Crystal Concept Extension to Geant4

E. Bagli

INFN Section of Ferrara

Why

- Some developments have required to extend the concept of materials. Indeed, the G4Material is meant to describe amorphous/gaseous material and not to take into consideration the microscopic structure of the material itself

- Some examples of these extensions are:
  - G4DNA
  - G4CMP (Phonon & charge carrier propagation)
  - Channeling
  - NCrystal (Neutron diffraction)

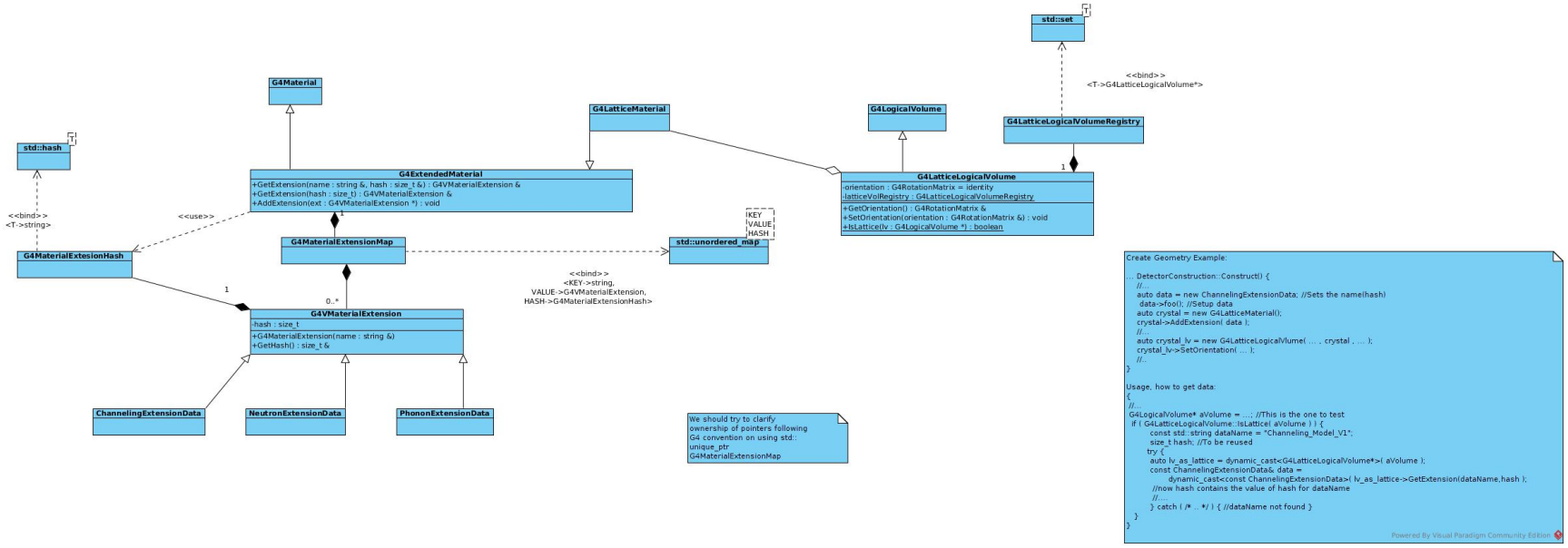
How

- Constraints:
  1. **Preserve G4Material class as it is**.
  2. The new material has to be compatible with the existing code
  3. A material can have more than one extension at once
  4. The extension has to be general in order to be used for other purposes.

