Avaliação de interfaces baseadas em técnicas de visão computacional

Danilo Eiji Seki
Thiago Schumacher Barcelos
Orientador: Prof. Dr. Carlos Hitoshi Morimoto

27 de fevereiro de 2003

Monografia de conclusão
do
Bacharelado em Ciência da Computação
do
Instituto de Matemática e Estatística
da
Universidade de São Paulo

Sumário

1 Introdução

Uma das áreas de pesquisa de grande interesse no campo da Interação Humano-Computador é o aumento da velocidade da interação. A entrada nos sistemas atualmente em uso é limitada à escrita através de um teclado e ao apontamento, através de dispositivos tais como o mouse, canetas ópticas ou *touch screens*. Tais métodos de entrada são insuficientes para atender às necessidades de novos dispositivos computacionais como, por exemplo, computadores pessoais portáteis. Ainda, novos paradigmas de interação, como a realidade virtual, criam novas necessidades no desenvolvimento de dispositivos de interação.

Uma possibilidade em estudo há vários anos é o desenvolvimento de rastreadores de olhar, ou seja, dipositivos capazes de indicar a posição observada pelo usuário, em geral sobre a tela de um monitor. Este trabalho baseia-se em um rastreador de olhar descrito em [1] e no *MAGIC Pointing* [2], que é uma aplicação do rastreamento de olhar nas interfaces WIMP (Windows, Icons, Menus and Pointing). O *MAGIC Pointing* usa a informação da posição observada pelo usuário para acelerar o controle do cursor do mouse.

Infelizmente, a precisão atualmente obtida pela tecnologia de rastreamento cria alguns obstáculos ao uso da informação da posição observada, exigindo o desenvolvimento de técnicas de ajuste dos dados fornecidos pelo rastreador. Esse é exatamente o objetivo do MAGIC Pointing – em [2] foram descritas duas estratégias de posicionamento do mouse a partir da posição observada. O objetivo desse trabalho foi a implementação e teste de uma nova estratégia de posicionamento que, com o objetivo de melhorar o ajuste, leva em consideração mais informações sobre o alvo de interesse usual do usuário nas interfaces atuais – os *widgets*, ou controles.

2 O problema e a proposta

2.1 A tecnologia de rastreamento de olhar

A tecnologia de rastreamento de olhar empregada neste trabalho usa uma câmera sensível à luz infra-vermelha que possui dois conjuntos de luzes: um próximo ao eixo óptico e um afastado. O conjunto central acende intermitentemente, gerando dois tipos de imagem de olho: um com a pupila iluminada pelo primeiro conjunto de luzes (como no efeito que conhecemos como o efeito de "olho vermelho" em fotografias), e um com a pupila escurecida. Um processo de subtração byte a byte das imagens é realizado, obtendo-se uma figura binária (contendo apenas cores pretas e brancas), onde aplica-se um

algoritmo de reconhecimento de formas, identificando a forma da pupila e do brilho ocular gerado pelo segundo conjunto de luzes.

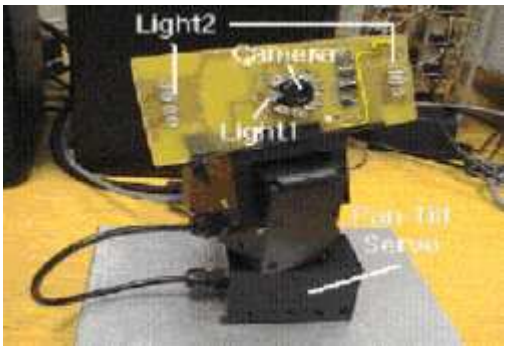

Figura 1: Câmera sensível a infra-vermelho usada no projeto

A partir da posição encontrada da pupila (P_{pupil}) e do brilho ocular (P_{glint}), é possível obter o vetor $P_{pupil} - P_{glint}$, que indica a posição observada pelo usuário, após um processo de calibragem: o usuário é solicitado a olhar para nove posições na tela do monitor, e a partir de um processo de aproximação usando um polinômio do segundo grau, obtem-se uma transformação de um vetor $P_{pupil} - P_{glint}$ para um ponto na tela do monitor.

A pergunta que surge é: como utilizar essa informação recém-gerada para melhorar a interação? A aplicação imediata que surge é usar essa informação para o controle do cursor do mouse. Estudos mostram que a aquisição de um alvo (*target acquisition*) de apontamento exige, dentre outras tarefas, uma busca visual do alvo desejado seguida por uma tarefa motora. Teoricamente, existindo uma maneira de eliminar a tarefa motora, uma parcela de tempo necessária para obter o alvo seria eliminada.

Porém, não é o que pode acontecer atualmente na prática por uma série de fatores. Antes de tudo, o olho não é uma "ferramenta" precisa como as mãos, por exemplo, estando sujeito a uma série de pequenos movimentos involuntários. Existe ainda um grau de imprecisão causado pela aproximação feita pelo processo de calibragem. Ainda, os modelos mais eficientes de cali-

Figura 2: Processo de subtração de imagens

bragem existentes atualmente exigem que a cabeça do usuário fique parada, causando um certo grau de desconforto no uso do sistema.

