1
Introdução

A área de renderização em tempo real sempre proporcionou grandes desafios aos pesquisadores interessados. Elaborar algoritmos capazes de processar uma cena foto-realística em uma taxa de quadros por segundo interativa não é tarefa fácil.

Apesar da grande evolução do hardware gráfico ao longo dos anos, a demanda por uma visualização mais imersiva é constante. Essa imersão significa, na maioria das vezes, mais realismo. Ou seja, texturas maiores, simulações físicas mais realísticas, modelos com grande quantidade de vértices e algoritmos mais complexos, entre outras consequências.

Para ajudar na elaboração de tais algoritmos, as placas de vídeo vêm evoluindo constantemente nos últimos anos. Até 2000 as placas de vídeo tinham um pipeline fixo que era apenas configurável [Wikipedia, 2010]. Ou seja, os algoritmos de computação gráfica eram restringidos a usar alguns recursos pouco flexíveis das GPUs.

A partir daí começou a surgir o pipeline programável. Em 2001 foi lançado o DirectX8 com suporte a programação de vertex shaders e pixel shaders em uma linguagem assembly. Esta versão tinha disponibilidade de registradores e acesso a texturas limitadas, mas foi a primeira versão que permitia programar o hardware gráfico. Foi chamado de Shader Model 1.0. Em 2002 foi lançado o DirectX9 com suporte ao Shader Model 2.0 e introdução de uma linguagem de shader de alto-nível chamada HLSL (High Level Shader Language). No Shader Model 2.0 foi introduzido o conceito de múltiplos alvos de renderização (multiple render-targets), texturas com precisão de ponto flutuante e mais registradores de acesso [Wikipedia, 2010].

Em 2004 o DirectX9 sofreu uma atualização grande, chamada de DirectX9.0c. Nesta atualização houve a introdução do Shader Model 3.0, permitindo uma série de instruções novas, incluindo acesso a textura no Vertex Shader. Porém, neste Shader Model ainda existiam apenas dois estágios programáveis no pipeline. Era possível apenas manipular vértices ou pixels, não era possível ainda criar informação em tempo de execução na GPU.

Em 2006 foi lançado o DirectX10 com suporte ao SM 4.0(Shader Model

4.0) [Limaye, 2009]. Foi introduzido um novo estágio no pipeline, o Geometry Shader. Pela primeira vez era possível criar vértices na GPU. Os programadores tinham acesso a cada triângulo que passava na placa gráfica, incluindo suas adjacências. Muitos pensaram que poderiam criar milhões de vértices em tempo de execução na placa gráfica. Porém, apesar de o Geometry Shader ser útil para uma série de algoritmos, ele é um estágio lento. O controle de fluxo ou criação de muitos vértices neste estágio faz com que a aplicação tenha uma queda de desempenho drástica, tornando inviável a renderização em tempo real.

Apesar da recente migração da interface de I/O de AGP para PCI-Express [Jim Brewer, 2004][Bhatt, 2004], o recurso mais caro em todo o pipeline gráfico hoje ainda é a largura de banda [Owens et al., 2008]. Enviar milhares de vértices a cada quadro para a placa de vídeo é um processo custoso. Por isso muitas aplicações de visualização 3D em tempo real ainda usam modelos com baixa quantidade de polígonos para garantir uma taxa de quadros por segundo alta.

Por outro lado, um dos recursos mais baratos no pipeline é a alta capacidade de processamento da GPU. Uma única placa de vídeo hoje tem até 512 processadores trabalhando em paralelo [NVIDIA, 2010], além do fato que múltiplas placas de vídeo podem ser combinadas montando clusters poderosos. Desta maneira, fica claro que uma troca de transferência de memória por operações aritméticas na GPU seria muito benéfica para a melhora de desempenho do pipeline gráfico.

Com base nisto, em 2008 foi proposto um novo pipeline gráfico baseado no DirectX11 [Drone et al., 2010] que introduz o Shader Model 5.0 com duas novas partes programáveis e uma parte fixa, permitindo assim a criação de primitivas em massa na GPU. Em Setembro de 2009 chegava ao mercado as primeiras placas com suporte ao DirectX11, foi a série HD5xxx da ATi. Mais tarde, em abril de 2010, a Nvidia lançou a série Geforce 4xx com suporte nativo ao DirectX11. Um gráfico da evolução do pipeline baseado nas placas da Nvidia e no DirectX é apresentado na Figura 1.1.

