Arquitetura para Realidade Aumentada em Plataformas Móveis

Daniel Ricardo Batiston, Dalton Solano dos Reis, Aurélio Hoppe

Departamento de Sistemas e Computação
Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

danielbatiston@yahoo.com.br, dalton.reis@gmail.com,
aurelio.hoppe@gmail.com

Abstract. *This article describes the architecture and development to built Augmented Reality applications on mobile device platforms. The architecture is based on OpenGL ES 1.1 library to create and draw some virtual objects, it uses also geolocation and features as images captured by camera, accelerometers and magnetometers to provide the user interaction. At the end we present an application example along with performance results running on iOS platfoms' devices, achieving acceptable results.*

Resumo. *Este artigo descreve a arquitetura e o desenvolvimento de uma biblioteca para a criação de aplicativos de Realidade Aumentada em plataformas móveis. Baseada na biblioteca OpenGL ES 1.1 para a construção e desenho de objetos virtuais, é utilizado também geolocalização e recursos como captura de imagens da câmera, acelerômetros e magnetômetros dos dispositivos para prover a interação com o usuário. Ao final é apresentado um exemplo de aplicação juntamente com os resultados de avaliações de desempenho feitas em dispositivos da plataforma iOS, atingindo resultados aceitáveis.*

1. Introdução

A Realidade Aumentada (RA) consiste numa interface computacional avançada, que promove em tempo real a exibição de elementos virtuais sobre a visualização de determinadas cenas do mundo real, oferecendo um forte potencial a aplicações industriais e educacionais, devido ao alto grau de interatividade [Kirner 2004]. Áreas como construção civil, jogos, medicina, publicidade, design, entre outras, vem aplicando este conceito para o aperfeiçoamento de suas técnicas. Esta crescente demanda torna o estudo da tecnologia indispensável para entender um pouco das técnicas e aplicações da RA, como registro e renderização de objetos em tempo real.

No desenvolvimento de aplicações de RA, são utilizadas bibliotecas computacionais que implementam a captura de vídeo, técnicas de rastreamento, interação em tempo real e os ajustes visuais das cenas do mundo real e sobrepõem esta visão com elementos virtuais [Hosn 2009].

Dispositivos do tipo Head Mounted Display (HMD) vem sendo amplamente utilizada para aplicações de RA [Bimber and Raskar 2005]. No entanto, esses dispositivos possuem grandes limitações ópticas (limitando o campo e o foco de visão), técnicas (resolução limitada e a instabilidade das imagens) e do fator humano (pelo elevado tamanho e peso do dispositivo) [Bimber and Raskar 2005]. Outros

dispositivos visuais que utilizam projeção em telas simples ou circunvizinhas também são utilizados para exibir conteúdo em RA mas fazem com que o usuário tenha sua mobilidade reduzida ou bloqueada pelo tamanho dos equipamentos [Hosn 2009]. Por outro lado, com os dispositivos móveis, o usuário possui maior controle em relação ao espaço onde terá sua RA, uma vez que poderá movimentar seu dispositivo em qualquer direção. Todavia, suas restrições estão no processamento, memória e bateria, impactando na renderização em tempo real que proporciona a visualização dos objetos virtuais e a sensação de imersão.

A RA em dispositivos móveis fornece basicamente dois tipos diferentes de experiências para o usuário: Geolocalização baseada em RA e Visão baseada em RA. A primeira utiliza recursos como Global Positioning System (GPS), bússola e outros sensores de um dispositivo móvel para fornecer alertas, interação e informações sobre pontos de interesse. Já a Visão baseada em RA utiliza muitos desses mesmos sensores para exibir conteúdo digital em contexto com objetos do mundo real baseando-se na leitura de marcadores impressos ou objetos rastreados pela câmera dos dispositivos móveis.

De maneira geral, as plataformas para dispositivos móveis tais como iOS [Apple 2012] e Android [Android 2012] possuem GPSs em seus dispositivos, permitindo o uso da localização geográfica para registro dos objetos virtuais. Também os sensores de movimento, como acelerômetros e magnetômetros (bússola), estão disponíveis nestas plataformas, de forma a auxiliar o rastreamento dos objetos virtuais na medida em que o dispositivo se move nos eixos de *azimuth, pitch* e *roll*.

Em dispositivos móveis, tais eixos constituem o sistema de coordenadas cartesiano tridimensional do dispositivo físico e servem de referência para a medição de movimentos de rotação capturados pelos acelerômetros, conforme ilustrado na Figura 1.