2.1.1 MAGIC Pointing

A proposta do *MAGIC Pointing* é então conjugar, nas ações do usuário necessárias para efetuar o apontamento, a busca visual do alvo e a tarefa motora, diminuindo a necessidade de realização dessa última em vez de eliminá-la. A informação do rastreamento de olhar é usada para determinar a região aproximada de interesse do usuário, enquanto que o mouse é usado para fazer o ajuste fino da posição do cursor.

Existem várias alternativas de definição da política de posicionamento do cursor a partir da posição indicada pelo rastreador. Já foram implementadas e descritas em [2] duas políticas: uma faz a movimentação do cursor para a posição observada somente quando o usuário movimenta o mouse, e outra movimenta o cursor sempre que a posição observada afasta-se além de um certo limite da posição do cursor do mouse.

2.2 Nossa proposta

As políticas já implementadas atuam sempre sobre a posição informada pelo rastreador, sujeita as várias formas de imprecisão descritas anteriormente. Com o objetivo de atenuar o efeito dessas imprecisões, a proposta deste trabalho é agregar ao MAGIC Pointing mais informação a respeito da interface sobre a qual ele atua.

Mais precisamente, a proposta é agregar ao sistema informações sobre os *widgets* que compõem a interface – afinal, é sobre cada ícone, botão ou caixa de texto que está focada a atenção visual do usuário ao realizar suas tarefas. Considerando que a dimensão da maioria dos widgets usuais está próxima dos limites de precisão do sistema, buscamos maior eficiência e conforto na execução de tarefas reais nas interfaces gráficas.

3 Organização

Como o projeto foi desenvolvido por uma equipe, é importante descrever a organização desse grupo de trabalho. Nossa equipe, na verdade uma dupla, foi formada pelos alunos Danilo Eiji Seki e Thiago Schumacher Barcelos. Todo o desenvolvimento do projeto foi feito em etapas de desenvolvimento paralelo com eventuais desenvolvimentos em conjunto.

Durante um período inicial, Thiago estudou a linguagem de programação C++ e principalmente todo o código-fonte da versão existente do programa. Daí para frente, alternamos os dois métodos de desenvolvimento. Nos reunimos para discutir o projeto e desenvolvermos em conjunto, lado a lado, no laboratório. Desse modo, pudemos nos comunicar, nos conscientizando das idéias um do outro e tomar decisões em conjunto, sempre discutindo e levando em consideração ambas as opiniões.

No final do dia, discutíamos o que foi feito e o que precisaria e poderia ser feito em paralelo. Normalmente, foram realizados nesses períodos pesquisas sobre a estrutura do sistema operacional, pesquisas e testes sobre a API (biblioteca de desenvolvimento) do mesmo e, logicamente, desenvolvimento em seções distintas do código, que poderiam ser facilmente unificadas. Após algum tempo trabalhando individualmente, marcávamos sempre que podíamos novas sessões de trabalho simultâneo, para discutimos e definir novas responsabilidades.

4 Ferramentas e técnicas

O projeto foi desenvolvido tanto no laboratório do IME como nas residências dos participantes. Os recursos de hardware no início do projeto eram uma câmera de espectro infravermelho com LED's infravermelhos e um computador no laboratório e, nas residências de cada um, seus respectivos computadores. Durante esse período, como o equipamento disponível não era "poderoso"suficiente para executar o aplicativo com desenvoltura, concentramos nossos esforços em aprendizado e pesquisa em casa. De acordo com o previsto, novos computadores foram adquiridos e instalados no laboratório. Por nós, foram utilizados principalmente dois computadores, Latin-1 e Latin-2, com poder computacional suficiente para executar o Eyetracker e o Magic Pointing sem problema algum.

Latin-1 e Latin-2 eram os nomes das máquinas no ambiente Windows XP, no qual o projeto foi desenvolvido. Além disso, ambas as máquinas possuíam configurações idênticas: processador AMD Athlon XP 1600+, 512 MB de RAM e placa de captura de vídeo da Pinnacle. Estando em rede e possuindo os mesmos aplicativos de desenvolvimento instalados, as máquinas foram utilizadas indistintamente. Havia preferência apenas pela máquina na qual estava, em cada momento, conectada a câmera e, nesta última, a fonte de energia, pois desse modo não haveria necessidade de se trocar as conexões.

Na aspecto de software, foram utilizados diversos programas, cada um com sua finalidade específica. Para programação e desenvolvimento (desenho) da interface, foi utilizado o programa Microsoft Visual C++. Os pro-

gramas Tera Term e puTTy foram utilizados para acessar a Rede Pró-Aluno do IME e para transferir arquivos dela e para ela. Finalmente, para navegar na Internet, ler e enviar eMails e, principalmente, consultar documentações on-line, foi utilizado o Microsoft Internet Explorer.

Outro aspecto muito importante de software é a biblioteca de reconhecimento de imagens. Versões antigas do programa utilizavam a Microsoft Vision SDK, uma biblioteca experimental desenvolvida por grupos de pesquisa da Microsoft. No entanto, ramificações mais recentes do programa já fornecidas para nós utilizam uma outra biblioteca, a OpenCV, da Intel. Dentre algumas vantagens podemos citar o fato da OpenCV ser disponibilizada sob uma licença de código aberto e ter uma versão para Linux, já que nosso orientador tem planos de portar o sistema para essa plataforma.

