Simulation Techniques for Animation: GPU Accelerated Mass Spring System

Joe Withers[*]

Abstract

During this project I explored the feasibility of offloading computation onto the GPU, in soft body Mass-spring system simulations, focusing primarily on techniques that make use of OpenGL compute shaders. This report documents my implementation of such techniques, as well as my findings.

1 Implementation Diary

I used C++ and OpenGL to develop my implementation, utilising the NCCA Graphics Library NGL [Macey 2014] to interact with OpenGL, and Qt5 to create the user interface. I based my implementation around a demo of a Mass Spring System using RK4 (Runge-Kutta 4th Order) integration [Macey 2015], that uses NGL and Qt5 for it's user interface. I was able to use many aspects this implementation as 'boilerplate code', such as the camera movement and passing of basic geometry to OpenGL, which saved me a lot of time and allowed me to focus on developing the simulation itself. This implementation also provided an example of RK4 integration and calculations of spring forces according to Hooke's law, though these needed to be altered to make them suitable for use with GPU computation.

Before starting to implement the system on the GPU, I first had to establish what the data structures for the Mass-spring system would be, and to do this I looked at tutorial material covering game physics systems [Fiedler 2004] hosted by Gaffer On Games. This provided me with an understanding of the necessary data I would need to store for each 'Mass' the system; the necessary data required to represent the 'Springs' was inferred from the previously mentioned Mass-spring implementation [Macey 2015]. I determined that each 'Mass' would be a struct of two vectors:

- Position $[x, y, z]$ - This stores the position of the mass in world space coordinates.
- Velocity $[x, y, z]$- This stores the combined speed and direction at which the mass is travelling.

For my first attempt at GPU implementation I decided to store each the attributes for each mass and each spring in openGL textures, and to use Image Load/Store commands for bindless textures for accessing and manipulating data. I implemented the following attributes as 1D textures: Mass positions (Vec3, RGBA32F, size = number of masses) Spring state position attribute (Vec3, RGBA32F, size = number of springs) Spring state velocity attribute (Vec3, RGBA32F, size = number of springs) Spring resting length attribute (Float, R32F, size = number of springs) The spring start and end index attributes serve as a pointer to an index in the mass positions texture which they manipulate.

2 Research Report

References

FIEDLER, G., 2004. Physics in 3d. `https://gafferongames.com/post/physics_in_3d/`.

MACEY, J., 2014. Ngl the ncca graphics library. `https://github.com/NCCA/NGL`.

MACEY, J., 2015. Mass spring system using rk 4 integration. `https://github.com/NCCA/MassSpring`.

[*]e-mail:joewithers96gmail.com