GPL - GPU (Accelerated) Particle Library

Table of Contents:

Purpose:

GPL is a library aimed to push the limits of particle systems implemented in game engines. GPL makes use of the CUDA GPGPU library to process particles in parallel using the Graphics Processing Unit (GPU). GPL takes advantage of the mass-parallel processing nature of the GPU to allow for fast processing of hundreds of thousands of particles at a smooth framerate. GPL takes advantage of CUDA-OpenGL interoperability to minimise GPU copy overhead when processing and drawing particles.

Requirements:

GPL requires the following libraries, APIs, and hardware to function correctly:

-OpenGL
-CUDA compatible GPU
-CUDA v10.1 Driver.

Distribution & Compilation:

GPL is distributed as a static library packages with the necessary header files to use the library. To compile the GPL library source code the CUDA GPU Computing Toolkit v10.1 is required, but using the pre-compiled static library does not require the toolkit.

Implementation of mass-multithreading:

GPU threads are arranged in blocks with sizes of 256, 512, 1024 threads etc. To use one thread for each particle GPL takes the amount of particles in a particle system and divides it to get the amount of thread blocks to use, if there are less particles than the block size one block will be used.

Integration:

GPL is best integrated as a component of a game engine. GPL provides classes for creating and updating GPU accelerated particle systems which can be adapted for use in any game engine using OpenGL for rendering.

GPL requires OpenGL to allocate particle memory buffers and to draw particles. A particle system will generate it's own vertex buffer used for drawing but a shader program must be supplied in order to draw the particles. Particles can be drawn by binding the Vertex Array Object (VAO) of the particlesystem and binding a compatible shader program, and using one of the particlesystems existing draw functions.

GPL provides a class called GPL_VertexAttributes for defining custom vertex attributes to add to the particle system's VAO.

GPL supplies the following functions for drawing particles:

SetMesh(GLuint glVBOHandle, GLuint glIndexBufferHandle, GLuint indexCount, GPL_VertexAttributes& attributes) - Set the mesh used for drawing each particle in the particle system.
Draw() - Draw particles using a supplied shader program and mesh (Mesh should be set during initialization stage).
DrawSimple() - Draw particles a point primitives using a supplied shader program.

GPL particle systems require behaviour programs in-order to determine how particles will behave. GPL provides some example behaviour programs to get a user started but it is unlikely these programs will provide the functionality the user desires.

GPL lets the user create their own behaviour programs contained within CUDA source files with the .cu extension. Users can use the **GPL Runtime Library** to assist in creating behaviour programs for use with GPL.

Additionally users can create custom CUDA header files to include within their behaviour programs using the GPL_BehaviourProgram class.

GPL_BehaviourProgram class:

The GPL_BehaviourProgram class allows users to load GPL behaviour program source code and compile it during runtime using the NVRTC library by Nvidia Corporation. The class also allows users to include source code from their own custom CUDA headers into a GPL behaviour program.

GPL Runtime Library:

The GPL Runtime Library is a tiny header library that provides users with a useful GPU compatible vector class for applying movement and other vector related effects to particles. GPL also provides some basic steering behaviour functions to move particles smoothly.