UNIVERSIDADE FEDERAL DE PERNAMBUCO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
CENTRO DE INFORMÁTICA

UM FRAMEWORK DE REALIDADE AUMENTADA PARA O DESENVOLVIMENTO DE APLICAÇÕES PORTÁVEIS PARA A PLATAFORMA POCKET PC

Trabalho de Graduação

Aluno: João Paulo Silva do Monte Lima (jpsml@cin.ufpe.br)
Orientadora: Judith Kelner (jk@cin.ufpe.br)
Co-Orientadora: Veronica Teichrieb (vt@cin.ufpe.br)

Recife, 29 de março de 2007

Resumo

Este trabalho tem como objetivo apresentar um *framework* que facilita o desenvolvimento de aplicações de realidade aumentada para a plataforma Pocket PC. Este *framework* foi projetado para propiciar a criação em alto nível de aplicações independentes de plataforma. Ele integra o OGRE, que é um engine gráfico *open source* de alto nível para renderização 3D em tempo real, e o OgreAR, uma biblioteca desenvolvida pelo autor que oculta os detalhes de implementação da aquisição de imagens da câmera e da detecção de marcadores. Além disso, o autor construiu um porte do *engine* OGRE para Pocket PC chamado OGRE4PPC, que é descrito detalhadamente. A arquitetura e o funcionamento do *framework* são apresentados ao longo deste trabalho. Finalmente, alguns estudos de caso são apresentados, onde o OGRE4PPC e o *framework* de realidade aumentada são avaliados utilizando-se um dispositivo móvel real.

Agradecimentos

Primeiramente, agradeço a Elidiane, meu amor, de quem tenho tanto orgulho e carinho, e com quem pretendo passar o resto da minha vida. Obrigado por me apoiar, por servir de estímulo em tudo que faço e por ser essa pessoa maravilhosa que você é. Te amo, pequena!

Agradeço a meus pais, Heraldo e Dileuza, por serem exemplos de pessoas dignas e por se esforçarem ao máximo para me dar a educação e a infra-estrutura necessária para que eu pudesse conquistar meus objetivos.

Agradeço a minhas irmãs queridas, Jennifer e Alessandra, por serem essas irmãs "babonas" que são, tentando sempre enxergar somente as virtudes desse irmão que possui um monte de defeitos. Obrigado por vocês estarem sempre ao meu lado, dispostas a me ajudar em tudo que preciso.

Agradeço aos meus avós, tios, primos e outros familiares, por sempre desejarem o melhor para mim.

Agradeço aos amigos do CIn, Roberto, Mozart, Cesar, Chico, entre tantos outros, pelos momentos de descontração e pelos "ping-pongzinhos" entre um projeto e outro. São todos "irmãos" que ganhei na faculdade e que nunca esquecerei.

Agradeço aos amigos do GRVM e do GPRT, Mouse, Joma, Gabriel, Curupa, entre tantos outros, com quem aprendi tantas coisas e também me diverti bastante. Um obrigado especial a Rodrigo Farias, por ser o criador do "momento Treloso", onde todos os colegas de grupo promovem uma confraternização regada a biscoitos de chocolate.

Agradeço à minha orientadora, a professora Judith Kelner, e ao professor Djamel Sadok, por terem me dado uma oportunidade única de crescer como profissional ao trabalhar num ambiente tão sadio e enriquecedor como é o GRVM.

Por fim, agradeço à minha co-orientadora, Veronica Teichrieb, por me ensinar tantas coisas e contribuir de maneira decisiva na minha formação profissional, além de ser a "chefe" mais gente boa que eu conheço!

Sumário

1. Introdução 7
1.1. Objetivos 7
1.2. Estrutura do documento 8
2. Conceitos básicos de RA 9
2.1. Captura de vídeo 11
2.2. Rastreamento de marcadores 11
2.3. Renderização 14
3. RA em dispositivos móveis 18
3.1. Exemplos de aplicações 18
3.1.1. Aplicações distribuídas para PDA 18
3.1.2. Aplicações autônomas para PDA 20
3.1.3. Aplicações distribuídas para celular 22
3.1.4. Aplicações autônomas para celular 23
3.1.5. Aplicações para consoles de jogos 25
3.2. Captura de vídeo 26
3.3. Rastreamento de marcadores 27
3.4. Renderização 27
3.5. Trabalhos relacionados 28
4. O *framework* de RA 29
4.1. A biblioteca OgreAR 29
4.2. A biblioteca OGRE4PPC: porte do OGRE para Pocket PC 31
4.2.1. Dependências 32
4.2.2. RenderSystem usando OpenGL ES 33
4.2.3. PlatformManager para Pocket PC 35
4.2.4. Implementação da matemática de ponto fixo 36
4.2.5. RenderSystem usando Klimt 37
4.2.6. RenderSystem usando aceleração em *hardware* 38
5. Estudos de caso e resultados 39
5.1. Avaliação do OGRE4PPC 39
5.2. Avaliação do *framework* de RA 41
6. Conclusões e trabalhos futuros 43
Referências 44

Índice de Figuras

Figura 1. Sistema de RA com registro do ambiente virtual e real usando GPS. 9
Figura 2. Fluxo típico de um sistema de RA baseado em vídeo. 10
Figura 3. Marcador do ARToolKit (esquerda) e aplicação de RA usando ARToolKit (direita). 12
Figura 4. Marcador do ARToolKitPlus que codifica o *ID* 0. 13
Figura 5. Robustez à oclusão parcial dos marcadores do ARTag (direita) em relação aos do ARToolKit (esquerda e centro). 13
Figura 6. Marcadores utilizados pelo MXR Toolkit. 14
Figura 7. Arquitetura do OGRE. 17
Figura 8. NaviCam utilizado para visualização de informações relativas a objetos da cena real. 19
Figura 9. Previsão de prédio a ser construído utilizando RA. 19
Figura 10. AR-PDA utilizado em aplicação de auxílio ao consumidor. 20
Figura 11. O jogo The Invisible Train para guiar trens de forma que não colidam. 20
Figura 12. O jogo Virtuoso para ordenar cronologicamente obras de arte. 21
Figura 13. O sistema SignPost para auxílio à navegação. 21
Figura 14. O jogo Kanji Teaching para aprendizado de caracteres chineses Kanji. 22
Figura 15. O álbum de recordações aumentado Digital Scrap Book. 22
Figura 16. ARPhone usado para visualização de interiores. 23
Figura 17. O gibi aumentado AR Comic Book (esquerda) e o AR Post-It para visualização de mensagens SMS (direita). 23
Figura 18. Mapa de ferrovia aumentado com informações sobre trens. 24
Figura 19. Nokia Virtual Video para visualização de informações sobre a cena real. 24
Figura 20. O jogo de ping-pong ARTennis (acima à esquerda), o jogo de montar Virtual Lego (acima à direita), o FPS Mozzies (abaixo à esquerda) e o jogo de disputa de pênaltis Kick Real (abaixo à direita). 25
Figura 21. A aplicação Vidente para visualização de elementos subterrâneos. 26
Figura 22. O jogo de estratégia Catapult. 26
Figura 23. Arquitetura do Studierstube. 28
Figura 24. Arquitetura do *framework* de RA para Pocket PC. 29
Figura 25. Aplicação de RA desenvolvida usando o OgreAR. 30
Figura 26. Arquitetura do OgreAR. 31
Figura 27. *Screenshots* dos demos do OGRE no Pocket PC: SkeletalAnimation (esquerda) e SkyBox (centro) usando o `RenderSystem` OpenGL ES e SkeletalAnimation usando o `RenderSystem` Klimt (direita). 40
Figura 28. *Frame rate* no dispositivo móvel usando diferentes `RenderSystems`. 40
Figura 29. Aplicação de RA no Pocket PC, desenvolvida com o *framework* de RA. 41
Figura 30. *Frame rate* no dispositivo móvel rodando a aplicação de RA e usando o `RenderSystem` baseado no Klimt. 42

Índice de Tabelas

Tabela 1. Mapeamento da entrada de teclado de aplicações do OGRE em dispositivos móveis. 36
Tabela 2. Mapeamento da entrada do *mouse* de aplicações do OGRE em dispositivos móveis. 36
Tabela 3. Mapeamento dos tipos matemáticos do OGRE em dispositivos móveis. 37

1. Introdução

Realidade Aumentada (RA) é a área da Realidade Mista (RM) que lida com a adição de elementos virtuais ao ambiente real em tempo real. Atualmente, RA é usada em diferentes campos de aplicação, tais como entretenimento, medicina, manutenção e educação [1]. Algumas dessas aplicações requerem que o usuário possa se mover livremente, exigindo a utilização de dispositivos portáteis [2]. Outra restrição comum neste contexto é a utilização de um dispositivo leve e compacto. Como decorrência deste fato, projetos de RA cujo alvo são dispositivos móveis tais como *Personal Digital Assistants* (PDAs) e *smartphones* estão se tornando populares [3].

As ferramentas disponíveis atualmente para construir aplicações de RA para plataformas embarcadas ainda não satisfazem as necessidades dos programadores. Um dos motivos é que as bibliotecas de renderização oferecem uma interface de programação de baixo nível, o que impacta a produtividade do desenvolvedor. Além disso, na maioria das vezes o mesmo programa desenvolvido para uma plataforma não pode ser usado em outras, visto que as bibliotecas utilizadas em cada plataforma são diferentes e incompatíveis.

1.1. Objetivos

Este Trabalho de Graduação apresenta um *framework* que possibilita a criação de aplicações portáveis de RA para a plataforma Pocket PC usando um *engine* de renderização de alto nível. O *engine* de renderização usado pelo *framework* é o *Object-oriented Graphics Rendering Engine* (OGRE), descrito em [4] e [5]. O OGRE é uma solução *open source* repleta de funcionalidades que simplificam o projeto de aplicações 3D em tempo real. A biblioteca é disponibilizada para várias plataformas, incluindo Windows, Linux e Mac OS. Os programas feitos com o OGRE podem ser executados em qualquer uma dessas plataformas sem mudanças.

Conseqüentemente, a biblioteca OGRE4PPC foi criada pelo autor, que é um porte do OGRE para a plataforma Pocket PC [6]. A biblioteca OgreAR também foi desenvolvida pelo autor para agir como uma camada de abstração para os recursos de RA usados no reconhecimento de marcadores e na aquisição de imagens [7]. Seu objetivo é garantir a portabilidade entre diferentes plataformas, que utilizam bibliotecas específicas para realizar tais tarefas. O ARToolKitPlus foi usado para rastreamento de marcadores no Pocket PC [8].

O *framework* implementado foi validado através de estudos de caso que contemplam tanto a renderização 3D como as tarefas relativas a aplicações de RA (rastreamento de marcadores e captura de vídeo). Foram comparados os resultados obtidos no Pocket PC ao se utilizar diferentes bibliotecas gráficas disponíveis para a plataforma.

1.2. Estrutura do documento

O capítulo 2 descreve os conceitos básicos relacionados a RA e enumera as principais ferramentas utilizadas na construção de aplicações nesta área. O capítulo 3 foca no desenvolvimento de aplicações de RA para dispositivos móveis, apresentando sistemas destinados a diferentes plataformas embarcadas, bibliotecas utilizadas na autoria de soluções e trabalhos relacionados ao *framework* proposto neste trabalho. O capítulo 4 explica como o *framework* está organizado e como foi o seu desenvolvimento. O capítulo 5 mostra estudos de caso, relativos a aplicações do OGRE e exemplos de RA, utilizando o Pocket PC, a fim de validar o *framework* proposto. As conclusões e trabalhos futuros são apresentados no capítulo 6.

2. Conceitos básicos de RA

O conceito de RM diz respeito à coexistência de elementos reais, pertencentes ao mundo em que os usuários vivem, e sintéticos, criados por computador, num mesmo meio. Dependendo da maneira com que tal combinação ocorre, em especial o grau de realidade e virtualidade presentes, as aplicações são classificadas em diferentes subáreas, entre elas Realidade Virtual (RV) e RA. Enquanto que em RV existe uma dominância por objetos virtuais, onde todo um mundo é modelado em computador, em RA o que se tem é o próprio mundo real mesclado com alguns artefatos sintéticos, adicionados em tempo real, de forma que os mesmos pareçam fazer parte do meio ao qual são introduzidos [9].

A tarefa que consiste em posicionar os objetos virtuais corretamente em relação ao ambiente real é chamada de registro. Para que a mesma possa ser realizada, torna-se necessário o uso de sensores que percebem as características do mundo e com base nesses dados determinam quando e como a cena deve ser apresentada. São utilizados para tais fins sensores de temperatura, ultrasom, *trackers* de movimento, sensores de vídeo, entre outros [10]. A Figura 1 mostra um sistema *outdoor* que utiliza um *Global Positioning System* (GPS) como sensor [11]. Enquanto o usuário anda pela rua vestindo um *Head Mounted Display* (HMD), são mostradas sobre a sua visão do mundo as coordenadas de sua posição, assim como gráficos informativos relacionados ao que está sendo visto. Além disso, as informações são atualizadas em tempo real e refletem a movimentação do usuário pelo ambiente real.

Figura 1. Sistema de RA com registro do ambiente virtual e real usando GPS.

Quanto à visualização do sistema pelo usuário, os sistemas de RA são classificados de duas maneiras: sistemas ópticos e sistemas baseados em vídeo. Nos sistemas ópticos o usuário utiliza um HMD óptico translúcido, obtendo uma visão direta do mundo sobre a qual a cena virtual é

renderizada. Nos sistemas baseados em vídeo o usuário enxerga, num monitor comum ou HMD opaco, o ambiente real através de uma filmagem feita por uma câmera, onde os elementos sintéticos são sobrepostos [10].

Atualmente, a grande maioria dos sistemas de RA utiliza vídeo tanto para sensoriamento como para visualização. A análise das imagens do mundo real pode se tornar uma tarefa complexa e custosa. Conseqüentemente, em muitos casos são utilizados marcadores fiduciais, que consistem em elementos sintéticos que são adicionados à cena real de forma a servir como referência para a posição e a orientação que a cena virtual deve possuir. Exemplos de marcadores podem ser vistos na subseção 2.2. Com o auxílio destes marcadores, é possível utilizar algoritmos de processamento de imagem para indicar a maneira como os objetos virtuais devem ser desenhados (qual objeto, em que posição e orientação) [10].

