GEANT-4 GPU Port:

Design Document: System Architecture

Team 8
Stuart Douglas – dougls2
Matthew Pagnan – pagnanmm
Rob Gorrie – gorrierw
Victor Reginato – reginavp

System Architecture: Version 0
January 8, 2016

Table of Contents

1 Introduction

1.1 Revision History

All major edits to this document will be recorded in the table below.

Table 1: Revision History

Description of Changes	Author	Date
Set up sections and filled out Introduction section	Matthew	2015-12-15

1.2 Terms Used

Table 2: Glossary

Term	Description
Geant4	open-source software toolkit for simulating particle interactions
G4-STORK	fork of Geant4 developed by McMaster's Engineering Physics department to simulate McMaster's nuclear reactor
GPU	graphics processing unit, well-suited to parallel computing tasks
CPU	computer processing unit, well-suited to serial tasks
CUDA	parallel computing architecture for GPU programming, developed by NVIDIA
NVIDIA	computer hardware and software company specializing in GPU's

1.3 List of Tables

1.4 List of Figures

2 Overview

2.1 Purpose of Project

The purpose of GEANT4-GPU is to reduce the computation times of particle simulations in Geant4 by parallelizing and running certain procedures on the GPU.

2.2 Description

The project aims to improve the computation times of Geant4 particle simulations by running certain parallel operations on a GPU. GEANT4-GPU will be a fork of the existing Geant4 system with the additional option for users with compatible hardware to run operations on the GPU for improved performane. This functionality will be available on Mac, Linux and Windows operating systems running on computers with NVIDIA graphics cards that support CUDA.

The design strategy for the project will be based on taking a specific, computationally heavy class from Geant4 and creating a class that fulfills the same interface but that runs on the GPU. This will be repeated for many classes until the project's deadline has been reached. The user will have the option of using the existing classes (running on the CPU) or the new ones (running on the GPU).

2.3 Document Structure & Template

The design documentation for the project is based off of WHAT TEMPLATES?????? and is broken into two main documents.

This system architecture document details the system architecture, including an overview of the modules that make up the system, analysis of aspects that are likely and unlikely to change, reasoning behind the high-level decisions, and a table showing how each requirement is addressed in the proposed design.

A separate detailed design document covers the specifics of several key modules in the project. This includes the interface specification and implementation decisions.

3 Important Design Decisions & Reasoning

3.1 GPU Computing for Geant4

3.2 CUDA

3.3 GPU Integration Approach

When considering how to integrate the GPU computations with the non-ported procedures that run on the CPU there were several options. The first option would be to port the entire Geant4 toolkit to run 100% on the GPU. This would see the biggest speedup, but is far beyond the scope of this project and the resources available. A more realistic alternative is to take several key classes and implement their functions on the GPU. The existing functions on the CPU would make the necessary calls to the GPU to execute the corresponding function and receive the result when the GPU finishes and return it to the caller. The advantage of this approach is that the work is easier to break down and distribute, and will focus efforts on the modules and functions within those modules that require speed improvements the most.

When a ported module is first initialized, it will also initialize a copy on the GPU, and every time data is updated those updates will be applied to the GPU class as well. Then, when a function is called, the GPU will already have all the data it requires and will be able to quickly return the result. If a given function is not ported or if the user has not enabled GPU computations the function will run the existing code on the CPU. The overhead of this approach is the time during initialization to initialize the GPU class as well, and when updates occur. We believe this overhead will be outweighed by the advantages of the parallel GPU computing. The initialization is only called once, and updates to the data generally target small subsets of the vector.

This same approach of taking a class and initializing a copy on the GPU with a subset of the functions ported to the GPU will be applied to other classes in the future.

3.4 G4NeutronHPVector

4 Likely and Unlikely Changes

4.1 Likely Changes

1.

4.2 Unlikely Changes

1.

4.3 Traceability of Likely Changes to Design Components

5 Module Hierarchy

5.1 Decomposition of Components

The project is based off of Geant4, an existing software system, and modifying specific modules of that system. As such, components do not need to be decomposed, instead the decomposition of the existing system will guide the modules in the project development.

Each module in Geant4 is represented by a C++ class, and the project will port a subset of said classes to the GPU. The decision of which classes to port to the GPU is derived from performance analysis of the program, with the most computationally time-consuming classes being the targets for GPU porting, such as G4NeutronHPVector as documented in section 3.4.