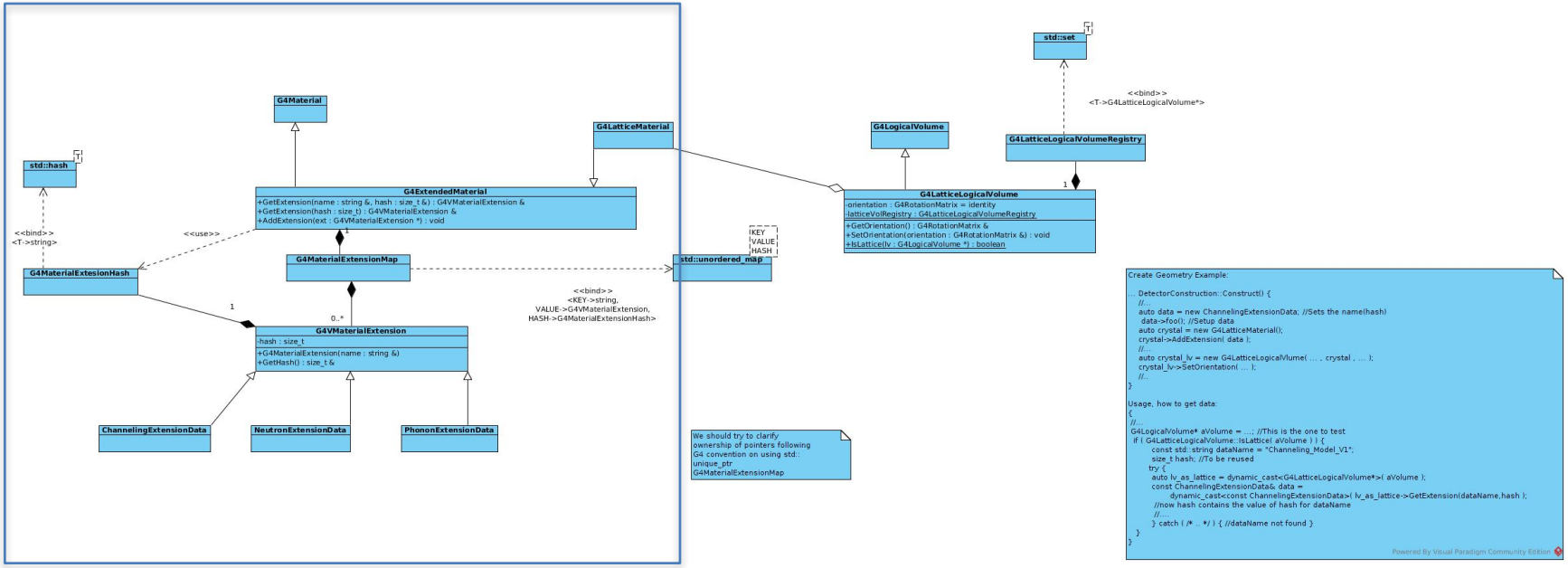
- Solution:
  - **G4ExtentedMaterial will be a derived class of G4Material**
  - G4ExtentedMaterial collects a map of G4MaterialExtensions*
  - G4CrystalMaterial will be a derived class of G4ExtentedMaterial

- Advantage:
  - No performance change for who does not use the extended material
  - More than one extension can be added to one extended material
  - Data for processes/models are stored in independent containers (G4MaterialExtensions)

