Synthèse des modifications du code Geant4

Enregistrement des photoabsorptions de primaires
dans le graphite et l'inox

Simulation MiniX – Cône concentrateur en graphite

February 9, 2026

---

1 Contexte et objectif

Le code existant comptabilisait les pertes de primaires par processus et par matériau dans des maps globales (`gLostByProc`, `gLostByMat`), imprimées en fin de run sous les tags `[LOSS][BY-PROC]` et `[LOSS][BY-MAT]`. Ce mécanisme fournissait des **totaux agrégés** mais aucune information événement par événement (position, énergie, volume logique exact).

Objectif : créer deux ntuples ROOT dédiés enregistrant chaque absorption photoélectrique individuelle d'un γ primaire :

- `abs_graphite` — absorptions dans le cône graphite (`logicConeCompton`)
- `abs_inox` — absorptions dans l'inox SS304 (porte-collimateur et enveloppe tube)

Chaque entrée conserve la position (x, y, z), l'énergie cinétique au moment de l'absorption, et l'historique Compton dans le cône (flag + compteur), permettant de corréler absorption et diffusion antérieure.

2 Fichiers modifiés

Trois fichiers sont concernés par cet ajout. Les modifications précédentes (tracking Compton via `MyTrackInfo`, cf. synthèse précédente) sont un prérequis.

Fichier	Rép.	Nature de la modification
`AnalysisManagerSetup.hh`	`include/`	Déclarations des getters des 2 nouveaux ntuples
`AnalysisManagerSetup.cc`	`src/`	Création des 2 ntuples + getters
`SteppingAction.cc`	`src/`	Remplissage des ntuples à la mort du primaire

Les fichiers `MyTrackInfo.hh/.cc` et `SurfaceSpectrumSD.cc` ne requièrent **aucune modification supplémentaire** par rapport à la synthèse précédente.

3 Détail des modifications

3.1 `AnalysisManagerSetup.hh / .cc` — Création des ntuples

En-tête (`.hh`)

Deux nouvelles déclarations de fonctions getter :

Ajouts dans `AnalysisManagerSetup.hh`

```
int GetAbsGraphiteNtupleId();
int GetAbsInoxNtupleId();
```

Source (.cc)

Deux variables statiques, deux blocs `CreateNtuple` et deux getters sont ajoutés. Le ntuple `abs_graphite` comporte 8 colonnes (indices 0–7) et le ntuple `abs_inox` en comporte 9 (indices 0–8), la colonne supplémentaire étant le nom du volume logique (pour distinguer porte-collimateur d'enveloppe).

3.2 `SteppingAction.cc` — Détection et remplissage

Nouvel include

Include ajouté

```
#include "AnalysisManagerSetup.hh"
```

Logique de remplissage

Un nouveau bloc `do {...} while(0)` est inséré dans la section `[TRACE] Fin de piste PRIMAIRE`, juste après le bloc `[LOSS]` existant (lignes ∼1104). Il se déclenche **uniquement** quand les conditions suivantes sont *toutes* réunies :

1. Le track est un primaire (`parentID == 0`) — déjà garanti par le bloc parent
2. Le track est mort (`fStopAndKill`) ou sort du monde
3. Le processus qui a tué le track est `"phot"` (effet photoélectrique)

Ensuite, le volume logique au pré-step détermine dans quel ntuple écrire :

- `namePre == "logicConeCompton"` → ntuple `abs_graphite`
- `materialPre == "StainlessSteel304"` → ntuple `abs_inox`