5 Implementação

Após a etapa de aprendizado e familiarização, iniciamos a implementação do projeto. Tendo recebido do nosso orientador uma versão do programa para modificar, começamos por implementar políticas de posicionamento do mouse. A partir dela, utilizando nossos conhecimentos, criamos então três políticas de posicionamento para testar o comportamento do sistema. Duas dessas políticas dividiam a tela em nove partições e, a partir da posição indicada pelo rastreador, moviam o cursor do mouse para o centro ou canto superior esquerdo de cada partição. Uma terceira política ainda movia o mouse apenas quando a posição observada estava a uma determinada distância da posição atual do mouse. Verificamos que o cursor do mouse permanecia muito "instável"com essas políticas, e que maiores refinamentos seriam necessários.

Logo depois tomamos contato com uma implementação anterior do MAGIC Pointing, que não sabíamos até aquele momento que existia. Essa implementação já tinha vários filtros dos dados oriundos do rastreador implementados, além das políticas descritas em [2]. Porém, sua implementação foi feita pensando-se na integração com uma versão mais antiga do rastreador. O código estava estruturado para compilar como uma DLL (Dynamic Link Library); após algum trabalho de adaptação, esse código estava em funcionamento com a nova versão do rastreador.

Partimos, então, para a implementação da nova extensão. O primeiro passo foi implementar um handler para um evento de janela gerado pelo sistema operacional. Esse handler é chamado sempre que uma janela é ativada, movida ou redimensionada, e é útil porque esse é o momento para obter novamente infomações sobre o posicionamento dos widgets. As informações sobre

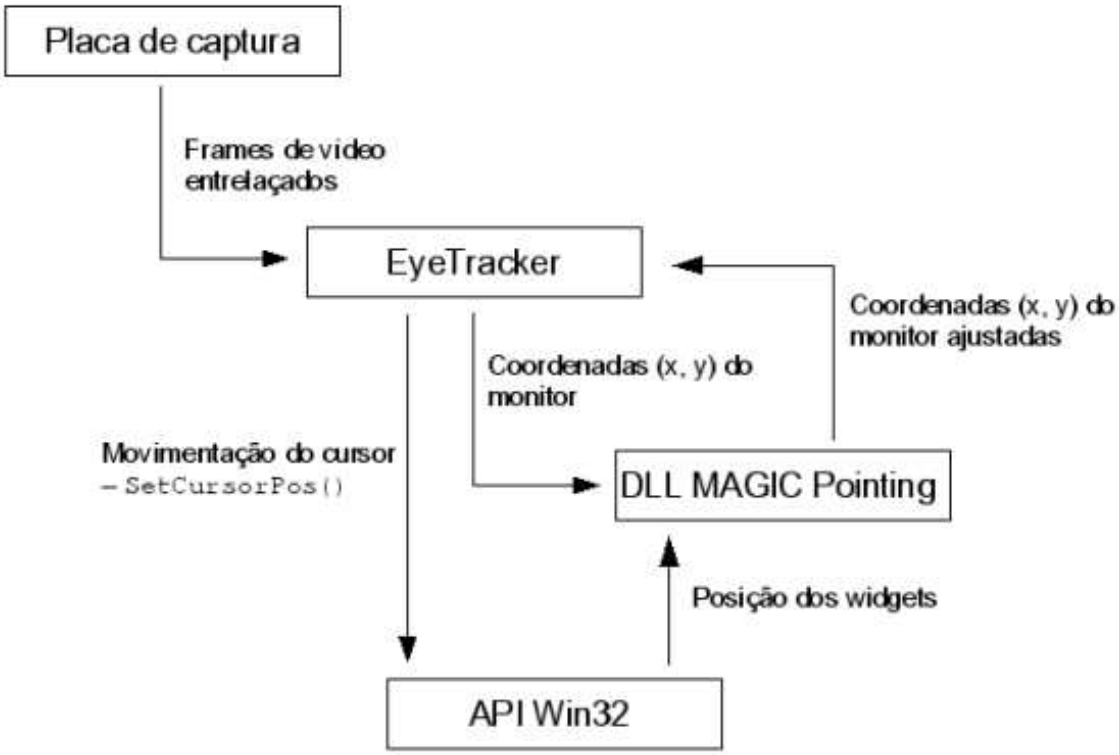

Figura 3: Diagrama de fluxo de dados entre os componentes do sistema

widgets são mantidas em uma lista linear, para que a cada dado recebido pelo MAGIC Pointing não seja necessário fazer uma nova chamada à API. Foi necessário manter a lista linear em uma região de memória compartilhada entre processos pelo sistema operacional devido a uma curiosa particularidade do Windows. Por estar contido no código de uma DLL, o código do handler é mapeado no espaço de memória do processo que possui a janela que gerou o evento. Desse modo, se a lista não estivesse em uma região de memória compartilhada, o programa rastreador não encontraria a lista. O espaço de memória que contém a lista ligada deve, então, ser acessível a cada processo que gerar um evento descrito acima.

