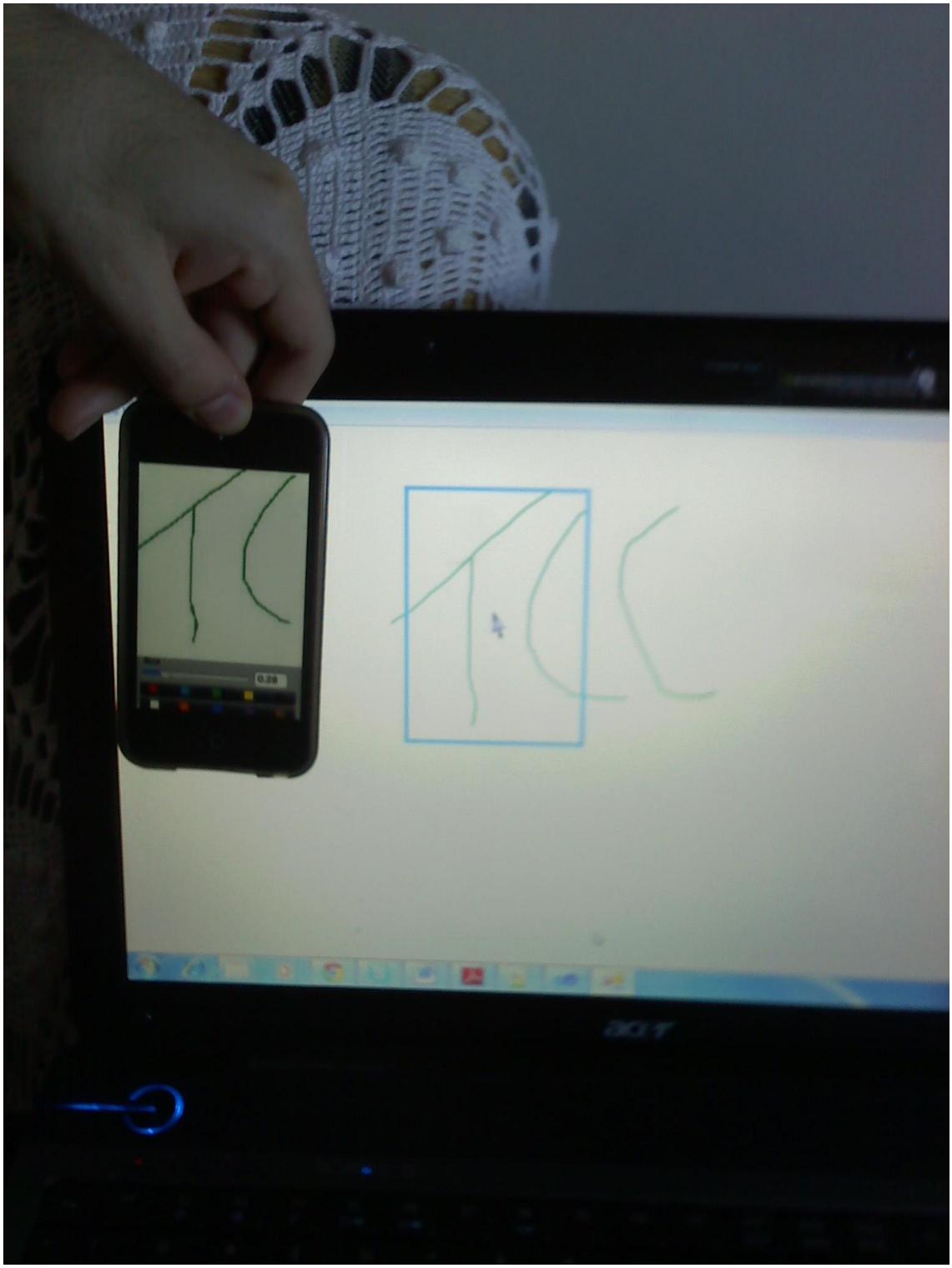

Figura 17: Imagem do protótipo em execução, mostrando a replicação no dispositivo

6.4 Avaliação

Para avaliar o protótipo desenvolvido foram realizados testes com um e com dois dispositivos conectados ao servidor.

Com apenas um dispositivo conectado ao servidor, este se comportou da forma esperada, movimentando o cursor com precisão e sem atraso na execução dos comandos enviados pelo dispositivo. Não foi observado atraso visível na replicação dos dados no dispositivo móvel.

Com dois dispositivos conectados ao servidor, este se comporta corretamente, representando corretamente os desenhos feitos nos dois dispositivos, e enviando as mensagens de replicação para os dispositivos de forma correta, mesmo nos casos onde os dois dispositivos estão desenhando na mesma área. Porém nos testes com dois dispositivos nota-se uma queda no desempenho da aplicação, em alguns momentos há um pequeno atraso no processamento dos comandos, mas nada que comprometa o funcionamento geral da aplicação. Também se observou um pequeno atraso na replicação dos dados.

7 CONCLUSÃO

Através do estudo da plataforma iOS obteve-se uma compreensão do funcionamento geral da sua arquitetura, descobrindo os seus pontos positivos e negativos. Estas descobertas permitiram que fossem identificadas as principais características do iOS que favorecem o estudo e a criação de técnicas de interação, e ficou evidente o enorme potencial que esta plataforma tem para o desenvolvimento nesta área.

A isso se soma o estudo das características do hardware do dispositivo móvel iPod, pois somente com estes conhecimento combinados é que se conseguiu extrair o máximo do dispositivo.