As aplicações de RA baseadas em vídeo que utilizam marcadores fiduciais para realizar o registro dos elementos virtuais comumente seguem o fluxo de funcionamento descrito na Figura 2. Primeiramente, a cena real é capturada por um sensor de imagem, como, por exemplo, uma *webcam*. Após isso, a imagem é obtida pela aplicação através de uma biblioteca de captura de vídeo. O próximo passo é rastrear a imagem em busca de marcadores fiduciais, utilizando técnicas de reconhecimento de padrões específicas. Uma vez que os marcadores são detectados, é feita a estimativa de posição e orientação dos respectivos objetos virtuais. Por fim, é realizada a renderização da cena virtual sobreposta à cena real, com o auxílio de uma biblioteca gráfica. O resultado final é exibido em um dispositivo de saída de vídeo, como um monitor ou um HMD. Na seqüência, cada uma destas etapas é descrita.

Figura 2. Fluxo típico de um sistema de RA baseado em vídeo.

2.1. Captura de vídeo

No ambiente Windows, a forma mais comum de acessar câmeras é utilizando o DirectShow, que consiste num subconjunto da *Application Programming Interface* (API) DirectX [12]. O DirectShow é responsável pela aquisição e manipulação de *streams* multimídia, se comunicando diretamente com o *device driver* da câmera e provendo ao desenvolvedor as funções necessárias para adicionar captura de vídeo à aplicação. O DirectShow utiliza o *Component Object Model* (COM), que possui um nível de programação notadamente baixo, sendo um fator dificultador na utilização da biblioteca.

Com o intuito de prover uma maneira mais prática de manipular vídeo, foi desenvolvido o DsVideoLib, que é um projeto *open source* baseado na API do DirectShow [13]. A biblioteca encapsula algumas características da interface de programação (programação de componentes do Windows), e disponibiliza uma API de alto nível que acelera o desenvolvimento de aplicações multimídia, sem que seja necessário o conhecimento prévio do modelo COM.

Uma outra solução utilizada por desenvolvedores de sistemas de RA é a API de captura de vídeo que faz parte da biblioteca de visão computacional OpenCV, da Intel [14]. Ela dá suporte atualmente às plataformas Windows e Linux, com integração direta com o sistema de janelas nativo do sistema, assim como os componentes de *Graphical User Interface* (GUI) oferecidos pelo próprio OpenCV.

Na plataforma Linux, ainda existe a opção do Video4Linux, que dá suporte a diversos modelos de câmeras e dispositivos de vídeo [15]. O Video4Linux possui uma forte integração com o *kernel* do sistema operacional, o que garante um bom desempenho.

Por fim, algumas câmeras possuem *Software Development Kits* (SDKs) proprietárias, que oferecem funcionalidades específicas do dispositivo em questão. Um exemplo é a API FlyCapture da câmera FireWire Dragonfly, da Point Grey [16].

2.2. Rastreamento de marcadores

O ARToolKit é uma biblioteca *open source* destinada à construção de aplicações de RA [17]. A biblioteca faz uso de marcadores físicos que consistem em cartões de papel com um quadrado de bordas pretas e um padrão desenhado no interior dos mesmos, como mostra a Figura 3, esquerda. Tais marcadores são denominados *template-based markers* (marcadores baseados em padrão). O ARToolKit usa uma técnica que calcula em tempo real a posição e orientação da câmera em relação aos marcadores posicionados no ambiente real. A partir dessas informações, o desenvolvedor pode sobrepor objetos virtuais a esses marcadores, como pode ser visto na Figura 3, direita.

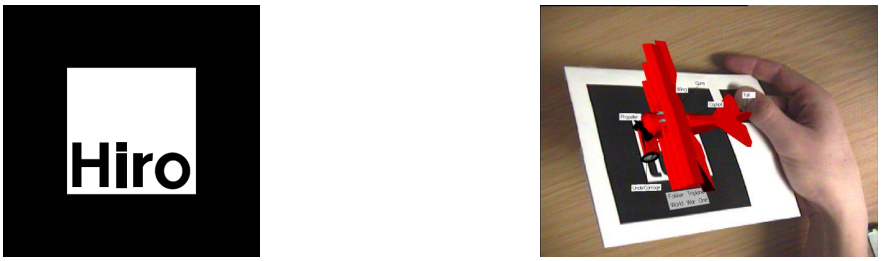

Figura 3. Marcador do ARToolKit (esquerda) e aplicação de RA usando ARToolKit (direita).

A utilização de câmeras baratas e marcadores simples são as principais vantagens do ARToolKit. Além do rastreamento de marcadores, o ARToolKit também provê funções para acesso à câmera e renderização gráfica. Outra característica importante da biblioteca é o suporte a uma gama de sistemas operacionais, como Windows, Linux, Mac OS e SGI IRIX. Como desvantagem, pode-se salientar que a utilização de *template based markers* acarreta muitas vezes em confusão entre marcadores com padrões semelhantes. Outro problema está relacionado à dificuldade em se detectar marcadores que aparecem com tamanho reduzido na imagem, pois a biblioteca não consegue distinguir o padrão presente no interior do marcador. O ARToolKit também é muito sensível à iluminação do ambiente onde a imagem é capturada. Marcadores inseridos em ambientes com pouca ou muita luz não são detectados pela biblioteca. Embora seja possível ajustar os parâmetros de detecção para se obter melhores resultados, tal processo é manual e precisa ser indicado pelo usuário.

Como uma forma de propiciar o desenvolvimento de aplicações de RA na linguagem Java, foi criado o jARToolKit, que consiste num *wrapper* das principais funções do ARToolKit [18]. O jARToolKit acessa o ARToolKit através da *Java Native Interface* (JNI). A biblioteca dá suporte a diferentes APIs de renderização (GL4Java, JOGL e Java3D). Por ser um simples *wrapper* do ARToolKit e por ser escrito em Java, uma linguagem portável, o jARToolKit é capaz de funcionar em qualquer plataforma onde o ARToolKit funciona.

O ARToolKitPlus é uma biblioteca de RA *open source* baseada no ARToolKit, embora se preocupe exclusivamente com a questão da detecção de marcadores, não oferecendo funções para captura de vídeo ou renderização de objetos 3D [8]. Os marcadores utilizados pela biblioteca são semelhantes ao do ARToolKit, com a diferença que o desenho no interior do quadrado preto consiste numa codificação do número identificador do marcador, como ilustrado na Figura 4. Esses marcadores são chamados de *ID-based markers* (marcadores baseados em *ID*). Tal codificação permite que o usuário possa utilizar até 4096 marcadores diferentes, ao contrário do ARToolKit, que só oferece a possibilidade de 50 marcadores distintos. Além disso, a ocorrência de confusão entre diferentes marcadores, comum no ARToolKit, é reduzida, devido ao uso de um algoritmo de

Cyclic Redundancy Check (CRC). O ARToolKitPlus suporta as plataformas Windows, Linux, Windows Mobile e o console de jogos Gizmondo.

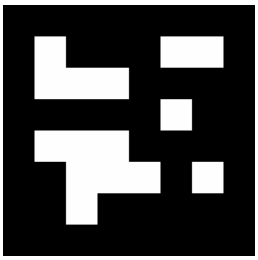

Figura 4. Marcador do ARToolKitPlus que codifica o *ID* 0.

O ARToolKitPlus utiliza a técnica de limiar adaptativo, que consiste em perceber alterações de iluminação no ambiente capturado pela câmera, de forma que a detecção dos marcadores não seja prejudicada. A biblioteca também oferece ao desenvolvedor a possibilidade de utilizar marcadores com borda de largura variável. Desta forma, pode-se diminuir a largura da borda para aproveitar mais espaço para a codificação, fazendo com que os *IDs* de marcadores pequenos possam ser distinguidos na imagem da câmera. Outra característica do ARToolKitPlus é a capacidade de realizar uma compensação sobre imagens de câmeras que geram uma queda radial de luminância. Esse fato ocorre em algumas câmeras que apresentam a imagem capturada mais escura nos cantos, fazendo com que os marcadores situados nessas regiões não sejam detectados.

Outra biblioteca de rastreamento de marcadores que utiliza *ID-based markers* é o ARTag [19]. Ela suporta 2002 marcadores diferentes. Uma característica marcante do ARTag é a robustez à oclusão parcial dos marcadores. Em bibliotecas como o ARToolKit e o ARToolKitPlus, basta que um pequeno pedaço do marcador não esteja visível na imagem para que o mesmo não possa ser identificado; no ARTag, marcadores ocludidos até um certo limite ainda são reconhecidos, como mostra a Figura 5. Outra vantagem do ARTag em relação ao ARToolKit diz respeito a uma menor sensibilidade a variações de iluminação no ambiente, devido a diferenças na estratégia de segmentação dos marcadores na imagem. O ARTag está disponível para uso nas plataformas Windows e Linux.

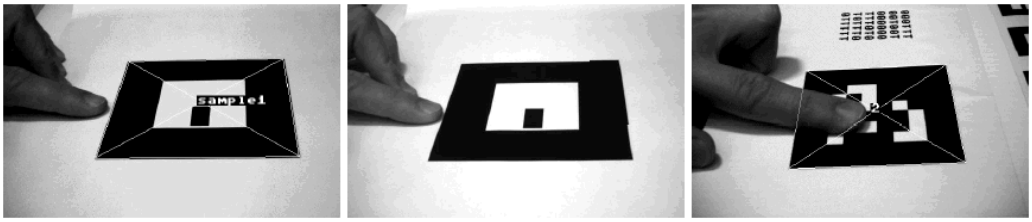

Figura 5. Robustez à oclusão parcial dos marcadores do ARTag (direita) em relação aos do ARToolKit (esquerda e centro).

O MXR Toolkit também é uma biblioteca de RA que utiliza marcadores quadrados [20]. Entretanto, a forma das marcas usadas é um pouco diferente quando comparada com a das APIs similares. O quadrado possui uma quebra em uma de suas arestas, com o intuito de identificar a orientação do marcador, evitando casamentos de padrões desnecessários. Tal fato também possibilita a utilização de um mesmo desenho rotacionado ou refletido para representar diferentes marcadores. Na Figura 6, os marcadores diferem entre si pela rotação do padrão interno, e o MXR Toolkit consegue distingui-los devido à quebra da aresta. O ARToolKit, por exemplo, consideraria as duas imagens como sendo referentes a um mesmo marcador. Por fim, para que um marcador seja detectado pelo MXR Toolkit, basta que a borda interior ou a borda exterior do quadrado esteja totalmente visível na imagem da câmera, ao contrário do ARToolKit, onde qualquer pequena oclusão resulta na não-identificação do marcador.

Figura 6. Marcadores utilizados pelo MXR Toolkit.

2.3. Renderização

A biblioteca gráfica OpenGL é bastante utilizada como solução gráfica em aplicações de RA [21]. Por exemplo, a versão corrente do ARToolKit possui funções de renderização baseadas em OpenGL. A biblioteca é desenvolvida desde 1992 pela OpenGL *Architecture Review Board* (ARB), fato esse que evidencia a maturidade da API e faz com que ela seja bastante estável e livre de *bugs*. OpenGL suporta a grande maioria das plataformas existentes, além de funcionar com diversos sistemas de GUI e linguagens de programação. Uma outra característica bastante importante de OpenGL é o suporte a extensões, que garantem a adequação da biblioteca aos avanços tecnológicos atingidos, como, por exemplo, o uso de programação em *Graphics Processing Unit* (GPU). A principal desvantagem de OpenGL é o seu baixo nível de programação, que obriga o desenvolvedor de RA a se preocupar com os mínimos detalhes relacionados com a renderização 3D de sua aplicação, reduzindo sua produtividade.

Uma opção alternativa ao OpenGL é a biblioteca Direct3D, desenvolvida pela Microsoft, que faz parte da API multimídia DirectX [22]. O Direct3D está disponível apenas para as plataformas Windows, além dos consoles de jogos Xbox e Xbox 360. Uma característica do Direct3D é que ele

pode ser utilizado em modo gerenciado quando associado ao .NET Framework. Entretanto, o nível de programação do Direct3D é tão baixo quanto o do OpenGL.

Devido ao baixo nível de programação oferecido por OpenGL, a SGI criou o Open Inventor, que consiste numa abstração orientada a objetos que roda sobre OpenGL [23]. O Open Inventor possui algumas funcionalidades implementadas que facilitam e aceleram o desenvolvimento de aplicações 3D, tais como grafo de cena e *mouse picking*. A versão oficial mantida pela SGI é *open source* e suporta as plataformas Linux e SGI Workstation. Existem algumas implementações alternativas, como o Coin3D, que consiste numa reimplementação *open source* do Open Inventor que suporta os ambientes Windows e Mac OS [24].

Para o desenvolvimento de aplicações gráficas num ambiente de alto nível, existe o Java 3D, que é utilizado através do ambiente Java [25]. A biblioteca é orientada a objetos e dá ao usuário a possibilidade de utilizar um grafo de cena para organizar os objetos 3D a serem renderizados. O uso de Java traz a vantagem inerente da portabilidade, visto que existem ambientes de execução Java disponíveis para várias plataformas. Em contrapartida, as aplicações Java notadamente apresentam problemas relacionados a desempenho.

Uma solução de renderização 3D largamente utilizada por desenvolvedores de RA é a *Virtual Reality Modeling Language* (VRML) [26]. O fato de o ARToolKit utilizar a biblioteca aberta OpenVRML para oferecer funções para carregamento e exibição de cenas em VRML incentivou a disseminação do formato na comunidade internacional de RA. VRML permite que mundos virtuais sejam criados a partir de código que descreve as propriedades dos objetos presentes, tais como vértices, materiais e texturas. O formato pode ser exibido em páginas da Internet através do uso de *plug-ins* associados ao *browser*. VRML é uma linguagem simples e fácil de usar, entretanto a interação com elementos específicos da cena é bastante limitada.

Em relação aos *engines* gráficos, pode-se destacar o OGRE, que oferece várias facilidades para o desenvolvedor de aplicações 3D em tempo real [4] [5]. O OGRE também é *open source* e está disponível para várias plataformas, como Windows, Linux e Mac OS. A implementação do OGRE para a plataforma Windows pode utilizar OpenGL ou Direct3D como sistema de renderização, que explora a implementação em *hardware* disponível na placa de vídeo para obter um *frame rate* mais elevado. Desta maneira, é possível criar aplicações que demandem alto processamento, pois com a utilização de uma GPU o processador fica livre da tarefa de renderização gráfica.

Ao se utilizar no OGRE o *Crazy Eddie's GUI System* (CEGUI), que é uma biblioteca que provê a construção de janelas e componentes gráficos, o desenvolvedor não desperdiça tempo em implementar um subsistema para criar uma interface com o usuário, agilizando o desenvolvimento da aplicação final.

