1
Introdução

O constante avanço das tecnologias de *hardware* gráfico propicia o desenvolvimento de um grande número de algoritmos para enriquecimento de ambientes virtuais, com o claro intuito de aproximá-los ao máximo da realidade. A visualização de ambientes complexos em tempo real requer um alto nível de sofisticação desses algoritmos de forma que seja mantida uma qualidade visual aceitável sem prejudicar a interação do usuário com a aplicação.

No entanto, um gargalo bastante comum em aplicações de computação gráfica é a geometria a ser renderizada, relacionada ao conjunto de vértices que constituem o modelo. Tal problema está diretamente relacionado às limitações impostas pela largura de banda da interface entre CPU e GPU (Owens, et al., 2008), que leva os desenvolvedores a reduzirem a quantidade de informações enviadas para a placa gráfica, sendo uma prática comum optar pela renderização de modelos com baixa quantidade de polígonos para favorecer o desempenho da aplicação.

Uma das formas mais comuns de contornar essa limitação é a utilização de algoritmos baseados em imagens para adição de detalhes sobre superfícies. Introduzido por Blinn (Blinn, 1978), o *bump mapping* permite o acréscimo de detalhes de rugosidade a um objeto através de uma implementação simples e de baixo custo computacional, mas, por ser apenas um truque de iluminação, não é capaz de representar efeitos mais complexos, como auto-sombreamento ou oclusões causados por interpenetrações (*self-shadowing* e *self-occlusion*), correção de perspectiva e detalhes da silhueta do objeto.

Displacement mapping, introduzido por Cook (Cook, 1984), possibilita a definição de detalhes geométricos através do deslocamento de pontos da superfície com base em informações de um mapa de altura. Dessa forma, garante-se a correta representação de detalhes das silhuetas e dos efeitos de *self-shadowing* e *occlusion*, ao custo de uma vigorosa subdivisão da malha a ser

renderizada em micropolígonos, o que torna esta técnica inadequada para aplicações em tempo real.

A partir disso, foi desenvolvido um número de técnicas de detalhamento envolvendo diferentes abordagens, com o intuito de aprimorar os efeitos do *bump mapping* e *displacement mapping* sem comprometer o desempenho. Algumas técnicas são baseadas em *ray tracing* (Pharr & Hanrahan, 1996) (Heidrich & Seidel, 1998) (Smits, Shirley, & Stark, 2000), distorção de imagens durante o processo de mapeamento para correção de perspectiva (Oliveira, Bishop, & McAllister, Relief texture mapping, 2000) e pré-computação de informações de visibilidade (Wang, et al., 2003) (Wang, et al., 2004). Os métodos baseados em *ray tracing*, devido ao alto custo computacional, são inadequados para aplicações em tempo real, enquanto os restantes, apesar de interativos, apresentam alto consumo de memória devido ao pré-processamento de um grande volume de informações.

Técnicas mais recentes que fazem uso de programação em GPU são capazes de reproduzir efeitos de sombreamento, oclusão e interpenetrações com maior desempenho e menores requerimentos de memória. Tais técnicas se baseiam principalmente em deslocamentos na imagem mapeada utilizando programas de *pixel* do *pipeline* gráfico (Kautz & Seidel, 2001) (Brawley & Tatarchuk, 2004) (Hirche, Ehlert, Guthe, & Doggett, 2004) (Policarpo, Oliveira, & Comba, 2005) (McGuire & McGuire, 2005) (Tatarchuk, 2006).

No entanto, apesar de reproduzirem detalhes de forma bastante convincente, tais técnicas não realizam modificações na geometria do objeto, não sendo adequadas a casos genéricos em que objetos podem ser vistos de diferentes ângulos e distâncias. Tal limitação resulta na representação incorreta de silhuetas, podendo afetar a qualidade da imagem final. Algumas abordagens propõem a geração de silhuetas através de deslocamentos de acordo com o ponto de vista do observador (Wang, et al., 2003) (Wang, et al., 2004) ou adequação de superfícies quádricas sobre o objeto para descrever, em nível de *pixel*, a curvatura da superfície (Oliveira & Policarpo, 2005), mas sofrem com problemas de *aliasing*.

Com as recentes atualizações na arquitetura dos *hardwares* gráficos, marcadas com os lançamentos das interfaces de programação (API) DirectX 11[1]

[1]SDK (junho de 2010) disponível para download em:
http://www.microsoft.com/download/en/details.aspx?id=6812.