O estudo da técnica lop-cursor permitiu a sua adaptação para o uso neste projeto, além disso, ficou visível o potencial desta técnica, bem como se mostra o quanto ela pode ser generalizada e aplicada a diversos outros estudos e cenários. Também foi criada uma extensão à técnica original que permite a replicação dos dados do display grande no dispositivo móvel. Esta extensão abre caminho para diversos novos usos da técnica do lop-cursor.

Com o protótipo desenvolvido, o objetivo deste estudo foi alcançado, pois o resultado nos testes mostra que a aplicação esta pronta para ser utilizada para testes com os usuários, e através destes poderá ser validada a extensão proposta da técnica do lop-cursor, bem como será possível verificar a utilidade prática do software desenvolvido.

Apesar do êxito no desenvolvimento da aplicação, ao longo deste trabalhou ficou evidente que existem diversas melhorias e novas funcionalidades que podem ser acrescentadas ao trabalho já realizado.

7.1 Trabalhos futuros

Para trabalhos futuros, seria interessante testar novas formas de representar e implementar a funcionalidade de múltiplos mouses no Windows, pois isto auxiliaria na escalabilidade e performance da aplicação

Outro trabalho futuro interessante seria refazer esta aplicação utilizando outros sistemas operacionais para dispositivos móveis como o Android ou o Windows Phone.

Também existem uma série de incrementos que poderiam ser feitos no trabalho que tornariam a aplicação mais útil. A seguir seguem as principais ideias surgidas ao longo deste trabalho:

- Permitir salvar nos dispositivos móveis os arquivos criados no display grande;

- Possibilitar a abertura no display de arquivos previamente salvos ou criados nos dispositivos;
- Aumentar o número de ferramentas para desenhos;
- Acrescentar outras funcionalidades à aplicação além da capacidade de desenhar, como por exemplo, permitir a escrita de documentos de forma coletiva.

REFERÊNCIAS

ANATEL. Disponível em : www.anatel.com.br . Acesso em Nov. 2011

BALLAGAS, R.,BORCHERS J., ROHS M., and SHERIDAN J.G. " **The Smart Phone: A Ubiquitous Input Device**". IEEE Pervasive Computing. 2006, Vol.5, 1

COCOS2D. Disponível em : http://www.cocos2d-iphone.org/ . Acesso em Nov. 2011

FINKE, M.,KAVIANE, N., WANG, I.,TSAO,V., FELS, S., Lea, R. **"Investigating distributed user interfaces across interactive large displays and mobile devices"**. In Proceedings of the International Conference on Advanced Visual Interfaces, ACM Press (2010), 413-413.

iOS App Programming Guide. Disponível em : http://developer.apple.com/library/ios/#documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html#//apple_ref/doc/uid/TP40007072-CH4-SW20 . Acesso Nov. 2011

iOS Human Interface Guidelines. Disponivel em: http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html Acesso Out. 2011

iOS Overview. Disponível em : http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/URL_iPhone_OS_Overview/_index.html . Acesso Nov. 2011

JEON, S., HWANG, J., KIM, G.J., BILINGHURST, M. **"Interaction techniques in large display environments using hand- held devices"**. In Proceedings of the ACM symposium on Virtual reality software and technology (VRST '06). ACM, New York, NY, USA, 100- 103.

JIN, C., TAKAHASHI, S., and TANAKA, J., **"Interaction Between Small Size Device and Large Screen in Public Space"**. KES (3) 2006. pp. 197-204.

KHRONOS Opengl ES. Disponível em : http://www.khronos.org/opengles/ . Acesso em Out. 2011

MADGWICK, S. O. H., HARRISON, A. J. L., and VAIDYANATHAN, R. **"Estimation of imu and marg orientation using a gradient descent algorithm"**. In Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on (29 2011-july 1 2011), 1–7.

Sarah M. Peck, Chris North, Doug Bowman, **"A multiscale interaction technique for large, high-resolution displays,"** 3dui, pp.31-38, 2009 IEEE Symposium on 3D User Interfaces, 2009

Shupp, L., R. Ball, B. Yost, J. Booker, and C. North**" Evaluation of**

viewport size and curvature of large, high-resolution displays", in

proc. Graphics Interface (GI'06). 2006, p. 123-130

W/MCCANN. Disponível em : http://www.slideshare.net/WMcCannBR/consumidor-mvel-2011

WEISER, Mark. **"The computer for the 21st century"**. Scientific American, September 1991.

WIKIPEDIA Android Operation System. Disponível em: http://en.wikipedia.org/wiki/Android_(operating_system). Acesso Dez. 2011

WIKIPEDIA iOS. Disponível em: http://en.wikipedia.org/wiki/IOS#cite_note-2 . Acesso em Out. 2011

WIKIPEDIA iOS Version History : Disponível em : http://en.wikipedia.org/wiki/IOS_version_history#References . Acesso Dez. 2011

WIKIPEDIA Siri(Software). Disponível em: http://en.wikipedia.org/wiki/Siri_(software). Acesso Dez. 2011

WIKIPEDIA Ubiquitious Compitong: Disponível em: http://en.wikipedia.org/wiki/Ubiquitous_computing . Acesso em Dez . 2011

. Acesso em Dez. 2011

WIKIPEDIA Windows Phone. Disponível em : http://en.wikipedia.org/wiki/Windows_Phone . Acesso Dez. 2011