2022 年度 卒業論文
sPHENIX 実験 INTT 検出器のための Event Display の開発

奈良女子大学 理学部
物理科学科 物理コース
高エネルギー物理学研究室
藤原愛実

2025 年 3 月 12 日

概要

現在、奈良女子大学高エネルギー物理学研究室では、アメリカブルックヘブン研究所 (BNL) にて 2023 年より実験開始予定の sPHENIX 実験で使用される中間飛跡検出器 (INTT) の開発を行なっている。INTT グループには奈良女子大学のほか、理化学研究所、立教大学、国立中央大学 (National Central University)、国立台湾大学 (National Taiwan University)、その他研究機関が参加している。
sPHENIX 検出器は、現在組み立て中であり、組み立てた後にケーブルの繋ぎ間違いなどのミスがないか、全てのラダーとロックが稼働するかなど、INTT が正常に動くかどうかのテストが必要である。
本研究では、sPHENIX 実験で用いる INTT 検出器が正常に稼働しているかどうかを実験中に確認するときに助けとなるようなツールとして Event Display を開発した。Event Display は ROOT に実装されている Event Visualization Environment, Eve を用いて開発した。開発した Event Display は検出器とヒット位置を 2 種類の方法 (3D,x-y 平面) で表示できる。開発した Event Display は sPHENIX の検出器シミュレーションを用いて動作確認を行なった。

目次

1　序論

1.1　素粒子とは

素粒子とは、物質を構成する最小単位であり、全ての物質は分解すると、素粒子にまで分けられると考えられている。素粒子はスピンによってフェルミ粒子とボーズ粒子に分けられ、フェルミ粒子はされにクォークとレプトンに分けられる。クォークはアップ (u), ダウン (d), ストレンジ (s), チャーム (c), トップ (t), ボトム (b) の 6 種類、レプトンは電子 (e), ミューオン (μ), タウオン (τ) とそれに対応するニュートリノ ν_e, ν_μ, ν_τ とそれぞれの反粒子に分類される。以下に素粒子標準模型で扱われる素粒子について示す。[1]

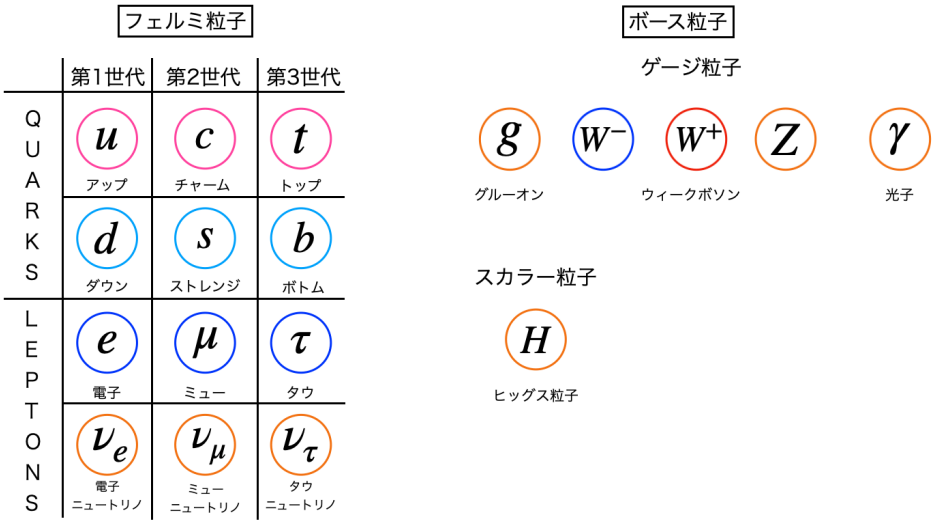


図 1　素粒子標準模型

1.2　クォークグルーオンプラズマ (QGP)

プラズマとは、固体、液体、気体に続く物質の第四相の名称である。プラズマは物質をイオン化エネルギーに相当する高温にした際に実現する。クォークとグルーオンは通常の温度では、強い相互作用により、単体に分けることができず、これをクォークの閉じ込めと呼ぶ。クォークグルーオンプラズマ (QGP) とは、クォークとグルーオンがプラズマ化し、クォークの閉じ込めから解放された状態で、図 2 で示すように、宇宙誕生直後に実現していたとされる状態である。[1]

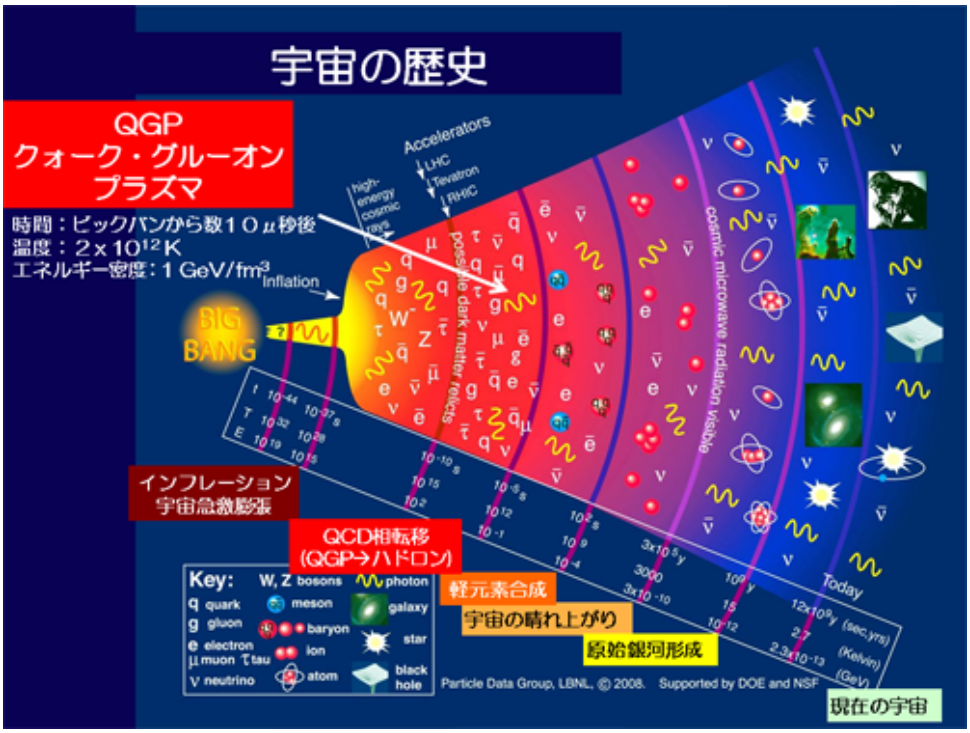


図 2　宇宙の歴史

1.2.1　重イオン衝突実験

QGP を実現するためには、超高温または超高密度な環境を作る必要があり、そのために金イオンや鉛イオンなどの重い原子核を光速近くまで加速して正面衝突させる。そのための実験設備である、重イオン加速器はアメリカにある RHIC、スイスとフランスの国境付近にある LHC がある。これらの加速器の衝突地点に各種検出器を配置し、QGP の性質を解明しようとしている。[1]

2　中間飛跡検出器 (INTT)