O handler então chama outra função que busca os widgets contidos na janela. A partir de um número inteiro, que é seu identificador único, utiliza funções da API para realizar um procedimento muito semelhante a uma busca em profundidade no que seria uma árvore hierárquica de widgets. Nesse momento os widgets que podem ser tratados pelo MAGIC Pointing são identificados e adicionados à lista. Note que no momento nosso programa ma-

peia apenas os objetos da janela ativa. Isso acontece porque assumimos que os widgets que não estão na janela ativa não serão apontados com grande freqüencia ou importância. Essa aparente negligência aumenta significativamente a eficiência do mapeamento, pois tratar os widgets de janelas de segundo plano tem vários complicadores, como objetos parcialmente visíveis, centros escondidos, prioridade para elementos de primeiro plano, etc.

A DLL tem quatro pontos de entrada, uma para cada política de posicionamento do cursor do mouse implementada.

Além disso tudo, fizemos uma alteração na calibragem do sistema. Pelo método original, o programa tentava capturar nove detecções válidas, ou seja, nove frames nos quais tanto o centro da pupila como o brilho ocular eram detectados com sucesso. O problema disso é que nem sempre é fácil conseguir uma detecção válida - devido principalmente a ruído na imagem e foco da câmera, quanto mais nove em seguida. Com isso, perdíamos mais tempo na calibragem do que no teste desejado. No método modificado, assumimos que próximo ao momento da detecção válida, houve pelo menos uma detecção válida anterior na qual o usuário estava observando o ponto em questão da calibragem. Com isso, reduzimos o número de erros na calibragem para zero, aumentando consideravelmente a eficiência de nossos esforços.

Finalmente, já preparando a fase de testes, alteramos levemente a interface gráfica do programa. Essencialmente, adicionamos a possibilidade de escolha da política de posicionamento sem a necessidade de se alterar o código fonte e recompilar o programa.

6 Estado final do projeto

Embora muito tenha sido feito, o projeto, infelizmente, não pôde ser concluído. Podemos dizer que vários fatores contribuíram para isso. Os mais significativos provavelmente são a má eficiência do uso tempo, o grande esforço desviado para outras disciplinas e talvez um objetivo um tanto mal dimensionado. Mas mesmo com tudo isso, várias etapas do projeto foram completadas, e tantas outras iniciadas ou já planejadas. Segundo o programa descrito no planejamento inicial, cumprimos a primeira e a segunda etapa. Respectivamente, nos familiarizamos com as ferramentas, o ambiente de desenvolvimento e o programa de rastreamento de olhar, e implementamos a extensão do MAGIC Pointer. Embora sejam apenas duas das oito partes que foram propostas, acreditamos que essas etapas representem grande parte do trabalho que havia sido previsto.

A primeira etapa, de aprendizado e familiarização com linguagens, ambiente e o programa em si, pode ser considerada totalmente completa. Após