Outra vantagem da utilização do OGRE como plataforma de desenvolvimento de aplicações gráficas é a fácil integração com ferramentas de modelagem 3D, que acelera a criação de cenários

e malhas exibidos pelo programa. Dentre as ferramentas que podem exportar recursos utilizados pelo OGRE estão o 3D Studio MAX, Blender, Maya, MilkShape3D, Wings3D e SOFTIMAGE|XSI. Juntamente com as malhas produzidas pela ferramenta de modelagem, podem ser também exportados quaisquer tipos de animações baseadas em *bones* (*Skeletal Animations*) e materiais utilizados nos modelos. Para conseguir essa integração, a plataforma utiliza *plug-ins* de exportação especializados para cada ferramenta de modelagem citada. Os *plug-ins* geram um formato intermediário descrito em *Extensible Markup Language* (XML), que posteriormente é transformado no formato proprietário do OGRE.

Existem algumas características presentes no OGRE que podem ser consideradas as principais razões para sua ampla aplicação. Uma delas é a orientação a objetos, que contribui para um desenvolvimento mais estruturado e dinâmico. Além disto, o fato de o OGRE ser formado por classes com interfaces bem definidas entre si favorece a abstração de implementação das diferentes partes do *engine*. Não existe uma dependência de como os módulos responsáveis por operações nativas e renderização 3D são codificados. Devido a isso, o OGRE se torna um *engine* independente de plataforma, sendo possível portá-lo para uma gama variada de ambientes.

O OGRE propõe-se a ser um *engine* flexível, capaz de trabalhar em conjunto com outras ferramentas externas, como bibliotecas de física e inteligência artificial. Outra vantagem do OGRE é o uso de *scripts* escritos em formato texto para o carregamento de opções da aplicação. O desenvolvedor pode modificar configurações como as texturas e materiais dos objetos sem ter de recompilar o programa.

A arquitetura do OGRE é mostrada na Figura 7. A principal classe do OGRE é a classe `Root`, responsável por instanciar as demais classes do *engine* e prover acesso às mesmas. O OGRE possui classes chamadas Managers, que gerenciam os diferentes elementos do sistema. Um `ResourceManager` supervisiona todos os recursos disponíveis para a aplicação. Existem vários tipos de `ResourceManager`, como o `MaterialManager`, responsável pelos materiais, o `FontManager`, responsável pelas fontes, e o `MeshManager`, responsável pelas malhas dos objetos. A classe a qual cabe a organização dos elementos da cena é chamada de `SceneManager`. O `SceneManager` possui uma referência para um objeto do tipo `RenderSystem`, ao qual é delegada a tarefa de renderizar a cena utilizando uma biblioteca gráfica. Outra classe importante é a `PlatformManager`, que encapsula todas as funcionalidades que são nativas do sistema operacional alvo da aplicação, tais como manipulação de entradas, temporização e gerenciamento de interfaces gráficas.

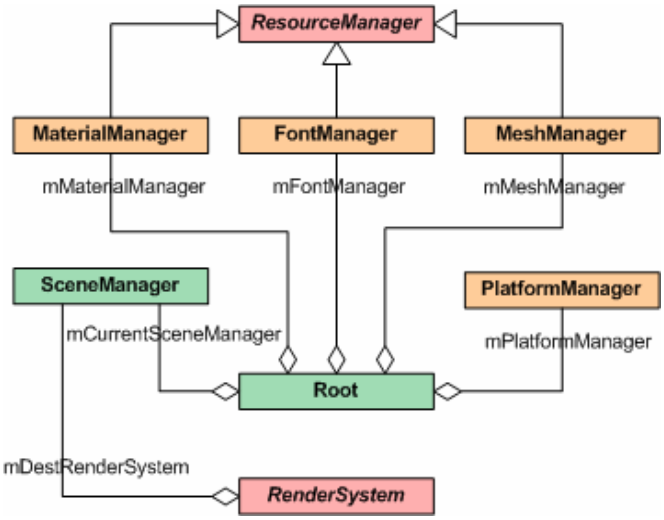

Figura 7. Arquitetura do OGRE.

3. RA em dispositivos móveis

Os sistemas de RA em dispositivos móveis podem ser divididos em duas categorias principais: distribuídos e autônomos. Os primeiros trabalhos de RA em dispositivos móveis eram distribuídos, devido à escassez de poder de processamento dos dispositivos da época. Nos sistemas distribuídos, uma parte das tarefas necessárias para aumentar o ambiente é realizada por um servidor, que troca dados com o dispositivo móvel. A detecção de marcadores fiduciais na imagem capturada e a renderização de objetos 3D é feita pelo servidor, e o dispositivo móvel é responsável apenas por capturar a imagem da câmera, enviá-la para o servidor, receber o resultado final e desenhá-lo na tela. Como o servidor normalmente possui um grande poder de processamento, ele pode executar operações mais complexas de maneira mais rápida que um dispositivo móvel. Além disso, o servidor também pode ter uma GPU repleta de funções à disposição, permitindo o uso de objetos 3D sofisticados. Entretanto, cada *frame* capturado pelo dispositivo móvel é transferido para o servidor, processado, e os resultados são então enviados de volta para o dispositivo, causando um atraso que impacta no *frame rate* das aplicações. Outro problema é relacionado à necessidade de se ter um computador servidor, o que prejudica a mobilidade das soluções.

Com o crescimento do poder de processamento dos dispositivos móveis, surgiram os sistemas autônomos, que não dependem de um servidor para gerar o resultado de RA, implementando todos os procedimentos necessários. Este fator permite o desenvolvimento de soluções de RA completamente móveis. Por outro lado, o poder de processamento baixo e a ausência de uma GPU são características comuns de um *handheld*, implicando na criação de aplicações menos complexas.

3.1. Exemplos de aplicações

A seguir serão apresentados alguns sistemas de RA destinados a dispositivos móveis. As aplicações são classificadas quanto ao dispositivo móvel de destino (PDA, celular ou consoles de jogos) e à categoria a qual elas pertencem (distribuídas ou autônomas).

3.1.1. Aplicações distribuídas para PDA

Uma das primeiras iniciativas em RA para dispositivos móveis foi o NaviCam, que é um dispositivo constituído de um PDA associado a uma câmera, conectados através de cabos com um servidor que detecta padrões de barras coloridas na imagem (ver parte inferior da imagem mostrada na Figura 8, direita) e envia para o PDA os dados a serem desenhados na tela do

dispositivo [27]. Devido à comunicação entre o dispositivo móvel e o servidor ser feita através de cabos, a mobilidade do sistema é prejudicada. Uma das aplicações do sistema é a obtenção de informações textuais relativas a objetos presentes na cena real, como pode ser visto na Figura 8.

Figura 8. NaviCam utilizado para visualização de informações relativas a objetos da cena real.

A aplicação detalhada em [28] serve para mostrar uma previsão da fachada de um prédio a ser construído, como mostra a Figura 9. Um marcador é posicionado no ambiente para determinar a posição do prédio. O PDA troca dados com o servidor usando a rede *Wireless Fidelity* (Wi-Fi), que possui raio de cobertura e taxa de transferência adequados para a aplicação. O fato de o sistema ser distribuído se deve principalmente à falta de recursos disponíveis no PDA para lidar com a renderização de objetos 3D de grande escala, como os utilizados na aplicação.

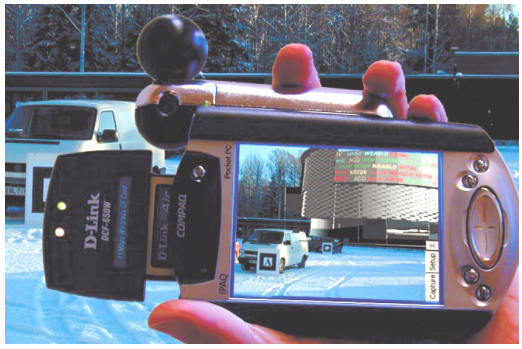

Figura 9. Previsão de prédio a ser construído utilizando RA.

O AR-PDA é mais um sistema distribuído onde a comunicação entre cliente e servidor é feita através da rede sem fio Wi-Fi [29]. O sistema é utilizado para auxiliar consumidores na aquisição de produtos, adicionando informações técnicas dos mesmos sobre a imagem. Na Figura 10, um fogão é aumentado com uma bandeja virtual. O rastreamento de objetos, que é feito sem a utilização de marcadores, é uma tarefa muito custosa para ser executada no PDA utilizado, justificando a abordagem distribuída do sistema.

Figura 10. AR-PDA utilizado em aplicação de auxílio ao consumidor.

3.1.2. Aplicações autônomas para PDA

The Invisible Train, ilustrado na Figura 11, consiste num jogo *multi-player* onde cada usuário é responsável por guiar um trem virtual que é desenhado sobre um trilho real de madeira [30]. O objetivo é evitar que os trens venham a colidir. Os jogadores têm a opção de alterar a velocidade do trem e desviar sua trajetória alterando a posição das *track switches* localizadas nas confluências dos trilhos. A posição dos trens é estimada através do rastreamento dos vários marcadores espalhados nas imediações dos trilhos. O estado do jogo é sincronizado entre os dispositivos envolvidos através da rede sem fio Wi-Fi.

Figura 11. O jogo The Invisible Train para guiar trens de forma que não colidam.

Virtuoso consiste num jogo que envolve conhecimento sobre artes, no qual deve-se ordenar cronologicamente obras exibidas sobre marcadores afixados na parede, de forma que as obras mais antigas fiquem dispostas sobre os marcadores mais à esquerda e as mais novas sobre os marcadores mais à direita, como uma linha do tempo [31]. Caso o jogador esteja em dúvida, ele

tem a opção de consultar o Sr. Virtuoso, um personagem virtual que traz informações a respeito das obras em questão, como mostra a Figura 12.

Figura 12. O jogo Virtuoso para ordenar cronologicamente obras de arte.

O sistema SignPost tem por finalidade prover orientações para o usuário chegar a um determinado local num ambiente, funcionando como um mapa interativo [32]. Para isso, marcadores são espalhados por pontos estratégicos do ambiente e, de acordo com o destino escolhido pelo usuário, indicam qual deve ser o caminho seguido através de setas, como pode ser observado na Figura 13.

Figura 13. O sistema SignPost para auxílio à navegação.

O Kanji Teaching é um jogo que testa os conhecimentos do usuário em relação ao significado dos caracteres chineses Kanji [33]. Vários cartões com um marcador impresso na frente e um caractere Kanji impresso no verso são emborcados sobre uma mesa. A cada jogada, é exibido um objeto 3D na tela e o usuário deve desvirar o marcador associado ao caractere Kanji cujo significado corresponde ao objeto em questão (Figura 14). Feito isso, o sistema detecta o marcador e verifica se o jogador acertou ou não, mostrando o objeto sobre o marcador em caso positivo.

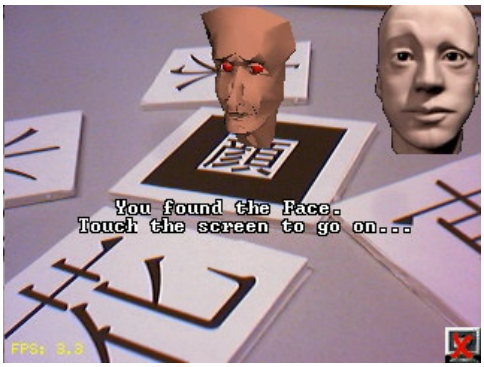

Figura 14. O jogo Kanji Teaching para aprendizado de caracteres chineses Kanji.

3.1.3. Aplicações distribuídas para celular

O Digital Scrap Book consiste num álbum de recordações em papel aumentado com mídias tais como som e vídeo [34]. É possível também digitalizar o conteúdo do álbum para compartilhamento, utilizando uma caneta digital para adquirir os manuscritos e uma câmera digital para obter as fotos. Marcadores são fixados às páginas do álbum e um celular com câmera é utilizado para visualizar vídeos sobre eles (Figura 15); desta forma, o vídeo parece fazer parte do álbum. A comunicação entre o dispositivo móvel e o servidor responsável pela detecção dos marcadores e aumento da cena se dá através de Bluetooth.

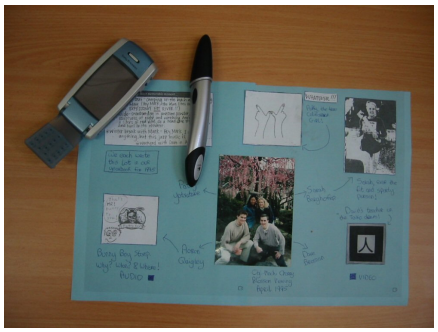

Figura 15. O álbum de recordações aumentado Digital Scrap Book.

O ARPhone consiste num sistema cliente-servidor que não utiliza marcadores para rastrear o ambiente, se baseando apenas nas informações dos elementos naturais presentes na cena [35]. O telefone envia os dados para o servidor através de Bluetooth. O sistema foi utilizado numa aplicação que posiciona móveis num ambiente (Figura 16).

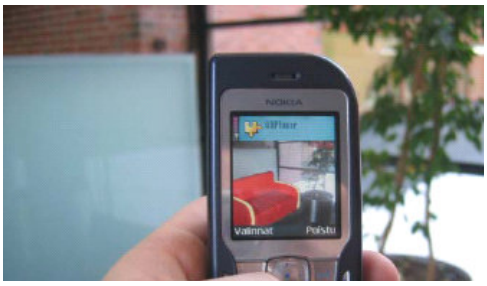

Figura 16. ARPhone usado para visualização de interiores.

Foram também desenvolvidas aplicações de RA para celular destinadas a crianças, das quais duas merecem destaque: AR Comic Book e AR Post-It [36]. O AR Comic Book consiste num gibi cujas páginas possuem marcadores, e ao ler-se o gibi através do celular, é possível visualizar modelos 3D dos personagens da história (Figura 17, esquerda). O AR Post-It permite a visualização de mensagens *Short Message Service* (SMS) sobre marcadores posicionados em locais nos quais o conteúdo da mensagem é relevante (Figura 17, direita). Ambas as aplicações se comunicam com o servidor através de Bluetooth.

