



Gustavo Bastos Nunes

Explorando aplicações que usam a geração de vértices em GPU

Dissertação de Mestrado

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática da PUC-Rio

Orientador: Prof. Alberto Barbosa Raposo
Co-Orientador: Prof. Bruno Feijó

Rio de Janeiro
Agosto de 2011



Gustavo Bastos Nunes

Explorando aplicações que usam a geração de vértices em GPU

Dissertação apresentada como requisito parcial para obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Alberto Barbosa Raposo

Orientador

Departamento de Informática – PUC-Rio

Prof. Bruno Feijó

Co-Orientador

Departamento de Informática – PUC-Rio

Prof. Waldemar Celes Filho

Departamento de Informática – PUC-Rio

Prof. Rodrigo Penteado Ribeiro de Toledo

UFRJ

Dr. Luciano Pereira Soares

PUC-Rio

José Eugenio Leal

Coordenador Setorial do Centro Técnico Científico – PUC-Rio

Rio de Janeiro, 16 de agosto de 2011.

Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, da autora e do orientador.

Gustavo Bastos Nunes

Graduou-se em Engenharia de Computação na Pontifícia Universidade Católica do Rio de Janeiro em 2008. De 2008 a 2010 trabalhou no laboratório de Computação Gráfica da PUC-Rio (TecGraf) desenvolvendo sistemas de realidade virtual e visualização científica. De Dez/2010 a Maio/2011 trabalhou na T&T desenvolvendo simuladores virtuais e serious games. A partir de Outubro/2011 começará na Microsoft(Redmond) como Software Development Engineer in Test na equipe do Microsoft SharePoint.

Ficha Catalográfica

Nunes, Gustavo Bastos

Explorando aplicações que usam a geração de vértices em GPU / Gustavo Bastos Nunes ; orientador: Alberto Barbosa Raposo ; co-orientador: Bruno Feijó. – 2011.

111 f. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2011.

Inclui bibliografia

1. Informática – Teses. 2. Hardware Tessellation. 3. Programação em GPU. 4. Geração de vértices. 5. DirectX11. 6. OpenGL4. 7. Shader Model 5.0. I. Raposo, Alberto Barbosa. II. Feijó, Bruno. I III. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. IV. Título.

CDD: 004

Agradecimentos

À Deus, que me deu força sempre.

Aos meus pais, Antonio e Maria, por me proporcionarem um ensino de qualidade. Às minhas irmãs, Silvia e Flavia, por sempre me incentivarem a estudar. À Marcela por sempre me apoiar. Aos meus amigos Alexandre e Rodrigo por partilharem seus conhecimentos comigo. Aos amigos do Siviep, Thiago, Pablo, Siviano, Henrique, Galak, Mariano e Galinha Velha, pelos bons momentos de TecGraf. Aos Professores Alberto e Bruno pela motivação a cada idéia que surgia na minha cabeça. À professora Karin pelo apoio em toda minha vida acadêmica. Ao professor Carlos Tomei por estar sempre disponível para tirar minhas dúvidas. Ao Professor Rodrigo pelas idéias e motivações. Aos professores Gattass e Waldemar, pelas excelentes aulas de Computação Gráfica que pude desfrutar. Aos demais professores do DI pela qualidade de ensino proporcionada. Aos meus amigos de Niterói: Falcão, Decão, Bê de óculos, Costela, Lopes, Dudu, Augusto, Daniel, André e respectivas pelas faltas nos churrascos.

Resumo

Nunes, Gustavo Bastos; Raposo, Alberto Barbosa; Feijó, Bruno. **Explorando aplicações que usam a geração de vértices em GPU**. Rio de Janeiro, 2011. 111p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Um dos maiores gargalos do pipeline gráfico hoje é a largura de banda disponível entre a GPU e CPU. Para minimizar esse gargalo funcionalidades programáveis foram inseridas nas placas de vídeo. Com o Geometry Shader é possível criar vértices em GPU, porém, este estágio da pipeline apresenta performance baixa. Com o lançamento das novas APIs gráficas (DirectX11 e OpenGL4) em 2009, foi adicionado o Tessellator, que permite a criação de vértices em massa na GPU. Esta dissertação estuda este novo estágio da pipeline, bem como apresenta algoritmos clássicos (PN-Triangles e Phong Tessellation) que originalmente foram feitos para CPU e propõe novos algoritmos (Renderização de Tubos e Terrenos em GPU) para tirar proveito deste novo paradigma.

Palavras-chave

Hardware Tessellation; Programação em GPU; Geração de vértices; DirectX11; OpenGL4; Shader Model 5.0.