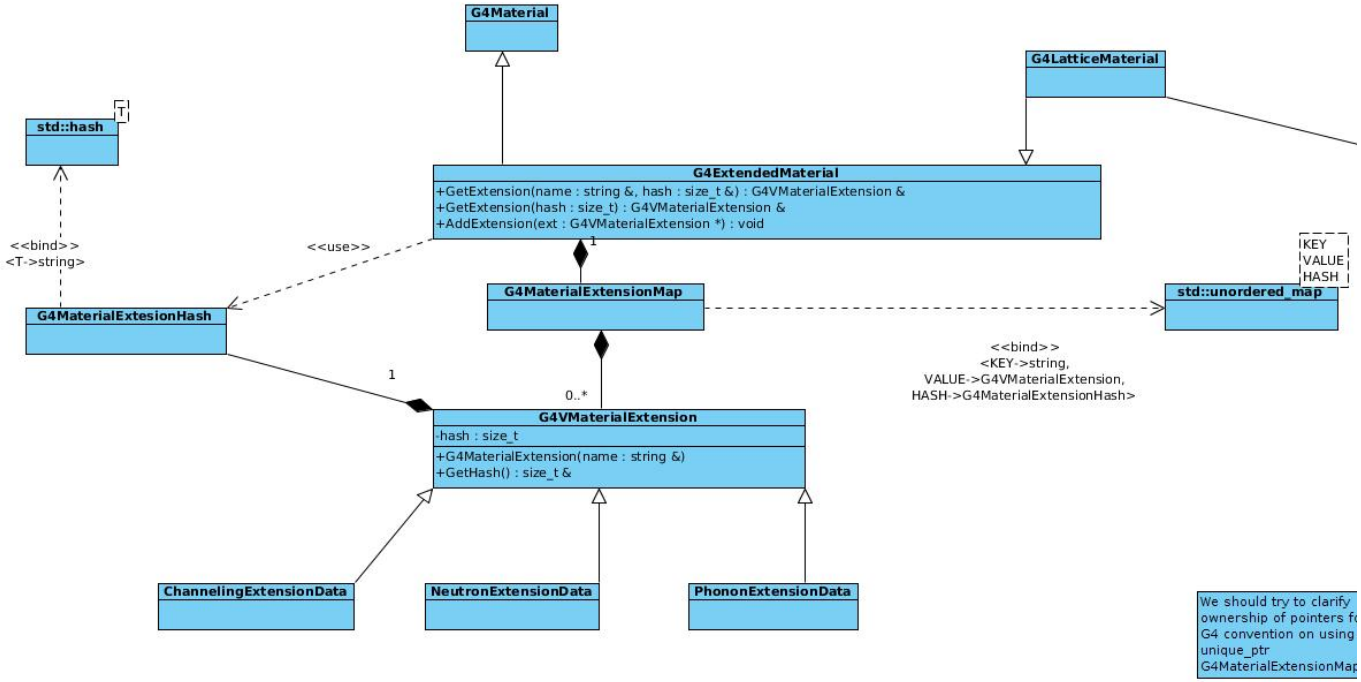
How


std::set
<<bind>>
<T->G4LatticeLogicalVolume*>
G4Material
G4LatticeMaterial
G4LogicalVolume
G4LatticeLogicalVolumeRegistry
std::hash
G4ExtendedMaterial
+GetExtension(name : string &, hash : size_t &) : G4VMaterialExtension &
+GetExtension(hash : size_t) : G4VMaterialExtension &
+AddExtension(ext : G4VMaterialExtension *) : void
G4LatticeLogicalVolume
-orientation : G4RotationMatrix = identity
-latticeVolRegistry : G4LatticeLogicalVolumeRegistry
+GetOrientation() : G4RotationMatrix &
+SetOrientation(orientation : G4RotationMatrix &) : void
+IsLattice(lv : G4LogicalVolume *) : boolean
<<bind>>
<T->string>
<<use>>
G4MaterialExtensionMap
std::unordered_map
KEY
VALUE
HASH
G4MaterialExtesionHash
<<bind>>
<KEY->string,
VALUE->G4VMaterialExtension,
HASH->G4MaterialExtensionHash>
G4VMaterialExtension
-hash : size_t
+G4MaterialExtension(name : string &)
+GetHash() : size_t &
ChannelingExtensionData
NeutronExtensionData
PhononExtensionData
We should try to clarify
ownership of pointers following
G4 convention on using std::
unique_ptr
G4MaterialExtensionMap
Create Geometry Example:
DetectorConstruction::Construct() {
//...
auto data = new ChannelingExtensionData; //Sets the name(hash)
data->foo(); //Setup data
auto crystal = new G4LatticeMaterial();
crystal->AddExtension(data);
//...
auto crystal_lv = new G4LatticeLogicalVlume(... , crystal , ...);
crystal_lv->SetOrientation(...);
//...
}
Usage, how to get data:
{
//...
G4LogicalVolume* aVolume = ...; //This is the one to test
if (G4LatticeLogicalVolume::IsLattice(aVolume)) {
const std::string dataName = "Channeling_Model_V1";
size_t hash; //To be reused
try {
auto lv_as_lattice = dynamic_cast<G4LatticeLogicalVolume*>(aVolume);
const ChannelingExtensionData& data =
dynamic_cast<const ChannelingExtensionData>(lv_as_lattice->GetExtension(dataName,hash);
//now hash contains the value of hash for dataName
//....
} catch (/* ... */) { //dataName not found }
}
}
Powered By Visual Paradigm Community Edition