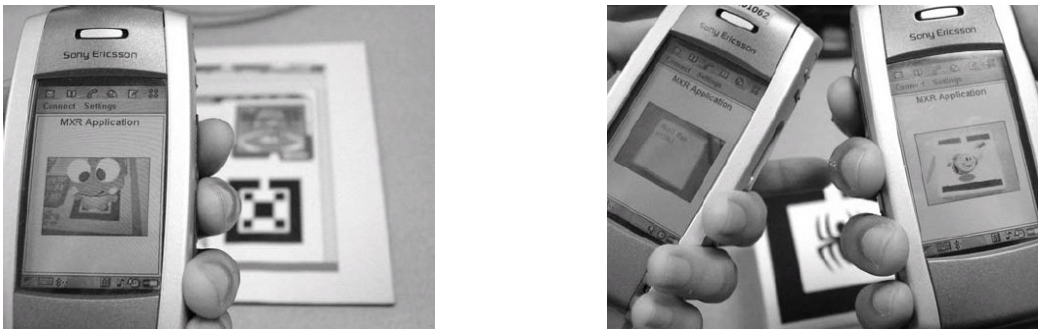

Figura 17. O gibi aumentado AR Comic Book (esquerda) e o AR Post-It para visualização de mensagens SMS (direita).

3.1.4. Aplicações autônomas para celular

A aplicação descrita em [37] tem por objetivo aumentar um mapa com informações de posicionamento dos trens de uma determinada ferrovia. Um marcador é colado sobre o mapa para que as vias possam ser reconhecidas pela aplicação, como pode ser visto na Figura 18, esquerda. De acordo com a hora do sistema e o horário de saída e chegada dos trens, a posição dos mesmos é calculada e indicada sobre a imagem do mapa, como ilustra a Figura 18, direita.

Figura 18. Mapa de ferrovia aumentado com informações sobre trens.

O Virtual Video, desenvolvido no projeto *Mobile Augmented Reality Applications* (MARA) da Nokia, dispõe de um celular equipado com GPS, acelerômetros e magnetômetro (Figura 19, esquerda) para determinar a posição e orientação da câmera em relação ao mundo real [38]. A partir disso, é possível sobrepor sobre a imagem da câmera informações sobre os elementos reais presentes na imagem da câmera. Na Figura 19, direita, é destacada a localização de uma pizzaria, assim como a distância entre o usuário e vários pontos do ambiente.

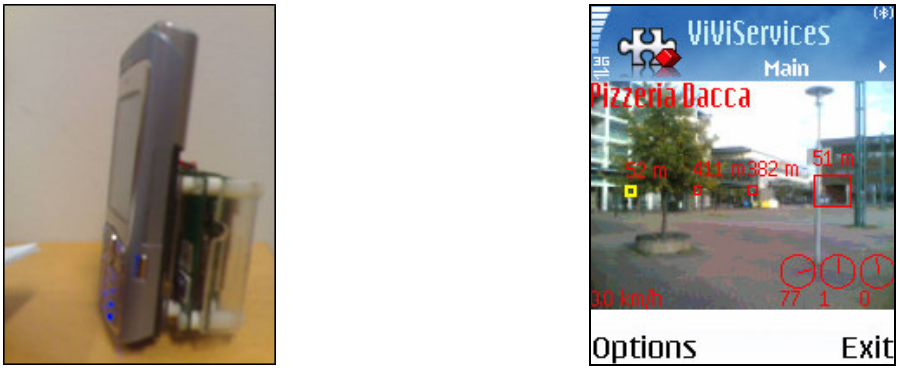

Figura 19. Nokia Virtual Video para visualização de informações sobre a cena real.

Existem vários jogos para celular que se valem da RA como principal forma de interação. O AR Tennis é um jogo de ping-pong onde os celulares dos dois jogadores se comunicam através de Bluetooth [39]. A posição das raquetes é detectada utilizando-se marcadores fixados aos celulares, fazendo com que o jogador mova o celular para modificar a posição de sua raquete. Existem também marcadores sobre a mesa para que a mesma possa ser rastreada (Figura 20, acima à esquerda). O Virtual Lego consiste no clássico jogo de montar utilizando RA no celular [40]. Marcadores são colocados sobre uma mesa para servir como base para encaixar as peças. O usuário pode transladar e rotacionar as peças utilizando os botões do celular, com o intuito de empilhá-las (Figura 20, acima à direita). Quando há um encaixe ou desencaixe de peças, o celular

vibra, de forma a dar um retorno háptico da ação ao jogador. O Mozzies é um jogo *First Person Shooting* (FPS) onde o objetivo é matar insetos virtuais que voam no mundo real [41]. Para isso, é necessário alinhar os insetos com a mira presente no centro da tela (Figura 20, abaixo à esquerda) e pressionar o botão do celular que atira. Para movimentar a mira, basta o jogador mover o celular, visto que é feita a detecção do movimento 2D da câmera com base na mudança da imagem capturada. O resultado é então refletido no jogo. O Kick Real [42] é um jogo de disputa de pênaltis, onde o jogador chuta a bola virtual com o próprio pé, como pode ser visto na Figura 20, abaixo à direita. Para que isso aconteça, é necessário que a câmera esteja apontada para o chão, que é uma posição comum de utilização do celular. Existem outros vários jogos para celular que utilizam RA sendo comercializados no site da Mauj [43].

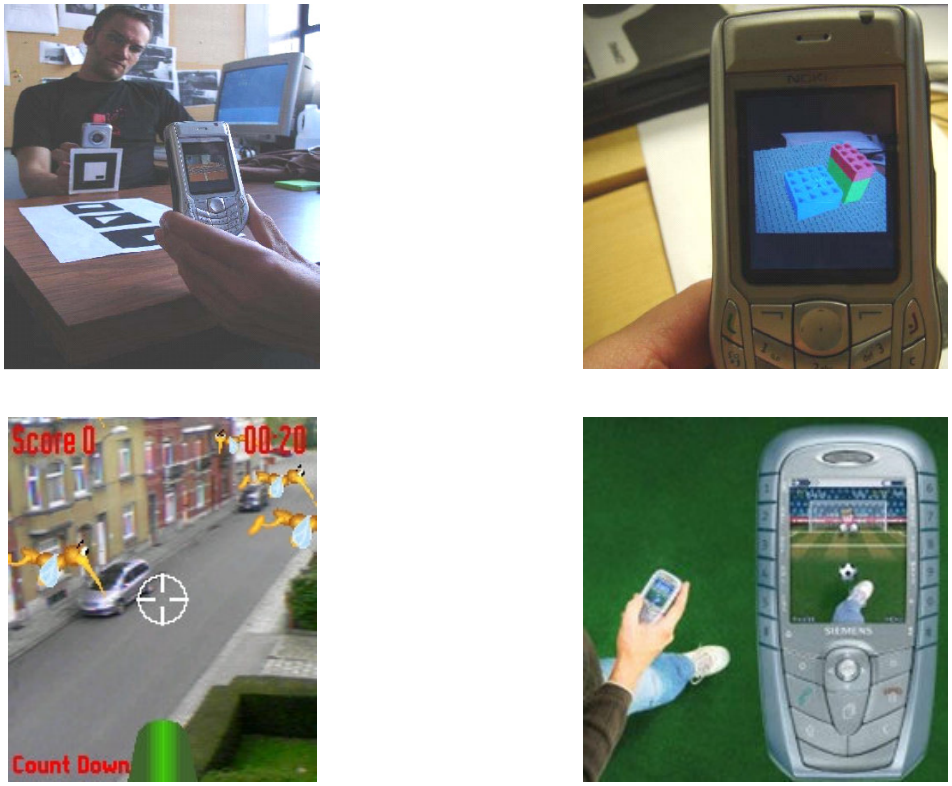

Figura 20. O jogo de ping-pong ARTennis (acima à esquerda), o jogo de montar Virtual Lego (acima à direita), o FPS Mozzies (abaixo à esquerda) e o jogo de disputa de pênaltis Kick Real (abaixo à direita).

3.1.5. Aplicações para consoles de jogos

Foram desenvolvidas aplicações autônomas de RA para o console de jogo portátil Gizmondo, que se aproveitam da aceleração 3D em *hardware* oferecida pelo dispositivo para obter um alto *frame rate*. Duas aplicações destinadas ao Gizmondo são detalhadas a seguir: Vidente e Catapult. O Vidente, ilustrado na Figura 21, consiste numa versão conceitual *indoor* de um sistema

de visualização de elementos subterrâneos [44]. O usuário pode ver, por exemplo, um raio-X dos cabos e encanamentos presentes sob as ruas da cidade.

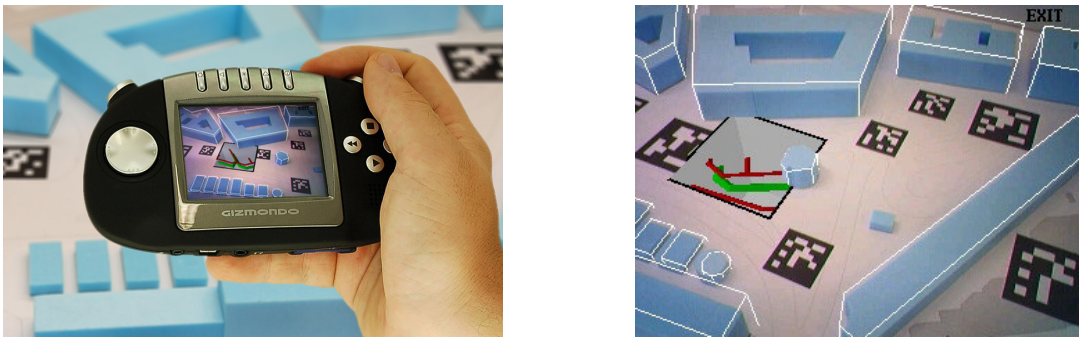

Figura 21. A aplicação Vidente para visualização de elementos subterrâneos.

O Catapult é um jogo de estratégia onde o objetivo é destruir as construções do inimigo utilizando catapultas [45]. As fortalezas são posicionadas sobre uma mesa, rastreada com o auxílio de um marcador, como ilustrado na Figura 22. O jogo permite também o modo *multi-player*, onde o Bluetooth é usado para sincronizar o estado dos jogadores.

Figura 22. O jogo de estratégia Catapult.

3.2. Captura de vídeo

Na plataforma Windows Mobile 5.0, o DirectShow encontra-se disponível para acesso a câmeras de dispositivos móveis através da Camera Capture API [46]. Pelos mesmos motivos existentes na plataforma *desktop* (ver subseção 2.1), foi criada uma versão da DsVideoLib para dispositivos móveis chamada DsVideoCE [47], que conta também com funções de conversão de *pixel format* eficientes.

Na grande maioria dos casos, a captura de vídeo em dispositivos móveis é realizada usando APIs específicas do sistema operacional, como a ECam do Symbian OS [48], e SDKs proprietárias dos fabricantes das câmeras, como a das câmeras *Secure Digital* (SD) ViCAM-III [49].

3.3. Rastreamento de marcadores

O ARToolKitPlus encontra-se disponível para as plataformas móveis Windows Mobile (tanto Pocket PC como *smartphone*) e Gizmondo [8]. O código fonte do ARToolKitPlus conta com várias otimizações, tais como a utilização de matemática de ponto fixo, com o intuito de gerar aplicações eficientes. Os dispositivos móveis não possuem uma *Floating Point Unit* (FPU) e todas as operações matemáticas baseadas em *hardware* são realizadas usando ponto fixo.

Existe também o porte do ARToolKit para a plataforma Symbian OS [37]. A biblioteca foi testada com êxito em celulares Nokia da Série 60. Entretanto, ainda restam problemas de desempenho, principalmente no que diz respeito às operações matemáticas, que são realizadas utilizando ponto flutuante.

3.4. Renderização

Recentemente, o crescimento do mercado de *handhelds*, PDAs e *smartphones* vem criando uma demanda por bibliotecas que tornem possível o desenvolvimento de aplicações para dispositivos móveis que usam gráficos 3D. A biblioteca mais popular para essa finalidade é o OpenGL ES, uma versão limitada de OpenGL direcionada para sistemas embarcados [50]. Existem basicamente duas versões sendo desenvolvidas em paralelo: a versão 1.x, projetada para trabalhar com *hardwares* de função fixa, e a versão 2.x, especificada para *hardwares* programáveis. Entretanto, a maioria dos dispositivos móveis atuais não possui aceleração 3D feita por *hardware*, situação esta que está mudando com o lançamento de dispositivos móveis com GPU integrada. Tanto OpenGL quanto OpenGL ES provêem uma programação de baixo nível, o que diminui a produtividade do desenvolvedor.

Outra solução gráfica 3D bastante usada por desenvolvedores de dispositivos móveis é o Klimt, que se propõe a prover uma interface similar a de OpenGL, cobrindo algumas das funcionalidades não suportadas por OpenGL ES [51]. O Klimt também possui algumas otimizações para os sistemas Pocket PC. A versão disponível é de código aberto e usa um mecanismo de renderização próprio. Devido à similaridade com OpenGL, o nível de programação é também muito baixo.

Especificamente para a plataforma Pocket PC, o sistema operacional Windows Mobile 5.0 suporta Direct3D Mobile, que é baseado no Direct3D 8 para *desktops* [52]. Quando combinado com o .NET Compact Framework, pode ser utilizado em modo gerenciado, provendo um desenvolvimento seguro e fácil. Por outro lado, o nível de abstração é quase o mesmo de OpenGL e OpenGL ES.

Em relação ao desenvolvimento em Java, existe uma API para renderização 3D chamada *Mobile 3D Graphics API for J2ME* (M3G) [53]. Ela é uma biblioteca gerenciada orientada a objetos que possui dois modos de operação diferentes: o *immediate mode*, que usa acesso de baixo nível semelhante ao de OpenGL e OpenGL ES, e o *retained mode*, que usa um grafo de cena, uma estrutura de mais alto nível para renderização 3D que também é utilizada no OGRE, tornando o processo de codificação muito mais intuitivo. A desvantagem das aplicações Java é o baixo desempenho, que causa uma queda no *frame rate*.

3.5. Trabalhos relacionados

O desenvolvimento de aplicações de RA portáveis para dispositivos móveis usando bibliotecas gráficas de alto nível é abordado pelo projeto Studierstube, que é um exemplo de trabalho similar ao apresentado neste Trabalho de Graduação [54]. Seus resultados, entretanto, ainda não foram totalmente publicados para tornar possível uma avaliação e comparação com o *framework* descrito neste trabalho.

