

Desenvolvimento de um Arcabouço para a Geração Procedural e Visualização de Terrenos em Tempo-Real

Fábio Markus Nunes Miranda
Orientador: Prof. Luiz Chaimowicz
Co-Orientador: Carlúcio Cordeiro

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Apresentação final - POC II

Sumário

1 Motivação

2 Metodologia

3 Proposta

4 Resultados

5 Conclusão e trabalhos futuros

6 Bibliografia

Motivação

- Atualmente, há uma necessidade de se criar modelos 3D cada vez maiores e com grande nível de detalhe.
- Porém, quanto maior e mais detalhado o modelo, mais tempo terá que ser gasto por um modelador para fazê-lo.
- Aí entra a geração procedural...

O que é geração procedural?

- Geração procedural é um termo genérico para descrever algoritmos que determinam características de efeitos ou modelos.
- Há diversos tipos de técnicas e algoritmos, cada um aplicado a uma determinada área:
 - L-System: geração de árvores e cidades.
 - Fractais e Perlin Noise: geração de terrenos e texturas

Vantagens da geração procedural

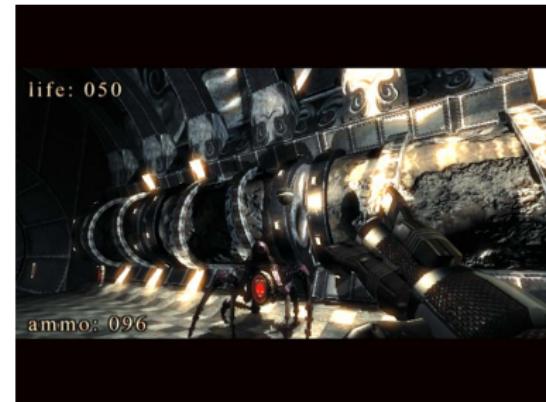
- Flexibilidade: alterando os parâmetros do algoritmo, é possível gerar um grande número de modelos.
- Espaço: não há necessidade de um grande espaço em disco, já que tudo será ditado por algoritmos.

Exemplos

- **.kkrieger**

Praticamente tudo gerado
proceduralmente

- Elite (1984)
- SpeedTree



Exemplos

- .kkrieger
- Elite (1984)
Oito galáxias, 256 planetas.
- SpeedTree



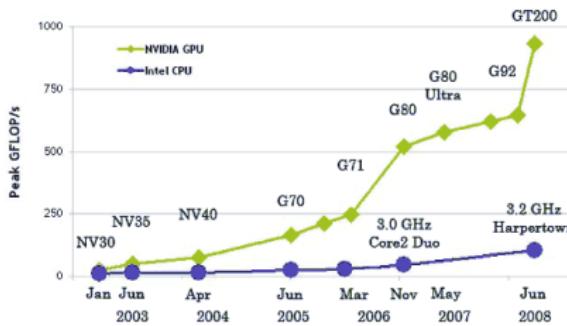
Exemplos

- .kkrieger
- Elite (1984)
- SpeedTree
Árvores geradas proceduralmente.



GPU

- As atuais placas de vídeo possuem *GPUs* com milhares de unidades de processamento.
- Construídas de forma a processar da melhor maneira possível um grande número de dados independentes entre si, como é o caso de vértices e *pixels*.



Sumário

1 Motivação

2 Metodologia

3 Proposta

4 Resultados

5 Conclusão e trabalhos futuros

6 Bibliografia

Metodologia

- Livro *Texturing and Modeling: A Procedural Approach* [1].
- Estudo das melhores formas de reduzir o gasto com memória através de estruturas de dados do OpenGL.
- Estudo da arquitetura da GPU.
- Implementação do sistema.