2.1　RHIC

Relativistic Heavy Ion Collider(RHIC) は、アメリカブルックヘブンに研究所の重イオン衝突型加速器である。上空からの写真を図 3 に示す。RHIC は周長 3.8km の 2 つの独立なビームラインを持つ。金原子核を光速近くまで加速し、衝突させることで、QGP を実現し、その性質を調べることを目的として、2000 年から稼働している。金原子核が衝突した際の核子対あたりの最大重心系エネルギーは 200GeV である。また、銅原子核や陽子など、さまざまな衝突実験が行われており、陽子・陽子衝突での最大重心系エネルギーは 510GeV である。[2]

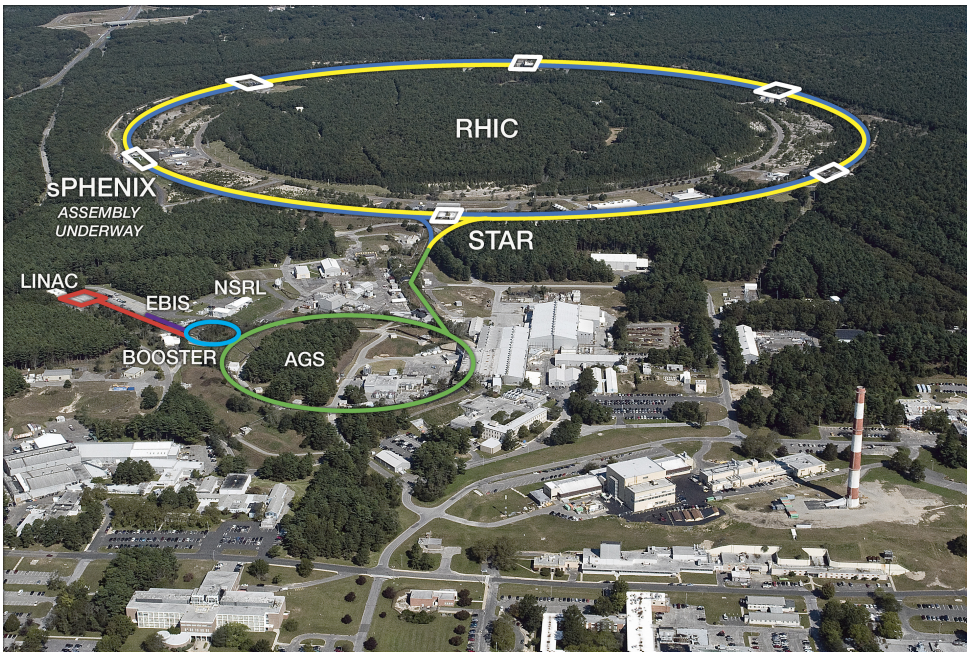


図 3　Relativistic Heavy Ion Collider(RHIC)

2.2　PHENIX 実験

PHENIX(the Pioneering High Energy Nuclear Interaction eXperiment) 実験は RHIC で 2000 年から 2016 年にかけて行われていた実験である。重イオンや陽子を高エネルギーで衝突させることで、QGP を実現し、その性質を調べることを目的としていた。[3]

2.3　sPHENIX 実験

前述の PHENIX 実験をさらに高度化した実験で、2023 年より稼働予定の実験である。sPHENIX では、ハドロンジェットや Υ 中間子を測定することで、QGP の性質をさらに詳しく調べることを目的としている。飛跡検出器は最内層から、MVTX,INTT,TPC が配置され、そのさらに外層にはカロリーメータとマグネットが設置される計画である。[3] 図 4 に各検出機とマグネットの配置を示す。

2.3.1　MVTX

Monolithic-Active-Pixel-Sensor-based VerTex Detector(MVTX) は最内層に配置されるピクセル型半導体検出器で、LHC 加速器を用いた ALICE 実験において開発された Monolithic-Active-Pixel-Sensor-based(MAPS) を用いる。衝突中心からビーム軸方向に$\pm$10cm 、方位角方向に対して 2π の範囲を覆っている。位置分解能が高いことが特徴であり、主に衝突点の再構成において大きな役割を担う。[4]

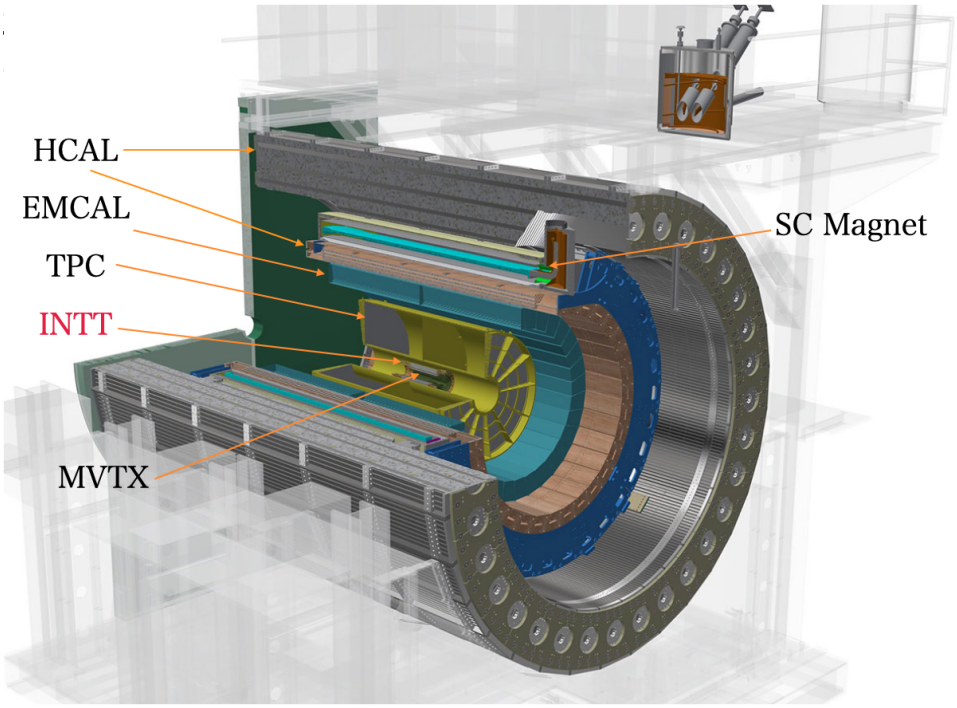


図 4　sPHENIX 検出器

2.3.2　TPC

Time Projection Chamber(TPC) は 3 つの飛跡検出器のうち、最外層 (ビームパイプから 20cm-78cm) に設置されたガス検出器である。衝突中心からビーム軸方向に±1m、方位角方向に対して 2π の範囲を覆っている。48 枚の読み出しパッドからなり、高い位置分解能をもつ。荷電粒子の飛跡を再構成し、曲率半径から運動量を計測する。また、$\frac{dE}{dx}$ を用いて、電子、π 中間子、K 中間子、陽子など、粒子の種類を特定する。[4] [5]

2.4　INTT

INtermediated Tracking detector(INTT) は sPHENIX 検出器において、2 層目 (ビームパイプから 6cm-12cm) に配置されたストリップ型シリコン検出器である。衝突中心からビーム軸方向に±23cm、方位角方向に対して 2π の範囲を覆っている。MVTX と TPC の間の飛跡を結び、粒子の運動量分解能を上げる。また、時間分解能が高いことも特徴であり、どのビーム交差で起きた粒子生成かを特定する役割を担う。図 5 で示すように、INTT はバレル状の 2 層構造になっており、内層 24 本、外層 32 本、計 56 本のラダーから構成されている。[4]