Extrait du bloc de remplissage (cas graphite)

```
if (!proc || proc->GetProcessName() != "phot") break;

const G4ThreeVector absPos = postPoint->GetPosition();
const G4double abs_ekin_keV = prePoint->GetKineticEnergy()/keV;

// Info Compton depuis MyTrackInfo
G4int had_compton = 0, n_compton = 0;
if (trackInfo) {
    had_compton = trackInfo->HasComptonInCone() ? 1 : 0;
    n_compton   = trackInfo->GetNComptonInCone();
}

if (namePre == "logicConeCompton") {
    const G4int id = GetAbsGraphiteNtupleId();
    man->FillNtupleIColumn(id, 0, eventID);
    man->FillNtupleIColumn(id, 1, track->GetTrackID());
    man->FillNtupleDColumn(id, 2, abs_ekin_keV);
    man->FillNtupleDColumn(id, 3, absPos.x()/mm);
    man->FillNtupleDColumn(id, 4, absPos.y()/mm);
    man->FillNtupleDColumn(id, 5, absPos.z()/mm);
    man->FillNtupleIColumn(id, 6, had_compton);
    man->FillNtupleIColumn(id, 7, n_compton);
    man->AddNtupleRow(id);
}
```

Le cas `abs_inox` est identique, avec en plus la colonne 6 de type `string` contenant le nom du volume logique (`namePre`), ce qui permet de distinguer le porte-collimateur de l'enveloppe inox.

4 Structure complète des ntuples

4.1 Ntuple `abs_graphite`

Table 1: Ntuple `abs_graphite` — Photoabsorptions de primaires dans le cône graphite (`logicConeCompton`).

Col.	Type	Nom	Unité	Description
0	int	eventID	—	Numéro de l'événement
1	int	trackID	—	Identifiant du track
2	double	ekin_keV	keV	Énergie cinétique juste avant l'absorption
3	double	x_mm	mm	Position x de l'absorption
4	double	y_mm	mm	Position y de l'absorption
5	double	z_mm	mm	Position z de l'absorption
6	int	had_compton_in_cone	—	1 si le γ avait subi ≥ 1 Compton dans le cône
7	int	n_compton_in_cone	—	Nombre de Compton dans le cône avant absorption

Volume concerné : `logicConeCompton` uniquement (G4Cons, graphite $\rho = 1.7\,\mathrm{g/cm^3}$, $z \in [1.90; 16.95]$ mm).

4.2 Ntuple `abs_inox`

Table 2: Ntuple `abs_inox` — Photoabsorptions de primaires dans l'inox SS304.

Col.	Type	Nom	Unité	Description
0	int	eventID	—	Numéro de l'événement
1	int	trackID	—	Identifiant du track
2	double	ekin_keV	keV	Énergie cinétique juste avant l'absorption
3	double	x_mm	mm	Position x de l'absorption
4	double	y_mm	mm	Position y de l'absorption
5	double	z_mm	mm	Position z de l'absorption
6	string	volume	—	Nom du volume logique (voir tableau 3)
7	int	had_compton_in_cone	—	1 si le γ avait subi ≥ 1 Compton dans le cône
8	int	n_compton_in_cone	—	Nombre de Compton dans le cône avant absorption

Table 3: Volumes logiques en inox SS304 distingués par la colonne `volume`.

Valeur de `volume`	Pièce	Description
`logicPorteCollimateur`	Porte-collimateur	Paroi interne $R_{\mathrm{int}} = 3.17$ mm
`logicEnveloppeGDML`	Enveloppe MiniX	Tube externe du collimateur

Note : le filtre se fait sur le matériau (`StainlessSteel304`), donc toute pièce en inox dans la géométrie est capturée. La colonne `volume` permet ensuite le tri.

5 Ntuple `plane_passages` (rappel, modification précédente)

Pour mémoire, le ntuple `plane_passages` (ScorePlane, $z = 18$ mm) a été étendu de 10 à 15 colonnes lors de la modification précédente (tracking Compton). Les colonnes 10–14 permettent de distinguer les primaires directs des primaires redirigés par Compton au plan de scoring.

Table 4: Ntuple `plane_passages` — structure complète (15 colonnes).