O Studierstube tem o seu foco em aplicações multiplataforma, e vários estudos foram realizados tendo como alvo a plataforma Pocket PC. Ele é composto de bibliotecas para renderização, gerenciamento de mídia (áudio, vídeo e câmera), rastreamento de marcadores e distribuição em rede, como destaca a Figura 23. Seu módulo de renderização é o Klimt. Sobre o Klimt, foi construída uma camada de abstração baseada no Open Inventor. Para a aquisição de vídeo foi utilizada a biblioteca DSVideoCE, que faz o acesso às câmeras através do DirectShow. A detecção de marcadores fica por conta do OpenTracker, um *wrapper* do ARToolKitPlus.

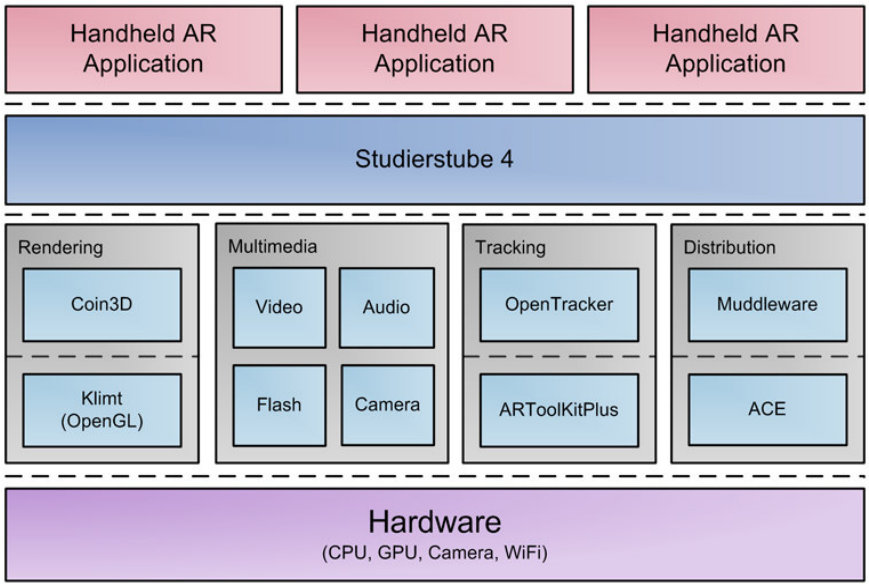

Figura 23. Arquitetura do Studierstube.

4. O *framework* de RA

O *framework* de RA proposto e implementado neste Trabalho de Graduação é genérico, podendo ser utilizado para desenvolver aplicações de RA tanto para *desktop* como para *handheld*. Os diferentes componentes do *framework* utilizados na plataforma Pocket PC são mostrados na Figura 24. As aplicações de RA rodam sobre o *engine* OGRE [4] [5], adaptado para a plataforma Pocket PC pelo autor originando a biblioteca chamada OGRE4PPC [6], e a biblioteca OgreAR [7]. O OGRE é responsável por desenhar as imagens do mundo real e os objetos virtuais sobrepostos a ele. Para realizar essa tarefa, o OGRE utiliza alguma biblioteca gráfica disponível na plataforma Pocket PC, como por exemplo, OpenGL ES ou Klimt. A biblioteca OgreAR é responsável por capturar a imagem da câmera e calcular as características geométricas dos marcadores presentes no *frame*. A aquisição de imagens pode ser implementada usando qualquer biblioteca de acesso à câmera e o rastreamento de marcadores independe de qualquer biblioteca de detecção. No cenário Pocket PC, o OgreAR utiliza a SDK proprietária da câmera, assim como a biblioteca ARToolKitPlus, visto que a câmera utilizada só pode ser acessada através de sua SDK e que o ARToolKitPlus é a única opção de rastreamento de marcadores disponível na plataforma em questão.

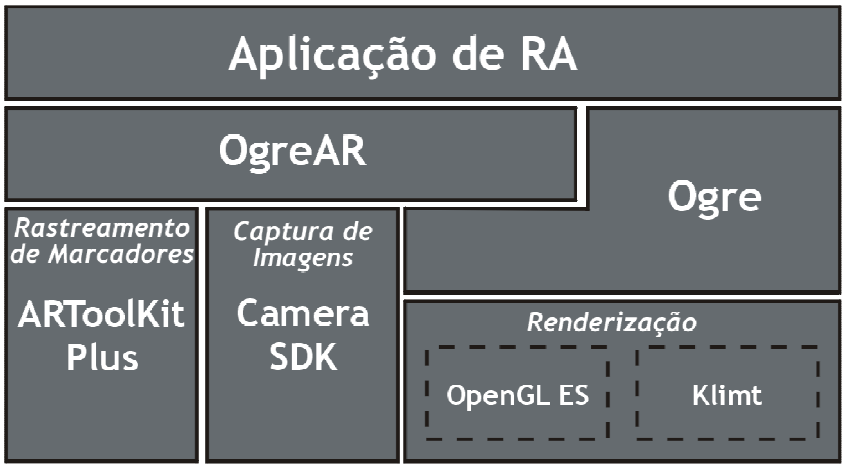

Figura 24. Arquitetura do *framework* de RA para Pocket PC.

4.1. A biblioteca OgreAR

O OgreAR é uma biblioteca desenvolvida pelo autor para prover uma interconexão de aplicações baseadas no *engine* de renderização OGRE e elementos básicos de RA, como aquisição de vídeo em tempo real e detecção de marcadores [7]. Ele disponibiliza uma interface aos recursos necessários para a construção de uma aplicação de RA, contando com a facilidade de utilização, a

legibilidade e a rapidez de um componente de abstração de alto nível. A biblioteca OgreAR permite que aplicações como a mostrada na Figura 25, onde um ninja dança sobre um marcador, sejam criadas em alguns minutos. Além disso, o OgreAR favorece a portabilidade das aplicações, visto que elas não dependem das bibliotecas que estão sendo utilizadas para realizar as funcionalidades de RA.

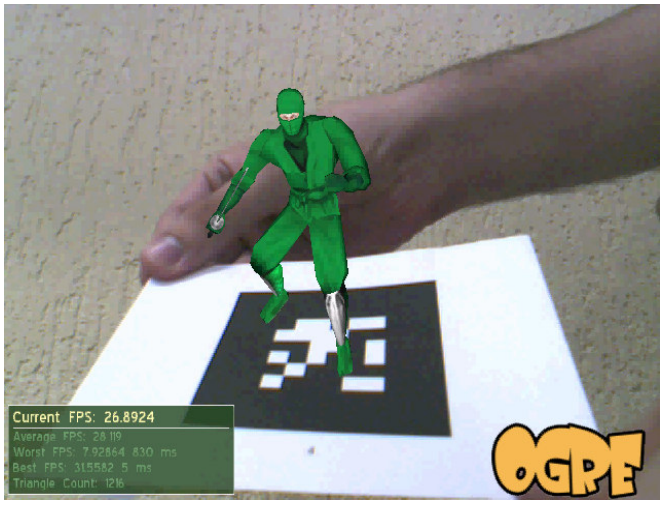

Figura 25. Aplicação de RA desenvolvida usando o OgreAR.

A Figura 26 descreve a arquitetura da biblioteca e a interconexão entre seus componentes. A detecção de padrões é independente de qualquer biblioteca de detecção de marcadores (ARToolKit, ARToolKitPlus, MXRToolkit etc.), pois ela foi concebida como uma camada de abstração do sistema de detecção, onde o usuário pode instanciar um `PatternDetector` baseado em qualquer técnica de reconhecimento de padrões, desde que esta seja encapsulada numa classe que estenda `PatternDetector`. A biblioteca já vem com uma implementação padrão para detecção de padrões baseados em marcadores genéricos (`ARToolkitAdapter`) e marcadores baseados em *IDs* (`ARToolkitPlusAdapter`). Além destes `PatternDetectors`, a biblioteca dispõe de um detector de padrão nulo, que serve apenas para testes do usuário ou exibição única de dados da câmera sem detecção de padrão.

A captura de imagem também é abstraída por uma interface chamada `FrameGrabber`, que é implementada usando alguma biblioteca de acesso à câmera, tais como o DsVideoLib ou a SDK proprietária da câmera.

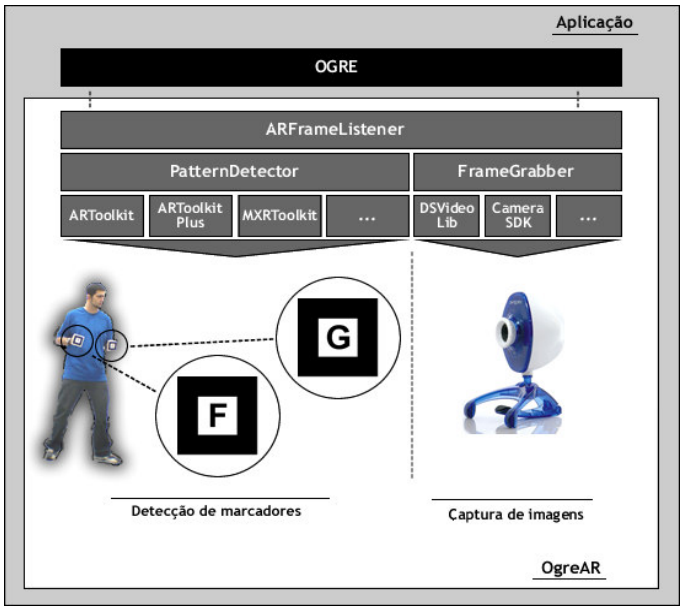

Figura 26. Arquitetura do OgreAR.

A captura do vídeo e sua exibição no *background* da aplicação são feitas através de um `FrameListener`, elemento básico de aplicações do OGRE. Para que este seja integrado no programa, basta apenas a adição do `ARFrameListener` à aplicação através do método `Root::addFrameListener`. O `ARFrameListener` deve ser associado sempre a um `PatternDetector`, que pode ser modificado pelo usuário a qualquer momento da execução do programa.

A detecção de padrões do OgreAR funciona baseada em eventos, assim como todo o sistema de interação do OGRE. Para receber estes eventos e manipular os dados dos marcadores identificados, o usuário deve implementar uma interface chamada `DetectionListener`. Esta possui apenas o método `itemDetected` que recebe um parâmetro do tipo `AREvent`, evento responsável por agregar informações como posição, orientação e identificação dos marcadores detectados.

4.2. A biblioteca OGRE4PPC: porte do OGRE para Pocket PC

A versão do OGRE portada para o Pocket PC se baseou na versão 1.2 (Dagon) do código oficial do OGRE. A primeira tarefa realizada para disponibilizar o OGRE para a plataforma Pocket PC foi portar todas as dependências do OGRE que não possuíam uma versão funcional para a mesma, e estender algumas bibliotecas que suportavam o sistema operacional para fornecer as funcionalidades necessárias, como, por exemplo, uma manipulação de arquivos mais completa. Estas dependências são bibliotecas utilizadas pelo *engine* para tarefas específicas, tais como

carregamento de texturas e gerenciamento de arquivos. O segundo passo consistiu em implementar um `RenderSystem` usando uma API gráfica disponível para Pocket PC, neste caso o OpenGL ES. Após isso, um `PlatformManager` para o sistema operacional teve que ser escrito. Depois, almejando melhorar o desempenho, o OGRE foi adaptado para trabalhar com rotinas matemáticas de ponto fixo. Além disto, foi implementado um segundo `RenderSystem` usando o Klimt, numa tentativa de obter um *frame rate* maior. Por fim, foi implementado um `RenderSystem` que explora a aceleração em *hardware* presente em alguns dispositivos móveis.

4.2.1. Dependências

As bibliotecas que não tinham uma versão funcional para a plataforma Pocket PC eram STLPort, DevIL, FreeType2 e Zziplib. Juntamente com sua finalidade, alguns detalhes de como o processo de porte foi feito são descritos a seguir.

O STLPort provê ao desenvolvedor todas as classes presentes na C++ *Standard Template Library* (STL) [55]. A STL é vastamente utilizada pelas dependências do OGRE e pelo próprio OGRE, mas os ambientes de desenvolvimento existentes para Pocket PC não oferecem uma implementação completa dela. A *Integrated Development Environment* (IDE) que implementa o maior número de funcionalidades da STL é o Microsoft Visual Studio 2005, então a versão portada do STL foi projetada para funcionar nesse ambiente. A versão Pocket PC do STLPort foi baseada no *release* 5.0 RC6, que sofreu pequenas modificações. Basicamente, foi preciso apenas adicionar um arquivo de cabeçalho adicional ao código fonte da biblioteca para definir as configurações da IDE.

O DevIL consiste numa biblioteca responsável pelo carregamento de texturas [56]. Ele suporta vários formatos de imagem, como bmp, gif, ico, jpeg, png, psd e tga. O DevIL para Pocket PC foi compilado baseado na versão 1.6.7 do código fonte da biblioteca. As modificações feitas na biblioteca original foram quase todas relacionadas a diferenças em assinaturas de funções. Na plataforma Pocket PC, as *strings* são codificadas utilizando o padrão *wide char* (*Unicode*), em que cada caractere é representado por 2 (dois) *bytes*. O formato usado na maioria das aplicações *desktop* é o *American Standard Code for Information Interchange* (ASCII), que usa 1 (um) *byte* para codificar um caractere. Algumas funções só recebem *strings* no formato *Unicode* e outras possuem uma versão para cada codificação. Como resultado, a codificação da *string* tem que ser checada para garantir que a função correta será chamada.

O FreeType2 é usado para carregar fontes em vários formatos, tais como TrueType e OpenType [57]. Todos os elementos de texto nas cenas do OGRE dependem desta biblioteca para definir seu *layout*. Depois de configurar o STLPort no ambiente Pocket PC, a versão 2.1.10 do código fonte da biblioteca FreeType2 pôde ser compilado sem grandes modificações.

O Zziplib é uma biblioteca que torna possível o acesso a arquivos que estão comprimidos em pacotes zip [58]. Os recursos da aplicação podem então ser empacotados em um ou mais arquivos zip, facilitando a distribuição posterior. A biblioteca também cria uma abstração quanto a maneira como o recurso será armazenado no disco. Arquivos que pertencem ao *filesystem* e arquivos que estão "zipados" são tratados da mesma maneira. Por exemplo, quando a aplicação tenta abrir um arquivo, a biblioteca procura por ele no caminho dado. Se ele não é encontrado, então ela procura em todos os arquivos zip armazenados no caminho. A versão 0.10.82 do Zziplib foi usada na plataforma Pocket PC, sendo apropriadamente compilada depois de se ter o STLPort configurado no ambiente de desenvolvimento.