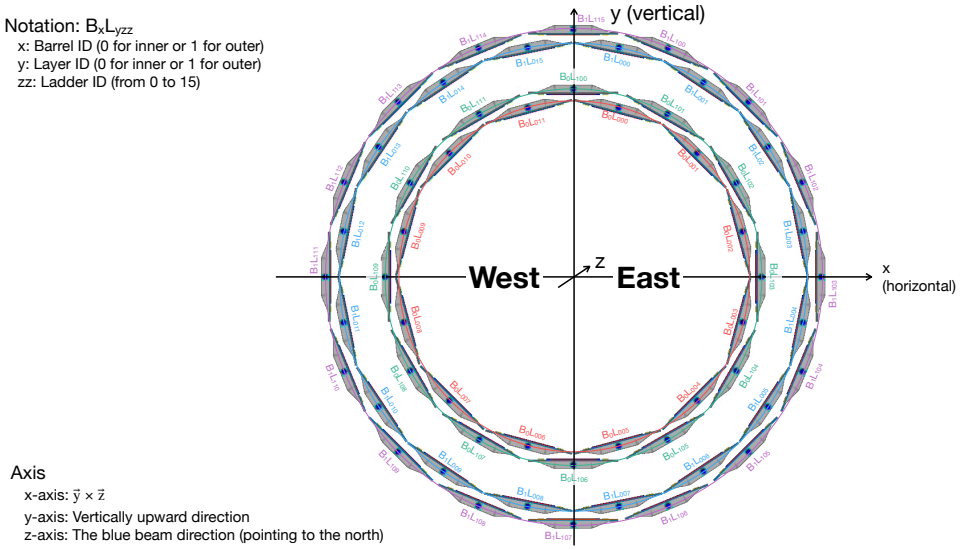


図 5　INTT の断面図

2.4.1　INTT 用シリコンストリップセンサー

INTT では、ストリップ長の異なる typeA、typeB、２種類のシリコンセンサーを使用している。各センサーのサイズは typeA が 128×19.9×0.32mm,typeB が 100×19.9×0.32mm である。また、INTT 用シリコンセンサーの構造を図 6 に示す。INTT ラダーは 2 つのシリコンセンサー、読み出しチップ、読み出し基盤とそれらを支えるステーブで構成されている。INTT ラダー上のシリコンセンサー、読み出しチップ、読み出し基盤は電気的に左右に分離されており、センサーで計測したデータは左右独立にデータを読み出している。この、独立した半分部分をハーフラダーと呼び、2 つのハーフラダーでフルラダーを構成している。INTT はこのフルラダー 56 本から構成されている。[6] [5]

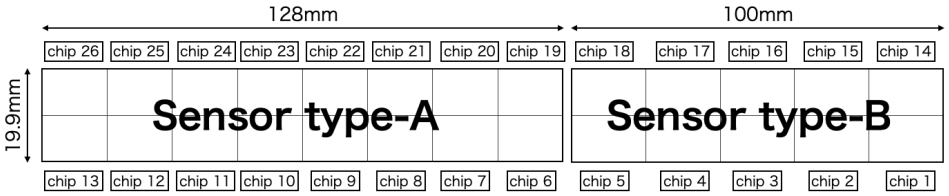


図 6　INTT 用ストリップセンサー

2.5　研究目的

2.5.1　INTT Event Display

INTT Event Display に求められる主な役割は、INTT 上のヒット位置など、INTT に関する情報をイベント毎に一目で確認できること、ラダーのアライメントを確認できること、INTT が正確に組み立てられていて、正常に動作していることを実験中に確認できることの 3 つである。
また、本研究ではこの要求を満たせるようなツールを作成することを目的としている。

2.5.2 INTT Event Display 開発に必要な機能

INTT Event Display 開発のためには、ワールド座標の設定、INTT の 3DCG モデルを作成、表示する機能、ヒット座標を表示する機能、3D を 2D に投影する機能などが必要である。

3 Event Display の開発

3.1 Event Display

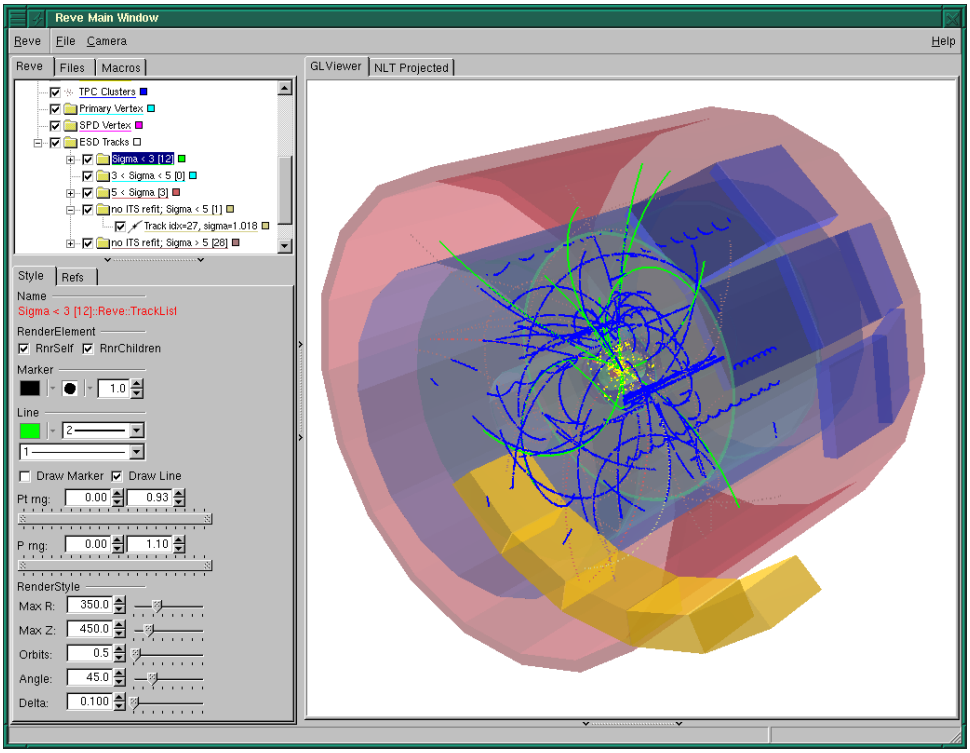


図 7　A simulated ALICE pp@14TeV event in 3D

Event Display とは検出器の Geometry と再構成した粒子の飛跡を同時に描画したもので、シミュレーションと再構成コードのデバック、実験データの可視化などを目的として用いられるツールである。例として、ALICE 実験のシミュレーションデータを可視化したものを図 7 に示す。[7]

3.2 ROOT

ROOT は CERN で開発されている、高エネルギー物理学の解析で用いられるソフトウェアである。グラフやヒストグラムの作成などのツールや、ユーザによって作成された高エネルギー物理学の解析に特化したライブラリやフレームワークが数多くある。

Event Display を作成するための機能として、ワールド座標を設定し、その上に INTT ラダーの 3DCG モデルやヒット座標を描画する機能が必要である。ROOT では、これらの機能を実装するために Eve や TGeoManager が用意されている。以下ではそれらを用いて INTT Event Display を作成した。

