京都教育大学
令和 2 年度 卒業論文

Windows 環境における Geant4 利用法の確立

2022 年 2 月 13 日

京都教育大学 理科教育専攻
学籍番号 181204

小山 繁継

序論

0.1 はじめに

福島第一原発事故や、カミオカンデでのニュートリノ振動の発見、CERN のアトラス実験によるヒッグス粒子の検出などにより、国内では放射線に対する注目が高まりつつある。しかし、放射線は目には見えないものであるためか、放射性物質から出る人体に有害のものであるという漠然としたイメージのみが先行している。放射線の正しい特徴・性質や、安全で効率的な利用法が存在することなど、放射線に関する正しい理解が求められている。

また、CERN の研究で用いるために開発された World Wide Web が一般にも広く普及した今日では、IT の発達が著しい。経済産業省がみずほ情報総研に委託して行った調査の結果によると、2030 年には最大 79 万人の IT 人材が不足する見通しであり [1]、更に学校教育でも文部科学省が GIGA スクール構想を打ち出すなど、プログラミング教育の推進も行われている状況である。

0.2 背景

放射線に対する関心が高まり、プログラミング教育が推進される中、2021 年に本校で行われた免許更新講習では、素粒子衝突シミュレーションソフトの Geant4 を用いて簡単なシミュレーションを行うというものだった。素粒子物理学界では、最もメジャーなシミュレーション方法として、Linux 系列の OS 搭載の PC において Geant4 を用いている事が多い。そのため、本講習でも Linux のディストリビューションである Ubuntu が搭載された PC で演習が行われた。なお、Geant4 の動作は、視覚的にわかりやすく表示させるため、Qt というアプリケーション開発フレームワークを用いて視覚化していた。しかし、Linux 系列の OS では、真っ黒な画面の CUI にコマンドを打ち込む操作を多用するため、プログラミング未経験者にとってはかなり難易度が高かったようである。

実際、私自身も VBA で簡単なエクセルのマクロを作ったぐらいのプログラミング経験しかなかったので、Geant4 の導入だけでも時間がかかった。更に、デフォルトで Linux 系列の OS がインストールされている個人用 PC は一般的ではなく、個人で Linux 系列の OS 搭載の PC を用意するのも難しいため、総合的に勘案してこの免許更新講習内容は、学校現場において実施するには難易度が高いと言える。よって、今日の素粒子物理学事情とプログラミング教育事情の両方の需要を満たすのには適していないと感じた。

0.3　目的

前述の 0.1 節や 0.2 節を承けて、**素粒子物理学では最もメジャーに用いられているシミュレーションソフトである Geant4 を、Qt というアプリケーション開発フレームワークを用いてわかりやすく視覚化し、それらの動作を最もユーザーの多い OS である Windows 環境で動作させることができる方法を確立し、日本の素粒子物理学とプログラミング双方の学びの一助となることを本研究の目的とする。コマンドの意味や用語を解説を逐次記述するので、この論文を見ながら作業するだけで、Geant4 を導入して用いるのに最低限の知識を得ることができ、Windows マシンで Geant4 が導入できるようになるように執筆する。**

想定する活用場面は、主に、高校物理の教員が、原子分野の内容を視覚的に生徒らに示してより身近に考えさせたいというときや、大学生らが、素粒子物理学に関する講義で学んだ内容を実際にシミュレートしたいというときなどが考えられる。このような場面で、本論文を読みながら作業することで、プログラミングに疎い方でも最低限の知識を得て、シミュレーションができるようになることを想定する。

0.4　Geant4 について

Geant4 とは、主に素粒子衝突をシミュレーションできる様々なライブラリで構成されているツールキットである。このソフトウェアは、物質中を通過する素粒子の飛跡や反応をシミュレーションすることができるため、主に高エネルギー物理学や、原子核実験や放射線治療の分野で用いられている。その他にも、放射性物質の崩壊の様子などもシミュレーションすることができる。また、オブジェクト指向プログラミング言語である C ＋＋を用いて作成されているので、ソフトウェアを構成するすべての仕組みを理解せずとも、必要な部分だけを書き換えるだけで実行できるため、比較的簡単に扱うことができる。また、Windows で一般的に用いられるアプリケーションは、主にホームページからダウンロードしたインストーラーを起動すれば自動的に必要なコンポーネントがダウンロードされ、質問形式のウィザードを進めていけば難なくインストールが完了し、すぐにアプリケーションを使うことができる。一方、Geant4 は、ソースファイル形式で配布されているため、シェルスクリプトやコマンドプロンプトなどでコマンドを用いてコンパイルやビルドを行う必要がある。どの OS でも Geant4 の動作は確認されているが、導入の一連の流れを行うに当たっては、Ubuntu などの Linux 系列の OS 上での操作が最も容易なため、Ubuntu 上で用いられることが多い印象である。Geant4 はコマンドを入力して動作させるが、その結果をビジュアライゼーション (可視化して表示) させるためには、グラフィックライブラリを用いる必要がある。主に用いられるものとしては、OpenGL か DirectX だが、今回は両方のライブラリに対応している Qt(キュート) というフレームワークを用いる。

0.5 Qt（キュート）について

Qt とは、クラスプラットフォーム対応のアプリ開発フレームワークである。Qt は、Linux、Windows、MacOS、Android、iOS など多くの OS 上で動作させることができ、開発したアプリのソースコードをそれぞれの OS 向けに書き直さずとも、動作させる OS に適したコンパイルをやり直すだけで同様の機能の動作が可能という互換性を持っている。プログラム自体は C ＋＋で記述されている。Qt では GUI を表示するのに OpenGL と DirectX の両方がサポートされている。今回は主に OpenGL のグラフィックライブラリを用いている。Qt の身近な導入例としては GoogleEarth が挙げられる。本研究では、この Qt のフレームワーク上で Geant4 を動作させる。

目次

1 序論

1.1 予備知識

本論文ではコンピューターに関する専門用語を用いる。そこで、PC でオフィスソフトやブラウザなどの操作やファイル操作を行ったことはあるが、コマンド操作や細かい設定まではしたことがない人、Linux の使い方を知らない人など、PC を普段遣いで最低限しか利用したことがないという人でも理解できるように、予めこのセクションで専門用語やシステムなどの仕組みの解説を行う。できるだけ網羅するように留意しているが、不明点があれば各自で調べると言ったスキルも必要だろう。また、コマンドやプログラムは英語をもとにしているので、用語は英語も共に覚えるようにしておくと理解しやすいと思われる。

1.1.1 Windows とは

Windows とは、Microsoft 社が開発している、主にラップトップ、デスクトップ向けの OS である。一般的に PC といえば Windows が搭載されている事が多く、読者の殆どが一度は利用したことがあると思われる。様々なバージョンが提供されてきており、現在の最新バージョンは Windows11 であるが、最も普及しているのは執筆段階で Windows10 である。1995 年にリリースされた Windows95 より前のバージョンでは、黒い画面に文字を打って操作する CUI（Character User Interface）という方式だったが、それ以降は現在の一般的な PC のようにグラフィカルに情報が描画される GUI（Graphical User Interface）という方式が採用されるようになり、直感的に操作が可能となった。また、Windows には、ユーザーの使い方に応じて主に 2 つのエディションが用意されている。一般ユーザー向けには Home。ヘビーユーザーやネットワーク機能を使うユーザー向けには Pro が用意されているが、両者に大きな差はないと考えて良い。更に、一度の情報を扱う量が多い 64bit 版と、その半分の 32bit 版の 2 種類があるが、現在の Windows10 の主流は 64bit である。手持ちの PC に搭載されているこれらの Windows の情報は, 設定の「システム」の「詳細情報」から確認が可能である。アプリによっては特定の Windows バージョンにしか対応していないものがあるので確認しておくとよいだろう。

1.1.2 Linux とは

Linux とは、リーナスによって開発されたオープンソースの OS である。また、その OS をエンジニアが改良し、新たに開発された OS も含めて Linux と呼ぶことが多い。こういった Linux をベースにした OS を Linux のディストリビューションと言う。今回の研究で用いる Ubuntu も Linux のディストリビューションの一つである。Linux は、一般的にはシステム開発やサーバー運用向けに用いられることが多い。また、PC を購入する際には既に Windows が搭載されているので知らない人も多いだろうが、Windows は正規で用いるための権利であるライセンス料は有料である。一方、Linux は無料で、ソースコードも公開されている。操作方法としては、近年の Windows と同様にグラフィカルに情報が描画される GUI を採用しているので、直感的に操作が可能である。しかし、アプリケーションは Linux 向けに開発されたものが圧倒的に少ないことに注意。