A proposta deste trabalho consiste em implementar algoritmos que tirem proveito da geração dinâmica de vértices na GPU e que anteriormente eram feitos em CPU, além de propor algoritmos novos que explorem o potencial do novo pipeline gráfico, e sempre avaliando o ganho de desempenho do Tessellator criando vértices na GPU ao invés de enviá-los da CPU para GPU. Esta dissertação usa como nomenclatura os termos da API do DirectX, porém, tudo descrito aqui pode ser feito com a API do OpenGL, apenas as nomenclaturas de alguns termos dessas APIs são diferentes.

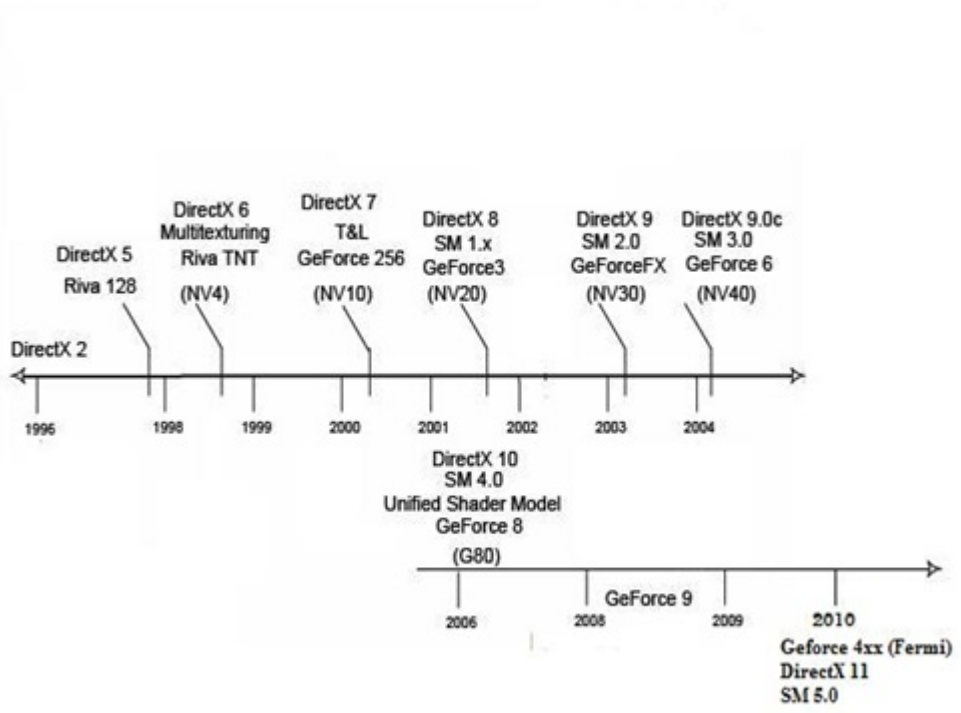

Figura 1.1: Linha do tempo da evolução do pipeline baseado nas placas Nvidia

A primeira parte do trabalho avalia o desempenho do Tessellator em comparação com o envio dos dados dos vértices pela CPU. Depois analisaremos o desempenho do PN-Triangles e do Phong Tessellation no Tessellator. Estes dois algoritmos têm como característica o uso de informações locais para o refinamento do malha. Em seguida são propostas duas novas soluções que usam o Tessellator: Renderização de Tubos e Terrenos em GPU.

Este documento está organizado como se segue. No capítulo que se segue, será apresentado o novo pipeline gráfico e discutida sua performance e economia de largura de banda. No capítulo 3 o novo pipeline será usado para implementar 2 algoritmos de subdivisão de superfícies: PN-Triangles e Phong Tessellation, que serão comparados entre si em termos qualitativos e quantitativos. Nos capítulos seguintes, serão mostrados dois novos algoritmos desenvolvidos para o novo pipeline: primeiramente um algoritmo para renderização de tubos com anti-aliasing e LOD contínuo e, no capítulo seguinte, um algoritmo para renderização de terrenos. Finalmente, o capítulo 6 apresenta as conclusões e trabalhos futuros.