Sumário

1 Motivação

2 Metodologia

3 Proposta

4 Resultados

5 Conclusão e trabalhos futuros

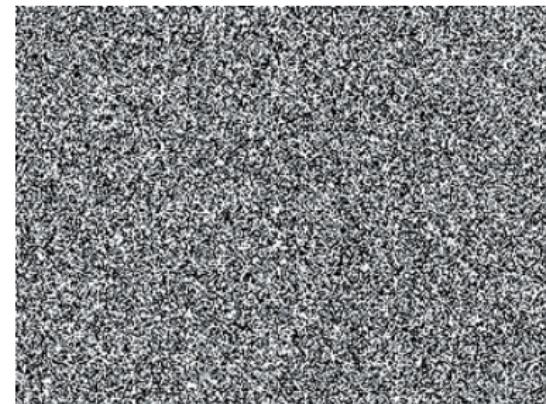
6 Bibliografia

Proposta

- O objetivo deste trabalho é construir um sistema em que seja possível gerar proceduralmente terrenos utilizando a *GPU* de placas gráficas e também na *CPU*.

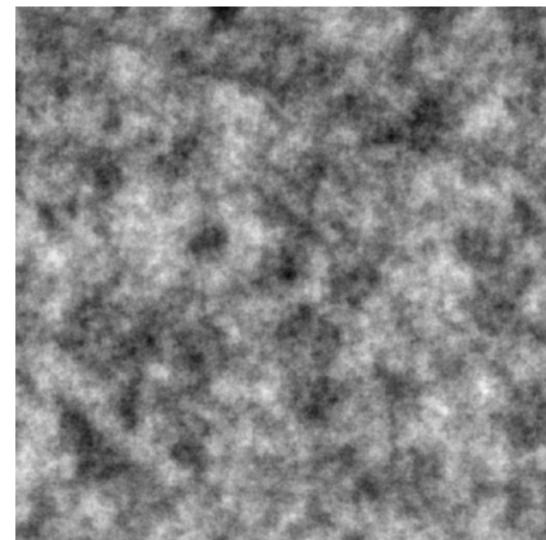
Proposta

- Os terrenos gerados serão baseados no algoritmo *ridged multifractal noise*.
- **Ruído de Perlin:**
Função que retorna um valor com certa consistência com os outros valores ao seu redor.
- Fractais:



Proposta

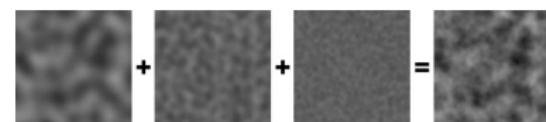
- Os terrenos gerados serão baseados no algoritmo *ridged multifractal noise*.
- **Ruído de Perlin:**
Função que retorna um valor com certa consistência com os outros valores ao seu redor.
- Fractais:



Proposta

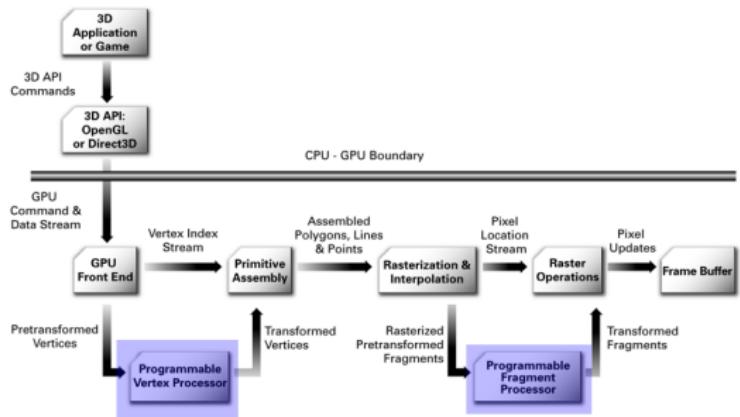
- Os terrenos gerados serão baseados no algoritmo *ridged multifractal noise*.
- Ruído de Perlin:
 - Fractais:

objetos geométricos complexos, na qual a complexidade surge da repetição de uma forma em uma extensão de escalas. [1]



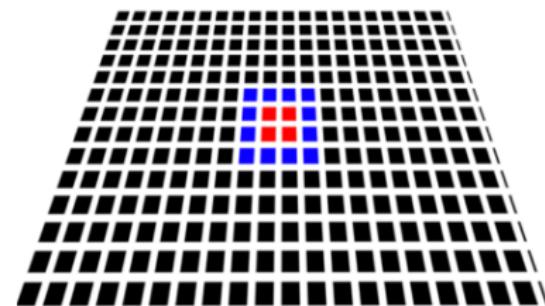
Geração na GPU

- A geração é feita através de conjuntos de instruções (shaders) executados na GPU.
- O *pipeline* gráfico das GPUs executa dois tipos de shaders:
 - Vertex shader: executado para cada vértice da cena.
 - Fragment shader: executado para cada fragmento dos polígonos (a saída do shader é uma cor, salva em uma textura).