1.1.3 OS とは

OS とは、Windows や Linux、Android、iOS など、コンピューターを動作させるために必要な重要なソフトウェアのこと。Operating System の略で、コンピューターに接続された CPU やメモリなどのあらゆる機器の管理や制御を行う基幹を担う。後述のカーネルやシェルなどの様々なソフトウェアやライブラリで構成されている。電源が入るとマザーボードが一番最初に起動し、次に OS が起動する。電源が切られるまで起動し続けて処理をし続ける。基本ソフトウェアとも言われ、OS 上で動かすアプリケーションは応用ソフトウェアとも言われる。コンピューターのハードウェアの仕様は数多くあるが、細かい違いによるエラーを起こさず応用ソフトウェアとハードウェアとのやり取りがスムーズに行えるのも OS のおかげである。この性質によって、ある OS 向けに開発された応用ソフトウェアは、その OS が搭載されていれば多くのハードウェアでも動作させることができる。

1.1.4 カーネル（kernel）とは

カーネルとは、OS の中でも中核を担う部分のことを指す。カーネルは動作している全てのソフトウェアの処理や状態を管理し、それらの処理に応じたハードウェアの制御、管理も行う。コンピューター上で応用ソフトウェアが動作するときには、その処理を行うためのメモリや CPU の領域をカーネルに割り当てて貰う必要がある。その際に応用ソフトウェアからカーネルに命令を送るための手段は予め決められており、その命令をシステムコールと言う。

1.1.5　シェル（shell）とは

シェルとは、OS を構成するソフトウェアの一つである。カーネルを包む殻（shell）のように捉えられることが多い。ユーザーがターミナルという種類のソフトウェアにコマンドを入力して直接コンピューターを操作しようとするとき、カーネルは 0 か 1 の 2 進数を使い、ユーザーは英語をもとにするプログラミング言語を使うため、それぞれの情報伝達にはそれぞれの言語を翻訳できる存在が必要である。そこで、ユーザーの入力したコマンドの意味を読み解き、それをカーネルに伝えたり、カーネルからの返答をユーザーに言語化して伝えたりしてカーネルとユーザーの仲介を行うのがシェルである。

なお、ターミナルというソフトウェアには、例えば Windows では「PowerShell」や「コマンドプロンプト」、Linux では図 1 の「端末」等がある。

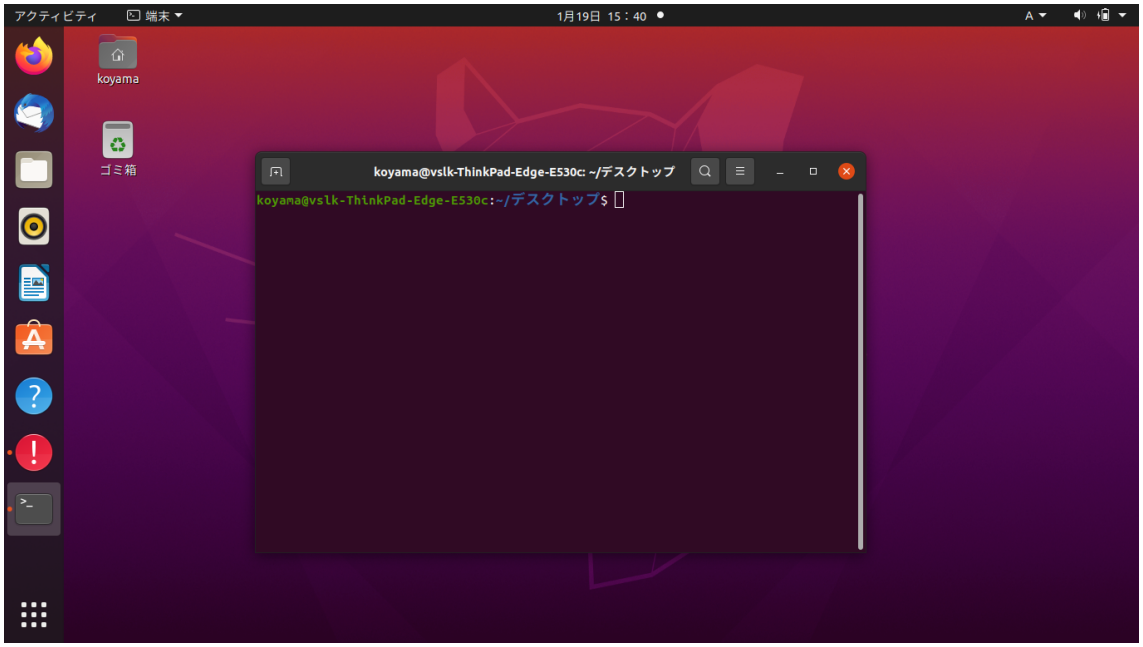

図 1　Linux のターミナル「端末」を開いたデスクトップ

1.1.6 コマンド（command）とは

コマンドとは、主に、ユーザーからコンピューターへ命令を行うときに使用される文字列のこと。基本的には半角英数字と記号で記述される。コマンドプロンプトなどのターミナルを起動し、黒い画面に予め定められた文字列を入力することで、シェルがコマンドを読み解き、カーネルに命令の実行をさせる。コマンドは基本的に英語をもとにする文字列で成り立っており、コマンド文の中には、操作の対象物の指定や、処理の詳細の記述、その他の様々なオプションを記述することができる。例えば、Linux のディレクトリ（フォルダのこと）の中身をまるごと他のディレクトリにコピーしたいとき、入力するコマンドの例は次のコマンド 1 のように記述できる。

コマンド 1　Linux でフォルダの中身をまるごとコピーするコマンド

```
1 cp -r /gazou /syasin
```

このコマンド文は、「cp」がコピーを実行させるコマンド、「-r」がフォルダ丸ごとをコピーするように指定するオプション、コピー元は「gazou」というディレクトリを指定し、コピー先を「syasin」に指定するという仕組みになっている。このコマンドを入力してからエンターキーを押すとコマンドが実行され、その結果、「gazou」というディレクトリが「syasin」というディレクトリの中に丸ごとコピーされる。なお、このコマンド文の順番は入れ替えられない。

このように、どのような処理をするかのコマンドを最初に記述し、オプションを指定し、操作対象を指定することによって殆どのコマンドが扱える。コマンドと各オプションや操作対象の記述の間には区切りのための半角スペースが必要である。また、大文字と小文字は区別するので注意が必要である。コマンドによって操作対象の指定方法や使えるオプションは異なるので、慣れないうちは調べながら用いると良いだろう。なお、コマンド文はエンターキーを押した時点で実行されてしまうので、一行で記述することになる。このとき、もしコマンド文が画面の端で折り返してしまっても、内部処理としては一行扱いなので気にする必要はない。コマンドで操作を行うときに注意するべきなのは、現在のディレクトリ（カレントディレクトリ）についてである。ターミナルでは、Windows のファイル閲覧・管理ソフトウェアのエクスプローラーのように、基本的には自分が現在操作対象としているディレクトリしか操作ができない。例えば、今のディレクトリが「`C:\Users\koyama\Music`」の場合、別のディレクトリ「`C:\Users\koyama\Documents`」のディレクトリを操作することは原則できない。

本研究ではコマンドを多用する。

1.1.7 ディレクトリ（directory）とは

ディレクトリとは、ストレージに複数のファイルを格納し、整理することができる場所のことを指す。Windows で一般的に用いられるフォルダの概念とほとんど同じである。HDD や SSD などのストレージを頂点とし、その中に新たなディレクトリが存在し、それらのディレクトリの中に更にディレクトリが存在するという、入れ子構造になっている。ディレクトリ全体を木に見立ててディレクトリツリーということもある。

コマンド内でディレクトリやファイルを指定する際は、絶対パスと相対パスの 2 種類の記述方法がある。絶対パスは、「`C:\Users\koyama\Music`」のように、C ドライブの中の Users ディレクトリの中の koyama というディレクトリの中の Music というディレクトリを指定する。ディレクトリツリーの頂点から一つずつ階

層を降りていって目的のディレクトリを指定するのである。対する相対パスは、カレントディレクトリにあるディレクトリやファイルしか指定できないが、カレントディレクトリまでの記述を省略できる。例えば、カレントディレクトリが「C:\Users\koyama」の場合、「C:\Users\koyama\Music」を相対パスで指定したいとき、「\Music」のように指定できる。また、相対パスでカレントディレクトリのひとつ上の階層を指定したい場合は、ドットマーク 2 つで「..\」と記述する。カレントディレクトリを指定するときはドットマーク 1 つで「.\」と記述する。慣れないうちは絶対パスで指定するとわかりやすいだろう。なお、ディレクトリの区切りには、Linux はスラッシュ「/」を用い、Windows ではバックスラッシュ「\」または円マーク「¥」を用いる。

1.1.8 ソースファイル（source file）とは

ソースファイルとは、何らかのプログラミング言語で書かれた、何らかの処理を行うことを目的としたプログラムのコードを保存したファイルのこと。平たく言えば、ソフトウェアの設計図である。通常、コンピューターは電気信号によって 0 か 1 かの 2 進数で動作しているが、人間が読んだり書いたりしやすい形でプログラムを作ることができるように、なんらかのプログラミング言語でプログラムを記述する。黒い画面に色々なコードを打ち込んでソフトウェアなどの開発をしている場面では、ソースファイルを作成しているということである。