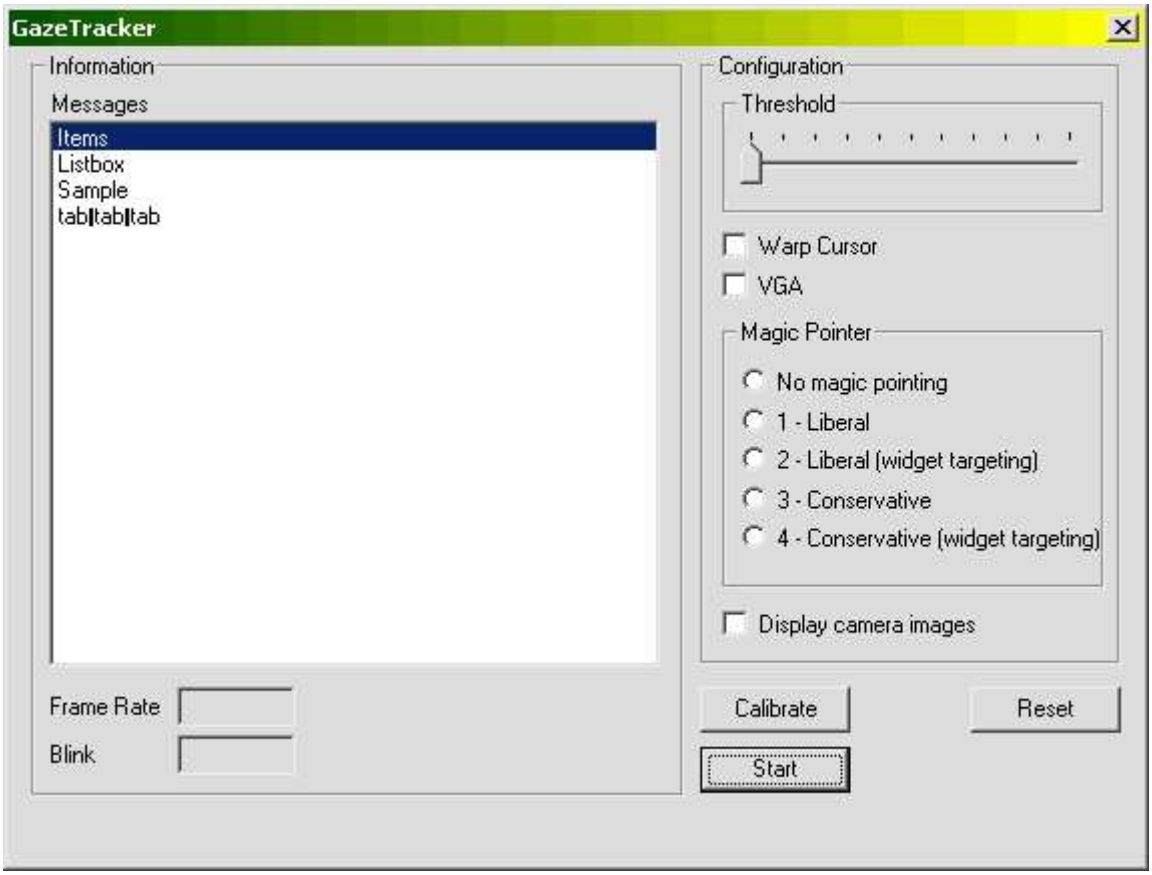

Figura 4: Nova interface do programa

vários meses em contato com o projeto, adquirimos proficiência básica a avançada em todas essas áreas de conhecimento. Por exemplo, a respeito da estrutura do Windows e seus métodos, a aquisição de conhecimento, a partir de agora, será através da leitura da documentação, com a qual já nos habituamos. Não nos esquecendo da linguagem C++, já conseguimos ler com certa facilidade programas escritos com ela e também programar utilizando alguns recursos e características específicas dela.

Quanto à implementação, segunda etapa do projeto, podemos resumir o que foi descrito na seção dedicada a esse assunto. Conseguimos modificar o código criando uma base para o reconhecimento de widgets que, apesar de estar reconhecendo e tratando um número limitado de widgets, pode ser facilmente extendida para responder perfeitamente a qualquer widget do sistema de janelas, desde que a API forneça informações suficientes sobre eles. Atualmente, reconhecemos botões, caixas de seleção (*list boxes*) e caixas de opção (*radio* e *check buttons*).

Apesar de iniciada a pesquisa das funções apropriadas da API, enfrentamos problemas em diferenciar algumas classes comuns de widgets. Em particular, não conseguimos diferenciar caixas de agrupamento, tratadas como botões pelo sistema de janelas Windows. Apesar de devidamente testado, o reconhecimento de widgets desabilitados ainda não foi incorporado ao código. Finalmente, o problema de manipular ícones da área de trabalho e botões de barras de ferramentas foi pesquisado e identificamos métodos que poderão resolver a questão, dependendo ainda de testes mais refinados.

As etapas seguintes à implementação seriam a modelagem de testes, a realiazação dos mesmos e a análise dos resultados. Embora não tenhamos iniciado oficialmente essa fase, já realizamos alguns trabalhos que a facilitarão. Por exemplo, foi feita uma rápida remodelagem da interface do programa e uma alteração no método de calibragem com o objetivo de deixar os testes menos confusos e, principalmente, menos cansativos para o usuário.

Também começamos a considerar algumas rotinas de teste que poderão ser utilizadas para avaliar o sistema. Basicamente, teríamos duas rotinas de teste. A primeira rotina verificaria a conformidade do apontamento auxiliado pelas informações sobre widgets com a já estabelecida Lei de Fitts. Para tanto, o usuário observaria (e posicionaria o mouse sobre) um objeto (provavelmente, um botão) no canto superior esquerdo na tela. Em seguida o usuário clicaria em outros objetos que apareceriam em posições com diferentes distâncias da posição original. Para testar o tamanho, exibiríamos objetos de diferentes tamanhos e em diferentes direções, mas a um mesmo raio de uma posição inicial. A segunda rotina consistiria de alguma tarefa usual do ambiente Windows (por exemplo, abrir uma aplicação ou copiar um arquivo), para que o usuário pudesse, com base em sua experiência anteri-

or no ambiente, dar sua opinião subjetiva sobre o novo sistema; se este lhe parece tornar a tarefa mais ou menos rápida, mais ou menos confortável...

Como se pode ver, o projeto está pronto para ser acabado, bastando algumas peças finais e, talvez, algum refinamento e ajuste das rotinas principais. Em seguida, já seria possível concluir as etapas de estudo de métodos de teste e os testes em si. Como é um projeto interessante e que, sobretudo, ainda lida com diversas tecnologias não abordadas no curso regular do Bacharelado em Ciência da Computação, acreditamos que futuros pesquisadores - ou alunos a procura de um trabalho de formatura - possam considerar a possibilidade de dar continuidade a ele, tanto ampliando a sua funcionalidade quanto desenvolvendo aplicações que utilizem o apontamento auxiliado pelo olhar.

7 Acompanhamento

O acompanhamento feito pelo professor doutor Carlos Hitoshi Morimoto teve um perfil liberal, ou seja, após uma discussão inicial sobre o projeto, o professor nos deixou livres para nos organizarmos. Essa abordagem permite, principalmente em trabalhos em grupo, o desenvolvimento da capacidade de planejamento pessoal e de atribuição de funções, em suma, da administração dos recursos humanos.

Mesmo com o acompanhamento liberal, o professor perguntou ocasionalmente sobre o andamento projeto e sobre as dificuldades, auxiliou no desenvolvimento do cronograma e marcou eventuais reuniões. Nessas reuniões, avaliamos o que já havia sido feito e o que restava fazer, e, quando necessário, discutimos e redefinimos etapas e cronogramas.