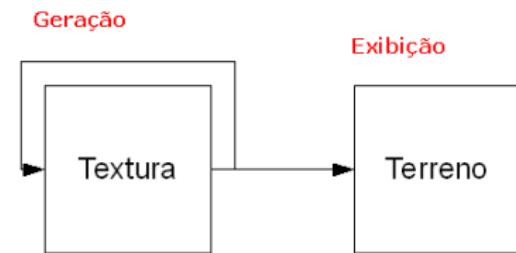
Terreno

- O terreno é dividido em um grid, sendo que cada *patch* é criado separadamente.
- As geometrias são criadas somente uma vez (no início do programa). Quando o jogador se move, apenas os mapas de altura são trocados.
- Os *patches* mais próximos da câmera possuem um maior número de vértices.



Geração na GPU

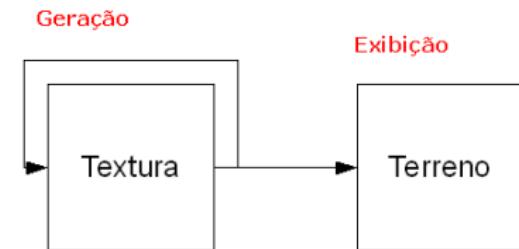
- Geração do terreno:
 - Fragment shader: calcular o novo terreno procedural e salva o resultado em um *frame buffer object* (FBO).
 - Geração incremental (*Ping-Pong*): a cada *frame*, uma iteração do *ridged multifractal noise* é feita. O *frame buffer* resultante servirá como entrada na próxima iteração.
 - Na última iteração, é calculado as normais de todos os vértices (iluminação).



Geração na GPU

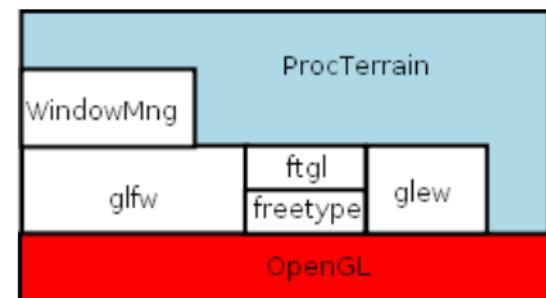
■ Visualização do terreno:

- Vertex shader: após o cálculo do terreno, o vertex shader irá consultar o que foi gerado, e ajustar a altura dos vértices do terreno.
- Fragment shader: Cálculo da iluminação (por pixel).



Sistema

- glfw, ftgl, freetype, glew: bibliotecas para OpenGL.
- WindowMng: camada que simula um aplicativo gráfico genérico.
- TerrainMng: gera e controla os terrenos.