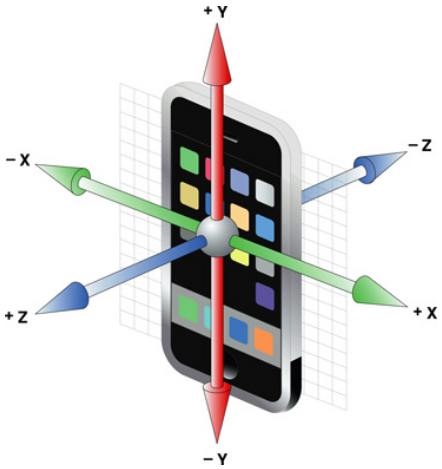

Figura 1. Eixos azimuth (x), pitch (y), e roll (z)

Além de possuir as limitações de hardware como processamento, memória e bateria, os dispositivos móveis oferecem um desafio grande para o desenvolvimento de aplicativos com RA. Por exemplo, as ferramentas de desenvolvimento incluem um simulador de dispositivo, mas que não faz a simulação da câmera e tão pouco dos sensores de movimento, tornando o simulador pouco útil para desenvolvedores de aplicativos de RA [Apple 2012].

É nesse contexto que este artigo apresenta o desenvolvimento de uma arquitetura para aplicações com RA dentro de dispositivos móveis, utilizando-se da câmera de vídeo para captar imagens do ambiente real, o GPS para registro dos objetos virtuais e os sensores de movimento para rastreamento da movimentação do dispositivo, bem com testes no equipamento da plataforma iOS.

O restante deste artigo está organizado da seguinte forma: a seção 2 apresenta trabalhos correlatos de RA com localização geográfica. A seção 3 detalha a arquitetura do aplicativo apresentado neste trabalho, explicando as soluções propostas e as tecnologias envolvidas. A seção 4 descreve as funcionalidades de simulação geradas por este trabalho para o desenvolvimento de aplicativos de RA com o uso dos simuladores de dispositivos. A seção 5 tem como foco demonstrar e analisar os resultados obtidos a partir de testes feitos na arquitetura do aplicativo. Por fim, a seção 6 apresenta considerações finais sobre o projeto e direções para trabalhos futuros.

2. Trabalhos correlatos

No desenvolvimento desta pesquisa, foram encontrados trabalhos semelhantes a este que tivessem suporte no desenvolvimento de aplicativos com RA para plataformas de mobilidade. Desta forma, iremos dar enfoque nos trabalhos correlatos de arquiteturas móveis de RA com localização geográfica. Para a escolha de trabalhos correlatos foram considerados projetos para dispositivos móveis que também possuem a estratégia de registro de objetos através de coordenadas geográficas e a estratégia de rastreamento dos objetos através dos sensores.

O framework Layar [Layar 2012] disponibiliza um conjunto de ferramentas que permite o desenvolvedor criar aplicações com RA baseadas em localização geográfica ou marcadores físicos como objetos ou imagens especificas impressas. Possui um software a ser instalado no dispositivo móvel que não necessita de customização por parte do desenvolvedor de aplicativo. Os objetos virtuais são exibidos em contexto com objetos do mundo real através das imagens da câmera do dispositivo que são captadas em tempo real. Possui também recursos para o desenho de objetos virtuais em 3D, que podem ser acompanhados de recursos de multimídia, proporcionando uma melhor interação com o aplicativo.

Magnitude [Magnitude 2012] é um projeto acadêmico open-source que visa fornecer serviços práticos de RA pelos alunos do INSA Toulouse, criando uma estrutura modular e reutilizável para outras aplicações de RA. Sua arquitetura possui uma aplicação para dispositivos móveis da plataforma Android e também um servidor de aplicação Java Enterprise Edition (JEE) que informa os pontos de interesse mais adequados.

O Augmented Reality Tool Kit (ARToolKit) [ARToolWorks 2012] é uma biblioteca de programação Open Source multi-plataforma desenvolvida em C e C++ que dá suporte a programadores para o desenvolvimento de aplicações de RA. Ela utiliza o rastreamento óptico, para identificar e estimar em tempo real a posição e a orientação de um marcador (moldura quadrada desenhada em um papel) em relação ao dispositivo de captura de vídeo. Com esses dados é possível estimar a posição e desenhar os objetos virtuais sobre a cena utilizando motor de renderização desenvolvido com a biblioteca OpenGL.