5.2 Uses Hierarchy

Since each modified module of the project will have a direct 1-to-1 relationship with an existing module in Geant4, the uses hierarchy with the new, GPU-enabled classes will be identical to the existing uses hierarchy of Geant4 with their corresponding existing classes.

Geant4 is an extremely large system with many modules, so its entire uses hierarchy will not be documented here. Instead, the following hierarchy just maps the dependencies of G4NeutronHPVector, which is currently the focus of the project. Note that modules used by system libraries are not included.

6 Traceability of Requirements and Design Components

The following section outlines each requirement and what section of the design document addresses it.

Figure 1: Uses Hierarchy for G4NeutronHPVector

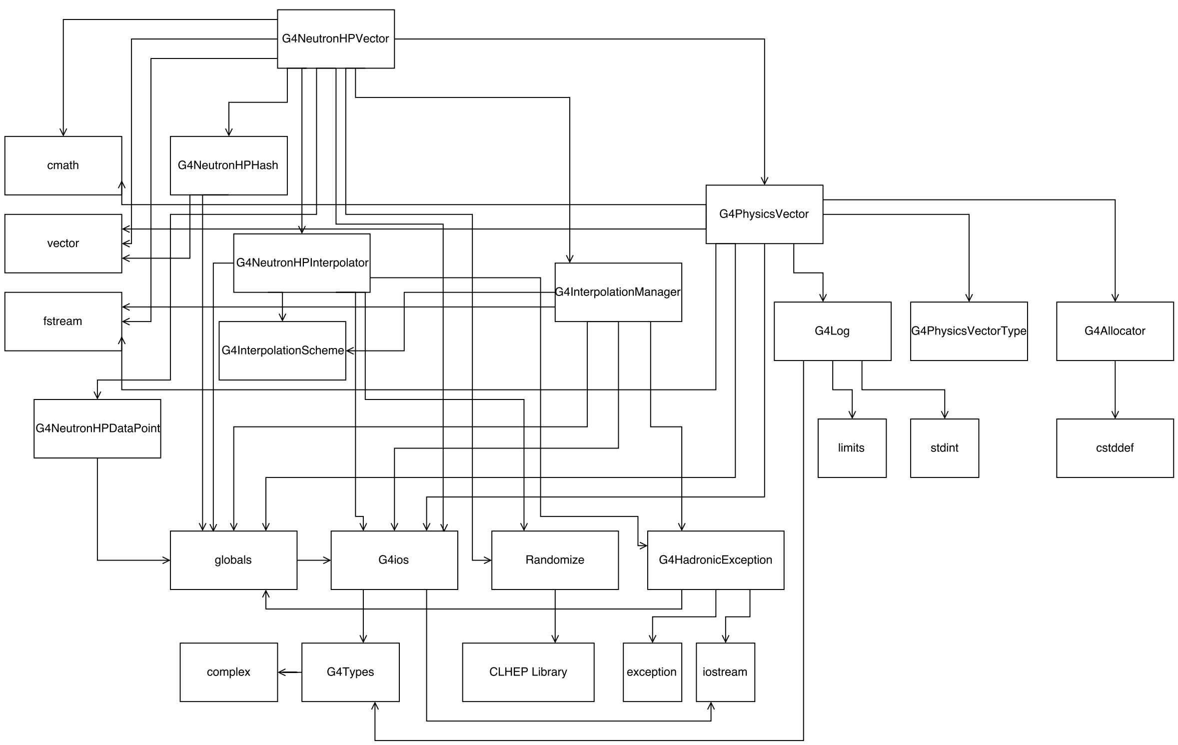

Table 3: Requirements and Design Relationship

Req.	Brief Description	Design Component
1	computations run on GPU	entire document
2	existing projects not affected	
3	by default simulation will run on CPU	
4	should detect if computer has compatible GPU	
5	enabling GPU computation on incompatible hardware not allowed	
6	enabling GPU functionality on existing projects easy	
7	runtime of simulation decreased with same output	
8	accuracy of results same as when run on CPU	
9	at least as stable as existing system	
10	errors will throw exceptions	
11	will work with last four versions of Geant4	
12	available on public repo with installation instructions	
13	new versions of product will be available on repo, won't break previous features	
14	all users have access to entire product	