Node.js による放射線シミュレーション Web アプリケーションの開発

医学研究科　松谷 秀哉

shu@hirosaki-u.ac.jp

背景

放射線は、医療分野では診断や治療に産業分野では非破壊検査をはじめとして我々の生活において重要な役割を担っている。一方で、東日本大震災における福島原子力発電所の事故以降、日本中で放射線に対して負のイメージが強くなり過剰とも思える拒否・アレルギー的な反応が多く見られるようになった。そのため文部科学省は正しい知識・認識の習得のために、初～高等教育（小中高等学校）における放射線教育に対しても強化・注力するようになった。

放射線の大きな特徴は、五感で感じる事ができない（目に見えない、臭わない、など）、生物などに大きな影響を及ぼす、といった点である。そのため、学習に際してイメージを持ちづらく敬遠され、結果的にアレルギー的な過度に脅威として感じる、といった傾向が見られる。そのため、放射線シミュレーションは放射線の理解や学習から研究に至るまで重要な役割を担っている。代表的な放射線シミュレーションとしては、EGS(Electron-Gamma Shower ver.5) [1]、Geant4 [2]、PHITS [3]などがあり、インターネットなどで公開されている。

ところで、この数年における HTML5 [4]をベースとする Web 技術の進歩は眼を見張るものがあり、Web ブラウザの可能性が大きく変化してきている。例えば WebGL [5]により、一般的な Web ブラウザでも三次元モデルを手軽に取り扱えるようになった。今回用いた Node.js [6] は HTML5 アプリケーションのひとつである。

目的

EGS はその前身を入れると 50 年以上の歴史があり、計算対象は光子（X・γ 線）と電子・陽電子であり、放射線関連の研究・開発のみならず放射線教育などでも用いられている。EGS の使用方法は、ユーザが用途に応じて FORTRAN 言語によりプログラム（**ユーザコード**と呼ぶ）を書いてコンパイルや実行する、といったコマンド操作が必要となるタイプで今ではやや古く感じられるスタイルである。日常的なパソコンの利用は GUI (Graphical User Interface) 環境による操作がほとんどである現在、コマンド操作の経験はほとんどない多くのユーザには EGS を使用する事はかなり厳しいものと思われる。

本件の目的は、EGS における FORTRAN プログラミングによる本来ある自由度や拡張性を制限する事と引き換えに、FORTRAN 言語やプログラミングの知識を必要とせず多くの人が使い慣れている GUI 環境である Web ブラウザの操作のみで EGS の利用できる環境を開発することである。

方法

EGS は、線源、体系、検出系で構成されている。線源はプログラム内で定められた変数名を用いて、線種（電荷）やエネルギーをはじめ位置や方向などの情報を設定する。体系や検出系は、配置する物質や幾何学的な形状や位置などの情報を定められた書式で別ファイルやプログラム内で設定する。そのため EGS の初学

者には、これらが分かりづらく戸惑いや混乱を生ずる。従来の EGS における構成と利用者の操作との関係を図1に示す。

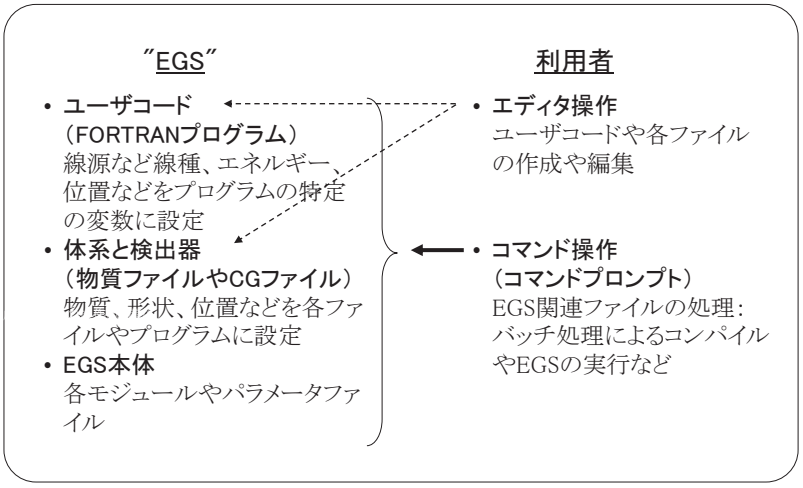

図1 EGSの構成と利用者の操作との関係

本法は、Web ブラウザなどの GUI アプリケーションを用いて EGS における変数に対して値の設定やファイルの修正をおこなうと共にコンパイルと実行を可能にしたものである。そのため、変数設定用のテンプレート用プログラムを作成し、処理用アプリケーションとして Node.js を用いた。従来と本法での EGS における大まかなシステム構成と関係を図2a,b に示す。