1.1.9 コンパイル（compile）とは

コンパイルとは、コンパイラというソフトウェアがソースファイルに記述された内容を読み取って、2 進数のコンピュータが理解できるデータ（バイナリデータ）に変換して、実行できる形（オブジェクトコード）にする作業のこと。

1.1.10 ビルド（build）とは

ビルドとは、記述されたソースファイルの内容を読み取って、動作の検証を行ってから問題なければコンパイルを行う一連の作業のことである。

1.1.11 ライブラリ（library）とは

ライブラリとは、ある汎用的なプログラムを他の様々なソフトウェアでも利用できるように部品化し、集めたファイルのこと。一般的には、ライブラリのファイルそのものは単体で実行は不可能で、他の実行ファイルとともに利用される。ライブラリには静的ライブラリと動的ライブラリの 2 種類がある。静的ライブラリは、スタティックリンクライブラリと言い、ソースコードのコンパイル時に実行ファイルに組み込むものである。動的ライブラリはダイナミックリンクライブラリと言い、実行ファイルが実行されるときに、必要になった場合に読み込むものである。

1.1.12 環境変数（environment variables）とは

環境変数とは、他のプログラムからコンピューター固有の様々な変数を参照することができるように、OSが保存している設定値のこと。環境変数は「PROCESSOR_IDENTIFIER=AMD64」のように、「変数名（アルファベット大文字＋記号）＝値」の形でセットで保存されており、ソフトウェアがOSに環境変数を変数名で要求すると、OSがその変数名に対応する値を応答する。基本的にはOSが自動的に値を設定するが、ターミナルからユーザーが任意の環境変数を設定することもできる。Windowsではコマンドプロンプトで「set」と入力してエンターキーを押すと、OSに保存されている全ての環境変数が表示される。変数名を手動でセットする場合はターミナルよりもWindowsの設定画面から「環境変数」を検索して、「システム環境変数の編集」に移動し、環境変数一覧から編集するほうが簡単である。

1.1.13 PATH（パス）とは

PATHとは、環境変数の一つ。1.1.7項で述べた絶対パス、相対パスに通じるところがある。通常は、ソフトウェアの名前だけではそのソフトウェアが存在するディレクトリの場所がわからないため、絶対パスか、ソフトウェアが存在するディレクトリに移動してから相対パスで指定する必要があるが、PATHを設定すると、ソフトウェアの名前を指定しただけでそのソフトウェアのディレクトリを特定し、ソフトウェアを起動することができる。PATHは、何らかのソフトウェアが絶対パスではなく、名前だけで指定されたときに絶対パスを補うための値を格納している。設定例としては、「Path=C:\Windows\system32;C:\Windows;C:\Windows\system32\Wbem」といった形で、複数の値がコロンで区切られて表されている。

例えば、電卓をターミナルから起動するとき、電卓の実行ファイルの絶対パスは「C:\Windows\System32\calc.exe」である。ここで、「calc.exe」と入力すると、環境変数のPATHを呼び出し、応答した全ての値を順番に当てはめていき、calc.exeが存在するディレクトリ「Path=C:\Windows\system32」を見つけて実行するのである。なお、PATHは様々なアプリを集めたフォルダに対して設定される。ソフトウェア名が指定されたときにこれらのパスを順番にソフトウェア名に当てはめていき、ソフトウェアが見つけられるまで試すのである。

1.2　よく使うコマンド

本実験では、ターミナルなどに様々なコマンドを記述して実行する。本論文に記載するコードをそのまま入力してもいいが、コードの実行内容や記述方法を理解しておくと応用が効くだろう。また、Linux の「端末」と Windows の「Developer Command Prompt VS 2019」とでは使えるコマンドが異なる点があるので注意。本研究では主に「端末」と「Developer Command Prompt VS 2019」を用いる。

1.2.1　ls（list segments）

「ls（エルエス）」は、ターミナルのカレントディレクトリに含まれるファイルやディレクトリを表示させるコマンドである。例えば、カレントディレクトリが「`C:\Users\shige`」のときに単に「ls」とだけ入力すると、図 2 のようにディレクトリが表示される。

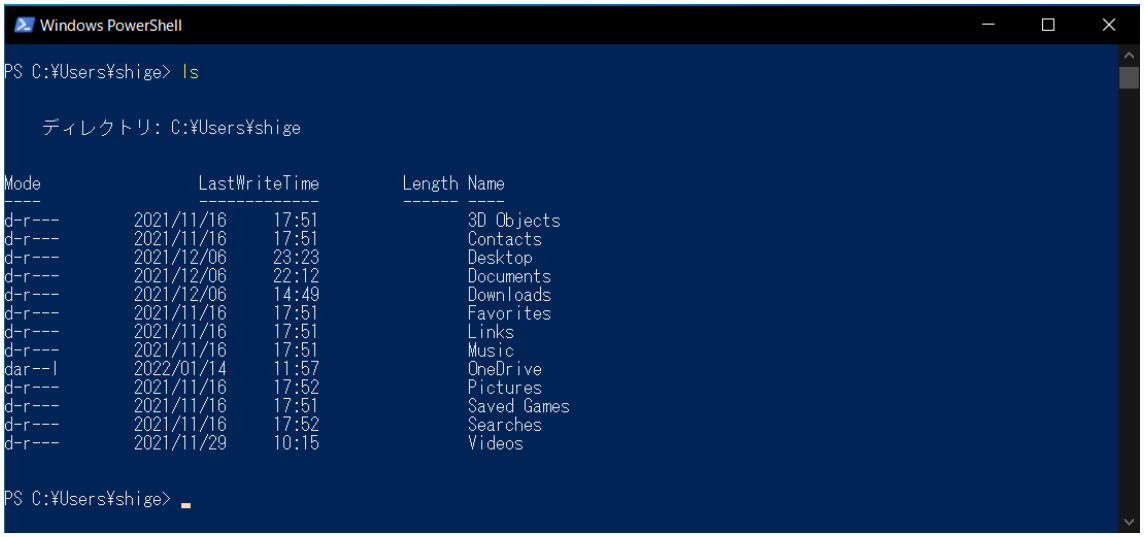

図 2　Linux の「端末」でコマンドを実行した画面

1.2.2　cd（change directory）

「cd（シーディー）」は、ターミナルのカレントディレクトリを変更するコマンドである。コマンドの入力方法は、「`cd C:\Users\shige\documents`」のように、cd の直後に移動したいディレクトリの絶対パスを入力するか、カレントディレクトリからの相対パスを入力すればよい。

1.2.3 sudo（superuser do）

「sudo（スードゥ）」は、主に管理者の権限で次に記述するコマンドを実行するコマンドである。記述方法は、「`sudo cd C:\Users\yamada\documents`」のように、sudo の直後に実行したいコマンドをそのまま入力するだけである。このコマンドを実行したときは、管理者アカウントのログインパスワードを入力しなくてはならない。

1.2.4 apt（advanced package tool）

「apt（アプト、エーピーティー）」は、インターネット上に公開されているソフトウェアを自動的にダウンロードし、インストールまで行う、アプリケーションパッケージツールを利用するコマンドである。記述方法は、「`apt install qt5`」のように、apt コマンドの直後にオプションの「install」を入力し、ソフトウェアの名前「qt5」を入力する。このコマンドで指定するソフトウェア名は複数指定可能であり、半角スペースで区切って記述する。

1.2.5 mkdir（make directory）

「mkdir」は、カレントディレクトリの中に新たにディレクトリを作成するコマンドである。記述方法は、「`mkdir atarasiiforuda`」のように、mkdir コマンドの直後に作成ディレクトリの名前を指定する。作成ディレクトリの名前は、「`mkdir C:\Users\shige\documents\atarasiiforuda`」のように、絶対パスか相対パスで指定することによって、カレントディレクトリ以外にもディレクトリが作成が可能である。

1.2.6 chown（change owner）

「chown（チェンジオーナー）」は、ファイルの所有者を変更するコマンドである。記述方法は、「`chown koyama C:\Users\yamada\documents\diary`」のように、chown コマンドの直後に変更後の所有者名を記述し、その次にファイルのディレクトリを絶対パスで指定する。空白で区切ると複数のファイルが指定可能である。このコマンドは管理者でしか実行できないことが多いので、sudo コマンドと併用するとよい。

