GigaVoxels/GigaSpace
Producers / Renderers - UML Design

Pascal Guehl , Fabrice Neyret

TECHNICAL REPORT

N° xxxx

13/01/2014

Project-Team Maverick

ISSN 0249-6399

GigaVoxels/GigaSpace

Pascal Guehl[1] , Fabrice Neyret [2]
Project-Teams Maverick

Technical Report N° xxxx — 13/01/2002014 —14 pages.

Abstract: GigaSpace UML design for Data Production management and Rendering system.

Key-words: insérez ici les mots-clés en anglais

[1] Pascal Guehl affiliation – pascal.guehl@inria.fr

[2] Fabrice Neyret affiliation – fabrice.neyret@inria.fr

GigaVoxels/GigaSpace

Résumé : UML Design of Data Production Management and Rendering Management in GigaSpace.

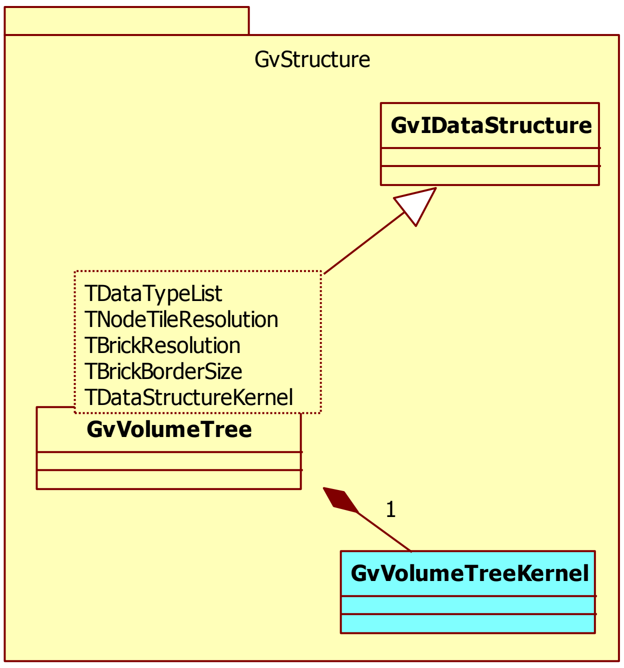
Mots clés : insérez ici les mots-clés en français

1. Introduction

This is the UML Design of Data Production and Rendering Management in GigaSpace.
Note : Each important class provides an associated device class that can be used in CUDA kernels.