Sumário

1 Motivação

2 Metodologia

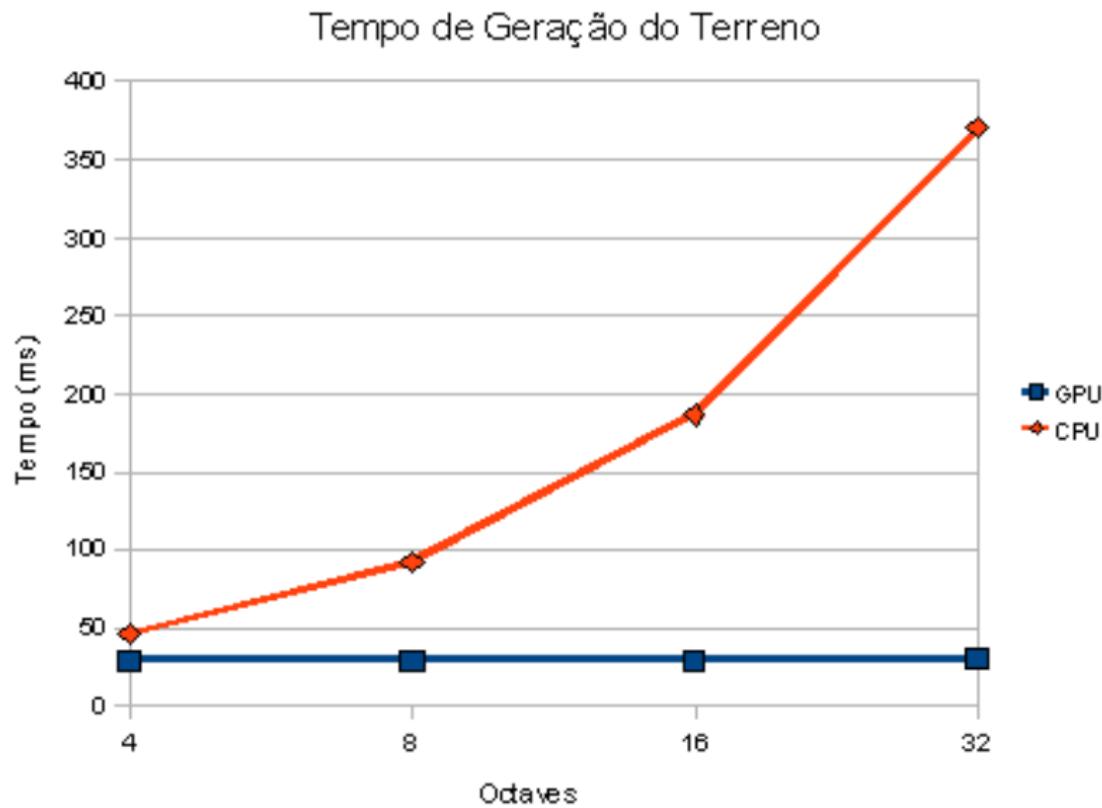
3 Proposta

4 Resultados

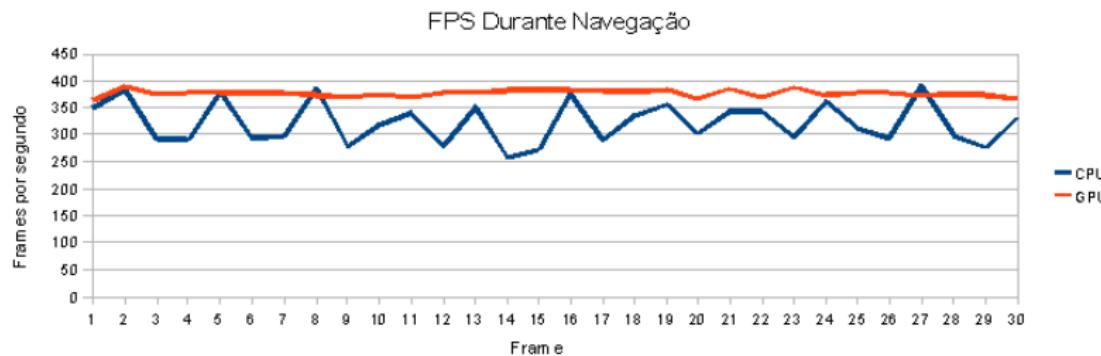
5 Conclusão e trabalhos futuros

6 Bibliografia

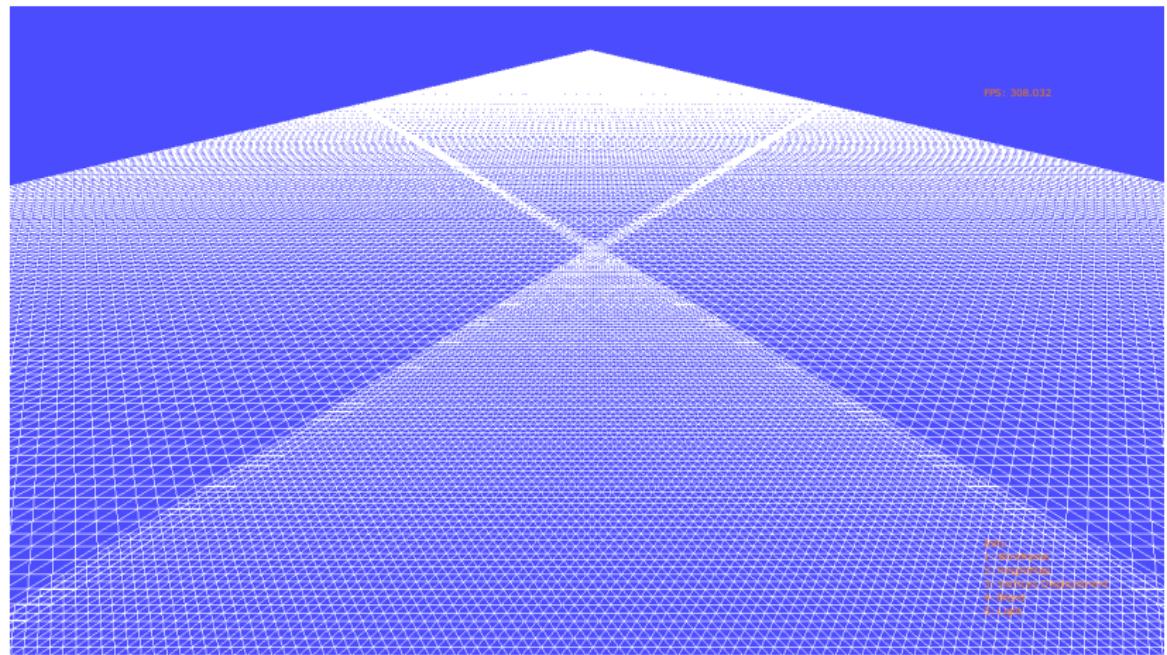