3.3 TGeoManager

TGeoManager は ROOT、TGeo での geometry 作成、視覚化、ファイル入出力など 3DCG のモデリングとファイルへの入出力をサポートしている。座標空間の定義、箱や筒などの簡単な形状の 3DCG の作成、オブジェクトの階層構造の構築、作成したオブジェクトの移動、回転、後述する EVE で使用できる形式でファイルに書き出すなどの機能がある。

3.3.1 Geometry 作成の例

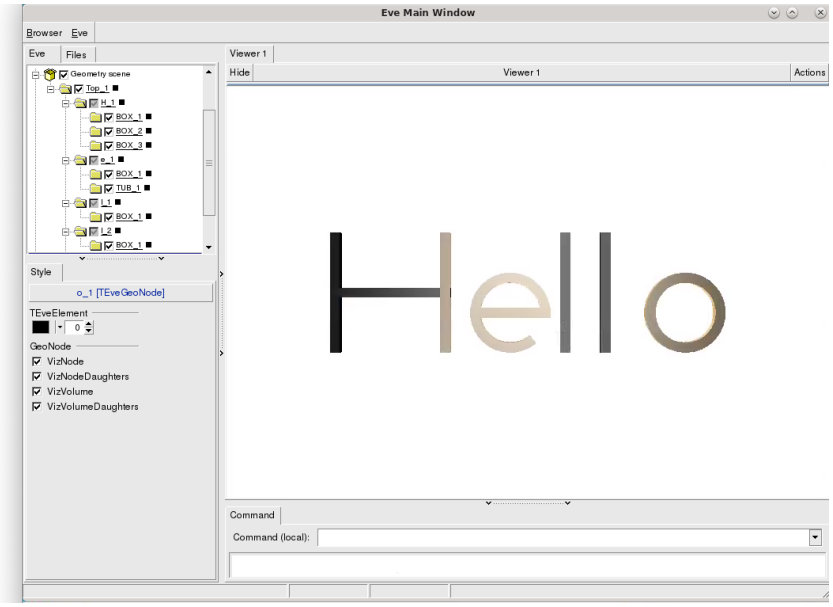


図 8　TGeoManager を用いて作成した Geometry の例

TGeoManager の具体的な機能や、階層構造について説明するために、図 8 のような Geometry を作成し、描画する。また、この Geometry を作成するソースコードの全文は付録 A を参照。
まず、Geometry ライブラリーをロードする。

```
2 gSystem->Load("libGeom");
```

次に、TGeoManager クラスのインスタンスを作成する。このクラスは、ジオメトリーの構築、視覚化を行い、カメラの移動やジオメトリーの表示/非表示の切り替えなどを行える UI を含んでいる。この後、構造体ポインター geom からジオメトリークラスにアクセスできる。

```
3 TGeoManager *geom = new TGeoManager("hello", "hello");
```

箱の形をした、ワールドボリュームを作成する。ボリュームは任意の形の 3DCG モデルのことである。ワールドボリュームの形状は箱形である必要はないが、箱型や筒型の方が処理が速いため、箱型や筒型に設定することが推奨される。また、デフォルトの単位は cm である。

```
6 TGeoVolume *top=geom->MakeBox("Top",NULL,100.,100.,100.);
```

先ほど作った、top ボリュームをワールドボリュームに設定する。この操作はジオメトリーを閉じる前に実行

する必要がある。また、ワールドボリュームはデフォルトで表示されない設定になっている。

7 geom->SetTopVolume(top);

ここからは、文字の形のボリュームを作成する。

10 TGeoVolume * H=geom-> MakeBox(”H”,NULL,100.,100.,100.);

先ほど作ったボリューム H を非表示にする。

11 H->SetVisibility(kFALSE);

棒状のボリューム bar1 を作成する。また、箱状のボリュームを作成する関数 MakeBox(dx,dy,dz) の dx,dy,dz は 3DCG で一般的な縦、横、高さの辺の長さを設定しているのではなく、箱の中心から x,y,z 方向への長さを設定していることに留意する。

12 TGeoVolume * bar1=geom->MakeBox(”BOX”,NULL,0.5,0.5,6.0);

回転角度を定義する。内部的には回転行列を生成している。

13 TGeoRotation * rot1 =new TGeoRotation(”rot1”,0,90,0);

移動先の座標と、作用させたい回転行列を定義する。

14 TGeoCombiTrans * combi1 =new TGeoCombiTrans(0,0,5.5,rot1);

ボリューム H に先ほど定義した、座標に移動、回転させたボリューム bar1 のノードを追加する。

15 H->AddNode(bar1,0,combi1);

以下同様に bar1 を移動、回転させ、ボリューム H にノードを追加することで H の形をしたボリュームを作成する。ボリューム H に bar1 のノードを追加することで、H という文字を作っている bar1 の相対的な位置を崩さずにまとめて移動、回転といった操作ができるようになる。

他の文字も同様の手順で作っている。

MakeTubs で角度を指定した、弧型のジオメトリーを作成できる。

31 TGeoVolume *tub=geom->MakeTubs(”TUB”,NULL,3.,4.,0.5,90.,40.);

MakeTube で円筒型のジオメトリーを作成できる。

38 TGeoVolume *tube=geom->MakeTube(”TUBE”,NULL,3.0,4.0,0.5);

移動先の座標を定義する。

42 TGeoTranslation * tr1 =new TGeoTranslation(0,0,-17.5);

ジオメトリー H に先ほど定義した、移動を適用して、top ノードに追加する。

43 top->AddNode(H,1,tr1);

ジオメトリーを閉じる。

52 geom->CloseGeometry();

TEveManager クラスのインスタンスを作成する。この後、ポインター gEve からイブマネジャークラスにアクセスできる。

54 TEveManager::Create();

TGeoManager で作成したジオメトリーを、TEveManager にインポートする。

55 gEve->AddGlobalElement(new TEveGeoTopNode(geom, geom->GetTopNode()));

図 8 のようなウィンドウを出力する。

56 gEve->Redraw3D(kTRUE);

また、図 8 のジオメトリーのノードツリーは図 9 のようになっている。[8]

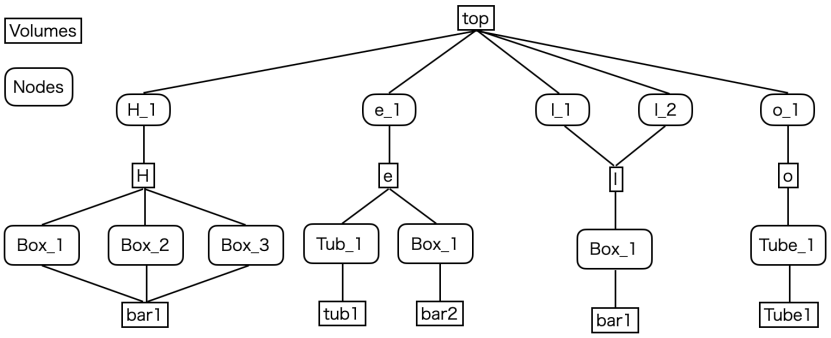


図 9　図 8 の Geometry のノードツリー

3.4　EVE

Event Visualization Environment(EVE) は ALICE 実験のために開発されたイベント可視化環境である。階層的なオブジェクトの作成、GUI、Geometry やヒット座標の視覚化、2D 投影の自動作成などの機能があり、オブジェクト管理のためのフレームワークとして機能する。[9]