O Junaio [Junaio 2012] é uma plataforma que permite aos usuários criar, explorar e compartilhar informações usando RA e baseados na localização de conteúdo. Contando com um aplicativo destinado a dispositivos móveis, permite que o usuário localize pontos de interesse em serviços de geolocalização como o Google Place [Google 2012] ou adquira informações sobre os objetos escaneados pela câmera do dispositivo sem serviços de busca ou canais da própria Junaio, mostrando em tempo real as informações relacionadas na tela.

O estudo de caso do presente artigo incorpora características presentes nos trabalhos correlatos como a busca online de informações utilizando um servidor de pesquisa web (Google Place) para fornecer os dados de pontos de interesse mostrados virtualmente ao usuário disponível no framework Junaio. Se assemelha ao Magnitude pois ambos fornecem uma biblioteca de desenvolvimento reutilizável para outras aplicações. E ao Layar por utilizar os conceitos de RA baseada em localização geográficas.

3. Arquitetura Proposta

A arquitetura proposta (Figura 2) possui uma organização baseada em componentes de forma a abstrair as camadas de hardware da camada que trabalha com a interface gráfica.

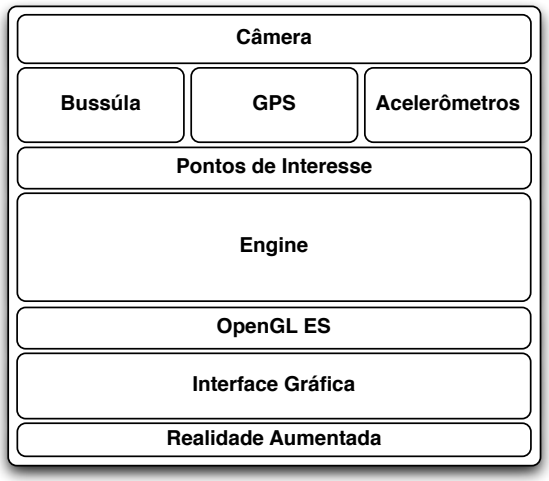

Figura 2. Arquitetura proposta

O componente chamado Engine é responsável por interagir com os recursos físicos do dispositivo fornecendo os valores e cálculos que representam cada ponto de interesse visualizado pelo observador. Estes cálculos são efetuados com base nas informações de mudança das coordenadas geográficas obtidas através do recurso do GPS do dispositivo, do ângulo da bússola que conseguimos através do recurso de magnetômetro e dos valores dos acelerômetros.

A Figura 3 mostra o ciclo de vida e os estados de monitoramento da aplicação. A Engine é iniciada com monitoramento através do recurso de GPS habilitado adquirindo assim a posição geográfica do dispositivo e o ângulo inicial da bússola. Neste momento a Engine passa a calcular o posicionamento em relação ao observador, de cada objeto virtual, incluindo aos objetos os valores de distância, ângulo, posição XYZ utilizadas pelo motor de renderização OpenGL ES, posteriormente.

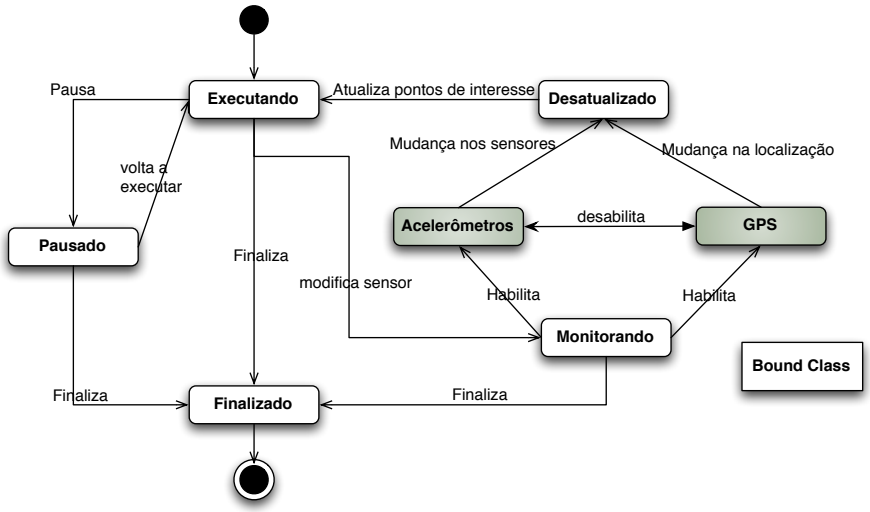

Figura 3. Ciclo de vida da aplicação e seus estados.