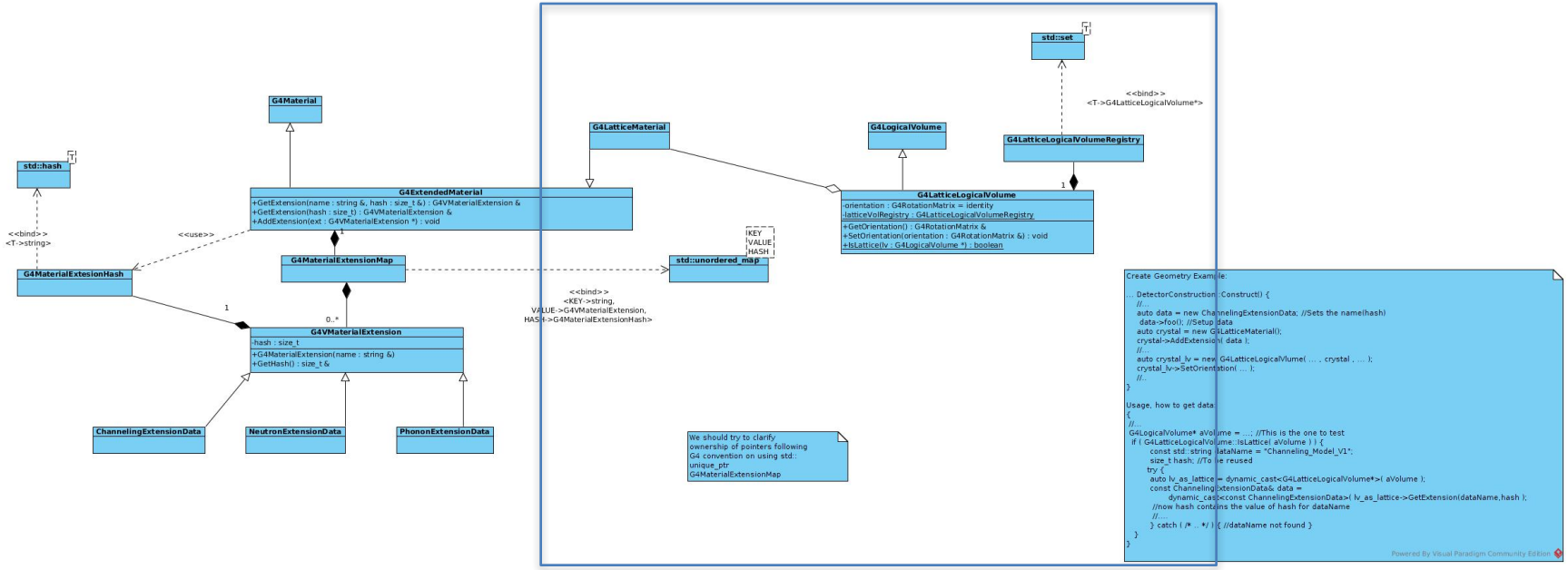
How


G4Material
G4LatticeMaterial
std::hash
G4ExtendedMaterial
+GetExtension(name : string &, hash : size_t &) : G4VMaterialExtension &
+GetExtension(hash : size_t) : G4VMaterialExtension &
+AddExtension(ext : G4VMaterialExtension *) : void
<<bind>>
<T->string>
<<use>>
G4MaterialExtesionHash
G4MaterialExtensionMap
std::unordered_map
KEY
VALUE
HASH
<<bind>>
<KEY->string,
VALUE->G4VMaterialExtension,
HASH->G4MaterialExtensionHash>
G4VMaterialExtension
-hash : size_t
+G4MaterialExtension(name : string &)
+GetHash() : size_t &
ChannelingExtensionData
NeutronExtensionData
PhononExtensionData
We should try to clarify
ownership of pointers following
G4 convention on using std::
unique_ptr
G4MaterialExtensionMap
std::set
<<bind>>
<T->G4LatticeLogicalVolume*>
G4LogicalVolume
G4LatticeLogicalVolumeRegistry
G4LatticeLogicalVolume
-orientation : G4RotationMatrix = identity
-latticeVolRegistry : G4LatticeLogicalVolumeRegistry
+GetOrientation() : G4RotationMatrix &
+SetOrientation(orientation : G4RotationMatrix &) : void
+IsLattice(lv : G4LogicalVolume *) : boolean
Create Geometry Example:
... DetectorConstruction::Construct() {
//...
auto data = new ChannelingExtensionData; //Sets the name(hash)
data->foo(); //Setup data
auto crystal = new G4LatticeMaterial();
crystal->AddExtension(data);
//...
auto crystal_lv = new G4LatticeLogicalVlume(... , crystal , ...);
crystal_lv->SetOrientation(...);
//...
}
Usage, how to get data:
{
//...
G4LogicalVolume* aVolume = ...; //This is the one to test
if (G4LatticeLogicalVolume::IsLattice(aVolume)) {
const std::string dataName = "Channeling_Model_V1";
size_t hash; //To be reused
try {
auto lv_as_lattice = dynamic_cast<G4LatticeLogicalVolume*>(aVolume);
const ChannelingExtensionData& data =
dynamic_cast<const ChannelingExtensionData>(lv_as_lattice->GetExtension(dataName,hash);
//now hash contains the value of hash for dataName
//....
} catch (/* ... */) { //dataName not found }
}
}
Powered By Visual Paradigm Community Edition