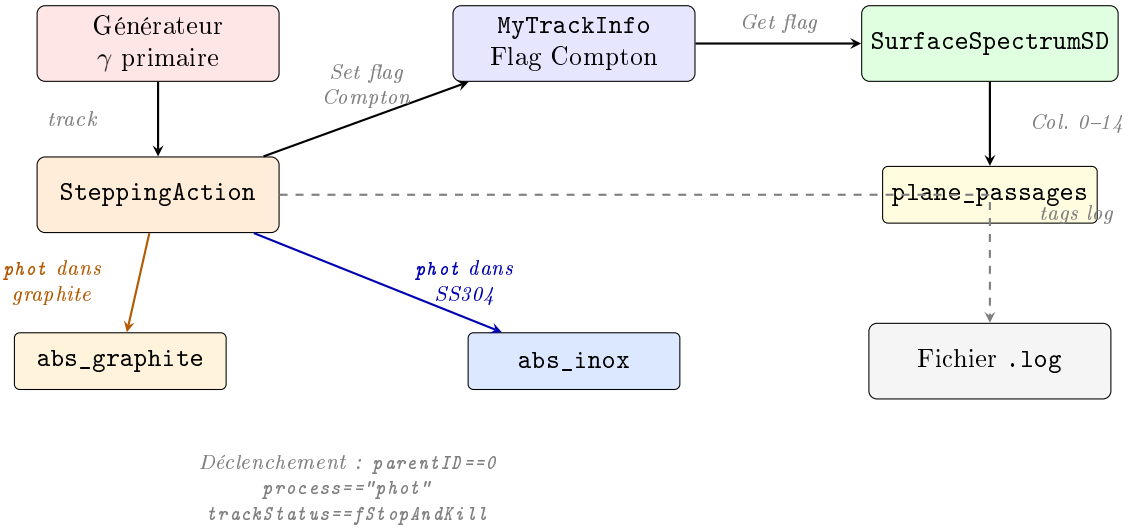
Col.	Type	Nom	Unité	Description
0	`int`	`pdg`	—	Code PDG
1	`string`	`name`	—	Nom particule
2	`int`	`is_secondary`	—	0 = primaire, 1 = secondaire
3	`double`	`x_mm`	mm	Position x au plan
4	`double`	`y_mm`	mm	Position y au plan
5	`double`	`z_mm`	mm	Position z au plan
6	`double`	`ekin_keV`	keV	Énergie cinétique
7	`int`	`trackID`	—	Identifiant du track
8	`int`	`parentID`	—	ID du parent
9	`string`	`creator_process`	—	Processus créateur
10	`int`	`compton_in_cone`	—	1 si Compton dans le cône
11	`int`	`n_compton_cone`	—	Nombre de Compton dans le cône
12	`double`	`compton_x_mm`	mm	x dernière diffusion Compton
13	`double`	`compton_y_mm`	mm	y dernière diffusion Compton
14	`double`	`compton_z_mm`	mm	z dernière diffusion Compton

6 Inventaire complet des ntuples dans le fichier ROOT

Table 5: Liste complète des ntuples dans le fichier de sortie ROOT. Les deux derniers (fond coloré) sont ajoutés par cette modification.

#	Nom ntuple	Nb col.	Source	Contenu
1	`plane_passages`	15	`SurfaceSpectrumSD`	Traversées ScorePlane ($z = 18$ mm)
2	`ScorePlane2_passages`	9	`ScorePlane2SD`	Traversées ScorePlane2 ($z = 28$ mm)
3	`ScorePlane3_passages`	9	`ScorePlane3SD`	Traversées ScorePlane3 ($z = 38$ mm)
4	`WaterRings_passages`	9	`ScorePlane4SD`	Traversées plan anneaux d'eau
5	`ScorePlane5_passages`	9	`ScorePlane5SD`	Traversées ScorePlane5 ($z = 70$ mm)
6	`abs_graphite`	8	`SteppingAction`	Photoabs. dans graphite
7	`abs_inox`	9	`SteppingAction`	Photoabs. dans inox SS304

7 Schéma synoptique du flux de données



8 Nouveaux tags dans le fichier `.log`

Table 6: Tags de log ajoutés pour le suivi des photoabsorptions.