A Engine monitora através do recurso de GPS se a posição geográfica do aparelho está sendo alterada (indicando movimentação do observador), caso o dispositivo esteja parado a Engine desabilita os recurso de GPS e o alinhamentos entre os objetos virtuais e a cena real passam a ser atualizados pelo recurso de Giroscópio do dispositivo. O GPS é habilitado em intervalos de tempo definidos pelo desenvolvedor a fim de recalibrar a posição geográfica do dispositivo e verificar se ele entrou em movimento.

Qualquer alteração nos valores de localização monitoradas pela Engine, delega ao motor de renderização OpenGL ES a atualização dos objetos na tela. Este processo é executado apenas se mudanças significativas de posição ocorrerem com o dispositivo. A necessidade de implementar uma máquina de estados se deu principalmente pela preocupação com o consumo de bateria e tempo de resposta dos sensores de movimento e GPS.

Outro componente da arquitetura é a interface gráfica (Figura 4), composta por três camadas sobrepostas, dando a impressão de RA ao observador quando inseridas as informações sintéticas à cena real. A camada mais inferior (câmera) tem o objetivo de mostrar em tempo real as imagens capturadas pela câmera do dispositivo. Esta camada trata das configurações e interações com o hardware, fornecendo uma camada de visualização dos frames capturados pela câmera. A camada intermediaria (OpenGL ES) desenha os objetos virtuais e a camada mais externa (Ferramentas) mostra as informações da bússola, pesquisa de Pontos de Interesse e opções de configuração da aplicação.

A camada (intermediária) que desenha os pontos de interesse utiliza a API da OpenGL for Embedded Systems 1.1 [Kronos-Group 2012], preparando o contexto de renderização dos objetos e configurando as propriedades e variáveis de estado da biblioteca. Também tem a responsabilidade de realizar todas as transformações e desenhar os objetos virtuais de cada ponto de interesse no seu devido lugar da tela, com base nas informações calculadas na Engine.

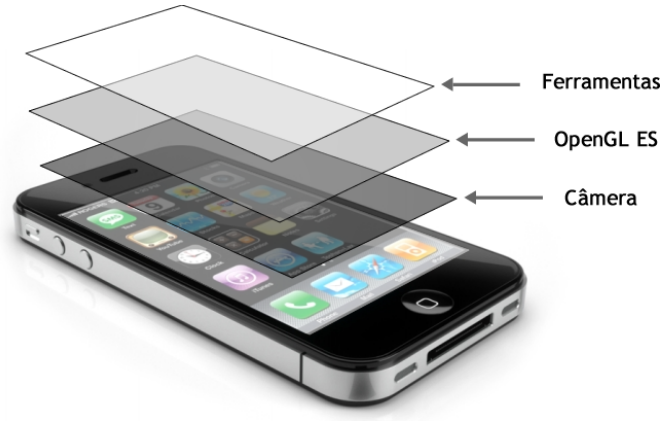

Figura 4. Camadas da interface

Dentre as funcionalidades para desenhar objetos disponíveis no OpenGL ES, foi escolhido para este trabalho, o desenho através de vértices, utilizando a projeção ortográfica. Apesar desta funcionalidade utilizar a memória principal do dispositivo, há a garantia dela ser suportada por todos os dispositivos que possuem OpenGL ES.

A arquitetura contempla ainda a funcionalidade de toque nos objetos virtuais através do visor do dispositivo. Para tal, foi utilizado para o algoritmo de Ray Picking que transforma o ponto do toque no plano bidimensional em uma reta no espaço tridimensional da OpenGL ES, permitindo assim o cálculo da colisão do toque com algum objeto virtual do espaço tridimensional [Zhao 2009].

Para este artigo criou-se uma aplicação de teste que desenha setas com o texto da distância entre o ponto de interesse e o dispositivo quando este estiver em modo ortogonal (Figura 5a) e painéis com texto do nome do ponto de interesse quando o dispositivo estiver em modo de perspectiva (Figura 5b). Estes objetos, são desenhados utilizando um conjunto de triângulos definidos em um vetor de vértices.

