Grupo 4 - Tiago Koji Castro Shibata (8988730), Adilson Torres Gregório de Souza (8988254)

Sistemas de Computação de Alto Desempenho
Projeto Final

Especificação Preliminar

- **Introdução**

A especificação preliminar possui como objetivo definir e validar o tema do projeto. Explicitamos a importância do algoritmo escolhido e indicamos quais serão os entregáveis finais.

- **Algoritmo escolhido**

O projeto final é a criação de um programa e otimização utilizando técnicas de programação paralela. O programa deve implementar um algoritmo que possa ser paralelizável. O relatório a ser redigido deve explicar o algoritmo escolhido e mostrar *benchmarks* do programa em sua versão serial e versões paralelas, indicando claramente ganhos de performance em diferentes condições (como diferentes níveis de paralelismo, tamanhos de entrada e parâmetros do algoritmo).

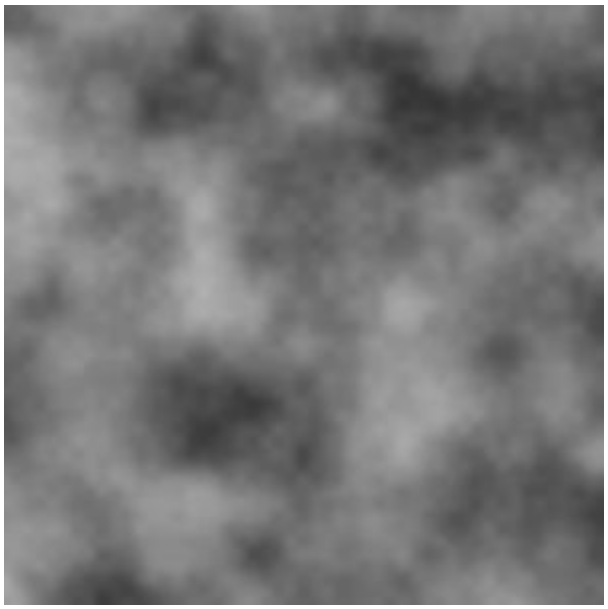
O algoritmo escolhido foi o **ruído Simplex**. Esse é um algoritmo de geração de ruído com aparência aleatória.

O algoritmo pertence aos geradores de ruído de gradiente: algoritmos que criam um *grid* de gradiente pseudo-aleatório, e, então, para qualquer ponto (x, y), calcula produtos escalares da posição (x, y) com os gradientes de pontos vizinhos do grid e realiza interpolação linear dos resultados. O resultado é um ruído suave e contínuo:

Exemplo de ruído de gradiente bidimensional

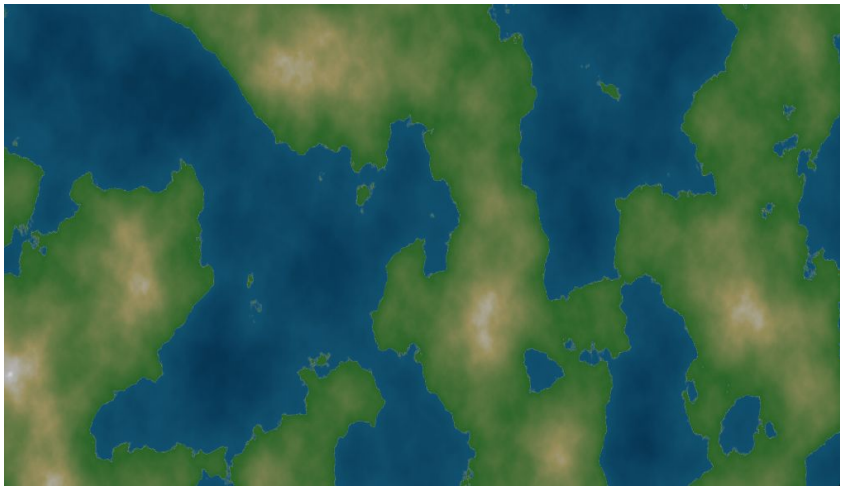
A principal aplicação do algoritmo é na geração procedural de texturas para processamento gráfico. No caso do algoritmo selecionado, como visto na amostra acima, seu ruído possui aparência aleatória, mas as “features” distinguíveis possuem tamanho similar. Esse ruído é frequentemente somado a texturas existentes (como paredes, fogo ou fumaça) para torná-los aparentemente mais realistas de maneira procedural e programática, servindo como pós-processamento de texturas.

Além de servir para pós-processamento de texturas existentes, o ruído pode servir para **geração** procedural de texturas totalmente novas. Somar o próprio ruído nele mesmo, após mudar sua escala de valores absolutos (escala de cada pixel/amplitude da imagem) e realizar *zoom* da imagem (mudança de frequência da imagem), gera efeitos visualmente agradáveis e convincentes. Por exemplo, criando ruído fractal (onde são geradas oitavas contendo a mesma energia, ou seja, múltiplas imagens de ruído dividindo-se a frequência por 2 e reduzindo a amplitude a cada divisão), adquire-se texturas que lembram fumaça ou nuvens:



Exemplo de ruído fractal

Se colorirmos o ruído de maneira conveniente, temos a formação de imagens que lembram fumaça, nuvens, relevos ou mapas. Por exemplo, gerando ruído com média próxima de 0 e colorindo valores positivos em um gradiente de verde e negativos em um gradiente de azul, podemos obter a seguinte imagem:



Ruído fractal em escala de cores conveniente
Gerado utilizando SimplexNoiseCImg

- **Entregáveis finais**

Ao final do curso, entregaremos programa com versão serial do algoritmo, versão paralela e *benchmark* de cada versão por captura do tempo consumido para geração do ruído. A versão paralela provavelmente utilizará OpenMP.

O programa deve ser capaz de gerar imagens em escala de cinza de ruído de gradiente e ruído fractal e salvá-las em disco.

Se tivermos tempo, também implementaremos rotinas para colorir o ruído em escalas de cores variadas para criar texturas convincentes (como, por exemplo, padrões para relevo em verde e azul ou padrões de água, fumaça, entre outros).