1.2.7 cmake

「cmake（シーメイク）」は、CMake というフリーソフトウェアを用いてビルドを自動で行うコマンドである。記述方法は、「`cmake -DCMAKE_INSTALL_PREFIX=/opt/geant4/geant4.10.06.p02-install -DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_QT=ON /opt/geant4/geant4.10.06.p02/`」のように、cmake コマンドの直後に様々なオプションを記述してから、ビルド対象にするソースファイルが格納されたディレクトリを指定する。この例に記述したオプションは、「`-DCMAKE_INSTALL_PREFIX=`」がインストール先のディレクトリを指定し、「`DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_QT=ON`」でオプションを指定している。このコマンドによってビルド専用のファイルが作成される、Windows の場合は「`cmake --build . --config Release`」のように、もう一度 cmake コマンドを実行すると、ビルドが行われ、更に「`cmake --build . --config Release --target install`」と再三 cmake コマンドを実行すると、ソフトウェアがインストールされる。Linux の場合は、ビルド専用ファイルの作成は cmake で行うが、ビルドとインストールは「cmake」ではなく、「make」というコマンドを用いる。なお、インストールではなく、実行ファイルを実行するだけのときは、その実行ファイル名を指定するだけで良い。cmake コマンドを用いるためには、「CMake」というソフトウェアをマシンにインストールする必要がある。

1.2.8 tar

「tar」は、複数のファイルをまとめて一つのアーカイブファイルにしたり、逆にアーカイブファイルを展開したりするコマンドである。記述方法は、「`tar zxvf geant4.10.06.p02.tar.gz`」のように、tar コマンドの直後にオプションを指定し、その次に対象のファイルを指定する。圧縮するか展開するかはオプションで指定でき、「zxvf」が展開、「zcvf」が圧縮である。圧縮または展開したディレクトリは、カレントディレクトリ直下に作成される。

1.3　実験環境

今回の実験を行うにあたって使用したのは、所属研究室が所有する 2 台のノートパソコンである。

まず、予備実験として Linux 系列の OS 搭載の PC が必要だったが、それには図 3 で示すノートパソコンを用いた。レノボ社製 ThinkPad Edge-E530c に Linux のディストリビューションの一つである Ubuntu20.04.2 LTS の 64 ビットをインストールしたマシンである。搭載 CPU はインテルの Core i5-3230M で、メモリ容量は 3.4GB で、GPU は Intel HD Graphics4000、ディスク容量は 250GB であった。

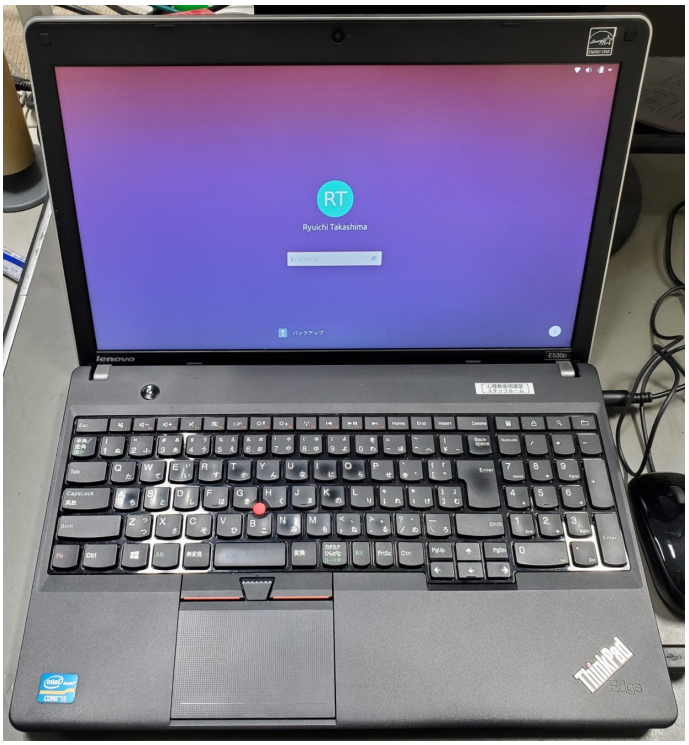

図 3　LinuxPC の外観

本実験で用いた Windows 搭載 PC は、図 4 で示すノートパソコンを用いた。NEC 社製の LaVie に Windows10 Pro バージョン 21H1 の 64 ビットをインストールしたマシンである。搭載 CPU はインテルの Core i7-3517U で、メモリ容量は 4.0GB で、GPU は Intel HD Graphics4000、ディスク容量は 120GB であった。

図 4　WindowsPC の外観

2　実験内容

このセクションでは、実際に私が行った操作とその結果の両方だけでなく、1.2 節で解説しなかったコマンドの大まかな意味や注意点、専門用語の解説や仕組みなども記述していく。コマンドプロンプトなどのターミナルなどに入力したコマンドは、使用する場面で逐一、そのまま記述していく。

2.1　Ubuntu での動作確認

まず、Windows での Geant4 の動作を行う前に、Linux 系列の OS でも動作させることができるかを確認した。ここで用いたコマンドは Qiita[2] を参考にした。

2.1.1　必要なソフトのインストール

Ubuntu では、Geant4 はソースファイルという形式で配布されているため、ソースファイルからビルドという作業を行ってコンピュータ上で処理できる状態にする必要がある。ビルドのためには、cmake という自動ビルドソフトが必要なため、このタイミングでインストールした。cmake と Qt は両方、Linux 標準のアプリケーション管理システムである APT を用いてインストールすることができる。APT 自体は初期状態の Ubuntu にすでにインストールされているので、下準備なしに利用が可能である。実際に cmake と Qt をインストールする方法は、Ubuntu のデスクトップ画面を右クリックして”端末”を選び、表示された黒い画面に以下のコマンド 2 を入力するだけである。

コマンド 2　cmake と Qt をインストールするコマンド

```
1 sudo apt install cmake qt5-default
```

コマンド 2 の意味としては、「sudo」で管理者の権限でコマンドを実行する。「apt install」で、APT を用いてソフトウェアをネット上から探してインストールする。という意味である。ここではインストールするソフトウェアを cmake とデフォルトの Qt5 の両方を一度に指定している。

以上の操作によって cmake と Qt がインストールされた。

2.1.2 Geant4 のインストール

Geant4 のソースファイルを Geant4 の公式サイト [3] からダウンローした。まず、Ubuntu に標準でインストールされている Firefox を起動し、サイトにアクセスし、「Source files」の欄の「GNU or Linux tar format…」と記載されている方のファイルをダウンロードした。

次に、ソースファイルを以下のコマンド 3 で展開した。

コマンド 3　ソースファイルを展開するコマンド

```
1 sudo mkdir /opt/geant4
2 sudo chown koyama /opt/geant4
3 cd /opt/geant4
4 mv ~/Download/geant4.10.07.p02.tar.gz .
5 tar zxvf geant4.10.07.p02.tar.gz
```

コマンド 3 に記述したコマンドは 1 行ごとにエンターキーを押す必要があることに注意。1 行目のコマンドで Geant4 のインストール用のディレクトリを作成し、2 行目のコマンドで所有権を変更する。4 行目のコマンドでダウンロードしたファイルを 1 行目で作成したディレクトリに移動し、5 行目のコマンドでファイルを展開する。

次に、展開したファイルを以下のコマンド 4 でコンパイルし、インストールした。

コマンド 4　ファイルをコンパイルするコマンド

```
1 mkdir geant4.10.07.p02-build
2 cd geant4.10.07.p02-build
3 cmake -DCMAKE_INSTALL_PREFIX=/opt/geant4/geant4.10.07.p02-install -DGEANT4_INSTALL_DATA=
      ON -DGEANT4_USE_QT=ON /opt/geant4/geant4.10.07.p02/
4 make -j8
5 make install
```

1 行目のコマンドで、コンパイルしたオブジェクトコードを保存するためのディレクトリを作成し、3 行目のコマンドで様々なオプションを指定してコンパイル、5 行目のコマンドでインストールを行う。

2.1.3 サンプルの実行

Geant4 がインストールされたので、実際にサンプルをコマンド 5 で実行した。

コマンド 5　サンプルを実行するコマンド

```
1 mkdir g4work
2 cd g4work
3 cp -r /opt/geant4/pro/share/Geant4-10.7.2/examples/basic/B1 .
4 mkdir B1-build
5 cd B1-build
6 cmake ../B1
7 make
8 ./exampleB1
```

1 行目でサンプル用のディレクトリを作成し、3 行目のコマンドで Geant4 のソースコード内のサンプルをコピーする。4 行目のコマンドでビルドしたオブジェクトコード用のディレクトリを作成し、6、7 行目のコマンドでビルドする。8 行目のコマンドで実行ファイルを指定し、Qt を有効にした Geant4 でサンプルを起動する。