Figura 5. Modos do dispositivo – a) Modo ortogonal – b) Modo perspectiva

Para fornecer os pontos de interesse mais próximos da localização geográfica do dispositivo, foi criado uma interação com o serviço de geolocalização do Google (Google Place). Através da camada Ferramentas, o usuário pode utilizar um campo de pesquisa digitando um texto que identifica o Ponto de Interesse desejado. Com esta informação é gerada uma requisição HTTP do tipo GET enviada ao Servidor de Pesquisa incluindo também a posição geográfica atual do dispositivo e o raio de alcance máximo em metros pré-configurado pelo usuário.

As informações de localização geográfica como latitude, longitude, nome e endereço dos Pontos de Interesse mais adequados correspondentes aos parâmetros de pesquisa enviados pelo dispositivo é retornado então pelo Servidor, no formato JSON.

O aplicativo no dispositivo móvel, no caso, conecta-se com o Servidor de Pesquisa através do pacote de dados ou da rede wireless configurada no dispositivo.

4. Ferramentas de Desenvolvimento

As plataformas para dispositivos móveis, como iOS, Android e Blackberry possuem ferramentas que auxiliam o desenvolvimento dos aplicativos de forma que o desenvolvedor de uma forma geral não necessite adquirir um dispositivo para testar o seu desenvolvimento. Porém os simuladores de dispositivos possuem recursos limitados quanto à simulação de alguns recursos de hardware, tais como a câmera e os sensores. No desenvolvimento de aplicativos com RA, o uso de tais recursos é crucial, tornando o simulador uma ferramenta incompleta para se testar este tipo de aplicativo.

Uma vez que a arquitetura deste artigo trabalha de forma independente da plataforma móvel que será utilizada, o uso de simuladores com as funcionalidades de simulação da câmera, dos sensores e da localização geográfica, torna-se fundamental para o desenvolvimento, evitando assim, a aquisição de um dispositivo para cada uma das principais plataformas móveis trabalhadas. Devido às limitações apresentadas foram pesquisadas outras alternativas para realizar a depuração da biblioteca desenvolvida.

Uma das alternativas encontradas foi o software Accelerometer Simulator [Chrons 2010]. Esta ferramenta consiste em duas partes: o cliente, que é executado no dispositivo e a biblioteca, que são dois arquivos a serem vinculados ao projeto no Xcode. Este software permite a simulação do sensor de movimento transferindo os dados do acelerômetro do dispositivo para o computador através do protocolo UDP.

Outra ferramenta encontrada foi o iSimulator [Vimov 2011]. A exemplo do Accelerometer Simulator, esta solução também possui um cliente que deve ser instalado no dispositivo e uma biblioteca que deve ser ligada do projeto no Xcode, porém mostra-se mais completa, pois além de simular os dados do acelerômetro, possui outros recursos como a simulação do serviço de localização, streaming de vídeo e multitoque.

Apesar das alternativas encontradas, optou-se por efetuar a depuração e a execução da biblioteca desenvolvida utilizando o dispositivo smartphone da fabricante Apple chamado iPhone 4, que possui o sistema operacional iOS e permite testar a biblioteca nas reais condições de uso.

Para a implementação da biblioteca RA foi utilizada a linguagem de programação Objective-C, que é uma linguagem derivada do C e acrescida de algumas capacidades do Smalltalk. O ambiente de desenvolvimento utilizado foi o Xcode, disponibilizado pela Apple no suíte de ferramentas de desenvolvimento com o mesmo nome.

A simulação da localização geográfica foi desenvolvida disponibilizando uma tela de pesquisa na plataforma Google Place, obtendo assim, as configurações de latitude e longitude de cada Ponto de Interesse.

5. Resultados dos Testes

Os resultados obtidos pela biblioteca desenvolvida foram medidos através de testes experimentais da sua capacidade de renderizar cenas. Entende-se que a velocidade de renderização dos objetos virtuais é um fator crítico para o sucesso da biblioteca, pois o

atraso no desenho da cena pode causar a perda da noção de realidade pelo usuário e a impressão de que os objetos reais e virtuais estão desalinhados. O valor utilizado para analisar a capacidade de renderizar cenas foi a taxa de Frames Por Segundo (FPS), tendo em vista que este valor esta diretamente relacionado com a percepção do movimento de animação.

Para se ter mais controle dos resultados obtidos optou-se em restringir os testes com simulações na variação da quantidade de objetos desenhados ao mesmo tempo. Num segundo momento, após esta validação inicial da biblioteca, pretendesse fazer testes num produto voltado para a realidade de mercado.