Resultados



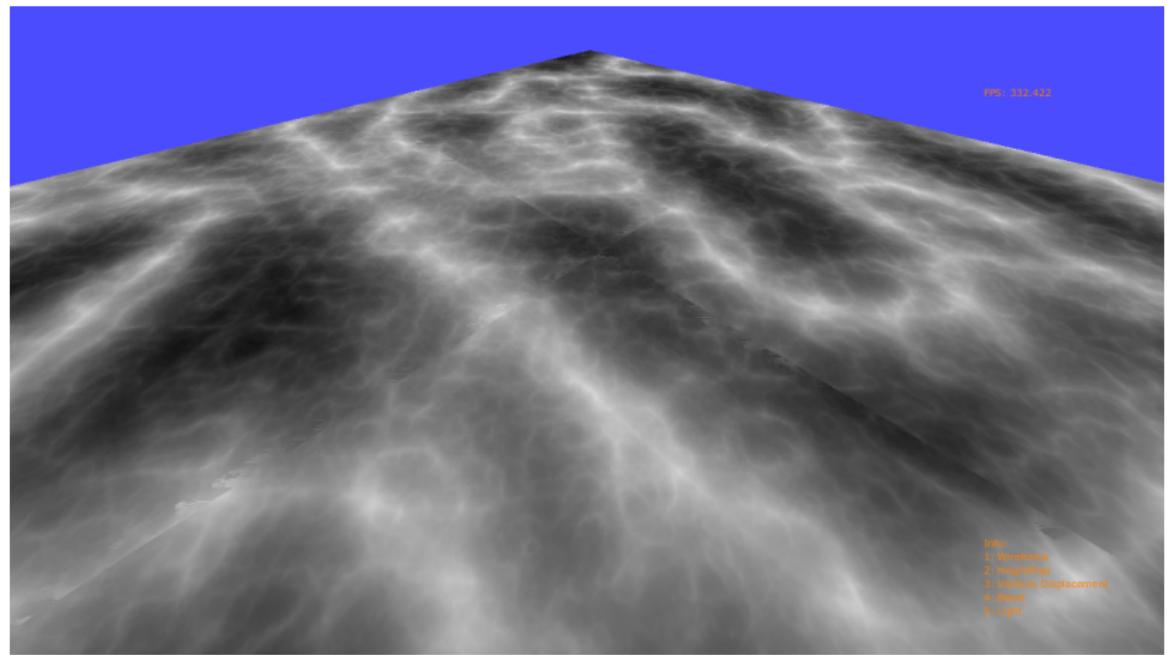
Resultados



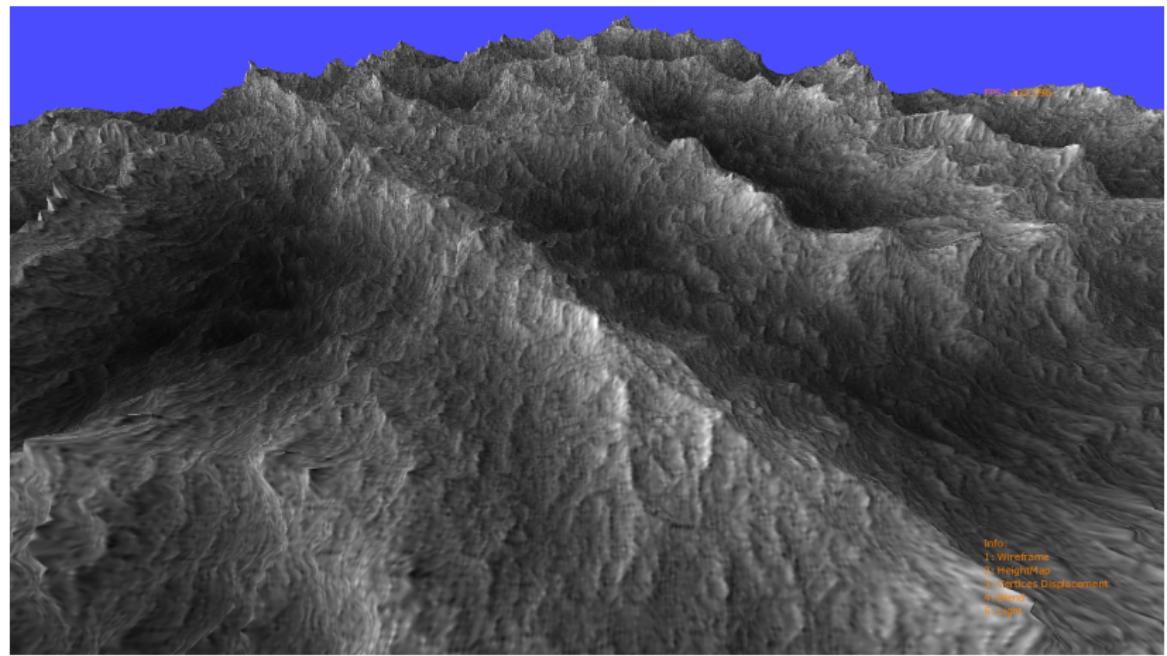
Resultados



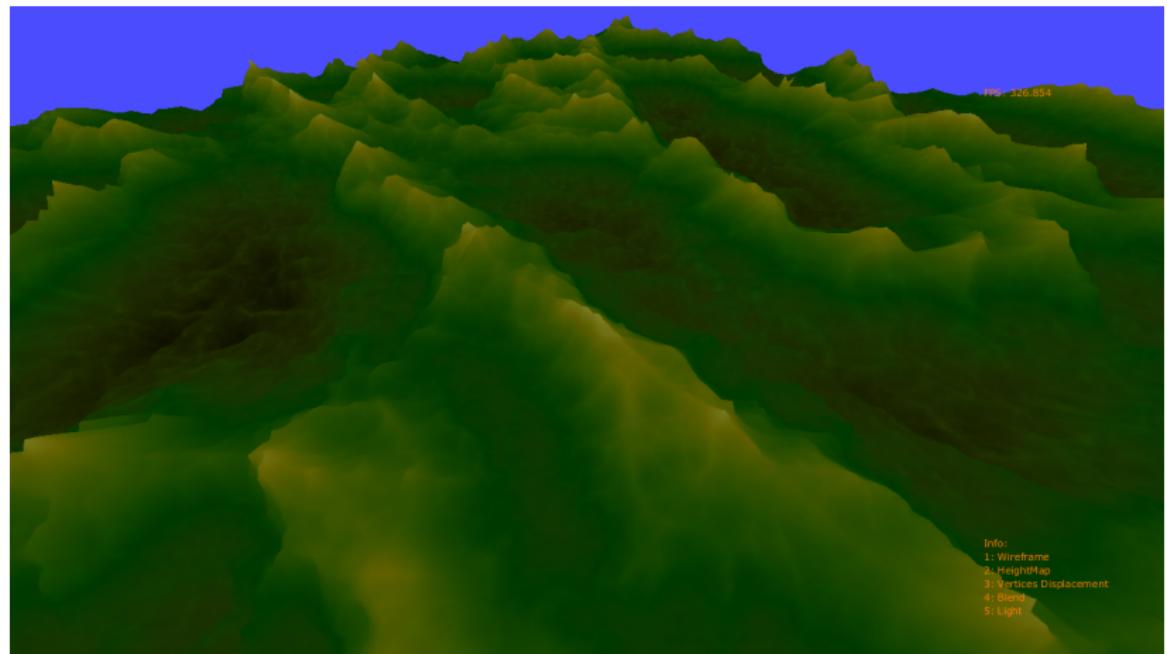
Resultados



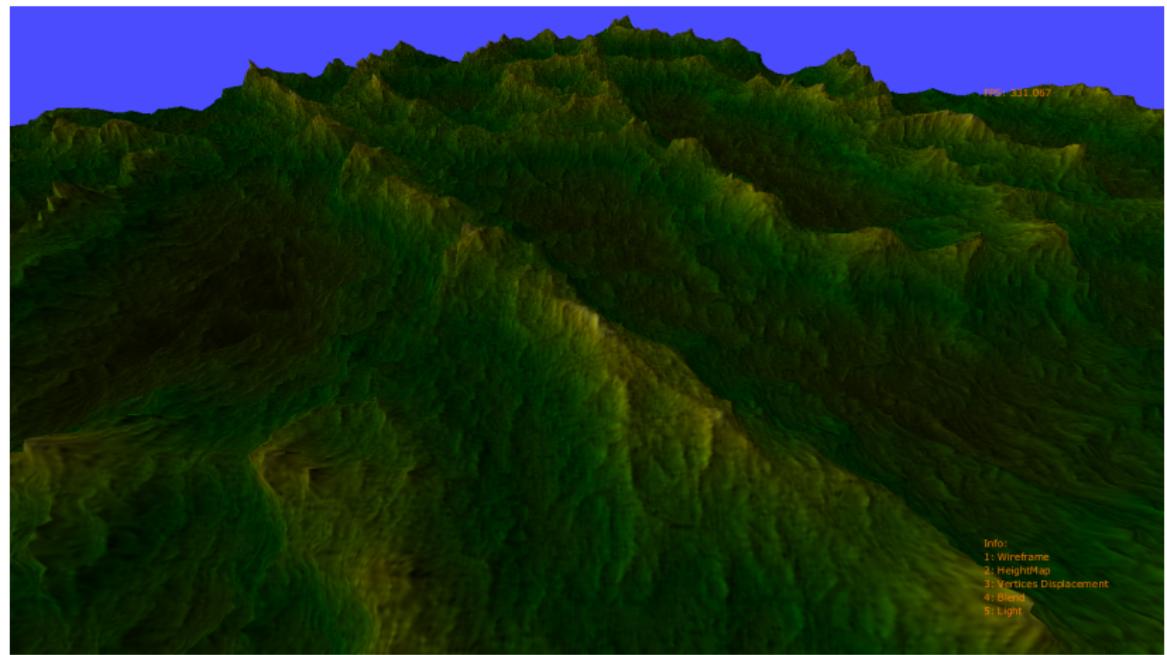
Resultados



Resultados



Resultados



Sumário

- 1 Motivação**
- 2 Metodologia**
- 3 Proposta**
- 4 Resultados**
- 5 Conclusão e trabalhos futuros**
- 6 Bibliografia**

Conclusão e trabalhos futuros

- Uma base de desenvolvimento já foi estabelecida.
- Já é possível visualizar terrenos, mas é preciso melhorar a transição entre eles.
- POC 2:
 - Terrenos esféricos.
 - Uso de texturas com sombras pré-calculadas.
 - Interface gráfica mais atrativa.