As bibliotecas que já tinham uma versão funcional para a plataforma Pocket PC e que foram estendidas para suprir as funcionalidades necessárias foram a LibCE, Zlib e OpenGL ES. Juntamente com sua finalidade, alguns detalhes de como o processo de porte foi feito são descritos a seguir.

O LibCE é uma biblioteca criada para preencher os espaços vazios presentes na biblioteca libc para o Pocket PC, que não implementa muitas funções presentes na versão Win32, como o acesso ao tempo do sistema [47]. A versão 1.0 do LibCE foi o ponto de partida para a biblioteca utilizada pelo OGRE. Alguns procedimentos foram adicionados para a manipulação de arquivos com parâmetros *Unicode* e outras funcionalidades extras que não estavam disponíveis na biblioteca, como a procura, remoção e renomeação de arquivos.

O Zlib implementa o algoritmo de compressão usado no formato de arquivo .zip [59]. O OGRE usa o Zlib para descompactar recursos "zipados" e disponibilizá-los para uso. Existia uma versão do Zlib para Pocket PC baseada na compilação 1.2.1 da biblioteca. Algumas pequenas correções foram feitas e ela foi então compilada com sucesso na IDE.

OpenGL ES é uma das APIs gráficas usadas na versão do OGRE para Pocket PC. Uma das implementações aplicadas no porte é chamada de Vincent, que não faz uso de processamento em *hardware* [60]. A versão 0.84 da biblioteca serviu como base para o projeto. As modificações feitas no código fonte da biblioteca foram relacionadas com as funcionalidades do *pixel buffer*, que não estavam funcionando corretamente para o processo de renderização usado no OGRE.

4.2.2. RenderSystem usando OpenGL ES

Na distribuição do código fonte do OGRE para Win32, existem três implementações do `RenderSystem`, cada uma usando uma biblioteca gráfica diferente: OpenGL, e Direct3D versões 7 e 9. Visto que nenhuma dessas bibliotecas suporta o Pocket PC, houve a necessidade de se implementar um `RenderSystem` usando uma biblioteca gráfica disponível para esta plataforma.

A biblioteca inicialmente escolhida foi o OpenGL ES 1.1, devido a sua similaridade com OpenGL, possibilitando que o `RenderSystem` original pudesse ser adaptado sem grandes mudanças. A biblioteca Vincent, uma implementação de código aberto do OpenGL ES, foi usada no projeto [60]. A versão atual é completamente baseada em *software*, sem utilizar aceleração em *hardware*, o que resulta em problemas de desempenho. O Vincent foi utilizado porque o dispositivo inicialmente utilizado no projeto (HP iPAQ H5500) não possui placa aceleradora. A implementação de OpenGL ES da Hybrid para Win32, descrita em [61], também foi usada, pois o `RenderSystem` implementado foi primeiramente testado no ambiente *desktop* para então ser usado na plataforma Pocket PC.

A tentativa de tornar a biblioteca OpenGL ES a menor possível para se encaixar nas restrições dos dispositivos móveis ocasionou a falta de muitas funcionalidades presentes em OpenGL. Devido a isso, ações do `RenderSystem` do OGRE que requerem, por exemplo, texturas 1D ou 3D ou *cube maps* não podem ser realizadas e irão levantar uma exceção de "funcionalidade não suportada". OpenGL ES também possui outras limitações, como a inabilidade de ler o conteúdo de uma textura e a utilização de índices de 32 *bits* para a renderização de polígonos.

O projeto inicial do `RenderSystem` OpenGL para Win32 foi modificado de forma a usar OpenGL ES. Primeiro, todo o código relativo a *shaders* foi removido, visto que esta funcionalidade requer processamento em GPU e não está disponível no Vincent.

A função de *startup* do `RenderSystem` foi modificada para inicializar as capacidades corretamente. O Vincent não suporta algumas funcionalidades como *mipmapping* em *hardware*, anisotropia, *cubemapping* e *vertex/fragment shading*.

Muitas das assinaturas de funções de OpenGL ES são equivalentes as de OpenGL, contribuindo para diminuir o número de modificações necessárias no código original. Alguns nomes de função possuem pequenas diferenças, como `glClearDepth` (OpenGL) e `glClearDepthf` (OpenGL ES). O `RenderSystem` também usou extensões do OpenGL que foram substituídas por funções equivalentes do OpenGL ES. O código relativo à criação e gerenciamento de janelas, que é específico de plataforma, contém algumas chamadas para procedimentos do WGL, a API de OpenGL para suporte nativo no Win32. Elas foram todas substituídas por funções do EGL, a interface de OpenGL ES responsável pelo sistema de janelas.

A biblioteca *OpenGL Utilities* (GLU) é usada no `RenderSystem` para a criação de *mipmaps* de textura. Como não há GLU para OpenGL ES disponível, a implementação do SGI GLU para OpenGL foi adaptada para funcionar com OpenGL ES.

4.2.3. PlatformManager para Pocket PC

Depois de estender a bilbioteca LibCE, todas as funções Win32 ausentes na API do Pocket PC estavam disponíveis para uso. Por essa razão, o `PlatformManager` para Pocket PC consistiu numa adaptação da versão Win32.

O `PlatformManager` consiste em cinco classes: `ConfigDialog`, `ErrorDialog`, `Timer`, `Input` e `PlatformDll`. O `ConfigDialog` é responsável por mostrar as opções de configurações para o usuário quando a aplicação começa, possibilitando que ele mude opções como profundidade de cores, freqüência de exibição e resolução de tela. O `ErrorDialog` possui como missão advertir o usuário sobre qualquer erro que ocorra na execução da aplicação. As exceções lançadas pelo OGRE são muito esclarecedoras, indicando o arquivo e a linha onde o erro ocorreu, juntamente com uma descrição detalhada do que aconteceu. O `Timer` provê funções para acessar o tempo do sistema operacional, de grande utilidade para várias ações tomadas pelo *engine*, tais como medir o *frame rate* da aplicação. O `Input` captura a interação feita pelo usuário usando qualquer dispositivo de entrada, como os botões do *hardware* e a caneta. O `PlatformDll` age como uma interface para todas as funcionalidades implementadas pelo `PlatformManager`, acessando os respectivos métodos de cada um.

Essas classes permaneceram praticamente as mesmas da plataforma Win32 no `PlatformManager` para Pocket PC, com pequenas mudanças relacionadas à conversão de *strings Unicode*.

A classe `Input` demandou um trabalho mais cuidadoso, visto que a interação do usuário no ambiente *desktop* é completamente diferente de um dispositivo móvel. Quando sentado em frente a um PC, o usuário comumente tem um teclado com 101 teclas e um *mouse* com dois ou três botões. Em muitos dispositivos móveis, na maioria das vezes a única maneira que o usuário tem para interagir com o sistema é através de alguns poucos botões de *hardware* e de uma caneta. Alguns PDAs e *smartphones* possuem um pequeno teclado com letras, dígitos e pontuação.

No desenvolvimento deste projeto, a interação feita pelo teclado foi mapeada para os botões de *hardware* do dispositivo e a caneta funciona como um *mouse*. O modelo de interação padrão de uma aplicação do OGRE é detalhado nas Tabelas 1 e 2, juntamente com o mapeamento entre as entradas no *desktop* e no dispositivo móvel. Algumas teclas do teclado não existem em dispositivos embarcados e não puderam portanto ser mapeadas. Entretanto, se o desenvolvedor possui alguma outra tecla disponível que não está sendo usada na aplicação, ele pode atribuir o comportamento desejado a mesma. Sobre o mapeamento de *mouse* para caneta, o movimento pode ser relacionado com a caneta sendo arrastada pela tela sensível a toque. Devido ao fato de o botão direito não poder ser diretamente relacionado com nenhuma entrada, o desenvolvedor tem de escolher uma tecla para realizar a ação.

Tabela 1. Mapeamento da entrada de teclado de aplicações do OGRE em dispositivos móveis.

Entrada do *Desktop*	Entrada do Dispositivo Móvel	Ação Padrão
Tecla para Cima Tecla W	Botão de *hardware* para Cima Tecla W (se presente)	Move a câmera para frente
Tecla para Baixo Tecla S	Botão de *hardware* para Baixo Tecla S (se presente)	Mova a câmera para trás
Tecla para Esquerda Tecla A	Botão de *hardware* para Esquerda Tecla A (se presente)	Move a câmera para a esquerda
Tecla para Direita Tecla D	Botão de *hardware* para Direita Tecla D (se presente)	Move a câmera para a direita
Tecla Page Up	Não mapeada	Move a câmera para cima
Tecla Page Down	Não mapeada	Move a câmera para baixo
Tecla F	Tecla F (se presente)	Oculta/Mostra *overlays*
Tecla R	Tecla R (se presente)	Muda modo de renderização
Tecla P	Tecla P (se presente)	Muda filtragem de textura
Tecla T	Tecla T (se presente)	Oculta/Mostra coordenadas da câmera

Tabela 2. Mapeamento da entrada do *mouse* de aplicações do OGRE em dispositivos móveis.

Entrada do *Desktop*	Entrada do Dispositivo Móvel	Ação Padrão
Movimento do *mouse*	Arrastar da caneta	Move a câmera
Movimento + botão direito pressionado	Não mapeada	Rotaciona a câmera

No `PlatformManager` para Pocket PC, a captura da entrada do teclado e da caneta é feita através de *polling*. Para cada *frame* que é renderizado, os estados do teclado e do *mouse* são rastreados e os resultados são retornados para a aplicação. O programa pode checar se uma tecla está pressionada ou não ou se a caneta está tocando a tela, assim como verificar quantos *pixels* a caneta se moveu sobre a tela.

4.2.4. Implementação da matemática de ponto fixo

Objetivando propósitos de otimização, o código original do OGRE foi modificado para ser capaz de usar matemática de ponto fixo em seus cálculos. A arquitetura extensível do OGRE provê uma maneira elegante para se obter isto, já que o tipo que será usado para realizar as operações matemáticas pode ser escolhido pelo desenvolvedor em tempo de compilação, modificando-se apenas duas ou três linhas de código.

A Tabela 3 descreve as representações matemáticas usadas pelo OGRE e os tipos padrão responsáveis por implementá-las nas plataformas *desktop* e dispositivo móvel. O tipo `Real` representa um número real, o tipo `Real32` representa um número real de 32 *bits* e o tipo `RealGPU` representa o número real usado em operações matemáticas da GPU. No dispositivo móvel, a classe de matemática de ponto fixo `FixedPoint` representa os tipos `Real` e `RealGPU`. Inicialmente, a classe usava a biblioteca Intel *Graphics Performance Primitives* (GPP), que serve para prover as funções matemáticas apropriadas [62]. Entretanto, o GPP utiliza um *buffer* de 32 *bits* para realizar as operações, causando *overflow* quando uma cena grande tem de ser renderizada. Devido a isso, a classe `FixedPoint` utiliza um *buffer* de 64 *bits* para operar, suportando cenas mais complexas e uma matemática mais precisa durante as computações.

Entretanto, o tamanho do ponto fixo de OpenGL ES é 32 *bits*. Logo, após terminar os resultados da computação, a classe `FixedPoint` tem de reduzi-los para 32 *bits*. O tipo `RealGPU` é representado pelo tipo `float` em ambas as plataformas, visto que o Vincent ainda usa `floats` para operações relacionadas com GPU, como *geometry buffers*.

Tabela 3. Mapeamento dos tipos matemáticos do OGRE em dispositivos móveis.

Representação Matemática	Tipo do *Desktop*	Tipo do Dispositivo Móvel
Real	double	FixedPoint
Real32	float	FixedPoint
RealGPU	float	float

O próximo passo tomado foi corrigir alguns módulos do OGRE que usavam o tipo de ponto flutuante diretamente, em vez do tipo `Real`. Isso foi necessário para garantir que todas as partes do código estavam usando a definição matemática correta.

Depois disso, foi criada uma versão do `RenderSystem` baseada em OpenGL ES usando matemática de ponto fixo. Logo, passaram a existir duas implementações diferentes de `RenderSystem` disponíveis na plataforma Pocket PC: uma versão com ponto flutuante e uma versão com ponto fixo. Ambas usam OpenGL ES como API gráfica para operações de renderização. Foi dada preferência à implementação que utiliza ponto fixo, devido ao ganho em performance obtido.

4.2.5. RenderSystem usando Klimt

O Klimt é uma biblioteca 3D de código aberto para dispositivos móveis que é muito similar a OpenGL e OpenGL ES. Ela não usa aceleração em *hardware*, então o processo completo de renderização é feito em *software*. Apesar disso, ela tenta ser o mais eficiente possível na plataforma Pocket PC.

A estratégia usada pelo Klimt para aumentar a velocidade das aplicações é baseada principalmente em usar a biblioteca Intel GPP para matemática de ponto fixo e a biblioteca PocketHAL para uma implementação de um *frame buffer* linear, que acaba sendo mais rápido que o existente na própria API nativa do Pocket PC [63].

O `RenderSystem` usando Klimt foi compilado através do `RenderSystem` usando OpenGL ES, dado que as chamadas de funções e arquitetura como um todo são praticamente as mesmas. O projeto foi primeiramente testado na plataforma Win32, suportada pelo Klimt, e então validada no Pocket PC.

4.2.6. RenderSystem usando aceleração em *hardware*

Com o objetivo de melhorar a performance das aplicações que utilizam o OGRE4PPC, foi implementado um `RenderSystem` que explora a aceleração 3D em *hardware* oferecida pelo dispositivo móvel. O dispositivo móvel utilizado no desenvolvimento do porte (Dell Axim x51v) possui uma placa aceleradora (Intel 2700G) que implementa tanto OpenGL ES como Direct3D Mobile em *hardware*. Desta forma, iniciou-se o desenvolvimento de `RenderSystems` baseados em OpenGL ES e Direct3D Mobile acelerados em *hardware*. O `RenderSystem` baseado em OpenGL ES foi concluído, enquanto que o desenvolvimento do `RenderSystem` baseado em Direct3D Mobile está em andamento. A seguir, são descritas as tarefas realizadas na implementação de ambos os módulos.