すると、以下の図 5 のような画面が表示された。水に満たされた空間内に骨を模した 2 つの物体が存在するというサンプルである。

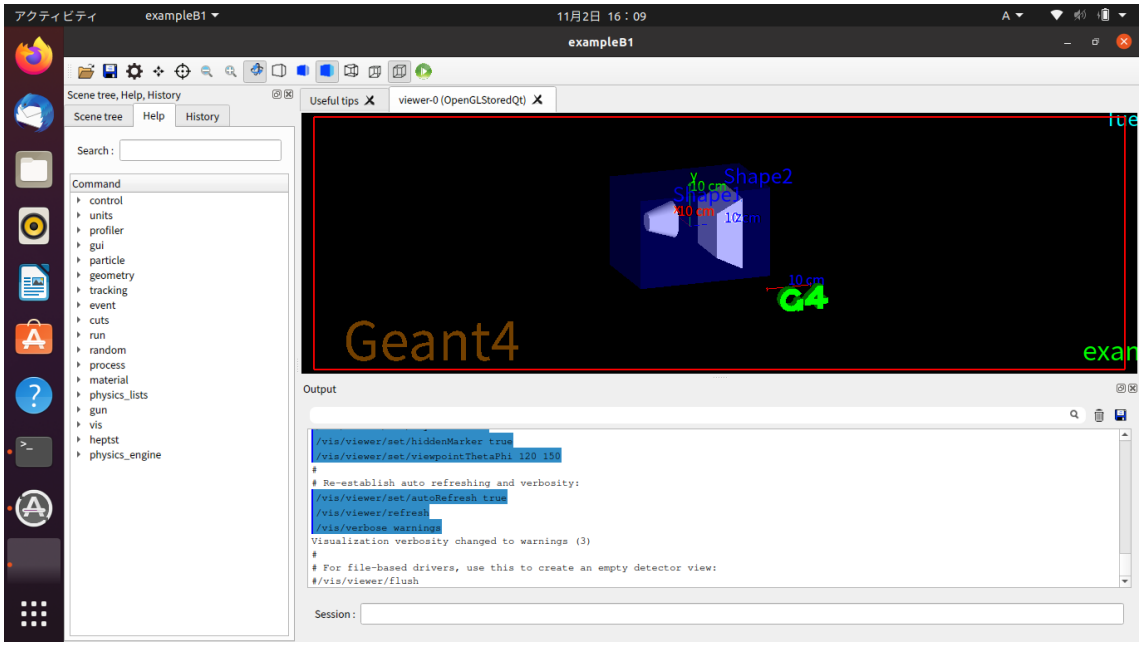

図 5　Linux 上で Qt を有効にした Geant4 でサンプルを起動した画面

これで Linux での Qt を有効にした Geant4 を起動することに成功した。以下のコマンド 6 をセッションボックスに入力して、正しく動作するかを確かめた。

コマンド 6　サンプルに光子を照射するコマンド

```
1 /gun/particle gamma
2 /gun/energy 100 MeV
3 /run/beamOn 10
```

すると、以下の図 6 のように、透過したり反射したりした光子の軌跡が表示された。

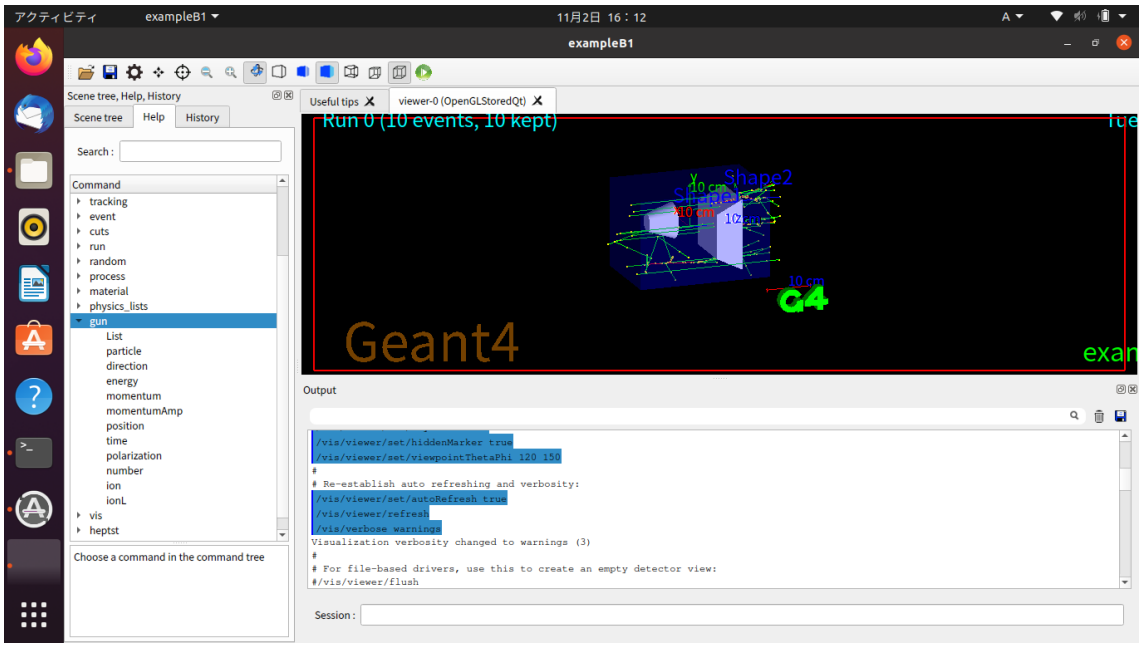

図 6　Linux 上で Qt を有効にした Geant4 で起動したサンプルに光子を照射した画面

なお、この画面はマウス操作で図 7 のように視点を滑らかに移動させることができた。

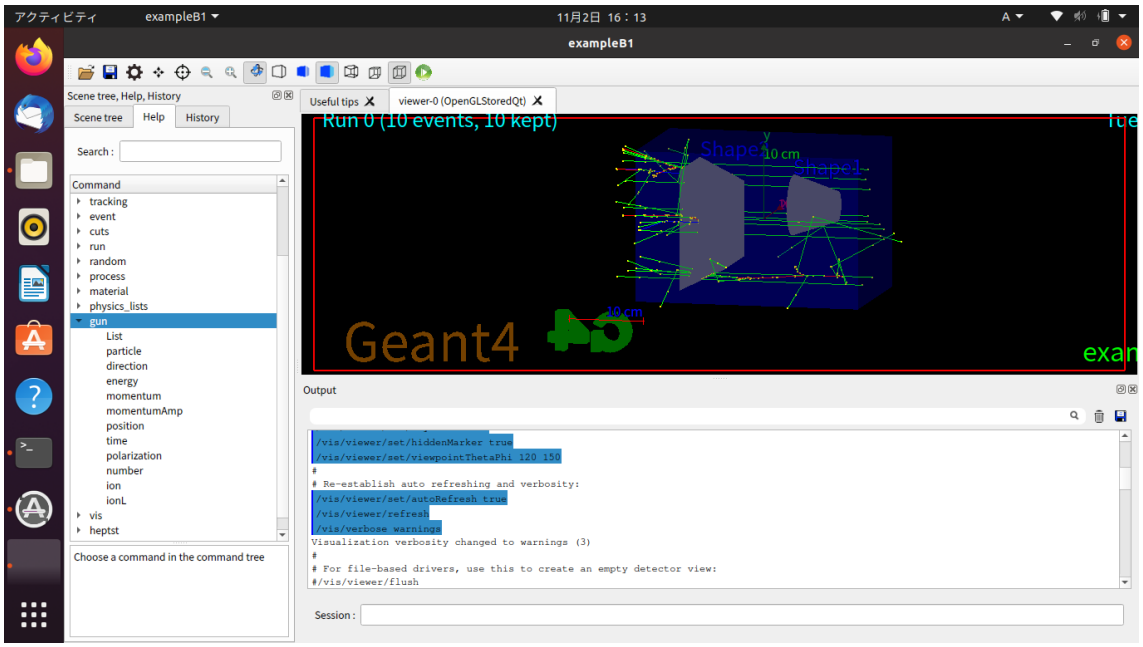

図 7　図 6 の視点を移動した画面

2.2　Windows での導入

Linux での Qt を有効にした Geant4 が動作したので、次は Windows マシン上でも実現できるかを試した。Windows では、Linux のように APT で Qt を入手するのではなく、公式サイトからインストーラーをダウンロードしてインストールする必要があった。また、ビルドを行う際に、Microsoft Visual Studio 2019 が必要なので、これも事前にインストールする必要があった。MinGW や MicrosoftC++ コンパイラでの動作は保証されていないようである。これらの情報は Geant4 ホームページのインストールガイド [5] や Qiita[6] を参考にしている。

2.2.1　Microsoft Visual Studio 2019 Comunity のインストール

ソースファイルを Windows 向けにコンパイルする際には、VisualStudio に組み込まれている CMake のコマンドラインツールを使用することが推奨されている [5] ので、本実験では、「Microsoft Visual Studio 2019 Comunity」をインストールし、「Developer Command Prompt VS 2019」をターミナルとして使用した。

「Microsoft Visual Studio 2019 Comunity」のインストーラーを公式サイト [7] からダウンロードし、起動した。画面の指示に従って進めていくと、途中でインストールするコンポーネントを選択する画面が表示され

たので、ここでは「C++ によるデスクトップ開発」を選択してインストールした。インストールが完了したら、Windows アプリケーションの一覧に「Developer Command Prompt VS 2019」が追加されていることを確認した。