Sumário

1 Motivação

2 Metodologia

3 Proposta

4 Resultados

5 Conclusão e trabalhos futuros

6 Bibliografia

-  David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley.
Texturing and Modeling: A Procedural Approach.
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.
-  Benoit B. Mandelbrot.
The Fractal Geometry of Nature.
W. H. Freeman, August 1982.
-  Przemyslaw Prusinkiewicz and Aristid Lindenmayer.
The algorithmic beauty of plants.
Springer-Verlag New York, Inc., New York, NY, USA, 1996.
-  geo-spatial data acquisition home.
Disponível em: <http://emrl.byu.edu/gsda/>. Acessado em: 23 nov. 2008.
-  Pixar animation studios.
Disponível em: <http://www.pixar.com/>. Acessado em: 23 nov. 2008.
-  Ian bell's elite pages.
Disponível em: <http://www.iancgbell.clara.net/elite/>. Acessado em: 23 nov. 2008.
-  Procedural content generation.
Disponível em: <http://lukehalliwell.wordpress.com/2008/08/05/procedural-content-generation/>.
Acessado em: 23 nov. 2008.
-  Acmc projects ,cg rendering of coral at the university of queensland.
Disponível em: http://www.acmc.uq.edu.au/Projects/CG_Rendering.html. Acessado em: 23 nov. 2008.
-  Object oriented framework development.
Disponível em: <http://www.acm.org/crossroads/xrds7-4/frameworks.html>. Acessado em: 23 nov. 2008.

-  **Glfw - an opengl framework.**
Disponível em: <http://glfw.sourceforge.net/>. Acessado em: 23 nov. 2008.
-  **Anttweakbar gui library to tweak parameters of opengl and directx applications.**
Disponível em: <http://www.antisphere.com/Wiki/tools:anttweakbar>. Acessado em: 23 nov. 2008.
-  **Devil - a full featured cross-platform image library.**
Disponível em: <http://openil.sourceforge.net/>. Acessado em: 23 nov. 2008.
-  **Gamedev.net - 'slope lighting' terrain.**
Disponível em: <http://www.gamedev.net/reference/articles/article1436.asp>. Acessado em: 23 nov. 2008.
-  **Stefan Greuter and Jeremy Parker.**
Undiscovered worlds - towards a framework for real-time.
In *In Proc. of the Fifth Intern. Digital Arts and Culture Conference*. Press, 2003.
-  **Yoav I H Parish and Pascal Müller.**
Procedural modelling of cities.
In *In Proc. ACM SIGGRAPH, (Los Angeles, 2001)* ACM Press, pages 301–308, 2001.
-  **George Kelly and Hugh McCabe.**
Citygen: An interactive system for procedural city generation.
In *Game Design & Technology Workshop*, 2006.
-  **Stefan Greuter, Jeremy Parker, Nigel Stewart, and Geoff Leach.**
Real-time procedural generation of 'pseudo infinite' cities.
In *GRAPHITE '03: Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia*, pages 87–ff, New York, NY, USA, 2003. ACM.
-  **Jacob Olsen.**
Realtime procedural terrain generation.

In *Department of Mathematics And Computer Science (IMADA)*, 2004.



Lukas Zimmerli and Paul Verschure.

Delivering environmental presence through procedural virtual environments.

In *PRESENCE 2007, The 10th Annual International Workshop on Presence*, 2007.



Farès Belhadj.

Terrain modeling: a constrained fractal model.

In *AFRIGRAPH '07: Proceedings of the 5th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa*, pages 197–204, New York, NY, USA, 2007. ACM.



Speedtree — idv, inc.

Disponível em: <http://www.speedtree.com/>. Acessado em: 23 nov. 2008.



Mojoworld generator.

Disponível em: <http://www.mojoworld.org/>. Acessado em: 23 nov. 2008.



Infinity.

Disponível em: <http://www.infinity-universe.com/Infinity/>. Acessado em: 23 nov. 2008.



The official spore and spore creature creator site.

Disponível em: <http://www.spore.com/>. Acessado em: 23 nov. 2008.



Ken perlin's homepage.

Disponível em: <http://mrl.nyu.edu/~perlin/>. Acessado em: 23 nov. 2008.

Dúvidas?