GEANT-4 GPU Port:

Design Document: Detailed Design

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

Detailed Design: Version 0
January 10, 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
Added sections for Errors and Key Algorithms	Stuart	2016-01-08

1.2 Document Structure & Template

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

The 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.

This detailed design document covers the specifics of several key modules in the project. For each module, an MIS is given fully detailing the interface of the module. Then, the methods for handling errors within the module are discussed, and finally the main algorithms and data structures used by the module are presented.

1.3 List of Tables

1.4 Note About G4 variables

Geant4 uses its own basic types for standard C++ types (G4int, G4bool, G4double, etc). These types are currently just `typedefs` to the respective type as defined in the system libraries.

2 G4NeutronHPDataPoint

2.1 Description

2.2 MIS (Module Interface Specification)

2.2.1 Access Program Syntax

Table 2: G4NeutronHPDataPoint – access program syntax

Routine Name	Input	Output	Exceptions
G4NeutronHPDataPoint			
G4NeutronHPDataPoint	G4double, G4double		
=	G4NeutronHPDataPoint		
GetEnergy		G4double	
GetXsection		G4double	
SetEnergy	G4double		
SetXsection	G4double		
GetX		G4double	
GetY		G4double	
SetX	G4double		
SetY	G4double		
SetData	G4double, G4double		

2.2.2 Access Program Semantics

Note that hyphens in routine names and inputs are just for linebreaks due to the table size. The actual routine names and inputs do not have hyphens.

Table 3: NeutronHPDataPoint – access program semantics

Routine Name	Input	Semantics
G4NeutronHPDataPoint		instantiates the class, setting `energy` and `xSec` to 0
G4NeutronHPDataPoint	G4double, G4double	instantiates the class with the inputted `energy` and `xSec`
=	G4NeutronHP-DataPoint	sets the `energy` and `xSec` of the instance to those of the input
GetEnergy		returns `energy` of the instance
GetXsection		returns the `xSec` of the instance
SetEnergy	G4double	sets `energy` of instance to the argument
SetXsection	G4double	sets `xSec` of instance to the argument
GetX		returns the `energy` of the instance
GetY		returns the `xSec` of the instance
SetX	G4double	sets `energy` of instance to the argument
SetY	G4double	sets `xSec` of instance to the argument
SetData	G4double, G4double	sets instance's `energy` and `xSec` to the passed arguments

2.2.3 State Variables

The following variables maintain state for the class, and are all private to the module.

Table 4: G4NeutronHPDataPoint – state variables

Variable	Type	Description
`energy`	G4double	
`xSec`	G4double	

2.2.4 Environment Variables

There are no environment variables for this module.

2.2.5 Assumptions

It can be assumed that the class will be initialized. As such, all getter methods will return a non-null value.

2.3 Error Handling

2.4 Key Algorithms

3 G4NeutronHPVector

3.1 Description

3.2 MIS (Module Interface Specification)

Note that hyphens in routine names, inputs, outputs, and exceptions are just for line-breaks due to the table size. The actual routine names, inputs, outputs, and exceptions do not have hyphens.

3.2.1 Access Program Syntax

Table 5: G4NeutronHPVector – access program syntax

Routine Name	Input	Output	Exceptions
G4NeutronHPVector			
G4NeutronHPVector	G4int		
SetVerbose	G4int		
Times	G4double		
SetPoint	G4int, G4NeutronHPDataPoint		
SetData	G4int, G4double,G4double		
SetX	G4int, G4double		
SetEnergy	G4int, G4double		
SetY	G4int, G4double		
SetXsec	G4int, G4double		
GetEnergy	G4int	G4double	
GetXsec	G4int	G4double	
GetXsec	G4double	G4double	
GetXsec	G4double,G4int	G4double	
GetX	G4int	G4double	
GetY	G4double	G4double	
GetY	G4int	G4double	
GetVectorLength		G4int	