Abstract

Nunes, Gustavo Bastos; Raposo, Alberto Barbosa (advisor); Feijó, Bruno (co-advisor). **Exploring applications that use vertex generation on GPU.** Rio de Janeiro, 2010. 111p. MSc. Dissertation – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

One of the main bottlenecks in the graphics pipeline nowadays is the memory bandwidth available between the CPU and the GPU. To avoid this bottleneck, programmable features were inserted into the video cards. With the Geometry Shader launch it is possible to create vertices in the GPU, however, this pipeline stage has a low performance. With the new graphic APIs (DirectX11 and OpenGL4) a Tessellator stage that allows massive vertex generation inside the GPU was created. This dissertation studies this new pipeline stage, as well as presents classic algorithms (PN-Triangles and Phong Tessellation) that were originally designed for CPU and proposes new algorithms (Tubes and Terrain rendering in the GPU) that takes advantage of this new paradigm.

Keywords

Hardware Tessellation; GPU Programming; Vertex Generation; DirectX11; OpenGL4; Shader Model 5.0.

Sumário

1. Introdução	14
2. O Novo Pipeline Gráfico	17
2.1 Possibilidades no novo pipeline gráfico.....	18
2.2 Introdução ao novo pipeline.....	20
2.3 Performance do Tessellator.....	26
2.4 Discussão.....	33
3. PN-Triangles vs Phong Tessellation	36
3.1 Introdução.....	36
3.2 Trabalhos Relacionados.....	37
3.3 Continuidade de Superfícies.....	39
3.4 Métodos Aproximativos vs Interpolativos.....	41
3.5 Avaliação da superfície.....	42
3.6 PN-Triangles.....	43
3.7 Phong Tessellation.....	53
3.8 Resultados.....	57
4. Renderizando Tubos a partir de Curvas Discretas com Anti-Aliasing e LOD contínuo usando Tecelagem em Hardware	71
4.1 Introdução.....	71
4.2 Trabalhos Relacionados.....	73
4.3 O algoritmo de geração de tubos.....	73
4.4 Implementação.....	77
4.5 Resultados.....	83
4.6 Melhorando o consumo de memória por frame ainda mais.....	84
4.7 Limitações.....	85
5. Renderização de terreno usando processo local paralelo em GPU	88
5.1 Introdução.....	88
5.2 Trabalhos Relacionados.....	89
5.3 Visão Geral.....	91

5.4 Análise do mapa de altura.....	93
5.5 View-dependent LOD.....	94
5.6 Implementação.....	97
5.7 Resultados.....	99
5.8 Discussão.....	100
6. Conclusão.....	104
6.1 Contribuições.....	105
6.2 Trabalhos Futuros.....	105
Referências bibliográficas.....	106

Lista de figuras

1.1 Linha do tempo da evolução do pipeline baseado nas placas Nvidia	16
2.1 Modelo com grande quantidade de polígonos	19
2.2 Modelo com diferentes níveis de detalhe produzidos na GPU	19
2.3 O novo pipeline gráfico	20
2.4 Patch com 16 pontos de controle representando uma superfície de Bézier	21
2.5 Economia ao animar modelo em baixa frequência	22
2.6 O Hull Shader	23
2.7 Fases do Hull Shader	24
2.8 Exemplos de uso do Tessellator em diferentes domínios	25
2.9 Fluxograma da nova parte do pipeline	26
2.10 Superfícies paramétricas geradas em GPU	32
2.11 Gráfico representando o ganho de performance do Tessellator	35
3.1 Foto do jogo MAFIA II	38
3.2 Duas curvas que não são contínuas	39
3.3 Continuidade C^0	40
3.4 Continuidade C^1	40
3.5 Continuidade C^2	41
3.6 Continuidade C^3	41
3.7 Continuidade C^4	42
3.8 Interpolativo vs aproximativo	42
3.9 Máscara para um algoritmo hipotético de subdivisão	43
3.10 Pontos de controle do patch de geometria	44
3.11 Pontos de controle do patch de normal	45
3.12 Interpolação linear das normais (acima) e variação quadráticas (embaixo)	46
3.13 Reflexão da normal no meio da aresta pelo plano perpendicular a ela	47
3.14 Normais variando linearmente (esquerda) e quadraticamente (direita)	48