Mas esse modelo, como qualquer outro, possui seus defeitos. Como trabalhamos em dupla, discutimos a maior parte dos problemas entre nós mesmos, pecando por não consultar suficientemente o professor. Dessa forma, chegamos a despender tempo e energia para resolver problemas que já haviam sido resolvidos em outras ocasiões.

Levando-se em consideração os prós e os contras de tal tipo de acompanhamento, concluímos que essa abordagem é melhor do que auxiliar o estudante em cada detalhe, na medida que aumenta a flexibilidade e a independência do estudante, características muito importantes profissionalmente.

8 O BCC e o trabalho de formatura

Sendo um tema que exige uma análise pessoal e subjetiva, cada um dos integrantes do grupo de trabalho dá sua opinião separadamente nesta seção.

8.1 Desafios e frustrações

Danilo:

> Acredito que a maior dificuldade e o maior desafio tenha sido lidar com o sistema operacional Microsoft Windows. A totalidade das disciplinas e dos professores lidam apenas com sistemas do tipo Linux, então foi meu primeiro contato com aspectos mais estruturais do sistema operacional Windows. A grande dificuldade foi navegar e entender as estruturas e a forma de funcionamento das API's do sistema operacional, devido à má qualidade da documentação e às estruturas extremamente sem lógica e esquisitas.
>
> Outro aspecto importante foi o contato com a linguagem C++. Felizmente, por ter conhecimentos de C e orientação a objetos (sob a forma de Java), não houve um choque muito grande quando entrei em contato com a nova linguagem. A maior frustração é justamente nesse aspecto, pois apesar de considerar o conhecimento de C++ muito importante para minha vida profissional, não acredito que tenha obtido uma habilidade muito grande com a linguagem. Felizmente esse fato não teve um impacto significativo sobre o projeto.
>
> Não posso deixar de mencionar a dificuldade de se trabalhar em grupo, ainda com várias versões de código envolvidas, sem a existência de um sistema de controle de versão. Aprofundarei esse tema mais adiante na seção sobre trabalho em equipe.

Thiago:

> A maior dificuldade que enfrentamos, na minha opinião, foi de ordem técnica. O rastreador de olhar exige um computador rápido para processar a entrada de vídeo em tempo real (30 frames por segundo), e no início do projeto só tínhamos em nosso laboratório computadores com processador Pentium III equipados com Windows 98. Com eles o programa processava em média 18 frames por segundo, o que era insuficiente para conduzir o desenvolvimento. Com a promessa da instalação de computadores

mais rápidos, passei cerca de um mês e meio estudando o funcionamento do código; cheguei até a construir uma versão do programa que trabalhasse com bitmaps carregados do disco ao invés do sinal da câmera, apenas como exercício para adquirir familiaridade com o código.

Em agosto foram instalados dois computadores com processador Athlon XP e 512 Mb de RAM cada equipados com Windows XP, que permitiram que o programa processasse 30 frames por segundo; então, o trabalho de programação se intensificou. Daí para frente as dificuldades se concentraram em duas frentes: a API do Windows e o sistema de captura de vídeo.

A API do Windows trouxe alguma dificuldade ao projeto devido a diferenças entre a documentação e a implementação de algumas funções. Obter informações úteis sobre os widgets contidos em uma determinada janela não é uma tarefa muito trivial, e nem muito utilizada da API, por isso a documentação oficial da Microsoft era várias vezes vaga e confusa a esse respeito. Mais de uma vez constatamos, para nossa surpresa, que algumas funções simplesmente não devolviam os valores citados na sua especificação!

Por ser minha primeira experiência com um sistema de visão computacional, algum tempo foi gasto aprendendo como conseguir condições ideais de captura de imagens – descobri que um mínimo ajuste de foco ou um cabo de vídeo diferente podem causar mudanças sensíveis na precisão do sistema.

Para mim a maior frustração foi não ter conseguido conduzir experimentos mais aprofundados a tempo de serem incluídos nessa monografia. Infelizmente perdemos mais tempo do que o previsto para concluir a implementação, além de termos que lidar com aquela constumeira sobrecarga de tarefas do BCC ao final do segundo semestre.

8.2 Disciplinas relevantes

Danilo:

Apesar do projeto ter um aspecto mais prático do que foi normalmente encontrado por mim no curso, pude ver exatamente o que significa conhecer a base e a teoria para aplicá-la nas mais diversas situações. Tudo começa com as matérias básicas, como Introdução à Computação, Princípio de Desenvolvimento

de Algoritmos e Estrutura de Dados, indispensáveis para bons métodos de programação, principalmente documentação, capacidade de leitura e interpretação de código e, não menos importante, conhecimento sobre os mais diversos tipos de estruturas de dados.

Outra disciplina importante foi Sistemas Operacionais, que facilitou o entendimento de toda a estrutura do sistema Microsoft Windows, ambiente de desenvolvimento do projeto. Ter cursado a disciplina me deu experiência e uma visão global do funcionamento de triggers, handlers e outros elementos que foram utilizados.

Não posso esquecer também que como o projeto foi desenvolvido em C++, linguagem não utilizada anteriormente, devo dizer que o conhecimento da linguaguem C adquirida nas disciplinas de introdução mencionadas anteriormente, juntamente com o conhecimento da linguagem Java, orientada a objetos, adquirida em trabalhos das disciplinas Engenharia de Software, Sistemas de Objetos Distribuídos e Princípios de Interação Homem-Computador.