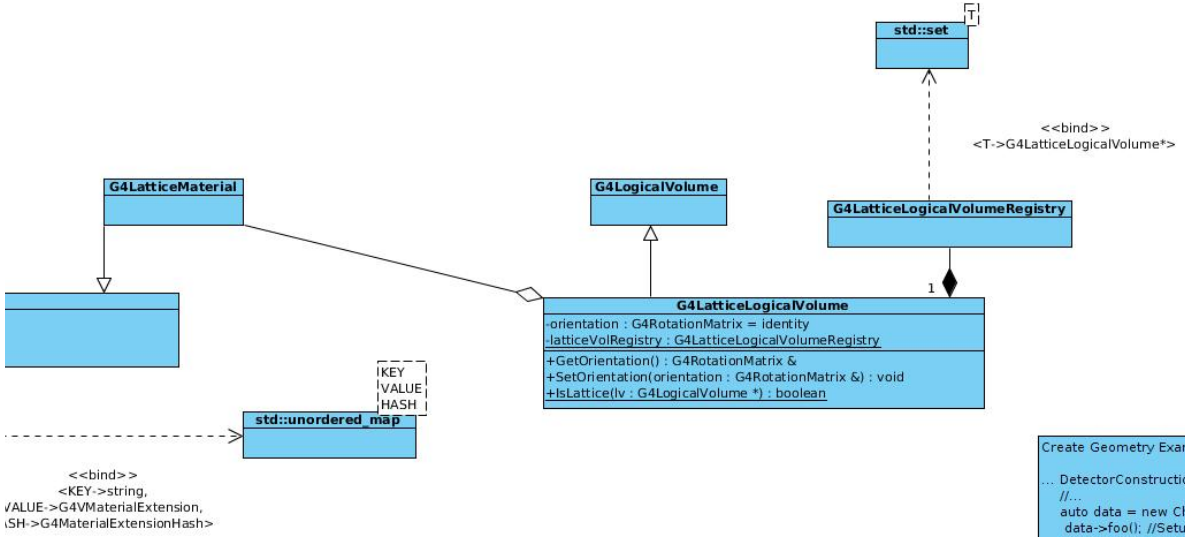
How


G4Material
G4LatticeMaterial
std::hash
T
G4ExtendedMaterial
+GetExtension(name : string &, hash : size_t &) : G4VMaterialExtension &
+GetExtension(hash : size_t) : G4VMaterialExtension &
+AddExtension(ext : G4VMaterialExtension *) : void
<<bind>>
<T->string>
<<use>>
1
KEY
VALUE
HASH
G4MaterialExtesionHash
G4MaterialExtensionMap
std::unordered_map
<<bind>>
<KEY->string,
VALUE->G4VMaterialExtension,
HASH->G4MaterialExtensionHash>
1
0..*
G4VMaterialExtension
-hash : size_t
+G4MaterialExtension(name : string &)
+GetHash() : size_t &
ChannelingExtensionData
NeutronExtensionData
PhononExtensionData
We should try to clarify
ownership of pointers f
G4 convention on using
unique_ptr
G4MaterialExtensionMap