(Valdetaro, Nunes, Raposo, & Feijó, 2010) e OpenGL 4.0[2], foram introduzidos dois novos estágios programáveis no *pipeline*, correspondentes aos *tessellation shaders*, que possibilitam a criação de grande volume de vértices na GPU. Como esse dispositivo é dotado de grande capacidade de processamento em paralelo, a geração massiva de primitivas pelo *hardware* gráfico pode ser utilizada para suprir a deficiência geométrica do modelo que é enviado à GPU, além de oferecer melhores meios para controlar o seu nível de detalhes, amenizando o gargalo provocado pela transferência de muitas informações para a placa gráfica. Um exemplo de aplicação de tesselação pode ser visto no jogo *Tom Clancy's H.A.W.X. 2*, que foi um dos primeiros jogos a utilizar este recurso por meio da API DirectX 11, apresentando terrenos bastante realistas com controle dinâmico do nível de detalhes. Uma comparação entre os detalhes de terrenos do jogo com tesselação desativada e ativada pode ser vista na Figura 1.

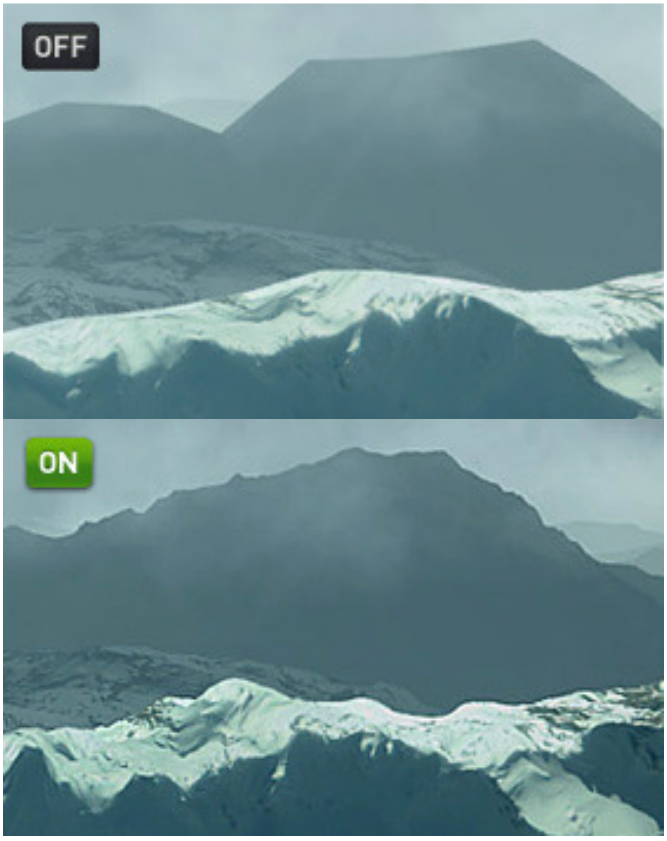

Figura 1: Tesselação de terrenos no jogo Tom Clancy's H.A.W.X. 2 (HAWX 2, 2010).

Com isso, surge a possibilidade de exploração deste *pipeline* para geração de detalhes sobre superfícies através da utilização de mapas de deslocamento na

[2] Informações sobre a versão mais recente disponíveis em http://www.opengl.org/.

GPU para realizar mudanças geométricas nas malhas subdivididas durante o estágio de tesselação. Tal procedimento não sofreria com as limitações dos métodos baseados em imagens, permitindo a visualização correta dos detalhes do objeto independente de ângulo ou distância de visualização.

A proposta desta dissertação, portanto, consiste em realizar um aparato sobre algumas técnicas de detalhamento baseadas em imagens comumente utilizadas em aplicações de tempo real e apresentar uma implementação que toma vantagem dos recursos recém-introduzidos no *pipeline* de renderização para adicionar detalhes geométricos autênticos a superfícies. Resultados experimentais demonstram vantagens em termos de desempenho e qualidade visual em relação a uma técnica de detalhamento baseada em imagens que é utilizada frequentemente em jogos eletrônicos para enriquecimento do ambiente virtual, a qual também foi implementada para que pudessem ser verificados os resultados deste trabalho. A análise dos resultados tem como principal objetivo identificar vantagens e desvantagens de cada abordagem, bem como avaliar propostas de trabalhos futuros que porventura aprimorem os resultados apresentados.

O Capítulo 2 apresenta rapidamente um conjunto de técnicas baseadas em imagens cujos objetivos são similares aos deste trabalho. O Capítulo 3 descreve os fundamentos do novo *pipeline*, tendo como foco o recurso de tesselação, e uma breve introdução à geração de texturas procedimentais. O Capítulo 4 revisita de forma mais detalhada algoritmos de algumas técnicas de detalhamento baseadas em imagens e expõe os critérios que levaram à escolha da técnica de *parallax occlusion mapping* para avaliar comparativamente os resultados deste trabalho. O Capítulo 5 expõe em detalhes a implementação proposta neste trabalho, apresentando as características de cada *shader* utilizado para tesselação e deslocamento de vértices. O Capítulo 6 apresenta uma análise dos resultados obtidos, realizando uma comparação do desempenho e da qualidade visual oferecida por cada implementação. O Capítulo 7, por fim, apresenta as conclusões e propostas de trabalhos futuros relacionados com o método descrito.