Finalmente, a disciplina Princípios de Interação Homem - Computador teve um destaque final como uma área que me interessou e acabou servindo como objetivo principal no desenvolvimento do projeto em questão.

Thiago:

Acredito que a primeira disciplina a ser citada é MAC 446 - Princípios de Interação Humano Computador, por ter me ajudado a ver e rever vários conceitos importantes para a análise dos resultados da implementação. Em alguns momentos foi necessário aplicar conceitos vistos em MAC 422 - Sistemas Operacionais, como o funcionamento do gerenciamento e proteção de memória em sistemas operacionais modernos. Sendo um trabalho de programação efetuado em dupla, as disciplinas que trataram tanto da prática quanto do planejamento do desenvolvimento foram importantes – ou seja, MAC 332 - Engenharia de Software, MAC 211 - Laboratório de Programação e MAC 242 - Laboratório de Programação II, além de MAC 110 e MAC 122.

O ambiente e a dinâmica do desenvolvimento foram muito semelhantes às da elaboração de EPs do curso; se por um lado isso facilitou o desenvolvimento, por outro me privou de uma oportunidade de experimentar outras maneiras de desenvolver.

8.3 Trabalho em equipe

Danilo:

Eu acredito que não houve nenhum problema nesse aspecto. Mesmo que, apesar de colegas, nunca tivesse trabalhado com o Thiago em projetos, acadêmicos ou não, considero que tivemos uma boa comunicação e conseguimos trabalhar como um time. Mantendo sempre o outro a par da situação e das atividades sendo realizadas, conseguimos nos direcionar e planejar adequadamente evitando desperdício de tempo e energia. Conseguimos também trocar idéias livremente. Isso se deve ao fato de tanto eu como o Thiago, sob meu ponto de vista, estarmos abertos às opiniões um do outro e sempre as considerarmos seriamente nas tomadas de decisões.

Em minha curta experiência profissional (não-acadêmica), pude perceber que esse tipo de abordagem sempre favorece o produto final, tanto no aspecto de qualidade como no aspecto de custo. Naquele ambiente empresarial, que inibia o compartilhamento e o aproveitamento de idéias individuais, percebia-se claramente que o produto final acabava sendo prejudicado. Presenciei diversas vezes idéias boas que diminuiriam drasticamente o custo de sistemas ou melhorariam muito sua usabilidade ou funcionalidade serem descartadas. Normalmente isso acaba acontecendo por timidez da pessoa que teria a sugestão, por lapso dos gerentes ou até mesmo por disputas políticas internas.

Deixando o relacionamento com a equipe de lado, gostaria de ressaltar a importância de um sistema de controle de versão quando desenvolvendo projetos de médio ou grande porte. Tive meu primeiro contato com o sistema CVS (*Concurrent Versions System*) no desenvolvimento do projeto da disciplina Sistemas de Objetos Distribuídos. Prevendo o tamanho do projeto, nosso grupo foi incentivado pelo membro Rodrigo Vieira Couto a desenvolvê-lo num ambiente de CVS. Com aquele projeto, eu, que era muito cético quanto aos benefícios de algo do tipo, vi o quanto tão sistema melhorava a produtividade do grupo. Aplicando mais algumas vezes no trabalho ou em projetos pessoais, hoje acredito que a habilidade em utilizar sistemas desse gênero seja essencial para a formação de um profissional da área. Tudo isso está sendo relatado aqui porque a existência de uma central de controle de versões definitivamente teria aumentado muito a produtividade deste projeto. Recomendo, inclusive, que tal gênero

de conhecimento seja incluído em alguma disciplina do currículo, Laboratório de Programação, por exemplo.

Em suma, fiquei muito feliz por ter conseguido participar desse modo da equipe, mas consciente de que o nosso caso não é algo comum. Afinal, já não é fácil encontrar equipes com apenas dois membros, muito menos com tais membros compartilhando dessa mesma ideologia. Para finalizar, gostaria de enfatizar a importância do uso de sistemas de controle de versão, já que a capacidade de tabalhar em equipe é algo extremamente importante em qualquer campo de atuação, seja ele acadêmico ou não.

Thiago:

Posso dizer que, ao contrário de outros trabalhos em equipe no curso, foi uma experiência muito positiva trabalhar com o Danilo (sem rasgação de seda... `:-)`). Eu fui o primeiro a entrar em contato com o nosso orientador e saber da possibilidade de projeto com o EyeTracker; desde esse momento eu achava que, pela dimensão inicial e pela própria natureza do projeto, contar com alguém para discutir caminhos e soluções do projeto poderia melhorar bastante a qualidade do resultado final. Ainda, com a intenção inicial de efetuar testes com usuários, seria possível obter mais dados, já que cada um de nós poderia conseguir mais "cobaias" nas horas livres.