3.15	Projeções e interpolações do Phong Tessellation.....	54
3.16	Modelo original sem tecelagem.....	58
3.17	Modelo usando o algoritmo Phong Tessellation.....	59
3.18	Modelo usando o algoritmo PN-Triangles e normais quadráticas.....	61
3.19	PN-Triangles com interpolação linear das normais.....	63
3.20	PN-Triangles com interpolação quadrática das normais.....	63
3.21	Modelo original sem tecelagem.....	64
3.22	Modelo usando o algoritmo Phong Tessellation.....	64
3.23	Modelo usando o algoritmo PN-Triangles e normais quadráticas.....	65
3.24	PN-Triangles com interpolação linear das normais.....	65
3.25	PN-Triangles com interpolação quadrática das normais.....	66
3.26	Modelo original sem tecelagem.....	66
3.27	Modelo usando o algoritmo Phong Tessellation.....	67
3.28	Modelo usando o algoritmo PN-Triangles e normais quadráticas.....	67
3.29	PN-Triangles com interpolação linear das normais.....	68
3.30	PN-Triangles com interpolação quadrática das normais.....	68
3.31	Demonstrativo da diferença de performance entre o Phong Tessellation e o PN-Triangles para o modelo da Pessoa.....	69
3.32	Demonstrativo da diferença de performance entre o Phong Tessellation e o PN-Triangles para o modelo do Tigre.....	70
4.1	Tubos 3D renderizados com a técnica proposta em um visualizador de campos de petróleo.....	72
4.2	Grupo de pontos em preto representando o caminho do tubo. Em azul a linha central do tubo.....	74
4.3	Seleção de pontos com as tangentes, normais e bi-normais Associadas ao longo da curva. Áreas com derivada numérica alta requerem mais pontos para a reconstrução precisa.....	75
4.4	A) Um caso de sequência de pontos ruim. B) A sequência ruim após a aplicação da interpolação de Catmull-Rom. C) A seleção de pontos da sequência e cálculo das tangentes, normais e bi-normais. D) Cada 64 pontos implica em uma primitiva (patch). E) O patch é uma primitiva do tipo quad com 64xLOD linhas. F) Uma linha do quad com o número de pontos definido pelo LOD.	

G) Cada ponto da linha é transformado em um círculo no espaço 3D no plano $y=0$. H) usando as normais e bi-normais cada ponto é transformado do círculo para a seção de corte do tubo.....	86
4.5 Esquerda – Tubo com LOD baixo. Direita – Tubo com LOD alto.....	86
4.6 Mesma cena: na esquerda com 16x GPU anti-aliasing. Na direita com a correção de aliasing proposta.....	87
4.7 Gráfico mostrando a porcentagem de ganho em FPS do nosso algoritmo com e sem a correção de aliasing proposta comparado comparado com a abordagem em CPU sem anti-aliasing.....	87
5.1 Pontos de referência de mosaico (quatro pontos no meio dos contornos e um no centro do patch). V_i é um vértice do patch.....	92
5.2 A área no topo possui primeira e segunda derivada pequenas. A parte inferior em maior valor de primeira e derivada e menor de segunda derivada. Já a área do meio apresenta maior segunda derivada. A segunda derivada denota locais que precisam de refinamento	93
5.3 TessFactor como uma função de HAM. G não é definido por $T < 1$ porque o intervalo do TessFactor é $[1,64]$	95
5.4 Processo de cálculo do T_{min} que para quando o erro projetado é igual ao erro permitido. h é o valor máximo de altura.....	96
5.5 TessFactor como uma função de distância da câmera d (equação 5-1).....	98
5.6 Frames por segundo (fps) para o caso das Figuras 5.7c e 5.7d. “Straight View-dependent LOD” não usa T_{max} , “No Lod with Tessellator” utiliza o TessFactor 64 para toda a malha, “No LOD, without Tessellator” é apenas uma referência (o caso onde toda a malha é transferida do CPU para o GPU). “Our technique FC” mostra a performance com o frustum culling ligado.	100
5.7 Modelos de terreno em wireframe e visualização final correspondente gerados pela técnica proposta rodando em uma Nvidia GTX480 e um Intel core i7. (a) e (b): mapa de altura de 65536x65536, área de $8 \times 10^{12} \text{ m}^2$ com precisão de 40mx40m, à 52-109fps. (c) e (d): mapa de altura de 2048x2048, área de $4.4 \times 10^{10} \text{ m}^2$ com precisão	

de 100mx100m, à 347-399fps.....	101
5.8 Contagem de triângulos para o mesmo caso da Figura 5.6.....	102
5.9 Frames por segundo (fps) para o caso 65536x65536 nas Figuras 5.7(a) e 5.7b.....	103

Lista de tabelas

2.1 Ganho de performance do Tessellator.....	33
2.2 Tabela mostrando a economia de memória com o uso do Tessellator (em Megabytes).....	34
3.1 FPS do modelo da Pessoa usando o Phong Tessellation e o PN- Triangles.....	60
3.2 FPS do modelo do Tigre usando o Phong Tessellation e o PN- Triangles.....	60
3.3 Ganho percentual e absoluto do Phong Tessellation sobre o PN-Triangles para o modelo do Tigre.....	62
3.4 Ganho percentual e absoluto do Phong Tessellation sobre o PN-Triangles para o modelo da Pessoa.....	62
4.1 Comparação de FPS das técnicas.....	84
4.2 Comparação consumo de largura de banda CPU-GPU.....	85