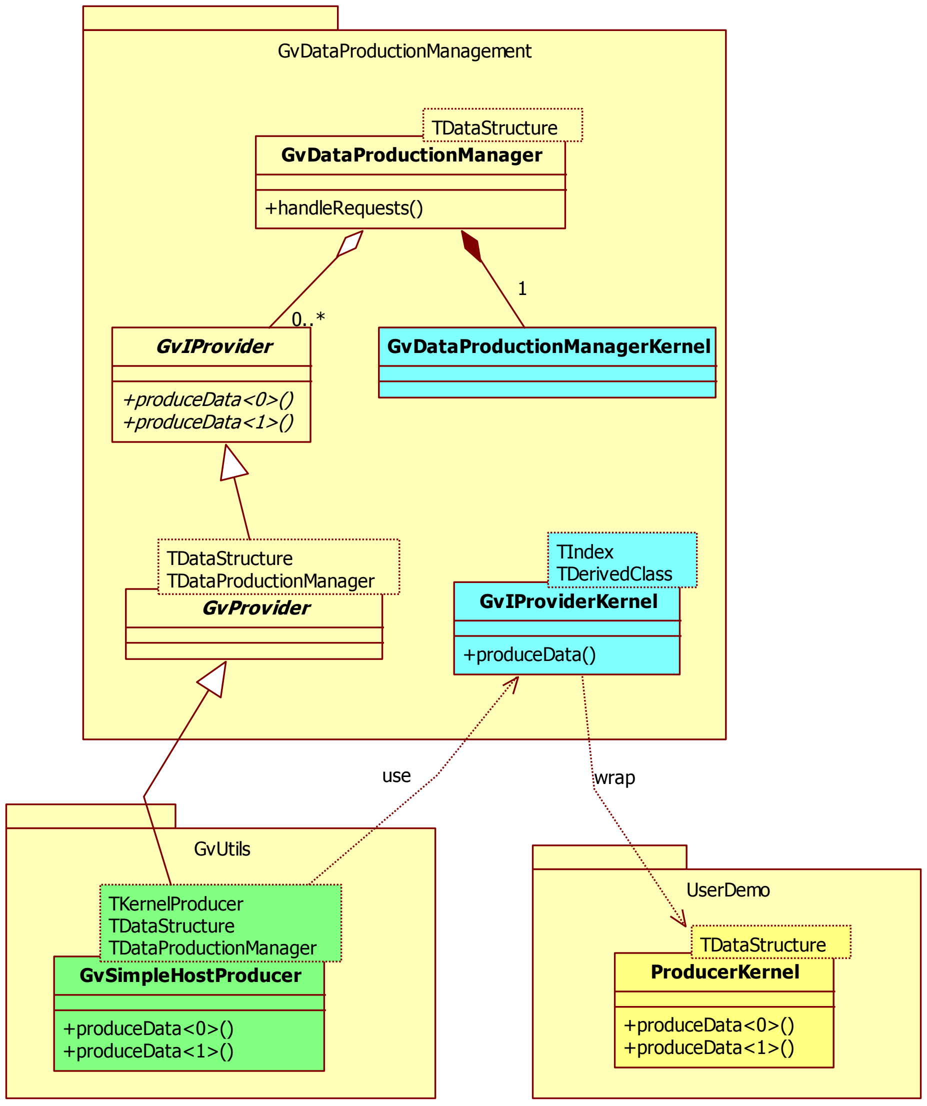
2. Data Structure

The data structure "GvVolumeTree" provides an interface to user data.
"TDataTypeList" template parameter defines the content of data as a list of types (ex: float4, uchar, etc...)
"TNodeTileResolution" defines the topology of the associated spatial partionning structure as a generalized N3-tree of nodes (ex: octree, etc...). Each node contains one brick of data (ex: voxels, ...)
"TBrickResolution" defines how many voxel are stored per node.
"TBrickBorderSize" specifies the number of voxels added as border in nodes. This is used during rendering stage to handle multi-resolution. By default, 1.
"TDataStructureKernel" specifies the associated class that will be used in device code (i.e. CUDA kernels).



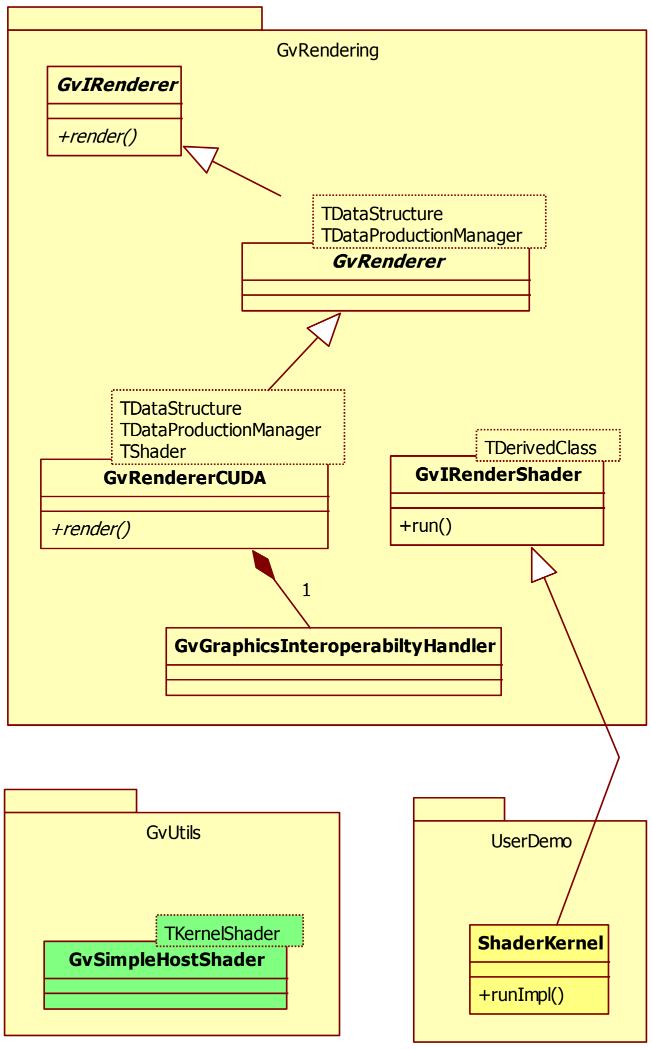
3. Producers

Producers provide an interface to produce data in data structures.
Production refers to tree refinement (node subdivisions) and filling node data (procedural generation, data streaming from hard disk, etc…)
"GvDataProductionManager" is the main scheduler of GigaSpace. It is in charge of handling the requests emitting during the rendering stage (requests stands for node and brick production). It contains a list of producers (actually only 1).
produceData() methods have an ID referring to nodes production (<0>) and bricks of data production (<1>).
"GvSimpleHostProducer" is a useful class that provides common behaviors of producers and fits inside the GigaSpace pipeline flow sequence. Therefore, users should inherits from that class and modify virtual methods if needed. As usual, it is template by its associated device-side class that will be used in device code (i.e CUDA kernels).



4. Renderers

Renderers provide an interface to produce visualize user data.
Besides rendering data, the Rendering stage emits requests of nodes production to refine the tree structure and fill nodes data when empty.
"GvRendererCUDA" is a useful class that provides common behaviors of renderers and fits inside the GigaSpace pipeline flow sequence. Therefore, users should inherits from that class and modify virtual methods if needed. As usual, it is template by its associated device-side class that will be used in device code (i.e CUDA kernels). It also takes "Shader" as template parameter. This class is used during the ray-marching process when the renderer needs to visualize a node. Typically computing ADS lighting model (ambient, diffuse, specular) and fetching data from volume.



Conclusion

The Renders package should be redesign to handle user customization, multi-pass framework, more OpenGL interoperability, etc...

Bibliography

[1] NVIDIA – CUDA C Programming Guide – *doc*

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr
ISSN 0249-6399