Apesar de ainda não termos conseguido efetuar os testes, e de que o nosso tempo livre foi, para variar, *bem* menor do que eu imaginava, minha intuição inicial se concretizou. No começo do projeto, quando ainda nem tínhamos acesso ao código fonte do MAGIC Pointing em si, tivemos que "explorar" bastante o código fonte para entender o funcionamento do programa, e nossas discussões evitaram muita perda de tempo que poderia acontecer se eu se estivesse trabalhando sozinho e entendesse errado o funcionamento de algum trecho de código. Quando o trabalho de desenvolvimento da extensão do MAGIC Pointing se intensificou, a divisão de tarefas não foi difícil, porque apesar de não ter feito nenhum trabalho com o Danilo anteriormente, já sabíamos com quais tarefas cada um teria mais afinidade e, portanto, mais chance de cumprir com sucesso. Eu já tinha algum conhecimento sobre a API do Windows, portanto ficou comigo a tarefa de descobrir funções que ajudassem a nossa implementação, e testá-las antes do uso no código do projeto (como já citei anteriormente, nem sempre a documentação cumpre o que promete), enquanto

que o Danilo cuidou mais de implementação das funcionalidades no código do projeto. Acredito que essa estratégia de testar algumas funcionalidades fora do código do projeto nos poupou tempo precioso, ou pelo menos algumas dores de cabeça. Como não pudemos contar com o apoio de um CVS (o "sonho dourado" do Danilo), tentamos organizar o desenvolvimento através de um *log* detalhado de alterações e funcionalidades a implementar. Assim nós dois sabíamos o que cada um modificava no código, e sempre que retomávamos o trabalho depois de alguns dias estudando para provas ou fazendo algum EP, lembrávamos rapidamente qual era o próximo passo.

8.4 Conclusões pessoais

Danilo:

Acredito que avaliando o que foi descrito nos tópicos anteriores, é possível perceber que quase a totalidade das tecnologias e conhecimentos utilizados foram adquiridos, de certa forma, durante o desenvolvimento do projeto. Portanto, apesar de eu ter utilizado em grande escala os conhecimentos acadêmicos adquiridos ao longo do curso, aprendi muito sobre áreas diferentes que não foram diretamente abordadas. No meu ponto de vista, isso é bom e está de acordo com o objetivo do curso que, no meu entender, é dar ao aluno uma base genérica e ampla de conhecimentos com o intuito de facilitar o entendimento e aprendizado de qualquer tecnologia, nova ou antiga.

Algo definitivamente valioso foi a experiência de se trabalhar com projetos científicos. O ponto decisivo que me convenceu a escolher uma iniciação científica como projeto de formatura e não um estágio (minha segunda opção), foi o fato de que ter a oportunidade de se trabalhar com algo do gênero não é comum quando se está no mercado de trabalho real. Ou seja, resolvi fazer uma iniciação científica tendo em vista que atividades do gênero de um estágio seriam muito comuns e freqüentes no futuro, mas que iniciações científicas não, aproveitando portanto a oportunidade que se mostrava disponível.

Finalmente, apreciei bastante a disciplina e considero sua presença no currículo como essencial para a formação de um bacharel em ciência da computação. No entanto, acredito que o benefício seria relevantemente maior se o curso fosse extendido por um

quinto ano no qual a única atividade seria o projeto.

Thiago:

> Este foi para mim um projeto muito gratificante. Antes de tudo, desenvolver um projeto na área de Interação Humano-Computador era um desejo que tinha desde que entrei no BCC. Ainda, como procurei mostrar nessa segunda parte do trabalho, as dificuldades encontradas foram uma grande motivação para manter organizado e produtivo um projeto de duração razoavelmente longa, se comparada com a duração média do esforço dispendido nas tarefas do BCC.
>
> Por fim, como pretendo seguir meus estudos nessa área em nível de pós graduação, a disciplina proporcionou uma excelente oportunidade de conhecer de perto a linha de pesquisa no departamento.

A Agradecimentos

O Danilo e o Thiago gostariam de agradecer

Ao nosso orientador, por ter nos proporcionado a oportunidade de trabalhar em um projeto de formatura interessante;

Aos nossos amigos do BCC, que sempre deram sugestões ao projeto, e ajudaram dando conselhos em áreas que entendiam mais do que a gente. Em especial ao Rodrigo Moreira Barbosa, por ter nos ajudado quando a rede pifou, e ao André Gustavo Andrade, por ter sido uma cobaia paciente e "modelo" de uma das fotos de pupila dessa monografia;

Ao TCB (powered by SMB), a "maior e melhor"filosofia de desenvolvimento de software do mundo, que nos proporcionou muitas risadas durante o curso, e vai continuar proporcionando;

Aos nossos pais, familiares e amigos, que sempre acharam nosso trabalho intrigante e sempre perguntaram se o modo de clicar com o olho é "dando uma piscadinha"

A você, que leu esse trabalho até aqui!

Referências

[1] C. H. Morimoto, D. Koons, A. Amir e M. Flickner. Frame-rate pupil detector and gaze tracker. Em *To be presented at: ICCV'99 - Frame-rate computer vision applications*, Corfu, Greece, Setembro 1999 (URL: `http://wwww.ime.usp.br/~hitoshi/framerate`)

[2] S. Zhai, C. H. Morimoto e S. Ihde. Manual and gaze input cascaded (MAGIC) pointing. Em *Proc. of the ACM SIGCHI - Human Factors in Computing Systems Conference*, páginas 246-253, Pittsburgh, PA, Maio 1999.