3.5　開発環境

ROOT のバージョンが 6.24/06。コンパイラーのバージョンは g++ (GCC) 8.3.0。リモートデスクトップソフトウェア NoMachine を通して実行する。

3.6　INTT Geometry の作成

3.6.1　INTT Geometry 作成の流れ

各ラダーのモデルはセンサー A/B のサイズに合わせた箱型ボリュームによって構成されている。各ラダーの位置はシミュレーションマクロで用いられている座標情報を取得し、使用している。位置の取得については次章で説明する。

3.6.2　各センサーの中心座標の取得

シミュレーションマクロ内で設定されている各センサーの中心座標を取得し、ファイルに書き出した。各センサーの位置を取得するためには MVTX から数えたレイヤー数 (INTT は 3 層から 7 層の 4 層に当たる)、`ladder_z_index`、`ladder_phi_index`、を指定する必要がある。その 3 つの変数を以下のプログラムに通すと、ladderLocation という配列にセンサーの座標が代入される。

```
ladderLocation[3] = {0.,0.,0.};
auto genhitkeyintt = InttDefs::genHitSetKey
(uint8_t lyr,uint8_t ladder_z_index,uint8_t ladder_phi_index,int time_bucket);
auto surfintt = m_tGeometry->maps().getSiliconSurface(genhitkeyintt);
m_geom->find_segment_center(surfintt,m_tGeometry,ladderLocation);
```

また、`ladder_z_index` の番号の振り方については、ladder z id は横軸が z 軸、縦軸が y 軸として図 10 のように取られている。`ladder_phi_index` は横軸が x 軸、縦軸が y 軸として、図 11 のように、反時計回りに番号が振られている。

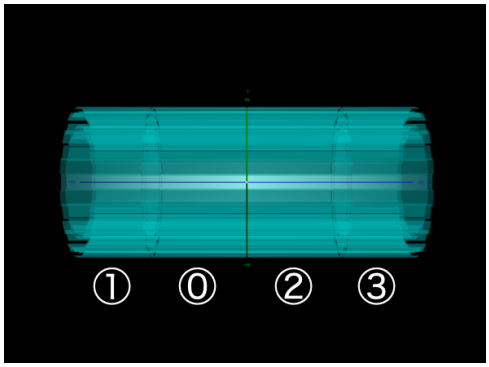


図 10　ladder z index

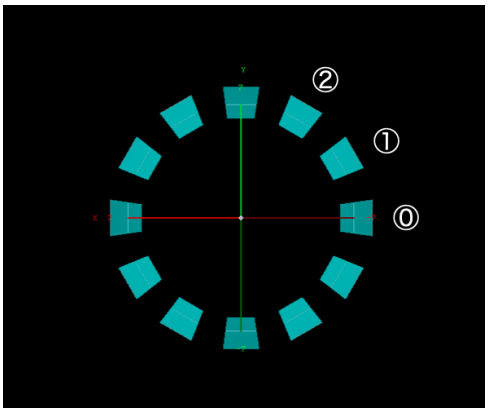
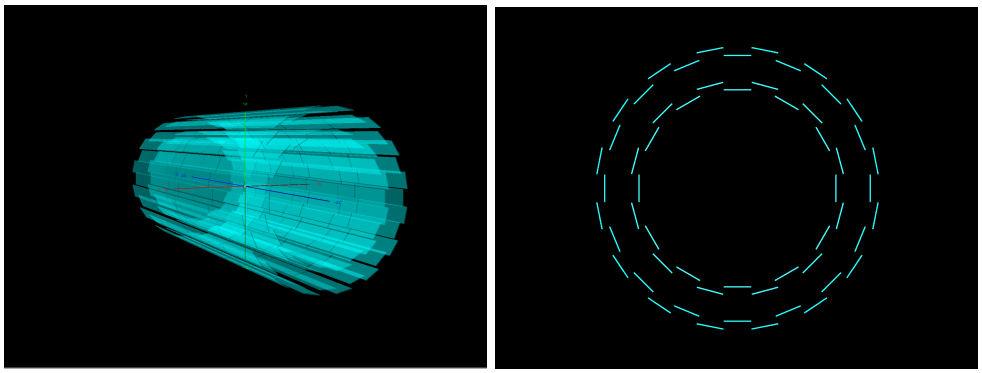


図 11　ladder phi index

3.6.3　INTT Geometry

前述のコードで取得した各センサーの座標にセンサーと同じサイズの箱型の Geometry を作成し、円の接線の角度と同じ角度に設定することで INTT の Geometry を作成した。図 12 に作成した Geometry の画像を示す。



(a) 3D　　(b) r-ϕ プロジェクション

図 12　INTT Geometry

3.7　ヒット位置の描画

3.7.1　ヒット位置の取得

sPHENIX のデータフォーマットである DST 形式の root ファイルからヒット位置を読み出す。ヒット位置は TrkrClusterNode に格納されており、ヒット位置を読み出し 3 次元ベクターの配列に入れ直して後述の

ヒット位置を描画する関数に渡している。

3.7.2 ヒット位置の表示

ヒット位置の表示には TEvePointSet クラスを用いている。ヒット位置を入れた 3 次元ベクターの配列からヒットの座標を 1 つずつ読み、SetNextPoint (Eve で描画する点のリストに点の座標を追加する) 関数に渡し、そのリストにある座標に点を描画している。

3.8 実装されている機能

3.8.1 3D 表示

Geometry とヒット位置を 3D で表示し、ユーザーがカメラを回転させて確認できるモード。TEvePointSet クラスを利用して、ヒット座標を可視化している。その、ヒット座標を TEveGeoTopNode クラスを利用して、TGeo を用いて作成した Geometry を保存したファイルからインポートした INTT の Geometry とともに表示している。ここでは、π 粒子が原点で崩壊したイベントを表示している。

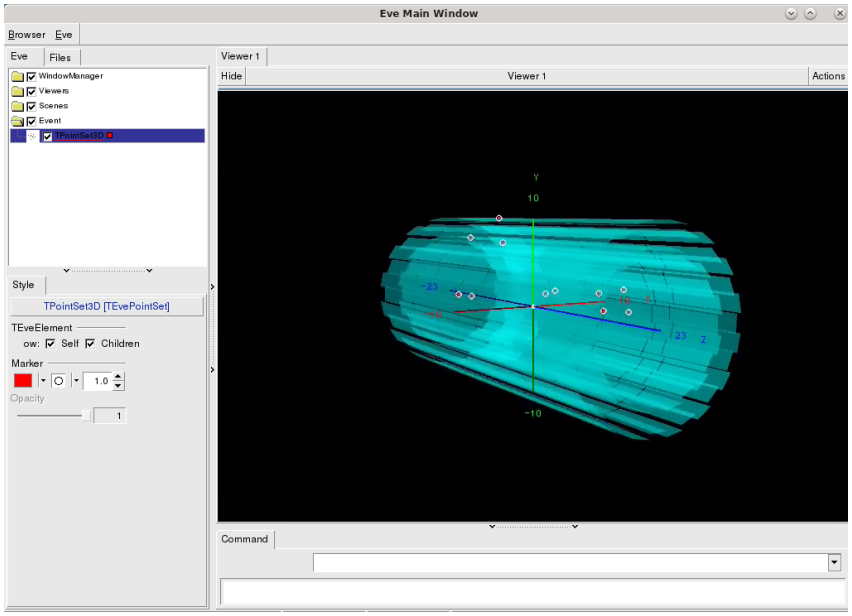


図 13　Simple Event のイベントディスプレイ表示 (3D)