GetPoint	G4int	const G4NeutronHPDataPoint&
Hash		
ReHash		
InitInterpolation	istream	
Init	istream,G4int, G4double, G4double	
Init	istream, G4double,G4double	
ThinOut	G4double	
SetLabel	G4double	
GetLabel		G4double
CleanUp		
Sample		G4double
Debug		G4double *
Merge	G4NeutronHPVector *, G4NeutronHPVector *	
Merge	G4InterpolationScheme, G4double, G4NeutronHPVector *, G4NeutronHPVector *	
SampleLin		G4double
IntegrateAndNormalise		
Integrate		
GetIntegral		G4double
SetInterpolationManager	const G4InterpolationManager &	
SetInterpolationManager	G4InterpolationManager &	
G4InterpolationManager		const G4InterpolationManager &
SetScheme	G4int,const G4InterpolationScheme &	
GetScheme	G4int	G4InterpolationScheme
GetMeanX		G4double
GetBlocked		vector<G4double>
GetBuffered		vector<G4double>
Get15percentBorder		G4double

Get50percentBorder		G4double	
Check	G4int		G4Hadronic-Exception
IsBlocked	G4double	G4bool	

3.2.2 Access Program Semantics

Note that hyphens in routine names and inputs are just for linebreaks due to the table size. The actual routine names and inputs do not have hyphens.

Table 6: G4NeutronHPVector – access program semantics

Routine Name	**Input**	**Description**
G4NeutronHPVector		
G4NeutronHPVector	G4int	
SetVerbose	G4int	
Times	G4double	
SetPoint	G4int, G4NeutronHP-DataPoint	
SetData	G4int, G4double,G4double	
SetX	G4int, G4double	
SetEnergy	G4int, G4double	
SetY	G4int, G4double	
SetXsec	G4int, G4double	
GetEnergy	G4int	
GetXsec	G4int	
GetXsec	G4double	
GetXsec	G4double,G4int	
GetX	G4int	
GetY	G4double	
GetY	G4int	
GetVectorLength		
GetPoint	G4int	
Hash		
ReHash		
InitInterpolation	istream	
Init	istream,G4int, G4double, G4double	

Init	istream, G4double,G4double
ThinOut	G4double
SetLabel	G4double
GetLabel	
CleanUp	
Sample	
Debug	
Merge	G4NeutronHPVector*, G4NeutronHPVector*
Merge	G4InterpolationScheme, G4double, G4NeutronHPVector*, G4NeutronHPVector*
SampleLin	
IntegrateAndNormalise	
Integrate	
GetIntegral	
SetInterpolation- Manager	const G4Interpolation- Manager&
SetInterpolation- Manager	G4Interpolation- Manager&
G4Interpolation- Manager	
SetScheme	G4int, G4Interpolation- Scheme&
GetScheme	G4int
GetMeanX	
GetBlocked	
GetBuffered	
Get15percentBorder	
Get50percentBorder	
Check	G4int
IsBlocked	G4double

3.2.3 State Variables

The following variables maintain state for the class, and are all private to the class.

Note that hyphens in variable names and types are just for linebreaks due to the table size. The actual variable names and types do not have hyphens.

Table 7: G4NeutronHPVector – state variables

Variable	Type	Description
theLine	G4NeutronHP-Interpolator	
totalIntegral	G4double	
theData	G4NeutronHP-DataPoint*	
theManager	G4Interpolation-Manager	
theIntegral	G4double*	
nEntries	G4int	
nPoints	G4int	
label	G4double	
theInt	G4Neutron-Interpolator	
Verbose	G4int	
isFreed	G4int	
theHash	G4NeutronHP-Hash	
maxValue	G4double	
theBlocked	vector <G4double>	
theBuffered	vector <G4double>	
the15percent-BorderCash	G4double	
the50percent-BorderCash	G4double	

3.2.4 Environment Variables

There are no environment variables for this Module.

3.2.5 Assumptions

It can be assumed that the module will be initialized before other functions are called.

3.3 Error Handling

3.4 Key Algorithms

4 CMake Files

4.1 Description

4.2 MIS (Module Interface Specification)

4.2.1 Access Program Syntax

NA

4.2.2 Access Program Semantics

NA

4.2.3 State Variables

Table 8: CMake Files – state variables

Variable	Type	Description
`useCuda`	Boolean	if set to true, the makefiles generated by CMake will include directives to compile and link the CUDA code and will execute ported procedures on the GPU. Default is false.

4.2.4 Environment Variables

- NeutronHPVectorGPU.cu : CUDA file with GPU code

4.2.5 Assumptions

No assumptions need to be made for CMake.

4.3 Error Handling

4.4 Key Algorithms