How


G4Material
std::hash
G4ExtendedMaterial
+GetExtension(name : string &, hash : size_t &) : G4VMaterialExtension &
+GetExtension(hash : size_t) : G4VMaterialExtension &
+AddExtension(ext : G4VMaterialExtension *) : void
<<bind>>
<T->string>
<<use>>
G4MaterialExtesionHash
G4MaterialExtensionMap
G4VMaterialExtension
-hash : size_t
+G4MaterialExtension(name : string &)
+GetHash() : size_t &
ChannelingExtensionData
NeutronExtensionData
PhononExtensionData
G4LatticeMaterial
std::unordered_map
KEY
VALUE
HASH
<<bind>>
<KEY->string,
VALUE->G4VMaterialExtension,
HASH->G4MaterialExtensionHash>
G4LogicalVolume
G4LatticeLogicalVolume
-orientation : G4RotationMatrix = identity
-latticeVolRegistry : G4LatticeLogicalVolumeRegistry
+GetOrientation() : G4RotationMatrix &
+SetOrientation(orientation : G4RotationMatrix &) : void
+IsLattice(lv : G4LogicalVolume *) : boolean
std::set
<<bind>>
<T->G4LatticeLogicalVolume*>
G4LatticeLogicalVolumeRegistry
We should try to clarify
ownership of pointers following
G4 convention on using std::
unique_ptr
G4MaterialExtensionMap
Create Geometry Example:
DetectorConstruction::Construct() {
//...
auto data = new ChannelingExtensionData; //Sets the name(hash)
data->foo(); //Setup data
auto crystal = new G4LatticeMaterial();
crystal->AddExtension(data);
//...
auto crystal_lv = new G4LatticeLogicalVlume(... , crystal , ...);
crystal_lv->SetOrientation(...);
//...
}
Usage, how to get data
{
//...
G4LogicalVolume* aVolume = ...; //This is the one to test
if (G4LatticeLogicalVolume::IsLattice(aVolume)) {
const std::string dataName = "Channeling_Model_V1";
size_t hash; //To be reused
try {
auto lv_as_lattice = dynamic_cast<G4LatticeLogicalVolume*>(aVolume);
const ChannelingExtensionData& data =
dynamic_cast<const ChannelingExtensionData>(lv_as_lattice->GetExtension(dataName,hash);
//now hash contains the value of hash for dataName
//....
} catch (/* ... */) { //dataName not found }
}
}
Powered By Visual Paradigm Community Edition

How



We should try to clarify ownership of pointers following G4 convention on using std::unique_ptr G4MaterialExtensionMap

```
Create Geometry Example:

... DetectorConstruction::Construct() {
    //...
    auto data = new ChannelingExtensionData; //Se
     data->foo(); //Setup data
    auto crystal = new G4LatticeMaterial();
    crystal->AddExtension( data );
    //...
    auto crystal_lv = new G4LatticeLogicalVlume( ...
    crystal_lv->SetOrientation( ... );
    //..
}

Usage, how to get data:
{
 //...
 G4LogicalVolume* aVolume = ...; //This is the one
  if ( G4LatticeLogicalVolume::IsLattice( aVolume ) )
        const std::string dataName = "Channeling_M
        size_t hash; //To be reused
       try {
        auto lv_as_lattice = dynamic_cast<G4Lattice
        const ChannelingExtensionData& data =
              dynamic_cast<const ChannelingExtens
         //now hash contains the value of hash for c
         //....
        } catch ( /* .. */ ) { //dataName not found }
   }
}
```

Code (preliminary)

Detector Construction

```
//** Amorphous Definition **//
G4Material* Si = G4NistManager::Instance()->FindOrBuildMaterial("G4_Si");

//** Crystal Definition **//
G4MaterialCrystal* SiCrystal = new G4MaterialCrystal(Si,"SiCrystal");

//** Set Crystal Properties **//
SiCrystal->SetUnitCell(new G4CrystalUnitCell(...));
SiCrystal->AddExtension(new MaterialExtensionData(...));
```

Code (preliminary)

Physics Process

```
G4LogicalVolume* aLV = aTrack.GetVolume()->GetLogicalVolume();

if(G4LogicalCrystalVolume::IsLattice(aLV)){

     G4LogicalCrystalVolume* aLCV = (G4LogicalCrystalVolume*)aLV;
     G4cout << aLCV->GetName() << " is a Crystal!" << G4endl;

     MaterialExtensionData* data =
          (MaterialExtensionData*) aLCV->GetExtension(fProcessExtensionName);
     if(data != NULL){
          G4cout << "Data with name " << fProcessExtensionName << G4endl;
     }
}

else{
     G4cout << aLV->GetName() <<  " is an Amorphous Material..." << G4endl;
}
```

Conclusions

- New developments require the extension of the G4Material class
- G4ExtendedMaterial allows to preserve from any modification the G4Material class
- More than one process can add extension to a material
- The extended material is general and can be used for various purposes other than for crystalline material