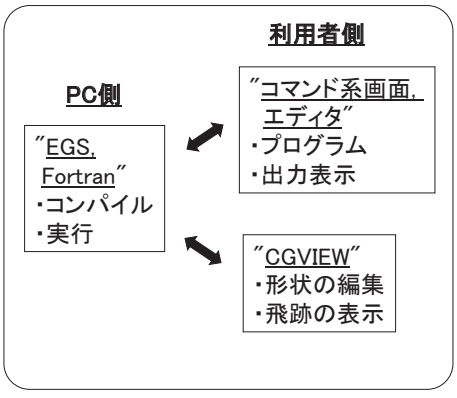

図2a 従来のEGSにおけシステムる構成

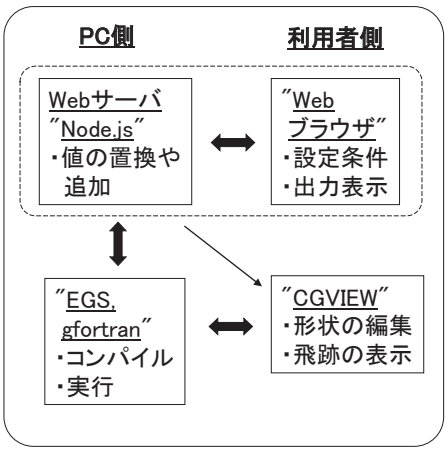

図2b 本法におけるシステム構成（点線部が改良した部分）

作成したテンプレート用プログラムは、EGS のチュートリアル用に公開されている代表的なユーザコード "ucnaicgv.f"をベースにして、線源の幾何学的条件（等方的点線源／平行ビーム、方向など）、エネルギー、線種などの項目に対して設定できるようにした。線源は、単色エネルギーの光子と電子・陽電子をはじめとして放射線同位体として ^{60}Co、 ^{90}Sr、 ^{90}Y、 ^{192}Ir を選べるようにした。放射線同位体のエネルギースペクトルは文献

[7] のデータを用いた。また物質データについても同様で、よく用いられる空気、水、アクリル、Ge、NaI、アルミ、鉄、鉛、Xe ガス、鉛ガラスを用意して利用者が選べるようにした。

Node.js はサーバサイドの Web サーバであることからパソコン内のファイル操作やコマンドを実行してその結果を利用できる。さらにこれらの処理に対してきめ細かくプログラミングできるために、Web ブラウズをインターフェイスとしていろいろな操作や結果に対する対応が可能となる。本法では、上記を実現するための EGS 処理用の Node.js プログラム(JavaScript)を作成した。ちなみに Node.js は非同期処理が大きな特長であるが、今回は操作と結果に対するインターフェイスを意図しているため同期処理でおこなった。

体系(Geometry)は基本図形(球や直方体など 14 種)の組み合わせで構成する"Combinatorial Geometry"として、幾何学的な設定や編集は GUI で作業や表示できる Cgview [8]を使用した。出力量は入射粒子の線種毎の粒子数とエネルギースペクトル、設定区画毎の反応数と吸収エネルギーの割合、さらに可視化用の飛跡データである。飛跡の可視化には体系(Geometry)で用いる Cgview を使用した。

本法での具体的な処理の流れを図3に示す。

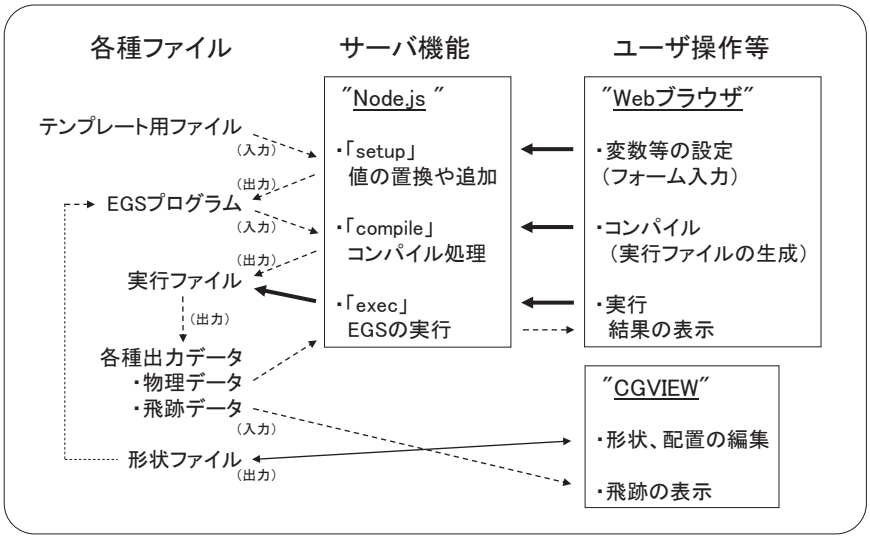

図3 本法における具体的な処理の流れ
(実線は操作、破線は入出力。"Node.js"の「」内の名称は実際のプログラムに対応)