3.8.2 r-ϕ プロジェクション

Geometry とヒット座標を輪切りにしたような 2D 投影ビューで描画し、ヒット位置の位置関係を容易に確認できるモード。TEveProjectionManager を利用して、前述の 3D 表示と同じように作成した、3D 空間での INTT Geometry とヒット座標を 2D に変換している。

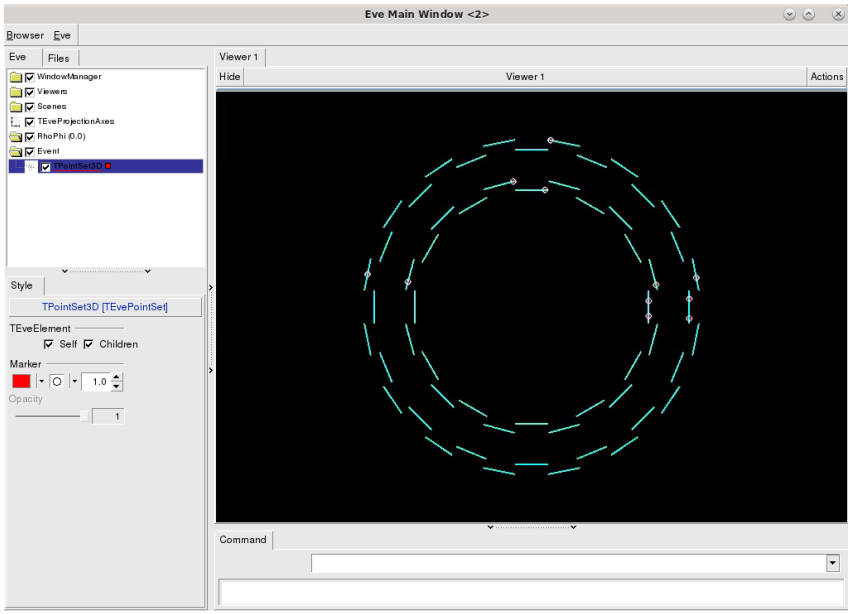


図 14　Simple Event のイベントディスプレイ表示 (r-ϕ)

3.9　イベントディスプレイの使用方法

イベントディスプレイを使用する前に、表示するデータをシミュレーションで生成する必要がある。このイベントディスプレイでは、sPHENIX のシミュレーションを使用して生成した、DST ファイルを読み込んで表示することができる。動作確認のために原点で π^- 5 個が崩壊したイベントを表示した。
イベントディスプレイを表示するには以下のコマンドを順に実行する。

1. root Loadfile.C

データファイルを読み込むマクロを実行し、DST データをイベントディスプレイを表示するマクロ (INTTEventDisplay.cc) に渡す。

2. intteventdisplay->DrawHits() or `DrawHit_rphi()`

intteventdisplay というポインタにアクセスし、DrawHits() もしくは `DrawHit_rphi()` という関数を実行する。DrawHits() は 3D 表示するモードで、`DrawHit_rphi()` は r-ϕ プロジェクションを表示するモードである。

3. 次のイベントを表示したい場合には、se->run(1) を実行してから、2 の操作をもう一度行う。

4　まとめ

本研究では、組み立てた後の INTT が正常に稼働しているかを実験中にその場で確認する助けになるツールである Event Display の 1st Version を開発した。
今後も、ρ-z プロジェクションや粒子の飛跡を描くなど、Event Display に必要な機能を随時実装して、バージョンアップしていく予定である。

謝辞

本研究を進めるにあたって、多くの方々にお世話になり、支えていただきました。
指導教員の蜂谷先生には、ターミナルを使ってのデータの取り扱いなどといった初歩的な点から、他者に共有することを考えたソフトウェアの開発の際に留意するべき点などさまざまなことをお教えいただきました。初めは、右も左もわからず、本当に私に Event Display が開発できるのか不安でしたが、わかりやすくお教えいただき、また、定期的に進捗を聞いていただいたおかげでここまで至ることができました。
また、INTT グループの秋葉さん、中川さん、糠塚さんにも心より感謝申し上げます。同じ研究室の林井先生、宮林先生、下村先生及び高エネルギー研究室の皆様にも感謝申し上げます。特に、高濱先輩には、自身の研究もあって忙しい中、ベースとなるコードを作成していただき、コードの動かし方などを教えていただき、数々の質問に対して真摯にお答えいただいたことに感謝申し上げます。また、同学年の皆さんにはいつも励ましてもらいました。
最後に、この大学に通わせてくれて、毎日支えてくれた家族に感謝します。

参考文献

[1] 秋葉康之. クォーク・グルーオン・プラズマの物理. 共立出版, 2014.

[2] 高濱瑠菜. 重イオン衝突実験における multi-parton interaction の測定. https://webhepl.cc.nara-wu.ac.jp/old_HP/thesis/4kaisei/2020/takahama_b4thesis.pdf, 2021. 2023 年 2 月 13 日参照.

[3] 鈴木彩香. Rhic-sphenix 実験における intt シリコンモジュールの性能評価 ビームテスト実験のデータ解析. https://webhepl.cc.nara-wu.ac.jp/old_HP/thesis/master/2019suzuki/master_thesis_suzuki.pdf, 2020. 2022 年 11 月 15 日参照.

[4] 杉山由佳. Rhic-sphenix 実験における 中間飛跡検出器 intt 用シリコンセンサーでの エネルギー損失測定の評価. https://webhepl.cc.nara-wu.ac.jp/old_HP/thesis/4kaisei/2021/sugiyama_b4thesis.pdf, 2022. 2023 年 2 月 13 日参照.

[5]

[6] 西森早紀子. sphenix 実験における中間飛跡検出器 intt の 宇宙線を用いた検出効率の研究. https://webhepl.cc.nara-wu.ac.jp/old_HP/thesis/4kaisei/2020/nishimori_b4thesis.pdf, 2020. 2022 年 10 月 13 日参照.

[7] Event display. https://root.cern.ch/doc/master/group__TEve.html. 2023 年 4 月 14 日参照.

[8] The geometry package. https://root.cern/doc/master/group__Geometry.html. 2023 年 3 月 29 日参照.

[9] Matevz Tadel. Eve - event visualization environment of the root framework. http://pos.sissa.it//archive/conferences/070/103/ACAT08_103.pdf, 2008. 2023 年 3 月 27 日参照.