2.2.2　Qt のインストール

ブラウザで Qt の公式サイト [4] にアクセスし、オンラインインストーラーをダウンロードし、起動した。すると、インストーラーが起動し、図 8 のような画面が表示されるので、画面の表示に従って操作を行った。なお、ここでは Qt のアカウントを作成することが求められたので、Web メールのアドレスが必要であった。

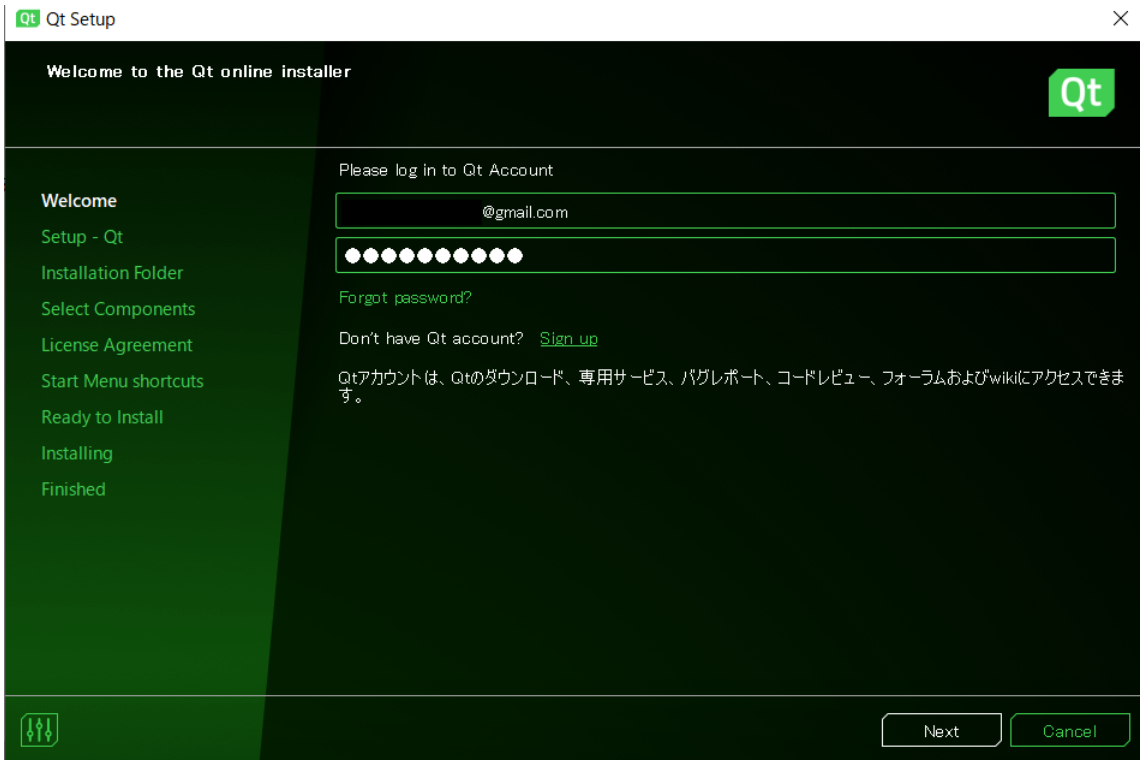

図 8　Qt のオンラインインストーラーを起動した画面

インストーラーの操作を進めていき、「Insatllation Folder」の項目で、「Custom Installation」を選択した。ここでこのオプションを選択することで、次の画面でインストールするパッケージを選択することができる。次の画面では、図 fig:QtInstallpack で示すように、「Qt 5.15.2」のツリーの中から「MSVC 2021 64-bit」と、「Qt」から始まるパッケージをすべて選択した。

図 9　Qt でインストールするパッケージを選択した画面

オンラインインストーラーのウィザードをそのまま進めていき、インストールを終了させた。なお、この処理には 1 時間程度を要した。

2.2.3　Qt のパスの設定

次に、Qt のオブジェクトコードが、Geant4 のビルドのときに正しく参照できるようにするために、環境変数の 1 つである「Path」を設定する。

Geant4 をビルドするときには、Qt のライブラリを参照して、その内容を組み込んだオブジェクトコードを作成するが、そのときには、コンパイラーが、Qt のオブジェクトコードの格納されている場所を見つける必要がある。そのフォルダまでの道筋を示すための環境変数が、「Path」である。

操作方法としては、Windows の設定を開き、「システム」、「詳細情報」、「システムの詳細設定」、「環境変数」と移動して、「システムの環境変数」の一覧の「Path」をダブルクリックして、「新規」から Qt の MSVC 用のバイナリが格納されているフォルダの絶対パス「`C:\Qt\5.15.2\msvc2019_64.bin`」を追加した。

2.2.4 Geant4 のインストール

まず、Geant4 の公式サイト [3] の「Source files」から Zip ファイルをダウンロードし、Geant4 をインストールするための作業フォルダを任意の場所に作成した。本研究では、「C:\Users\shige\Documents\Geant4_BaseDir」を作成した。エクスプローラーで Zip ファイルをダブルクリックするか、ブラウザのダウンロードダイアログからファイルをクリックして、Windows 標準の機能で先ほど作成したインストール用の作業フォルダに解凍すると、「Geant4_10_07_p03」というフォルダができていることを確認した。このフォルダに Geant4 ソースファイルが格納されている。Geant4 インストール用の作業フォルダにビルド用フォルダ「C:\Users\shige\Documents\ Geant4_BaseDir\geant4-build」とインストール用フォルダ「C:\Users\shige\Documents\ Geant4_BaseDir\geant4-install」を作成した。Windows のデスクトップ画面左下の検索ボックスから「Developer Command Prompt VS 2019」を検索して、管理者として実行し、ターミナルを開いた。コマンド 7 をコマンドラインに入力することで、Geant4 をインストールした。

コマンド 7　Geant4 をインストールするときに用いたコマンド

```
1 cd C:\Users\shige\Documents\Geant4_BaseDir\geant4-build
2 cmake -DCMAKE_INSTALL_PREFIX=C:\Users\shige\Documents\Geant4_BaseDir\geant4-install -
      DGEANT4_INSTALL_DATA=ON -DGEANT4_USE_QT=ON -DCMAKE_PREFIX_PATH=C:\Qt\5.15.2\
      msvc2019_64 C:\Users\shige\Documents\Geant4_BaseDir\Geant4_10_07_p03
3 cmake --build . --config Release
4 cmake --build . --config Release --target install
```

コマンド 7 では、1 行目でビルド用フォルダに移動し、2,3,4 行目で CMake を用いてビルドとインストールを行っている。2 行目の cmake コマンドの記述としては、「-DCMAKE_INSTALL_PREFIX=」によってインストールフォルダを指定し、「DGEANT4_INSTALL_DATA=ON」によって必要なデータセットのダウンロードをし、「-DGEANT4_USE_QT=ON」によって Qt を用いて可視化することを指定し、「-DCMAKE_PREFIX_PATH=」によって Qt のパスを設定している。最後にソースファイルが格納されているフォルダ「C:\Users\shige\Documents\Geant4_BaseDir\Geant4_10_07_p03」を指定している。3 行目では CMake で実行ファイルを作成し、4 行目で実行してインストールを行っている。

図 10 のように、エラーが生じずに実行が終われば成功である。

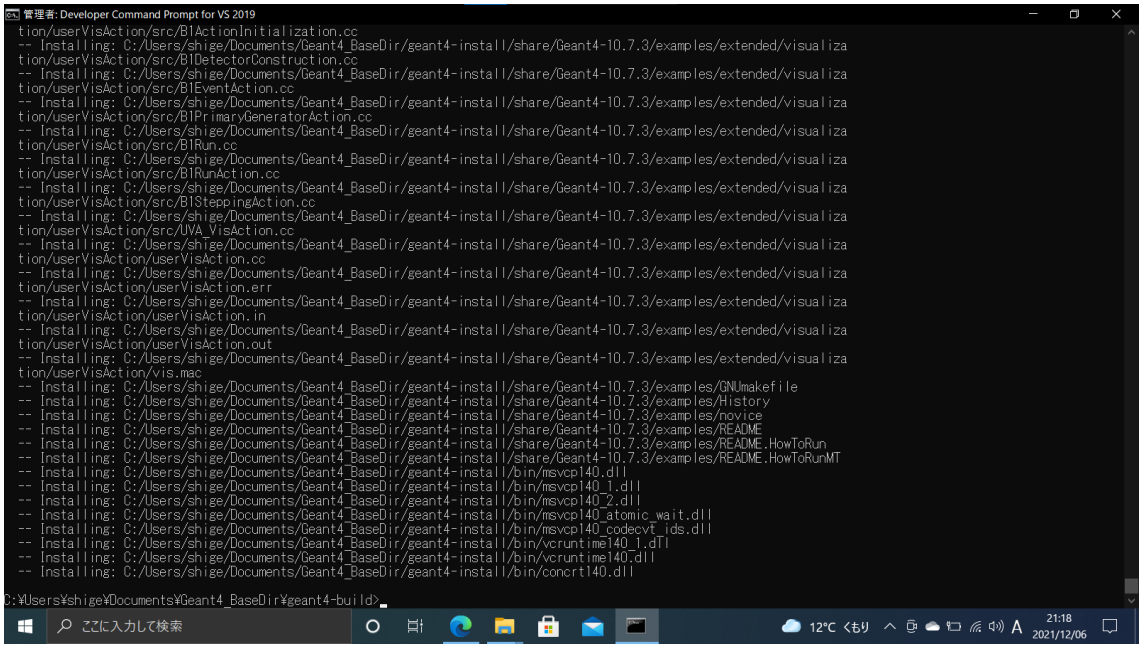

図 10　Geant4 のインストールに成功したときのコマンド画面

2.2.5　Geant4 のパスの設定

Geant4 も、Qt のときと同様、Geant4 のパスを設定する必要がある。まず、システムの環境変数の「Path」に、Geant4 の実行ファイルが格納されているフォルダの絶対パス「`C:\Users\shige\Documents\Geant4_BaseDir\geant4-install`」を追加した。

