COMPUTING TRAJECTORIES USING OPENCL

DTS

Abstract. This tool demonstrates how to execute a simple game physics engine for computing trajectory of a projectile within a framework of OpenCL kernels.

1 Introduction

To compute projectile trajectory coordinates using parametric representation, when target and launch point are at the same level, we use,

$$\begin{aligned} x(t) &= (v_0 \cos(\theta))\, t \\ y(t) &= (v_0 \sin(\theta))\, t - \frac{gt^2}{2} \\ v_x(t) &= v_0 \cos(\theta) \\ v_y(t) &= v_0 \sin(\theta) - gt \\ \|\boldsymbol{v}(t)\|_2 &= \sqrt{v_0^2 - 2gtv_0 \sin(\theta) + (gt)^2} \end{aligned} \tag{1}$$

where v_0 is the initial speed. When a projectile is dropped from a moving system,

$$\begin{aligned} x(t) &= v_0 t \\ y(t) &= h_0 - \frac{gt^2}{2} \\ v_x(t) &= v_0 \\ v_y(t) &= -gt \\ \|\boldsymbol{v}(t)\|_2 &= \sqrt{v_0^2 + (gt)^2} \end{aligned} \tag{2}$$

where h_0 is the initial height.

2 Compute Kernels

To implement an OpenCL kernel for computing trajectories of a projectile, we will use parametric representations of a trajectory. For this tool there will be two OpenCL kernels: one will implement parametric set of equations in (1) and the other implements the set in (2).

The inputs to both kernels are the constants of initial time t_0, initial velocity v_0, and time difference Δt. The first kernel also takes as an input the initial angle of launch θ_0, whilst the second kernel takes the initial height h_0.

Date: 2009-04-26.

For both kernels, the total number of points along a timeline is computed according to,

$$n = \frac{|t_n - t_0|}{\Delta t_m}, \tag{3}$$

where $|t_n - t_0|$ is the duration of computation in seconds, with $t_0 < t_1 < \cdots < t_n$ the time interval, and time change $\Delta t_m = t_m - t_{m-1}$ for all $m = 1, \ldots, n$. Since we are considering equally spaced time intervals (3), all Δt_m will be equal, and as such will be denoted by Δt.

Furthermore, the outputs for both kernels are:

- position vector $\boldsymbol{r}(t) = x(t)\hat{\boldsymbol{e}}_x + y(t)\hat{\boldsymbol{e}}_y$
- velocity vector $\boldsymbol{v}(t) = v_x(t)\hat{\boldsymbol{e}}_x + v_y(t)\hat{\boldsymbol{e}}_y$
- speed $\|\boldsymbol{v}(t)\|_2 = \sqrt{v_x(t)^2 + v_y(t)^2}$

Lastly, the set of equations in (1) is implemented in a compute kernel as,

$$f_k(t_0, \Delta t, v_0, \theta; \boldsymbol{r}(t), \boldsymbol{v}(t), \|\boldsymbol{v}(t)\|_2) = \begin{cases} t_1 = t_0 + k\Delta t \\ v_1 = gt_1 \\ v_2 = v_0 \cos(\theta) \\ v_3 = v_0 \sin(\theta) \\ v_4 = 2v_3 \\ v_5 = v_4 - v_1 \\ v_6 = v_0^2 - v_1 v_5 \\ x(t) = v_2 t_1 \\ y(t) = \dfrac{1}{2} v_5 t_1 \\ v_x(t) = v_2 \\ v_y(t) = v_3 - v_1 \\ \|\boldsymbol{v}(t)\|_2 = \sqrt{v_6} \end{cases} \tag{4}$$

with the set of equations in (2) implemented in a compute kernel as,

$$f_k(t_0, \Delta t, v_0, h_0; \boldsymbol{r}(t), \boldsymbol{v}(t), \|\boldsymbol{v}(t)\|_2) = \begin{cases} t_1 = t_0 + k\Delta t \\ v_1 = gt_1 \\ v_2 = v_0^2 + v_1^2 \\ x(t) = v_0 t_1 \\ y(t) = h_0 - \dfrac{1}{2} v_1 t_1 \\ v_x(t) = v_0 \\ v_y(t) = -v_1 \\ \|\boldsymbol{v}(t)\|_2 = \sqrt{v_2} \end{cases} \tag{5}$$

for $k = 0, \ldots, n$. Note that, one may recover the equations in (1) and (2) by elementary algebraic substitutions.

3 Build Requirements

To successfully build the Xcode project, containing the sources and the kernel, use the following:

- Mac OS X v10.6 or later.
- Xcode v3.2 or later.

4 Runtime Requirements

On Mac OS X v10.6 or later, to use NVidia GPU as a compute device, use one of the following hardware configurations:

- MacBook Pro w/NVidia GeForce 8600M
- Mac Pro w/NVidia GeForce 8800GT

5 Packaging List

- ReadMe.txt
- Docs
  - Trajectories.pdf
- Sources
  - Kernel
    - TrajectoriesKernel.cl
  - Main
    - Trajectories.cpp
  - OpenCL
    - Headers
      - OpenCLBuffer.h
      - OpenCLFile.h
      - OpenCLKernel.h
      - OpenCLKit.h
      - OpenCLProgram.h
    - Sources
      - OpenCLBuffer.cpp
      - OpenCLFile.cpp
      - OpenCLKernel.cpp
      - OpenCLProgram.cpp
  - Trajectory
    - Trajectory.cpp
    - Trajectory.h
- Trajectories.xcodeproj