Tag	Source	Fréquence	Contenu
`[STEP][ABS_GRAPHITE]`	`SteppingAction`	50 premiers, puis 1/10 000	Événement, E, position, flag Compton
`[STEP][ABS_INOX]`	`SteppingAction`	50 premiers, puis 1/10 000	Événement, E, volume, position, flag Compton

9 Exemples d'analyse ROOT

9.1 Ntuple `abs_graphite`

```
// Spectre en energie des gamma absorbes dans le graphite
abs_graphite->Draw("ekin_keV")

// Carte (r, z) des absorptions dans le cone
abs_graphite->Draw("z_mm:sqrt(x_mm^2+y_mm^2)",
    "", "COLZ")

// Gamma ayant fait un Compton avant d'etre absorbes
abs_graphite->Draw("ekin_keV",
    "had_compton_in_cone==1")

// Profil z des absorptions directes vs post-Compton
abs_graphite->Draw("z_mm",
    "had_compton_in_cone==0", "", "SAME")
abs_graphite->Draw("z_mm",
    "had_compton_in_cone==1", "", "SAME")
```

9.2 Ntuple `abs_inox`

```
// Spectre dans le porte-collimateur uniquement
abs_inox->Draw("ekin_keV",
    "volume==\"logicPorteCollimateur\"")
```

```
// Spectre dans l'enveloppe uniquement
abs_inox->Draw("ekin_keV",
    "volume==\"logicEnveloppeGDML\"")

// Carte (x, z) de toutes les absorptions inox
abs_inox->Draw("z_mm:x_mm", "", "COLZ")

// Comparaison porte-collimateur vs enveloppe
abs_inox->Draw("ekin_keV",
    "volume==\"logicPorteCollimateur\"")
abs_inox->Draw("ekin_keV",
    "volume==\"logicEnveloppeGDML\"", "", "SAME")

// Gamma ayant suivi un Compton avant absorption inox
abs_inox->Draw("z_mm",
    "had_compton_in_cone==1")
```

9.3 Analyses croisées

```
// Comptages totaux
cout << "Abs graphite : "
     << abs_graphite->GetEntries() << endl;
cout << "Abs inox     : "
     << abs_inox->GetEntries() << endl;

// Fraction des gamma ayant subi un Compton avant absorption
cout << "Graphite post-Compton : "
     << abs_graphite->GetEntries("had_compton_in_cone==1")
     << " / " << abs_graphite->GetEntries() << endl;

// Rapport avec les transmis au ScorePlane
cout << "Transmis directs     : "
     << plane_passages->GetEntries(
        "pdg==22 && is_secondary==0 && compton_in_cone==0")
     << endl;
cout << "Transmis Compton     : "
     << plane_passages->GetEntries(
        "pdg==22 && is_secondary==0 && compton_in_cone==1")
     << endl;
```

10 Compatibilité et remarques

- **Prérequis :** les modifications `MyTrackInfo` (flag Compton) doivent être en place pour que les colonnes `had_compton_in_cone` et `n_compton_in_cone` soient renseignées. Sans ces modifications, ces colonnes vaudront toujours 0.
- **Ntuples existants :** les 5 ntuples de passages (plans 1–5) restent inchangés dans leur structure.
- **Taille du fichier ROOT :** l'impact dépend du nombre d'absorptions. Pour 50M événements, on attend ∼30M entrées dans `abs_inox` et ∼17M dans `abs_graphite` (d'après les bilans `[LOSS][BY-MAT]` précédents), soit ∼200–300 Mo supplémentaires.
- **Multi-thread :** le remplissage utilise `G4AnalysisManager` qui gère en interne la sérialisation des ntuples en mode MT. Les compteurs de log sont `static` dans des fonctions appelées par thread (pas de race condition sur le compteur de log car l'ordre d'affichage n'est pas critique).
- **Recompilation :** un `make` incrémental suffit (3 fichiers modifiés).

Récapitulatif global des modifications (les deux synthèses)

Au total, 6 fichiers ont été modifiés :

Fichier	**Rép.**	**Modif. 1** *(Compton)*	**Modif. 2** *(Photoabs.)*
`MyTrackInfo.hh`	`include/`	✓	—
`MyTrackInfo.cc`	`src/`	✓	—
`SteppingAction.cc`	`src/`	✓	✓
`SurfaceSpectrumSD.cc`	`src/`	✓	—
`AnalysisManagerSetup.hh`	`include/`	—	✓
`AnalysisManagerSetup.cc`	`src/`	✓	✓