Todos os testes foram executados da mesma maneira utilizando um dispositivo iPhone 4 sendo monitorado pela ferramenta de depuração Instruments existente no pacote de desenvolvimento da Apple. Foram realizadas medições da taxa de FPS em quantidades exponenciais de objetos sendo desenhados ao mesmo tempo. As medições foram realizadas utilizando objetos simples, formados por 2 faces e 6 vértices.

Para permitir ter controle dos testes realizados os recursos de captura de imagem da câmera foram desativados assim não interferiam na coleta de informações de FPS da Engine analisada, bem como as funções de atualização dos objetos com base na movimentação do dispositivo e sua localização geográficas foram substituídos por rotinas de atualização constante dos objetos em cena, tornando os testes padrões e controláveis.

Analisando a arquitetura implementada observou-se dois fatores que podem afetar o desempenho da biblioteca, sendo eles: a quantidade de objetos virtuais presentes na cena a ser renderizada, e a utilização de texturas. Os resultados obtidos são apresentados na Tabela 1.

Tabela 1. FPS em objetos com e sem textura

Quantidade de Objetos	FPS em Objetos Sem Textura	FPS em Objetos Com Textura
2	41	40
4	40	35
8	40	26
16	40	15
32	40	8
64	38	3
128	35	1
256	32	-
512	23	-
1024	12	-
2048	6	-
4096	3	-
8192	1	-

Para as medições com texturas, estas foram renderizadas antes do desenho de cada objeto e deletando a mesma ao final do desenho. Com base nas duas medições da Engine foi possível gerar o gráfico da Figura 6 que representa o desempenho obtido através da medição do FPS na vertical e da quantidade de pontos de interesse na horizontal.

Podemos observar que a quantidade de objetos tem impacto direto no desempenho do aplicativo em ambos os casos analisados, mas há uma queda significativa de desempenho ao ser utilizadas renderizações com texturas.

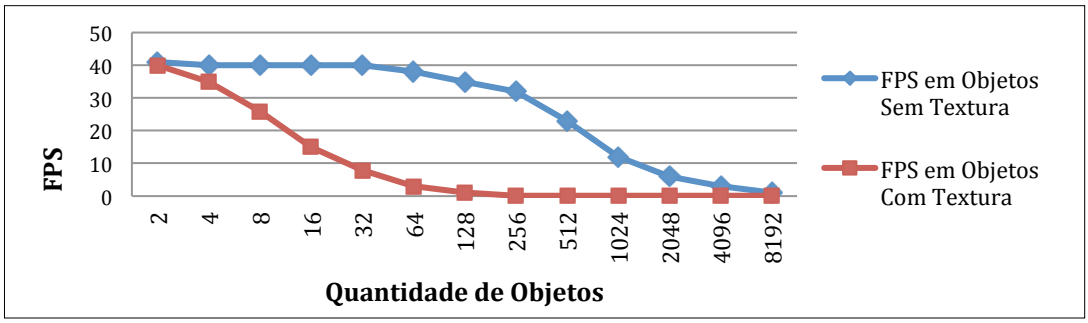

Figura 6. Comparativo de desempenho

A Engine se mostrou com alto desempenho gráfico ao serem utilizados objetos sem textura mesmo com 256 objetos desenhados simultaneamente. Por outro lado a renderização de texturas se mostrou um gargalo da Engine apresentando taxas satisfatórias com apenas 8 objetos renderizados, demonstrando que a estratégia de desenho pelo Open GL ES ainda precisa de otimizações.

6. Conclusão e Trabalhos Futuros

A conveniência e as funcionalidades oferecidas pelos dispositivos móveis, como Smartphones, tiveram como resultado a opção de muitas pessoas pelo uso da mobilidade. Com o crescimento do desenvolvimento para dispositivos móveis, inúmeras empresas e áreas de atuação estão incorporando aplicações móveis a seu dia a dia, afim de agilizar seus negócios, captar clientes, divulgar seus produtos e serviços.

O conceito de RA de adicionar elementos virtuais a um cenário real, utilizando-se de recursos tecnológicos como vídeo e monitoramento de movimentos dos dispositivos móveis, torna essa tecnologia aplicável a inúmeras áreas de conhecimento, contribuindo de maneira significativa na interação para os usuários.