付録 A　Geometry を作成するコード例

```
1  void Hello(){
2      gSystem->Load("libGeom");
3      TGeoManager *geom = new TGeoManager("hello", "hello");
4
5      //top create
6      TGeoVolume *top=geom->MakeBox("Top",NULL,100.,100.,100.);
7      geom->SetTopVolume(top);
8
9      //H
10     TGeoVolume * H=geom-> MakeBox("H",NULL,100.,100.,100.);
11     H->SetVisibility(kFALSE);
12     TGeoVolume * bar1=geom->MakeBox("BOX",NULL,0.5,0.5,6.0);
13     TGeoRotation * rot1 =new TGeoRotation("rot1",0,90,0);
14     TGeoCombiTrans * combi1 =new TGeoCombiTrans(0,0,5.5,rot1);
15     H->AddNode(bar1,0,combi1);
16     H->AddNode(bar1,1,0);
17     TGeoCombiTrans * combi2 =new TGeoCombiTrans(0,0,-5.5,rot1);
18     H->AddNode(bar1,2,combi2);
19
20     //l
21     TGeoVolume * l=geom-> MakeBox("l",NULL,100.,100.,100.);
22     l->SetVisibility(kFALSE);
23     TGeoCombiTrans * combi3 =new TGeoCombiTrans(0,0,0,rot1);
24     l->AddNode(bar1,0,combi3);
25
26     //e
27     TGeoVolume * e=geom-> MakeBox("e",NULL,100.,100.,100.);
28     e->SetVisibility(kFALSE);
29     TGeoVolume *bar2=geom->MakeBox("BOX",NULL,0.5,0.5,4.0);
30     e->AddNode(bar2,0,0);
31     TGeoVolume *tub=geom->MakeTubs("TUB",NULL,3.,4.,0.5,90.,40.);
32     TGeoRotation * rot2=new TGeoRotation("rot2",270,90,0);
33     TGeoCombiTrans *combi4=new TGeoCombiTrans(0,0,0,rot2);
34     e->AddNode(tub,1,combi4);
35
36     //o
37     TGeoVolume * o=geom-> MakeBox("o",NULL,100.,100.,100.);
```

```
38      TGeoVolume *tube=geom->MakeTube("TUBE",NULL,3.0,4.0,0.5);
39      TGeoCombiTrans *combi5=new TGeoCombiTrans(0,0,0,rot2);
40      o->AddNode(tube,0,combi4);
41
42      TGeoTranslation * tr1 =new TGeoTranslation(0,0,-17.5);
43      top->AddNode(H,1,tr1);
44      TGeoTranslation * tr2 =new TGeoTranslation(0,-2,-5.5);
45      top->AddNode(e,2,tr2);
46      top->AddNode(l,3,0);
47      TGeoTranslation * tr3 =new TGeoTranslation(0,0,4);
48      top->AddNode(l,4,tr3);
49      TGeoTranslation * tr4 =new TGeoTranslation(0,-2,12);
50      top->AddNode(o,5,tr4);
51
52      geom->CloseGeometry();
53
54      TEveManager::Create();
55      gEve->AddGlobalElement(new TEveGeoTopNode(geom, geom->GetTopNode()));
56      gEve->Redraw3D(kTRUE);
57  }
```

付録 B　INTT の Geometry を作成するコード

```
1   #include <TGeoManager.h>
2   #include <TGeoVolume.h>
3   #include <TAttLine.h>
4   // mode 0 default thickness of ladder is right
5   // mode 1 make geometry for r-phi projection thickness of ladder is 1mm
6   using namespace std;
7   void make_inttgeom(int mode = 0)
8   {
9       // read vector file
10      TFile *location = TFile::Open("segmentlocation.root", "READ");
11      vector<Double_t> segmentALocationX;
12      vector<Double_t> segmentALocationY;
13      vector<Double_t> segmentALocationZ;
14
15      vector<Double_t> *tmpxa;
16      vector<Double_t> *tmpya;
```

```
17      vector<Double_t> *tmpza;
18
19      vector<Double_t> segmentBLocationX;
20      vector<Double_t> segmentBLocationY;
21      vector<Double_t> segmentBLocationZ;
22
23      vector<Double_t> *tmpxb;
24      vector<Double_t> *tmpyb;
25      vector<Double_t> *tmpzb;
26
27      location->GetObject("segmentALocationX", tmpxa);
28      location->GetObject("segmentALocationY", tmpya);
29      location->GetObject("segmentALocationZ", tmpza);
30
31      location->GetObject("segmentBLocationX", tmpxb);
32      location->GetObject("segmentBLocationY", tmpyb);
33      location->GetObject("segmentBLocationZ", tmpzb);
34
35      location->Close();
36
37      segmentALocationX = *tmpxa;
38      segmentALocationY = *tmpya;
39      segmentALocationZ = *tmpza;
40
41      segmentBLocationX = *tmpxb;
42      segmentBLocationY = *tmpyb;
43      segmentBLocationZ = *tmpzb;
44
45          for (int i = 0; i < segmentALocationX.size(); i++)
46          {
47                  cout << segmentALocationZ[i] << endl;
48                  cout << segmentBLocationZ[i] << endl;
49          }
50
51      // makeinttgeom
52      TGeoManager *geom = new TGeoManager("geom", "geom");
53      TGeoVolume *inttgeom = geom->MakeBox("intt", NULL, 20, 20, 20);
54      geom->SetTopVolume(inttgeom);
55      inttgeom->SetVisibility(kFALSE);
56
```

```
57      // segment
58      double segment_thickness = 0.32; // mm
59      double segment_thicknessrphi = 1.0;
60      double segment_widthY = 19.9;   // mm
61      double segmentA_widthZ = 128.0; // mm
62      double segmentB_widthZ = 100.0; // mm
63      TGeoVolume *segmentA = geom->MakeBox("BOX", NULL, segment_thickness / 10 / 2,
64      segment_widthY / 10 / 2, segmentA_widthZ / 10 / 2);
65      TGeoVolume *segmentB = geom->MakeBox("BOX", NULL, segment_thickness / 10 / 2,
66      segment_widthY / 10 / 2, segmentB_widthZ / 10 / 2);
67      segmentA->SetLineColor(kCyan);
68      segmentB->SetLineColor(kViolet);
69
70      TGeoVolume *lyrA = geom->MakeBox("lyrA", NULL, 10, 10, 10);
71      lyrA->SetVisibility(kFALSE);
72      inttgeom->AddNode(lyrA, 1, 0);
73
74      TGeoVolume *lyrB = geom->MakeBox("lyrB", NULL, 10, 10, 10);
75      lyrB->SetVisibility(kFALSE);
76      inttgeom->AddNode(lyrB, 2, 0);
77
78      int sum_segmentA = segmentALocationX.size();
79      int sum_segmentB = segmentBLocationX.size();
80
81      TGeoRotation *rotA[sum_segmentA];
82      TGeoCombiTrans *combitranslyrA[sum_segmentA];
83
84      TGeoRotation *rotB[sum_segmentB];
85      TGeoCombiTrans *combitranslyrB[sum_segmentB];
86
87      double phi;
88      double dr = (segment_thicknessrphi - segment_thickness) / 2.0;
89
90      switch (mode){
91          case 0:
92              for (int i = 0; i < sum_segmentA; i++){
93                  phi = atan2(segmentALocationY[i], segmentALocationX[i]) * 180 / M_PI;
94                  rotA[i] = new TGeoRotation(Form("rotA[%d]", i), phi, 0., 0.);
95                  combitranslyrA[i] = new TGeoCombiTrans(segmentALocationX[i],
96                  segmentALocationY[i], segmentALocationZ[i], rotA[i]);
```

```
97                  lyrA->AddNode(segmentA, i, combitranslyrA[i]);
98              }
99
100             for (int i = 0; i < sum_segmentA; i++){
101                 phi = atan2(segmentBLocationY[i], segmentBLocationX[i]) * 180 / M_PI;
102                 rotB[i] = new TGeoRotation(Form("rotB[%d]", i), phi, 0., 0.);
103                 combitranslyrB[i] = new TGeoCombiTrans(segmentBLocationX[i],
104                 segmentBLocationY[i], segmentBLocationZ[i], rotB[i]);
105                 lyrB->AddNode(segmentB, i, combitranslyrB[i]);
106             }
107             break;
108
109         case 1:
110             for (int i = 0; i < sum_segmentA; i++){
111                 phi = atan2(segmentALocationY[i], segmentALocationX[i]) * 180 / M_PI;
112                 rotA[i] = new TGeoRotation(Form("rotA[%d]", i), phi, 0., 0.);
113                 combitranslyrA[i] = new TGeoCombiTrans(segmentALocationX[i] - dr * cos(phi),
114                 segmentALocationY[i] - dr * sin(phi), segmentALocationZ[i], rotA[i]);
115                 lyrA->AddNode(segmentA, i, combitranslyrA[i]);
116             }
117
118             for (int i = 0; i < sum_segmentA; i++){
119                 phi = atan2(segmentBLocationY[i], segmentBLocationX[i]) * 180 / M_PI;
120                 rotB[i] = new TGeoRotation(Form("rotB[%d]", i), phi, 0., 0.);
121                 combitranslyrB[i] = new TGeoCombiTrans(segmentBLocationX[i] - dr * cos(phi),
122                 segmentBLocationY[i] - dr * sin(phi), segmentBLocationZ[i], rotB[i]);
123                 lyrB->AddNode(segmentB, i, combitranslyrB[i]);
124             }
125             break;
126
127         default:
128             cout << "mode number is wrong" << endl;
129             break;
130         }
131
132         geom->CloseGeometry();
133         inttgeom->Draw();
134
135         string savefilename;
136
```

```
137         // file save
138         switch (mode)
139         {
140         case 0:
141             savefilename = "inttgeometry.root";
142             break;
143         case 1:
144             savefilename = "inttgeometry_rphi.root";
145             break;
146         default:
147             break;
148         }
149 
150         TFile *file = new TFile(savefilename.c_str(), "RECREATE");
151         geom->Write();
152         file->Close();
153 }
```