また使用したおもな環境などを以下にまとめた。

- 総合情報処理センター教育用PC:Intel Core i5 2.9GHz, 4GB, OS:Windows8.1
- 開発用 PC:AMD A10-5800K 3.8GHz, 8GB, OS:Windows10
- 開発環境:GNU Fortran ver. 4.8.1, MinGW ver. 4.8.1
- EGS 本体:egs5.160113.tar.gz(2016.1.13 版), ユーザーコード:ucnaicgv.f
- Cgview:Version 2.4(教育用パソコン使用時), Version 3.01(ローカル PC 使用時)
- Node.js:Version 6.2.2, 拡張モジュールは cheerio:DOM(Document Object Model)の操作, ejs:テンプレート処理, iconv-lite:データの文字コードを変換

ところで従来の EGS での処理手順は、ユーザが作成したプログラムと EGS 本体のプログラムを合体させて一つのプログラムにしてからコンパイルする。そのためユーザがプログラムを変更するたびに、変更を必要とし

ない EGS 本体もその都度コンパイルが必要となる。これは時間的にもシステム的にも効率的とはいえない。そこで本件では、事前に EGS 本体(図1参照)のライブラリ化をおこない、ユーザプログラムのみをコンパイルするように変更した。その際、EGS 本体で必要となる配列変数などは EGS 本体での最大値とした。

ところで、本法は医学部保健学科放射線科学専攻の 3 年次学生を対象とした「医用情報学演習」において実際に使用した。この授業は後期 15 回の構成であるが、EGS はその中の 3 回である。実際に授業で用いた EGS は、総合情報処理センターの教育用システムでの P ドライブ(「public」ドライブ)ー「松谷」ー「EGS」にあるので使用する事ができる(実際の簡単な使い方は「使い方.pdf」を参照)。

結果(使用状況)

本法により、EGS の利用において設定から実行までを Web ブラウザのみで行えるようになった。また体系(Geometry)の設定・編集および飛跡データの表示は Cgview でおこなうが、ともに GUI ベースであるため違和感はなく親和性も高い。設定などの様子とその結果の様子を図4a,b に示す。

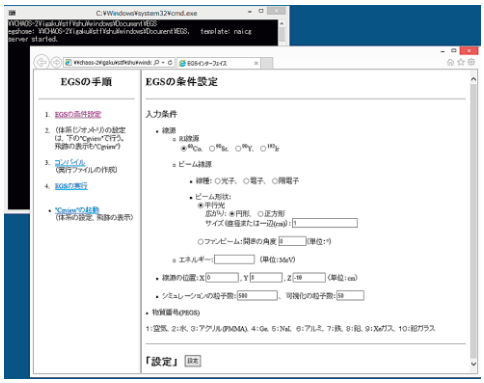

図4a Webブラウザによる設定とNode.js(コマンドプロンプト)

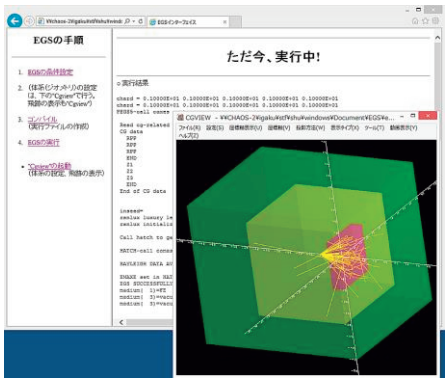

図4b 本法におけるEGSの実行時と結果の表示(WebブラウザとCgview)

実行時における Web サーバである Node.js の負荷は小さく PC 自身のレスポンスには影響しない。具体的には、Node.js 自身による消費メモリーは 20MB 程度、CPU の負荷もごく僅か上昇する程度である。実際、コンパイルや EGS の実行による負荷の方が短時間ではあるが大きい。ただ、入力などで想定外の設定値などの場合、コンパイルの失敗だけに留まらず Node.js が止まる(落ちる)事もあった。