それに加えて、Geant4 が実行されるときに、様々なデータセットを参照することができるようにするため、「環境変数」の「ユーザーの環境変数」から「新規」を押して、変数名と変数値を以下の図 11 で示すように 11 セットの環境変数を追加した。

変数名	変数値
G4ABLADATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4ABLA3.1
G4ENSDFSTATEDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4ENSDFSTATE2.3
G4INCLDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4INCL1.0
G4LEDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4EMLOW7.13
G4LEVELGAMMADATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥PhotonEvaporation5.7
G4NEUTRONHPDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4NDL4.6
G4PARTICLEXSDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4PARTICLEXS3.1.1
G4PIIDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4PII1.3
G4RADIOACTIVEDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥RadioactiveDecay5.6
G4REALSURFACEDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥RealSurface2.2
G4SAIDXSDATA	C:¥Users¥shige¥Documents¥Geant4_BaseDir¥geant4-install¥share¥Geant4-10.7.3¥data¥G4SAIDDATA2.0

図 11　Geant4 のデータセットの環境変数一覧

この図 11 で示した変数値に関しては、今回の研究を行った環境での一例であり、実際には、Geant4 のインストールフォルダの「`share\Geant4-10.7.3\data\`」のフォルダ内に存在するデータセットが格納されたフォルダの名前を確認して、バージョンの数字を書き換える必要がある可能性がある。また、Geant4 のバージョン自体もアップデートが行われる可能性があるので、確認して対処が必要である。

環境変数は PC の起動時に読み込まれるので、再起動を行ってから次の作業に取り掛かった。

2.2.6 サンプルの実行

最後に、サンプルの実行を行って動作を確かめた。まず、エクスプローラーでサンプルの作業用のフォルダを任意のフォルダに作成した。本研究では、「C:\Users\shige\Documents\work_dir」を作成した。Geant4 ソースファイルが格納されているフォルダ「C:\Users\shige\Documents\Geant4_BaseDir\Geant4_10_07_p03」を開き、「\examples\basic\B1」のフォルダまるごとをサンプルの作業用のフォルダにコピーした。

次に、ターミナルを開いて、以下のコマンド 8 を入力して、サンプルの実行を行った。

コマンド 8　Geant4 をのサンプルを実行するときに用いたコマンド

```
1 cd C:\Users\shige\Documents\work_dir
2 mkdir B1-build
3 cmake -DGeant4_DIR=C:\Users\shige\Documents\Geant4_BaseDir\geant4-install\lib\Geant4
      -10.7.3 ..\B1
4 cmake --build . --config Release
5 .\Release\exampleB1
```

コマンド 8 の 1 行目でサンプルの作業用のフォルダに移動し、2 行目でビルド用のフォルダ「B1-Build」を作成し、3 行目では、CMake コマンドで「-DGeant4_DIR=」で Geant4 のライブラリパスを指定し、最後にソースファイルの場所を指定している。4 行目でビルドを行い、5 行目で実行ファイルを実行する。

実行ファイルを実行すると、以下の図 12 のような画面が表示された。

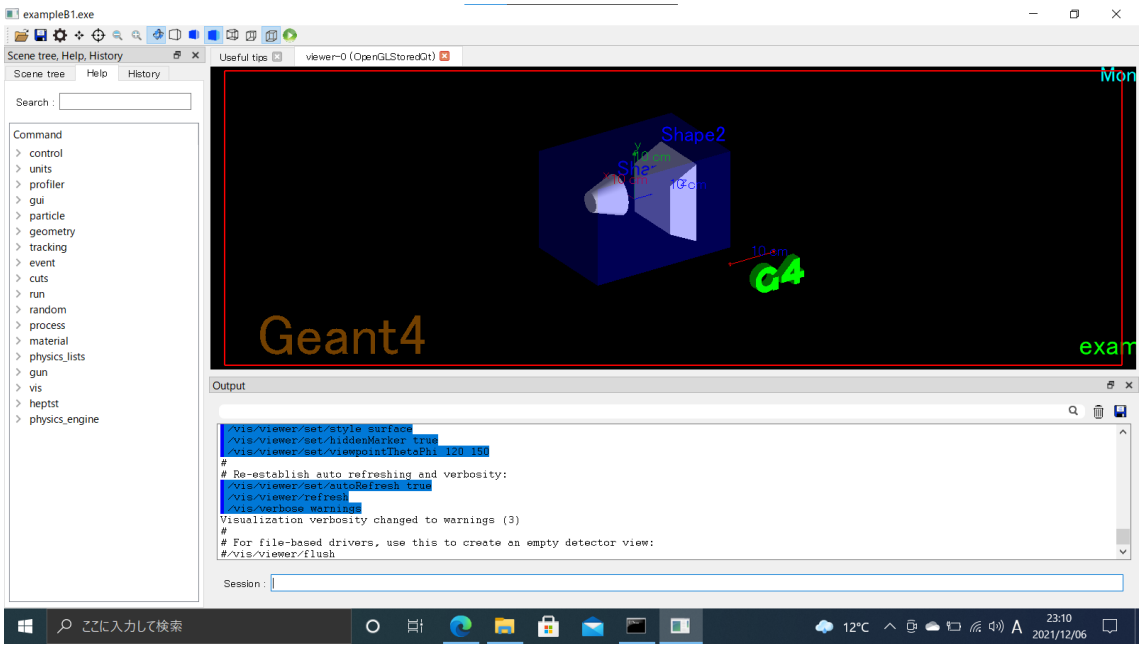

図 12　Windows 上で Qt を有効にした Geant4 でサンプルを起動した画面

また、Linux 同様、コマンド 6 をセッションボックスに入力して、正しく動作するかを確かめた。すると、以下の図 13 のように、透過したり反射したりした光子の軌跡が表示された。