O `RenderSystem` usando OpenGL ES acelerado em *hardware* foi baseado na versão que utiliza OpenGL ES implementado em *software*, ou seja, a biblioteca Vincent. Existem basicamente duas grandes diferenças entre os dois `RenderSystems`: a versão do OpenGL ES utilizada e o tipo matemático usado em operações relacionadas a GPU. Enquanto que o Vincent implementa o OpenGL ES 1.1, a placa aceleradora Intel 2700G traz uma implementação do OpenGL ES 1.0. Isso faz com que algumas funções suportadas pelo Vincent não estejam disponíveis na versão acelerada em *hardware*, como, por exemplo, as funções para obtenção de informações relativas a texturas. Foi necessário encontrar funções equivalentes ou estratégias que suprissem a falta dessas funções. No caso das texturas, as informações foram armazenadas numa tabela *hash* indexada pelo *ID* e pelo nível de *mipmap* da textura. Em relação ao tipo matemático da GPU, a implementação de OpenGL ES da Intel 2700G dá suporte somente a ponto fixo, em contraposição ao Vincent, que trabalha com ponto flutuante. Devido a isso, o tipo `RealGPU` teve que ser modificado para `int`, visto que os números no formato de ponto fixo são representados por um inteiro.

O `RenderSystem` usando Direct3D Mobile foi baseado no `RenderSystem` da plataforma *desktop* que utiliza Direct3D 9. Da mesma forma que ocorreu na implementação do `RenderSystem` usando OpenGL ES, todas as funções relativas a *shaders* tiveram que ser removidas, visto que o Direct3D Mobile ainda não dá suporte a programação em GPU. Outra mudança marcante diz respeito à inexistência de *vertex declarations* no Direct3D Mobile, que tiveram que ser substituídas por declarações em *Flexible Vertex Format* (FVF) para determinar as informações contidas na descrição dos vértices de uma geometria. Por fim, o Direct3D Mobile não dá suporte a múltiplos *vertex buffers* para uma mesma geometria, recurso esse muito utilizado pelo OGRE. Assim, é necessário que os vários *vertex buffers* sejam combinados em um único, além de que as modificações realizadas nos *buffers* de origem têm que se refletir no *buffer* combinado. Esta modificação ainda não foi concluída.

5. Estudos de caso e resultados

Dois estudos de caso foram realizados para avaliar o *framework* de RA descrito neste trabalho. No primeiro, apenas o OGRE4PPC foi testado, usando alguns programas de exemplo. No segundo, o *framework* por completo, incluindo captura de câmera e rastreamento de marcadores, foi testado com uma aplicação de RA simples.

O dispositivo móvel usado nos testes foi o PDA Dell Axim x51v. Ele possui um processador 624 MHz Intel PXA270 XScale, 256 MB de RAM, 64 MB de ROM e um display LCD com 16 *bit color depth* e 640x480 *pixels*. O PDA também possui um acelerador multimídia Intel 2700G com 16 MB de memória de vídeo. O sistema operacional é o Microsoft Windows Mobile 5.0 Pocket PC. A câmera usada no dispositivo móvel foi uma Spectec SD Camera SDC-001A, com resolução de 320x240 *pixels* e taxa de 15 *frames per second* (fps).

A ferramenta de desenvolvimento utilizada para o projeto foi o Microsoft Visual Studio .NET 2005 Professional Edition.

5.1. Avaliação do OGRE4PPC

Algumas aplicações foram usadas para validar o OGRE4PPC. As métricas de avaliação aplicadas nos testes foram o *frame rate*, a qualidade da imagem e a disponibilidade de funções. A ilusão do movimento é completamente relacionada com o *frame rate*, que é medido pelo número de quadros que são renderizados em um segundo (fps). Quanto mais alto o *frame rate* da aplicação, melhor será a animação resultante. A maneira como os polígonos e texturas são mostrados na tela é muito importante para o realismo da cena. A medida da qualidade da imagem é influenciada pela resolução da tela e dos algoritmos usados pela API de renderização. A aplicação móvel deve ter tantas funcionalidades como a da versão *desktop* do OGRE quanto possível. Contudo, algumas funções não serão acessíveis devido a restrições impostas pela plataforma, como a falta de processamento em GPU.

As aplicações usadas para verificar se o OGRE estava funcionando adequadamente foram os demos SkyBox e SkeletalAnimation, que fazem parte da distribuição oficial do OGRE. O código dos exemplos sofreu pequenas modificações relacionadas com a API de GUI do Pocket PC, cujas assinaturas de funções são diferentes das do Win32.

A Figura 27 ilustra alguns *screenshots* das aplicações de exemplo SkyBox e SkeletalAnimation, que fazem parte da distribuição oficial do OGRE, rodando no Pocket PC usando tanto o `RenderSystem` baseado em OpenGL ES como o baseado no Klimt.

Figura 27. *Screenshots* dos demos do OGRE no Pocket PC: SkeletalAnimation (esquerda) e SkyBox (centro) usando o `RenderSystem` OpenGL ES e SkeletalAnimation usando o `RenderSystem` Klimt (direita).

O *benchmark* da biblioteca foi feito usando uma aplicação onde uma nave é renderizada na tela. Os *frame rates* obtidos no Pocket PC são apresentados na Figura 28. Os valores foram obtidos de cenas com um número variado de triângulos e diferentes versões do `RenderSystem` foram utilizadas (OpenGL ES *software*, Klimt e OpenGL ES *hardware*).

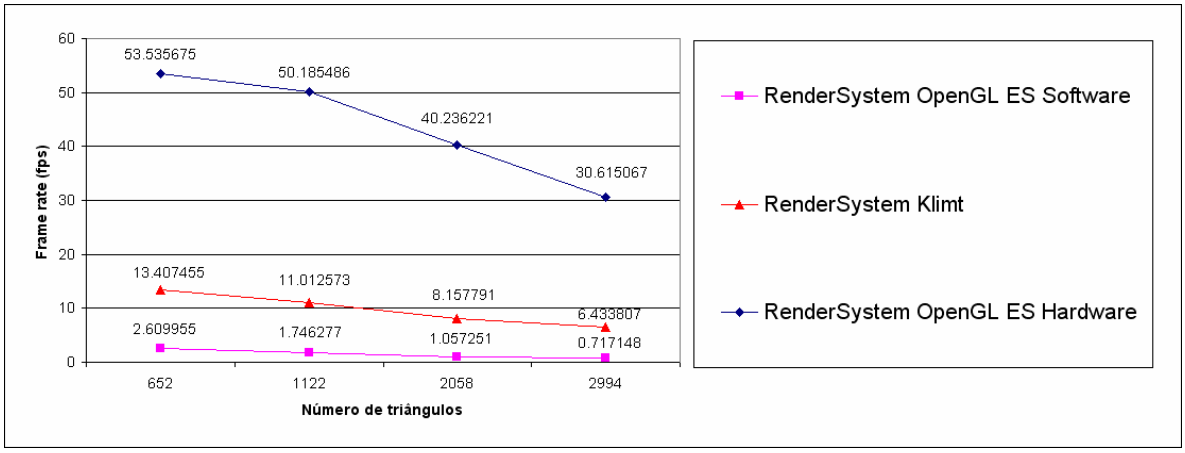

Figura 28. *Frame rate* no dispositivo móvel usando diferentes `RenderSystems`.

O *frame rate* do demo usando OpenGL ES *software* no Pocket PC variou entre 0.7 e 2.6 fps, o que é muito baixo para uma aplicação gráfica 3D em tempo real. A principal razão para este resultado pobre é o fato que a renderização do Vincent é emulada por *software*, sem uso de GPU. O resultado obtido usando o `RenderSystem` baseado no Klimt, que variou entre 6.4 e 13.4 fps, é bem melhor que o do OpenGL ES *software*, devido a otimizações presentes na biblioteca gráfica. Entretanto, está longe do *frame rate* obtido pelo `RenderSystem` usando OpenGL ES *hardware*,

que variou entre 30.6 e 53.5 fps, sendo este um resultado adequado para aplicações 3D em tempo real.

A qualidade da imagem apresentada pelo `RenderSystem` usando OpenGL ES (tanto *software* como *hardware*) foi superior ao do `RenderSystem` usando Klimt, que apresentou um pouco de *aliasing* nos objetos renderizados, embora seu aspecto tenha permanecido bastante aceitável.

Finalmente, existem algumas funcionalidades presentes nos `RenderSystems` baseados em *software* (tanto OpenGL ES como Klimt), porém ainda ausentes no `RenderSystems` OpenGL ES *hardware*, como, por exemplo, *skyboxes*.

5.2. Avaliação do *framework* de RA

Após se ter certeza que o porte do OGRE estava funcionando no Pocket PC, o *framework* de RA foi testado na plataforma. A métrica de avaliação utilizada neste estudo de caso foi o *frame rate*, visto que esse fator é muito importante para dar a sensação de aumento da realidade. Atrasos na apresentação da imagem do mundo real afetam as tarefas de interação com o usuário de sistemas de RA.

A aplicação de RA usada na avaliação do *framework* procura por marcadores na imagem capturada pela câmera e desenha um modelo 3D sobre eles. A versão 1.3.1 da biblioteca ARToolKitPlus foi utilizada para o rastreamento de marcadores. Os *screenshots* da aplicação podem ser vistos na Figura 29.

Figura 29. Aplicação de RA no Pocket PC, desenvolvida com o *framework* de RA.

Para medir o *frame rate* da aplicação, foram utilizados os `RenderSystems` baseados no Klimt e no OpenGL ES *hardware*, porque eles apresentaram os melhores resultados de *frame rate*

no primeiro estudo de caso. As quatro situações a seguir foram consideradas: sem marcadores na imagem (apenas exibição da imagem da câmera, sem reconhecimento de marcadores), e com 1, 2 e 3 marcadores na imagem. O modelo da nave foi desenhado em cada marcador detectado. A Figura 30 destaca os resultados obtidos.

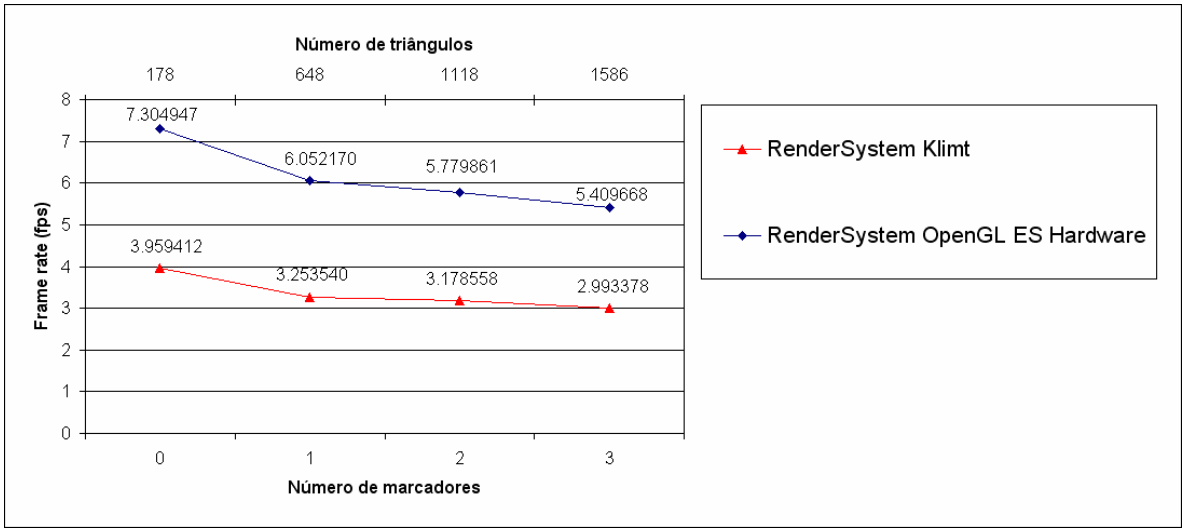

Figura 30. *Frame rate* no dispositivo móvel rodando a aplicação de RA e usando o `RenderSystem` baseado no Klimt.

Como esperado, os *frame rates* da aplicação usando Klimt foram mais baixos que os obtidos ao se usar OpenGL ES *hardware*. Com apenas a imagem da câmera sendo mostrada e sem objetos virtuais ou rastreamento de marcadores, o *frame rate* do `RenderSystem` usando Klimt foi de praticamente 4.0 fps, enquanto que o do `RenderSystem` usando OpenGL ES *hardware* foi de 7.3 fps. Quando o mundo real é aumentado, o *frame rate* do `RenderSystem` usando Klimt variou entre 3.0 e 3.3 fps, enquanto que o do `RenderSystem` usando OpenGL ES *hardware* variou entre 5.4 e 6.1 fps. Os resultados mostram que o *framework* já pode ser usado na prática em aplicações de RA no Pocket PC, principalmente quando a aceleração em *hardware* é utilizada.

6. Conclusões e trabalhos futuros

Um *framework* para a construção de aplicações de RA para a plataforma Pocket PC foi implementado, tornando a experiência de desenvolvimento mais fácil, produtiva e eficiente. A biblioteca OgreAR foi projetada de forma a possibilitar a criação de aplicações de RA portáveis. O *engine* OGRE foi portado com sucesso para a plataforma Pocket PC, provendo funcionalidades de renderização de alto nível para os desenvolvedores de dispositivos móveis. Finalmente, uma aplicação de RA foi desenvolvida usando o *framework* e testada em um dispositivo móvel.

O *framework* se mostrou viável para desenvolvimento de aplicações de RA na plataforma Pocket PC. As aplicações se mostraram portáveis, pois foi possível utilizar aplicações direcionadas a outras plataformas com alterações mínimas no código. Foi possível constatar a facilidade de utilização do *framework*, favorecida pela utilização de bibliotecas de alto nível, contribuindo para um desenvolvimento produtivo e focado na aplicação em si, sem preocupação com as bibliotecas de suporte. O desempenho do *framework* foi satisfatório ao se utilizar a aceleração em *hardware* presente em alguns dispositivos móveis, que, como esperado, foi bastante superior à abordagem baseada em *software*. Os estudos de caso realizados evidenciaram a necessidade de se utilizar aceleração gráfica para obter resultados adequados a aplicações de RA quando se utiliza um *framework* de alto nível.

Como trabalho futuro, assim que o projeto se tornar mais estável, seu código fonte será disponibilizado gratuitamente como um *add-on* do OGRE. Será feito também o refinamento do *framework* através da implementação de um `FrameGrabber` baseado no DirectShow. Desta forma, qualquer câmera compatível com o DirectShow poderia ser acessada pelo `FrameGrabber`. Também serão implementados novos `PatternDetectors`, utilizando outros conceitos, como, por exemplo, RA sem marcadores [65]. Algumas correções relativas ao `RenderSystem` usando OpenGL ES *hardware* se fazem necessárias, visto que algumas funcionalidades não estão funcionando na versão atual do *engine*, como os *skyboxes*. O `RenderSystem` baseado na biblioteca Direct3D Mobile será concluído. O compilador de C++ da Intel para Pocket PC pode também ser usado para melhorar ainda mais a velocidade de execução das aplicações, de acordo com [64]. Por fim, alguns aspectos de usabilidade relacionados à manipulação de entradas precisam ser investigados e avaliados durante a interação com o usuário.