付録 C　INTT Event Display を描画する関数のコード

C.1　3D

```
1     void AnaTutorial :: DrawHits()
2 {
3     TEveManager::Terminate();
4     TEveManager::Create();
5 
6     cout<<"TEve created"<<endl;
7 
8     int npoints = m_clusters.size();
9     cout<<"npoints = " <<npoints<<endl;
10    TEvePointSet* ps = new TEvePointSet(npoints);
11    ps->SetOwnIds(kTRUE);
12 
13    cout<<"new TEvePointSet"<<endl;
14 
15    int counter  = 0;
16    for(auto itr = m_clusters.begin(); itr != m_clusters.end();++itr ){
17      auto cluster = itr[0];
18      ps->SetNextPoint(cluster[0], cluster[1], cluster[2]);
```

```
19        ps->SetPointId(new TNamed(Form("Point %d", counter), ""));
20        counter ++ ;
21        cout<<"itr = "<<*itr<<endl;
22        cout<<"cluster[0] = "<<cluster[0]<<"   cluster[1] = "<<cluster[1]<<"
23        cluster[2] = "<<cluster[2]<<endl;
24      }
25
26      ps->SetMarkerColor(2);
27      ps->SetMarkerSize(1.0);
28      ps->SetMarkerStyle(4);
29
30
31      gEve->AddElement(ps);
32
33      //geometry load
34      gGeoManager = gEve->GetGeometry("/sphenix/u/mfujiwara/Documents/inttgeometry.root");
35      TEveGeoTopNode* geom = new TEveGeoTopNode(gGeoManager, gGeoManager->GetTopNode());
36      geom->CanEditMainTransparency();
37      geom->SetMainTransparency(50);
38      gEve->AddGlobalElement(geom);
39
40
41      //x,y,z axis show
42      TEveViewer *ev = gEve->GetDefaultViewer();
43      TGLViewer *gv = ev->GetGLViewer();
44      gv->SetGuideState(TGLUtil::kAxesOrigin, kTRUE, kFALSE, 0);
45      gEve->Redraw3D(kTRUE);
46
47      //Camera control
48      gSystem->ProcessEvents();
49      gv->CurrentCamera().RotateRad(0,-3.14/2);
50
51      gv->RequestDraw();
52
53  }
```

C.2 r-ϕ プロジェクション

```
1  void AnaTutorial :: DrawHit_rphi(){
```

```
2     TEveManager::Terminate();
3     TEveManager::Create();
4
5     //open geom file
6     gGeoManager = gEve->GetGeometry("/sphenix/u/mfujiwara/Documents/inttgeometry_rphi.root");
7     TEveGeoTopNode *geom = new TEveGeoTopNode(gGeoManager, gGeoManager->GetTopNode());
8     gEve->AddGlobalElement(geom);
9
10    // camera
11    TEveScene* s = gEve->SpawnNewScene("Projected Event");
12    gEve->GetDefaultViewer()->AddScene(s);
13    TGLViewer* v = gEve->GetDefaultGLViewer();
14    v->SetCurrentCamera(TGLViewer::kCameraOrthoXOY);
15    TGLOrthoCamera& cam = (TGLOrthoCamera&) v->CurrentCamera();
16    cam.SetZoomMinMax(0.2, 20);
17
18    // projections
19    TEveProjectionManager* mng =
20       new TEveProjectionManager(TEveProjection::kPT_RPhi);
21    s->AddElement(mng);
22    TEveProjectionAxes* axes = new TEveProjectionAxes(mng);
23    axes->SetTitle("TEveProjections demo");
24    s->AddElement(axes);
25    gEve->AddToListTree(axes, kTRUE);
26    gEve->AddToListTree(mng, kTRUE);
27
28    //hit point
29    int npoints = m_clusters.size();
30     cout<<"npoints = " <<npoints<<endl;
31     TEvePointSet* ps = new TEvePointSet(npoints);
32     ps->SetOwnIds(kTRUE);
33
34     cout<<"new TEvePointSet"<<endl;
35
36     int counter  = 0;
37     for(auto itr = m_clusters.begin(); itr != m_clusters.end();++itr ){
38       auto cluster = itr[0];
39       ps->SetNextPoint(cluster[0], cluster[1], cluster[2]);
40       ps->SetPointId(new TNamed(Form("Point %d", counter), ""));
41       counter ++ ;
```

```
42          cout<<"itr = "<<*itr<<endl;
43          cout<<"cluster[0] = "<<cluster[0]<<"   cluster[1] = "<<cluster[1]<<"
44          cluster[2] = "<<cluster[2]<<endl;
45        }
46
47        ps->SetMarkerColor(2);
48        ps->SetMarkerSize(1.0);
49        ps->SetMarkerStyle(4);
50        gEve->AddElement(ps);
51
52        geom->CanEditMainColor();
53
54        gEve->Redraw3D(kTRUE);
55    }
```

C.3 Event Display のソースコード全文へのリンク

https://github.com/sPHENIX-Collaboration/InttEventDisplay/tree/main