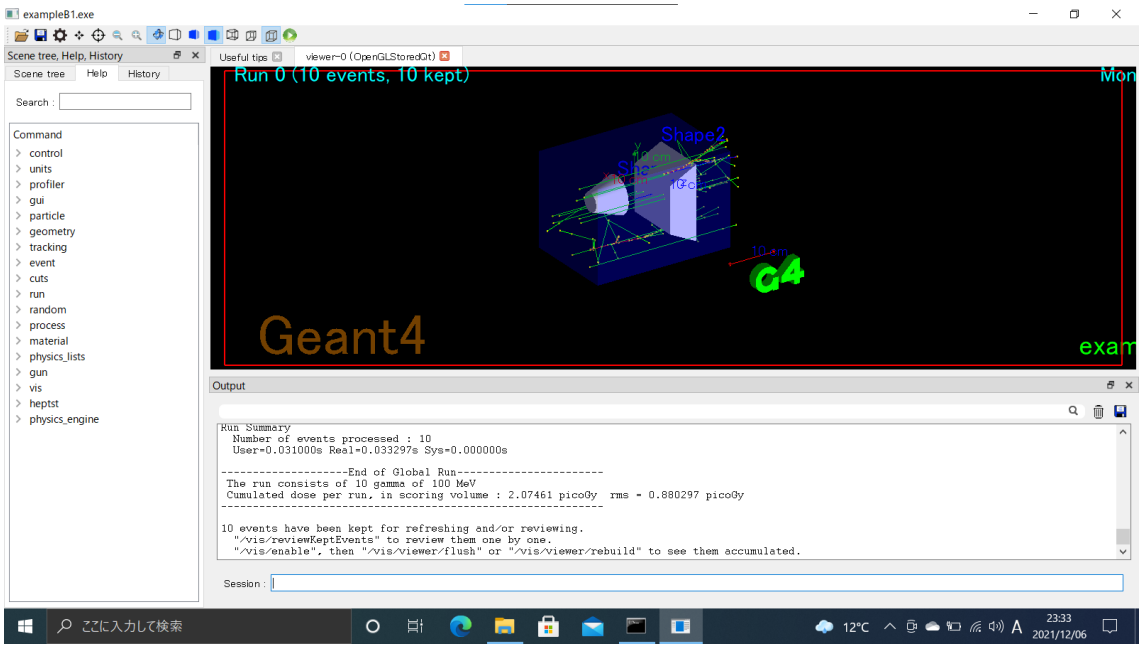

図 13　Windows 上で Qt を有効にした Geant4 で起動したサンプルに光子を照射した画面

この画面はマウス操作で図 14 のように視点を滑らかに移動させることができた。

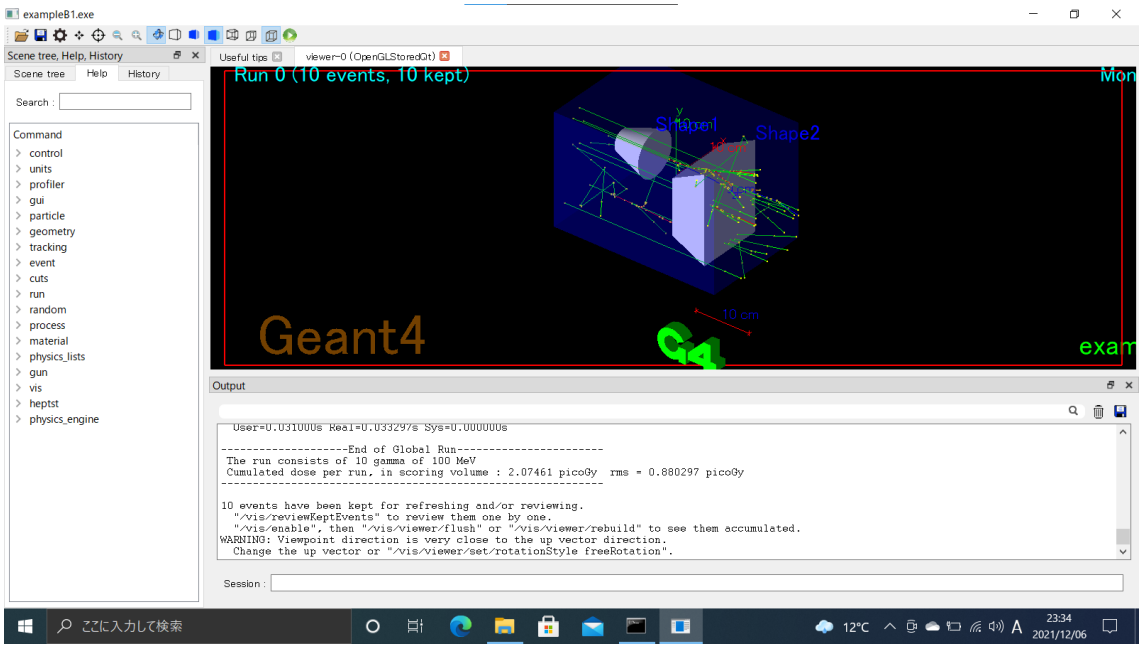

図 14　図 13 の視点を移動した画面

図 12、図 13、図 14 の画面は、Linux での Geant4 の実行結果とほとんど同じであるので、問題なく動作したと考えられる。

3　まとめ

3.1　Windows と Linux での違い

Windows と Linux で、Geant4 を導入する上でどのような利点や欠点があるかを図 15 にまとめた。

	Windows	Linux
利点	●Windowsマシンの用意が容易 ●コマンド操作が少ない	●cmakeやQtがAPTで追加できる
欠点	●必要なソフトウェアをブラウザから手動でダウンロードする必要がある ●環境変数の設定も手動でする必要がある ●エラーが起きる可能性が高い	●Linux搭載マシンの用意が難しい ●コマンド操作が多い

図 15　Linux と Windows の利点と欠点

図 15 に示すように、Windows では、個人用 PC としては最も普及している OS と思われるので、実行用の PC を用意することが容易であることや、コマンドの操作が比較的少ない点などが利点であるため、プログラミング初心者が素粒子物理学のシミュレーションを行いたいときに気軽に実施できると考えられる。一方、Windows の欠点としては、Linux では apt コマンド 1 つによって手軽にインストールできた Qt や CMake を、別途手動でインストールする必要があるという点。環境変数を計 13 セットも設定する必要がある点。環境変数の設定ミスなどにより、エラーが生じる可能性がある場面が多い点などがあげられる。1 点目の欠点は、Linux マシンを用意するハードルよりは低いものであろうと考えられる。2 点目の欠点に関しては、バッジファイルの活用により、大幅に省略できる可能性があるため、研究を続けて補いたい。3 点目の欠点は、本研究の環境と同じ状況で行えば、注意深く実行すればエラーは生じないと思われるが、Windows マシンにインストールされているアプリケーションや OS のバージョン、CPU のメーカーによっては、正しく動作しない可能性がある。

また、Linux では、Windows の利点と欠点の逆の利点と欠点が存在する。つまり、Linux を搭載したマシンを用意でき、Linux の操作に慣れてしまえば、Windows よりも手軽に Geant4 の導入ができると思われる。

3.2 本研究の評価

まず、本研究で達成できた点について考える。本研究の最大の目的である「Windows 上において Qt で視覚化した Geant4 を利用できるようにすること」は、Linux における Qt で視覚化した Geant4 を動作させた内容と同様の結果を Windows でも得られたので、達成したと考える。また、本研究で実際に行った手順を初心者でもわかりやすく記述したいという点に関しても、コマンドの説明や記述方法から、英語からの由来までも説明したり、用語の解説を行ったり、写真をふんだんに用いて手順を丁寧に解説したり、注意点も記述したりすることで、プログラミングや、高度な PC の操作を行ったことのない人でも理解し、実行できるような内容になっていると考える。

次に、達成できなかった点や課題に感じた点について考える。Geant4 そのものの使い方はあまり学ぶことが出来ず、それを活かして身近にある様々な現象のシミュレーションを行ったり、実際に実験を行って観測した実測値とシミュレーションでの結果を比較検証したりすることは出来なかった。今回実行したサンプル以外にも様々なシチュエーションの物があるので、時間に余裕があればもっとサンプルの実行を行いたかった。また、Geant4 や Qt や Windows の動作の詳しい仕組みも理解したかったが、それらの仕組みがあまりに膨大でブラックボックス化しており、更に、執筆者のプログラミングに関する知識があまりに不足していたため、理解には遠く及ばなかった。また、今回行った方法以外でも導入ができないかどうかも検証する余地があると感じた。例としては、WSL を導入して、Linux の仮想環境上で Linux での導入と同様の操作を行ったり、環境変数の設定をバッジファイルで行うことなどがあげられる。また、本研究で用いたマシン以外でも同様の操作を行い、エラーの有無を検証することも必要だと感じた。

4 謝辞

本研究を行うにあたって、基礎物理学研究室のメンバーや高嶋隆一教授には大変お世話になりました。本研究に関する知識が乏しく、右も左も分からなかった私に、基礎的な知識から応用まで幅広く助言を頂きました。Geant4 自体に関する知識、Windows マシンや Linux マシンの仕組み、CUI でのコマンド操作の方法などをご教授いただいたことにより、PC 操作のスキルや、わからないことでも自分だけで調べて理解する力、それを噛み砕いて説明する技能だけでなく、知識面でも大きく成長することができたと感じています。この場を借りて皆様に感謝を述べさせていただきます。本当にありがとうございました。

参考文献

[1]『IT 人材需給に関する調査』 経済産業省
(URL http://www.meti.go.jp/policy/it_policy/jinzai/houkokusyo.pdf) 2021/10/30

[2]『Geant4 のインストール』 Qiita
(URL https://qiita.com/niikura/items/6e34ae5d4bcb74a38669) 2021/10/30

[3]『geant4.web.cern.ch』 CERN
(URL https://geant4.web.cern.ch/support/download) 2021/10/30

[4]『Open Source Development — Open Source License — Qt』 Qt
(URL https://www.qt.io/download-open-source?hsCtaTracking=9f6a2170-a938-42df-a8e2-a9f0b1d6cdce\%7C6cb0de4f-9bb5-4778-ab02-bfb62735f3e5) 2021/12/18

[5]『Building and Installing from Source』 Geant4 Installation Guide 11.0 documentation
(URL https://geant4-userdoc.web.cern.ch/UsersGuides/InstallationGuide/html/installguide.html) 2021/12/20

[6]『Geant4 を Windows にインストールし、Qt を利用して表示する』 Geant4 Installation Guide 11.0 documentation
(URL https://qiita.com/m_moriti/items/cac199fd9a4d7c821044) 2021/12/20

[7]『Visual Studio 2019 の再配布 — Microsoft Docs』 Microsoft
(URL https://docs.microsoft.com/ja-jp/visualstudio/releases/2019/redistribution) 2021/12/20