授業での実施は、最初はやや戸惑いもあったが比較的短時間で慣れて全員が取り敢えずではあるがコンパイルし実行できた。一度慣れると以後の操作におけるつまずきはなくスムースにコンパイルや実行が可能となった。その際、コンパイルに必要な時間は 10 秒程度であり、ライブラリ化によりコンパイル時間は半分以下に短縮された。ただしコンパイルや実行中に Web ブラウザの表示が寂しいためか、待ちきれず(?)に何回もコンパイルや実行をクリックする人もいた。一方、従来の EGS では、最初に最もシンプルなチュートリアル用のユーザコード”tutor1”については全員が比較的スムースにコンパイル・実行することができた。ところが次に用いた”ucshield”では、プログラムに対して1行を不実行するための記号(コメント文)を1文字入力する指示(行頭に″C″を入力)をしたところ、ほとんど人がコンパイルでつまずき実行できた人は 2 割にも満たなかった。なお、授業における詳細については文献 [9] を参照して欲しい。

考察

放射線の初学者にとって手軽に放射線シミュレーションを使いその振る舞いを可視化できる事は、放射線の挙動や特性を理解する上において効果的である事は言うまでもない。そのためには、プログラミングが不要で柔軟性が高い、誰でもが使用できる環境が必須である。本件では、EGS の機能性や拡張性を限定（犠牲？）することによりプログラミング不要の GUI 環境を実現した。しかしこの限定化は、必ずしもマイナス要因を意味するものではない。何故なら放射線では専門分野ごとに必要とする機能や状況・内容などが大きく異なるために、視点を変えてみた場合、必ずしも EGS のすべての機能や拡張性を常に必要とするものではない、ともいえるからである。つまり専門分野や状況・内容に応じてこれらを整理・取捨選択する事により、それぞれにマッチした適切なカスタマイズが可能となる。これは、情報システムのカスタムエンジニアリング、と同じである。

従来の EGS ではコマンドでの使用を基本としているが、以前の計算機環境ではこれが一般的な利用環境であった。しかし現在では、パソコンやスマートフォンなどの利用では GUI での使用が一般的であり、そもそもコマンドの存在すら知らない人も多い。そのため以前からのアプリケーションの多くが GUI 対応になっている。一方で EGS のように GUI に未対応、さらには開発や維持などがすでに終了したものもある。今回用いた手法は、GUI に未対応のアプリケーションに対して一般的に適応できる手法であり、本件以外にも広く応用できる GUI 化手法ともいえる。ちなみに、Web ブラウザをユーザインターフェイスとして用いる方法は既に存在しており、Jupyter Notebook [10] は Python 言語 [11] に対する GUI として有名であるが、最近では R 言語 [12] の GUI としても利用できるようになった [13]。本法は Node.js を用いたが、Node.js 以外での可能性も示唆するものである。

ところで、JavaScript は Web ブラウザ上で動作するクライアントサイドのプログラミング言語であり、直接的にファイル操作やコマンドの実行などは出来ない。しかし Node.js の使用により、これらの制限が無くなり Web ブラウザが汎用的なユーザインターフェイスとして利用できる事になる。これは便利な反面、セキュリティ問題が生じる可能性が大きくなる。そのため、Node.js による利用は自 PC 内に留め、ネットワークサービスとしては制限するべきである。

利用時における問題点はふたつあった。ひとつはたまにではあるが Node.js が落ちる事、もうひとつはコンパイルや実行中の無駄なクリック、である。考えられるおもな要因は、前者は想定外の値の入力や設定であり、後者はユーザの操作習慣や不安の現れである。そのため今後は、想定外の値の入力や設定に対するチェックの強化、Web ブラウザの表示、繰り返しクリックの無効化、操作や処理の手順の改善や見直し、などが必要であり今後の課題である。

参考文献

[1] 高エネルギー研究所, Welcome to the Electron Gamma Shower (EGS) Web Page, http://rcwww.kek.jp/research/egs/

[2] Geant4, Geant4, http://www.geant4.org/geant4/

[3] 日本原子力研究開発機構, PHITS, https://phits.jaea.go.jp/indexj.html

[4] W3C, HTML5, https://www.w3.org/TR/html5/

[5] Khronos Group, WebGL – OpenGL ES 2.0 for the Web, https://www.khronos.org/webgl/

[6] Node.js Foundation, Node.js, https://nodejs.org/ja/

[7] RADAR – The Decay Data, http://www.doseinfo-radar.com/RADARDecay.html

[8] 高エネルギー研究所, Cgview, http://rcwww.kek.jp/research/egs/kek/cgview/

[9] 松谷 秀哉ほか, 放射線専攻学生を対象とした放射線シミュレーションの教育的有効性の検討, 教養教育, 1, 弘前大学 (2017) . (印刷中)

[10] Project Jupyter, Project Jupyter | Home, http://jupyter.org/

[11] Python Software Foundation, Welcome to Python.org, https://www.python.org/

[12] The R Foundation, The R Project for Statistical Computing, https://www.r-project.org/

[13] Thomas Kluyver, Philipp Angerer, Jan Schulz, GitHub – IRkernel/IRkernel; R kernel for Jupyter, https://github.com/IRkernel/IRkernel