Neste caminho de mobilidade cada vez mais interativa, este trabalho teve como objetivo a implementação de uma arquitetura juntamente com uma biblioteca computacional que auxilie os desenvolvedores na criação e implementação de recursos de RA em suas aplicações.

Este projeto concentrou-se em explorar a RA sobre dispositivos móveis, exibindo os recursos e restrições impostas por estes dispositivos em nível de poder de processamento e interface com o usuário, além de utilizar a biblioteca desenvolvida em uma aplicação com RA utilizando coordenadas geográficas para exibição de objetos virtuais, chamados pontos de interesse. A aplicação consistiu em capturar o vídeo da cena real através da câmera do dispositivo e aplicar sobre este os objetos virtuais, reagindo aos movimentos físicos realizados pelo usuário, acionados pelos recursos de bússola e acelerômetros do dispositivo.

As funcionalidades aqui apresentadas são semelhantes às dos trabalhos correlatos. Dentre elas pode-se destacar a determinação dos pontos de interesse através de coordenadas geográficas, acelerômetros e bússola; a visualização da realidade

através da câmera e da virtualidade através de objetos desenhados em um sistema em 3D; e a obtenção dos pontos de interesse através de um servidor web.

Foi possível demonstrar, a partir deste projeto, que é viável implementar algoritmos de RA em dispositivos móveis alavancando características e funcionalidades muito interessantes como acelerômetros, magnetômetros e câmera de vídeo. Também foi possível demonstrar que o uso da OpenGL ES pode ser usado para desenhar a imersão através de um espaço tridimensional, ainda que seu desempenho e otimização devam ser foco de estudo.

Como extensões deste trabalho é sugerida a utilização de VBOs (Vertex Buffer Object) como técnica de desenho, por se tratar de um método mais rápido e que permite otimizar o consumo de memória pela aplicação, pois neste caso é o OpenGL ES que gerencia a estrutura contendo os vértices do desenho. Outra extensão é portar a biblioteca desenvolvida para versão 2.0 (ou superior) do OpenGL ES com a finalidade de usufruir do ganho de desempenho proporcionado pela utilização de *shaders*. E ainda, poderia se pensar em portar a biblioteca desenvolvida para outros sistemas operacionais móveis (por exemplo Android ou Windows Mobile) e fazer testes comparativos.

Referências

Android. (2012), "Android 3.0 platform highlights", http://developer.android.com/about/versions/android-3.0-highlights.html, Outubro.

Apple. (2012), "iOS Dev Center - Apple Developer", http://developer.apple.com/devcenter/ios/index.action, Outubro.

ARToolWorks. (2012), "ARToolWorks Framework for mobile", http://www.artoolworks.com/products/mobile/artoolkit-for-mobile/, Novembro.

Bimber, O., Raskar and Spatial, R. (2005), "Augmented Reality: Merging Realand Virtual Worlds., Natick, MA, USA:A.K.Peters, Ltd.

Chrons, Otto. (2010), "Home of accelerometer simulator". http://code.google.com/p/accelerometer-simulator/wiki/Home, Novembro.

Google. (2012), "API do Google Places", https://developers.google.com/maps/documentation/places/?hl=pt-br, Setembro.

Hosn, R.N.A., Pereira, C. and Costa, J. (2009), "Immersive Virtual Reality. DISPOSITIVOS PARA REALIDADE VIRTUAL", http://www.oocities.org/raidhosn/cap04.htm, Novembro.

Junaio. (2012), "Junaio", http://www.junaio.com/, Novembro.

Kirner, C.; Tori, R. (2004), "Introdução a Realidade Virtual, Realidade Misturada e Hiper-realidade". In: Claudio Kirner; Romero Tori. (Ed.). Realidade Virtual: Conceitos, Tecnologia e Tendências. Sao Paulo, 2004, v. 1, p. 3-20.

Kronos-Group. (2012), "Open GL ES overview". http://www.khronos.org/opengles/, Novembro.

Layar. (2012), "An overview of the layar platform", http://www.layar.com/documentation/browser/, Outubro.

Magnitude. (2012), "Project magnitude", http://www.magnitudehq.com/, Outubro.

Vimov. (2011), “iSimulate: documentation”, http://www.vimov.com/isimulate/documentation/, Novembro.

Zhao, H., Jin, X., Shen, J., and Lu S. (2009), “Fast and reliable mouse picking using graphics hardware,” Int. J. Comput. Games Technol., vol. 4, pp. 1–7, Janeiro.