Referências

[1] Oliver Bimber, e Ramesh Raskar, "Spatial Augmented Reality: Merging Real and Virtual Worlds", 1. ed., Wellesley: A K Peters, 2005. 368 p.

[2] Veronica Teichrieb, Severino G. Neto, Thiago S. Farias, João M. Teixeira, João P. Lima, Gabriel Almeida e Judith Kelner, "Augmented Ambient: an Interactive Mobility Scenario", Lecture Notes in Computer Science, 2007 (a ser publicado).

[3] Ronald T. Azuma, Yohan Baillot, Reinhold Behringer, Steven Feiner, Simon Julier, e Blair MacIntyre, "Recent Advances in Augmented Reality", IEEE Computers Graphics & Applications, Volume 21, Número 6, Novembro de 2001.

[4] Thiago S. Farias, Daliton Silva, Veronica Teichrieb, e Judith Kelner, "Tutorial: O Engine Gráfico Ogre", Brazilian Symposium on Computer Games and Digital Entertainment, Recife, 08-10 de Novembro, 2006.

[5] OGRE 3D. Disponível em: site do OGRE Team. URL: http://www.ogre3d.org, visitado em Março de 2007.

[6] João P. Lima, Thiago S. Farias, Veronica Teichrieb, e Judith Kelner, "Port of the OGRE 3D Engine to the Pocket PC Platform", Proceedings of the Symposium on Virtual Reality, Belém, pp. 65-76, 02-05 de Maio, 2006.

[7] Thiago S. Farias, João P. Lima, Veronica Teichrieb, e Judith Kelner, "OgreAR: Construção de Aplicações de Realidade Aumentada Utilizando Bibliotecas de Alto Nível", Workshop de Aplicações de Realidade Virtual, Recife, 21-24 de Novembro, 2006.

[8] Daniel Wagner, e Dieter Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices", Proceedings of the Computer Vision Winter Workshop, St. Lambrecht, 06-08 de Fevereiro, 2007.

[9] Paul Milgram, e Fumio Kishino, "A Taxonomy of Mixed Reality Visual Displays", IEICE Transactions on Information Systems, Volume E77-D, Número 12, Dezembro de 1994.

[10] Ronald T. Azuma, "A Survey of Augmented Reality", Presence: Teleoperators and Virtual Environments, Volume 6, Número 4, Agosto de 1997.

[11] How Augmented Reality Will Work. Disponível em: site Howstuffworks. URL: http://computer.howstuffworks.com/augmented-reality.htm.

[12] DirectShow. Disponível em: site da Microsoft Developer Network. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directshow/htm/directshow.asp, visitado em Fevereiro de 2007.

[13] DsVideoLib. Disponível em: site do SourceForge. URL: http://sourceforge.net/projects/dsvideolib, visitado em Fevereiro de 2007.

[14] Open Source Computer Vision Library. Disponível em: site da Intel Corporation. URL: http://www.intel.com/technology/computing/opencv, visitado em Fevereiro de 2007.

[15] V4LWiki. Disponível em: site do LinuxTV project. URL: http://linuxtv.org/v4lwiki/index.php/Main_Page, visitado em Fevereiro de 2007.

[16] FlyCapture SDK. Disponível em: site da Point Grey Research, Inc. URL: http://www.ptgrey.com/products/pgrflycapture/index.asp, visitado em Fevereiro de 2007.

[17] Hirokazu Kato, e Mark Billinghurst, "Marker Tracking and HMD Calibration for a Video-Based Augmented Reality Conferencing System", Proceedings of the Workshop on Augmented Reality, San Francisco, pp. 85-94, 20-21 de Outubro, 1999.

[18] Christian Geiger, Christian Reimann, Jörg Stöcklein, e Volker Paelke, "JARToolKit – A Java Binding for ARToolKit", Proceedings of the Augmented Reality ToolKit Workshop, Darmstadt, 5 pp., 29 de Setembro, 2002.

[19] Mark Fiala, "ARTag Revision 1, a Fiducial Marker System Using Digital Techniques", Technical Report NRC/ERB-1117, Institute for Information Technology, National Research Council Canada, 2004.

[20] MXR ToolKit. Disponível em: site do SourceForge. URL: http://mxrtoolkit.sourceforge.net, visitado em Março de 2007.

[21] Dave Shreiner, Mason Woo, Jackie Neider, e Tom Davis, "OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 2", 5. ed., Boston: Addison-Wesley, 2005. 896 p.

[22] Direct3D Reference. Disponível em: site da Microsoft Developer Network. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/directx9_c/dx9_graphics_reference_d3d.asp, visitado em Dezembro de 2006.

[23] Josie Werneck, "The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2", 1. ed., Boston: Addison-Wesley, 1994. 560 p.

[24] Coin3D. Disponível em: site da Systems in Motion AS. URL: http://www.coin3d.org, visitado em Fevereiro de 2007.

[25] Java 3D API. Disponível em: site da Sun Microsystems, Inc. URL: http://java.sun.com/products/java-media/3D, visitado em Dezembro de 2006.

[26] Andrea L. Ames, David R. Nadeau, e John L. Moreland, "The VRML 2.0 Sourcebook", 2. ed., New York: John Wiley and Sons, 2001. 688 p.

[27] Jun Rekimoto, e Katashi Nagao, "The World through the Computer: Computer Augmented Interaction with Real World Environments", Proceedings of the Symposium on User Interface Software and Technology, Pittsburgh, pp. 29-36, 15-17 de Novembro, 1995.

[28] Wouter Pasman, Charles Woodward, Mika Hakkarainen, Petri Honkamaa, e Jouko Hyväkkä, "Augmented Reality with Large 3D Models on a PDA: Implementation, Performance and Use Experiences", Proceedings of the International Conference on Virtual Reality Continuum and Its Applications in Industry, Singapore, pp. 344-351, 16-18 de Junho, 2004.

[29] Jurgen Gausemeier, Juergen Fruend,, Carsten Matysczok, Beat Bruederlin, e David Beier, "Development of a Real Time Image Based Object Recognition Method for Mobile AR-devices", Proceedings of the International Conference on Computer Graphics, Virtual Reality, Visualization and Interaction in Africa, Cape Town, pp. 133-139, 03-05 de Fevereiro, 2003.

[30] Daniel Wagner, Thomas Pintaric, Florian Ledermann, e Dieter Schmalstieg, "Towards Massively Multi-User Augmented Reality on Handheld Devices", Proceedings of the International Conference on Pervasive Computing, Munich, pp. 208-219, 08-13 de Maio, 2005.

[31] Daniel Wagner, Mark Billinghurst, e Dieter Schmalstieg, "How Real Should Virtual Characters Be?", Proceedings of the Conference on Advances in Computer Entertainment Technology, Hollywood, 8 p., 14-16 de Junho, 2006.

[32] Daniel Wagner, e Dieter Schmalstieg, "First Steps Towards Handheld Augmented Reality", Proceedings of the International Symposium on Wearable Computers, New York, pp. 127-137, 21-23 de Outubro, 2003.

[33] Daniel Wagner, e Istvan Barakonyi, "Augmented Reality Kanji Learning", Demo, Symposium on Mixed and Augmented Reality, Tokyo, 07-10 de Outubro, 2003.

[34] Project Nightingale – Digital Scrap Book. Disponível em: site da University of Sidney. URL: http://praxis.cs.usyd.edu.au/~peterris/?Projects/Digital+Scrap+Book, visitado em Fevereiro de 2007.

[35] Charles Woodward, "Augmented Reality, Feature Detection – Applications on Camera Phones". Disponível em: site do VTT Technical Research Centre of Finland. URL: http://cic.vtt.fi/projects/vbe-net/Interactive_3D/Woodward_English.pdf, visitado em Fevereiro de 2007.

[36] Siddharth Singh, Adrian David Cheok, Guo Loong Ng, e Farzam Farbiz, "3D Augmented Reality Comic Book and Notes for Children Using Mobile Phones", Proceedings of the Conference on Interaction Design and Children, Maryland, pp. 149-150, 01-03 de Junho, 2004.

[37] Anders Henrysson, e Mark Ollila, "UMAR – Ubiquitous Mobile Augmented Reality", Proceedings of the International Conference on Mobile Ubiquitous Multimedia, College Park, pp. 41-45, 27-29 de Outubro, 2004.

[38] Markus Kähäri, e David J. Murphy, "MARA - Sensor Based Augmented Reality System for Mobile Imaging", Demo, International Symposium on Mixed and Augmented Reality, Santa Barbara, 22-25 de Outubro, 2006.

[39] Anders Henrysson, Mark Billinghurst, e Mark Ollila, "Face to Face Collaborative AR on Mobile Phones", Proceedings of the International Symposium on Mixed and Augmented Reality, Vienna, pp. 80-89, 05-08 de Outubro, 2005.

[40] Anders Henrysson, Mark Ollila, e Mark Billinghurst, "Mobile Phone Based AR Scene Assembly", Proceedings of the International Conference on Mobile and Ubiquitous Multimedia, Christchurch, pp. 95-102, 08-10 de Dezembro, 2005.

[41] Siemens SX1. Disponível em: site da Wikipedia, the free encyclopedia. URL: http://en.wikipedia.org/wiki/Siemens_SX1, visitado em Fevereiro de 2007.

[42] Christian Reimann, "Kick-real, a Mobile Mixed Reality Game", Proceedings of the International Conference on Advances in Computer Entertainment Technology, Valencia, pp. 387, 15-17 de Junho, 2005.

[43] Mobile Java Games – Augmented Reality. Disponível em: site da Mauj Mobile. URL: http://www.mauj.com/games/games.php?catid=105670, visitado em Fevereiro de 2007.

[44] Erick Mendez, Daniel Wagner, e Dieter Schmalstieg, "Vidente: A Glance at AR on the UMPC and Smartphone Platform", Demo, International Symposium on Mixed and Augmented Reality, Santa Barbara, 22-25 de Outubro, 2006.

[45] Catapult Screenshots. Disponível em: site do Gizmondo Forums. URL: http://www.gizmondoforums.com/forums/index.php?showtopic=3207&st=0&p=43778 &#entry43778, visitado em Fevereiro de 2007.

[46] DirectShow for Windows Mobile-based Devices. Disponível em: site da Microsoft Developer Network. URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/mobilesdk5/html/mob5oridirectshow.asp, visitada em Fevereiro de 2007.

[47] Handheld Augmented Reality. Disponível em: site da Graz University of Technology. URL: http://studierstube.icg.tu-graz.ac.at/handheld_ar, visitado em Maio de 2006.

[48] Richard Harrison, "Symbian OS C++ for Mobile Phones", 1. ed., Chichester: John Willey & Sons, 2004. 448 p.

[49] ViCAM-III Digital Imaging Engine. Disponível em: site da Vista Imaging, Inc.. URL: http://www.vistaimaging.com/vicam3.htm, visitado em Fevereiro de 2007.

[50] OpenGL ES Overview. Disponível em: site do Khronos Group. URL: http://www.khronos.org/opengles/index.html, visitado em Janeiro de 2006.

[51] Klimt – The Open Source 3D Graphics Library for Mobile Devices. Disponível em: site da Graz University of Technology. URL: http://studierstube.icg.tu-graz.ac.at/klimt, visitado em Maio de 2006.

[52] Direct3D Mobile. Disponível em: site da Microsoft Developer Network. URL: http://msdn.microsoft.com/library/default.asp?url=/library/enus/wcemultimedia5/html/wce50oriDirect3DMobile.asp, visitado em Janeiro de 2006.

[53] Qusay H. Mahmoud. "Getting Started with the Mobile 3D Graphics API for J2ME". Disponível em: site da Sun MicroSystems, Inc. URL: http://developers.sun.com/techtopics/mobility/apis/articles/3dgraphics/index.html, visitado em Janeiro de 2006.

[54] Dieter Schmalstieg, "The Studierstube Augmented Reality Project", Presence: Teleoperators and Virtual Environments, Volume 11, Número 1, MIT Press, Cambridge, pp. 33-54, Fevereiro de 2002.

[55] STLport. Disponível em: site do STLport. URL: http://www.stlport.com, visitado em Maio de 2006.

[56] DevIL – a Full Featured Cross-Platform Image Library. Disponível em: site do SourceForge.net. URL: http://openil.sourceforge.net, visitado em Maio de 2006.

[57] FreeType2 Overview. Disponível em: site do SourceForge. URL: http://freetype.sourceforge.net/freetype2/index.html, visitado em Maio de 2006.

[58] Zziplib – the Library. Disponível em: site do SourceForge. URL: http://zziplib.sourceforge.net, visitado em Maio de 2006.

[59] Zlib Home Site. Disponível em: site do Zlib. URL: http://www.zlib.net, visitado em Maio de 2006.

[60] Vincent, the 3-D Rendering Library for Pocket PCs and Smartphones based on the Published OpenGL. Disponível em: site do SourceForge. URL: http://ogl-es.sourceforge.net, visitado em Maio de 2006.

[61] Hybrid Graphics Ltd – Developer Tools. Disponível em: site da Hybrid Graphics. URL: http://www.hybrid.fi/main/products/devtools.php, visitado em Maio de 2006.

[62] Intel Graphics Performance Primitives Version 4.0 for Microsoft Windows Mobile 2003 Software. Disponível em: site da Intel Corporation. URL: http://www.intel.com/design/pca/applicationsprocessors/swsup/gppv40_windows.htm, visitado em Maio de 2006.

[63] PocketHAL – the Hardware Access Library. Disponível em: PocketHAL site. URL: http://pockethal.droneship.com, visitado em Maio de 2006.

[64] Daniel Wagner, e Dieter Schmalstieg, "ARToolKit on the PocketPC Platform", Technical Report, Vienna University of Technology, Austria, 2003, pp. 1-2.

[65] Andrew I. Comport, Eric Marchand, Muriel Pressigout, e Fraçois Chaumette, "Real-Time Markerless Tracking for Augmented Reality: The Virtual Visual Servoing Framework", IEEE Transactions on Visualization and Computer Graphics, 2006.

Judith Kelner

Veronica Teichrieb

João Paulo Silva do Monte Lima