OpenCL para simulação de partículas em múltiplas threads

Gustavo Roman Marangon, Rodrigo Nietiedt de Almeida

{gromanmarangon,rodrigonietiedt}@gmail.com

Abstract. *This paper has as objective to describe a basic model of implementation for a particles simulation using OpenCL. It also shows the results of a test of this implementation.*

Resumo. *Este artigo tem como objetivo descrever um modelo básico de implementação para uma simulação de partículas usando a OpenCL. O artigo também mostra os resultados de um teste de implementação do sistema para a simulação de grãos de areia.*

1. Introdução

Uma simulação de partículas é um problema ideal para ser resolvido através de programação em *GPU*, já que é altamente paralelizável. Cada partícula é processada em uma *thread* própria dentro de um *work group*, e dependendo da abordagem possui baixa necessidade de acesso às outras *threads*, como é o caso de simulações de partículas que não colidem entre si. Mas para simular-se partículas de areia é necessário computar colisões das partículas tanto com o ambiente quanto entre si.

Como ferramenta para a programação em *GPU* foi escolhida a *OpenCL*, e o fator decisivo para esta escolha foi a sua portabilidade. As principais fabricantes de hardware gráfico já suportam *OpenCL* através de implementações da sua especificação.

2. OpenCL

A OpenCL é uma linguagem que tem como objetivo principal o desenvolvimento através de periféricos dedicados. Esses periféricos costumam oferecer recursos que otimizam o processamento de certos tipos de dados. Um dos recursos principais é o uso de multiplas threads em dispositivos como placas de vídeo, que possuem otimização em hardware e costumam apresentar desempenho superior à de uma CPU em alguns casos.

Um programa em OpenCL deve ser compilado e executado pelo dispositivo dedicado. Para permitir acesso à esses recursos, a OpenCL oferece recursos ao como funções que permitem ao programa em CPU enviar um código ao dispositivo para ser compilado e executado. O programa em CPU também pode, através desses recursos da OpenCL, ler e escrever dados na memória do dispositivo.

A *OpenCL* foi escolhida por oferecer a possibilidade do processamento paralelo atrvés de um dispositivo dedicado, e o uso de uma *thread* exclusiva para cada partícula.

3. Simulação de Partículas

Partículas são estruturas constituídas por uma quantidade reduzida de informações, tendendo-se ao mínimo que o objeto real sendo modelado necessita para ser representado de maneira identificável. Objetos modelados como partículas possuem a característica de existirem em números massivos e de serem de dimensões pontuais.

A simulação de areia inclui a sua modelagem como um grupo de partículas sólidas que colidem entre si e que possuem baixo atrito, bem como sofrem influência constante da gravidade. Foi implementado um comportamento físico realístico para as partículas, onde elas são representadas por um Tamanho, uma Posição e uma Força. O Tamanho é utilizado para a detecção e tratamento de colisões. A Posição é utilizada tanto para a representação gráfica quanto para o processamento físico. Já a Força é utilizada somente para o processamento físico. Conforme o valor de força que uma partícula possui, ela tenta deslocar-se em uma direção. Nesta etapa, ela verifica se alguma das outras partículas existentes colidirá com ela caso ela realize o movimento intendido. Caso haja colisão, a partícula em movimento desloca-se até o ponto de contato com a partícula colidinte, e então atualiza o valor da sua força em função da posição da outra partícula. Caso não haja colisão, a partícula simplesmente move-se em função da sua força.

Este modelo de simulação física não foi atingido com sucesso, já que a questão do multiprocessamento e da otimização exigem soluções mais robustas do que foi conseguido realizar. O principal problema foi o tratamento da colisão, onde os cálculos utilizados para simular a interação entre as partículas não atingiram o ideal para uma simulação física realista.

4. Conclusão

Foi possível, com a *OpenCL*, simular o comportamento de partículas, calculando colisão entre objetos e aplicando força de forma paralela. Porém, não foram alcançadas otimização e velocidade de processamento, para a geração de partículas o suficiente para simular o comportamento de grãos de areia. Ainda foi possível perceber a velocidade de processamento do código paralelizado pela *OpenCL* em relação a um código comum serializado. Uma quantidade elevada de partículas conseguiram realizar seus cálculos individuais de movimento, ao mesmo tempo que acessavam as posições de todas as outras, a fim de processar colisões, e apesar de todo este processamento a aplicação continuou funcionando sem notar-se queda na taxa de quadros por segundo.

References

[1] Getting Started With *OpenCL* and *GPU computing*. Disponivel em: <http://www.thebigblob.com/getting-started-with-opencl-and-gpu-computing/>. Acesso em: 11 nov 2013

[2] BJURKOVSKI (*username-codeplex)* . **OpenCL Tutorials.** Disponível em: http://opencl.codeplex.com/wikipage?title=OpenCL%20Tutorials%20-%201. Acesso em: 11 nov 2013

[3] CHRIS MCCABE. **Setting up *OpenCL* in visual studio**. Disponível em: http://kode-stuff.blogspot.com.br/2012/11/setting-up-opencl-in-visual-studio_1.html. Acesso em: 10 nov 2013