物理エンジンを用いたプログラミング教育

箕原辰夫*1
Email: minohara@cuc.ac.jp

*1：千葉商科大学政策情報学部

◎Key Words 物理エンジン，プログラミング教育，３次元プログラミング

1. はじめに

近年ゲームライブラリを中心に、物理シミュレーションを行なうライブラリ、あるいは物理エンジンが用いられている。物理エンジンを使うと、物体や重力や風力などの環境を設定するだけで、場面に配置された後の物体の挙動は、自動的に物理エンジン側で計算され、物体の位置や姿勢が更新されていく。これまでも卒業研究の指導では、これらの物理エンジンを利用した形で Unity を利用した作品を扱ってきたが、昨年度の卒業研究指導では、航空力学を扱ったので、単純な物理計算以外に、翼断面による抗力・揚力の計算なども必要となり、物理エンジンをそのまま利用することができなかった。そこで、物理エンジンがどの程度の機能があり、教材としての構成要件を満たしているのか興味を持つこととなった。

この論文では、Bullet, PhysX, Havok などの良く使われている物理エンジンを採り上げ、Unity や Panda3D といったゲームエンジン・ライブラリの中に組み込まれた物理エンジンが、どの程度のシミュレーションの能力があるのかを試用し、それらをどのようにプログラミング教育に採り入れているべきかについて考える。

2. 各物理エンジンの機能

この論文では、教材を前提として学生がフリーで用いることができる物理エンジンの中から、いくつかの主要なものをピックアップして、比較を行なっていく。まず、採り上げた３つの物理エンジンを紹介する。もちろん、これ以外にもフリーのものが存在するが、学生が利用できる中でベストに近いものを選んだ。

2.1 Bullet Physics Engine

Bullet Physics Engine[(1)]は、オープンソースのライセンスで使用可能な C++で記述された物理エンジンで、衝突感知・剛体モデル・ソフトボディなどをサポートしている。レイおよび凸形状一斉テストによる、離散または連続の衝突判定を行なっている。衝突形状としては、凸形、凹形のメッシュ、およびその他の基本的な形状をサポートしている。それに加えて、拘束条件を追加することで、車（タイヤとボディから構成される）や関節を持つキャラクタ（人体）、スライダ、蝶番のような動きを持つ物体のシミュレーションができる。ソフトボディについては、布や紐、変形する物体といったソフトボディ同士、ソフトボディと剛体との相互作用が可能になっている。

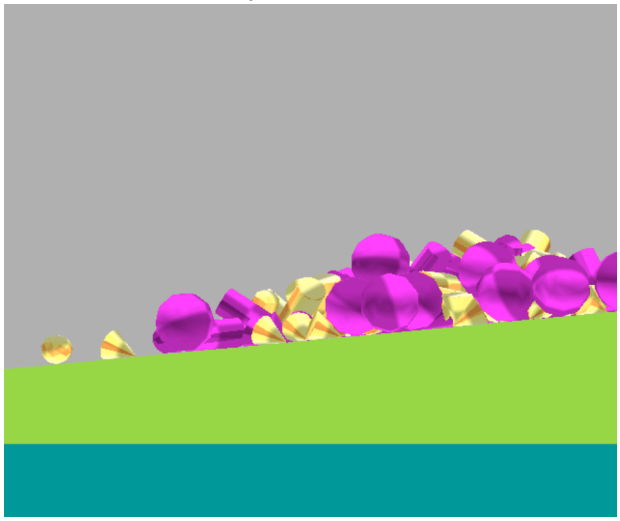

図 1 Bullet の回転オブジェクト（bulletphysics.org）

2.2 NVIDIA PhysX

PhysX[(1)]は、NVIDIA 社が開発して、主に自社製の GPU ボードで並列計算ができるような形で配布されている。多くの３次元グラフィックスを扱うライブラリ、エンジンに組み込まれて普及している。PhysX の機能としては、衝突判定の他に、剛体モデル、各種関節のサポート、人体関節、摩擦力の考慮、物理作用が機能する軸の制限、乗り物のための動力学の支援の他、布シミュレーションを含むソフトボディ、および流体シミュレーションを行なうことができる。

図 2 PhysX の布シミュレーション（developer.nvidia.com）

2.3 Havok Physics

Havok Physics[(1)]は、アイルランドの Havok 社で開発さ

れてきた物理シミュレーションのミドルウェアであるが、インテル社に買収されて、AMD/Intel 系列に入ることになった。C/C++で記述されたエンジンを中心に、統合的な開発環境まで用意されている。Havok Physics の物理エンジンとしての機能は、大量のメッシュに対しての衝突感知、機械などの制約のある運動、人体などの関節のあるキャラクタの運動、 布のシミュレーション、崩壊（destruction）のシミュレーション機能などがある。

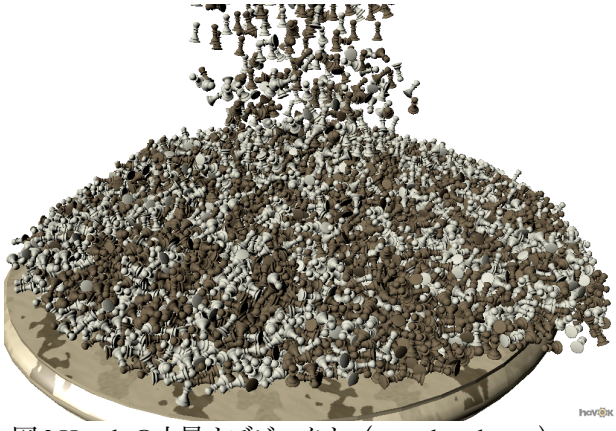

図 3 Havok の大量オブジェクト（www.havok.com）

3. 評価を行った開発環境

３つの物理エンジン評価するにあたって、使用した開発環境を紹介する。

3.1 Panda3D

Panda3D は、C++および Python 用に開発された、フリーの 3DCG ライブラリである。Panda3D の組込みの物理エンジンは機能が限られているため、1.5.3 版からは Object Dynamic Engine（ODE）を利用することが可能になっていた。これに加えて、1.8.9 版からは Bullet Physics Engine が標準で利用することができる。また、1.7.0 版からは開発元とは別の組織によって、PhysX エンジンが利用することが可能になっている（ただし、Panda3D 用の PhysX のバイナリの配布は Windows と Linux 版のみ）。そのため、同じプラットフォーム上で Bullet と PhysX の性能の違いを試すことが可能となった。

図 4 フライトシミュレータ作品 (www.panda3d.org)

3.2 Unity

Unity は、Unity Technologies 社が開発した 3DCG を伴うゲームを開発するための統合的な環境である。無償版と有償の Pro 版が用意されていて、Pro 版では更に高度な機能を用いることができる。3DCG モデルをシーンエディタ上で配置し、その後は、物理的な挙動が設定されていれば、それに従ってモデルが動くようになっている。また、各オブジェクトにはスクリプトを添付することができ、定期的にスクリプト内の関数（メソッド）が呼ばれて、オブジェクトを制御することができる。このスクリプトの中でも、剛体の挙動を知ることができ、それに応じたプログラミングが可能になっている。Unity でのスクリプトは、JavaScript、C#、Boo（Boo は Python に文法が似ている、型指定のできるスクリプト言語）のいずれかの言語を使って記述することが可能になっている。

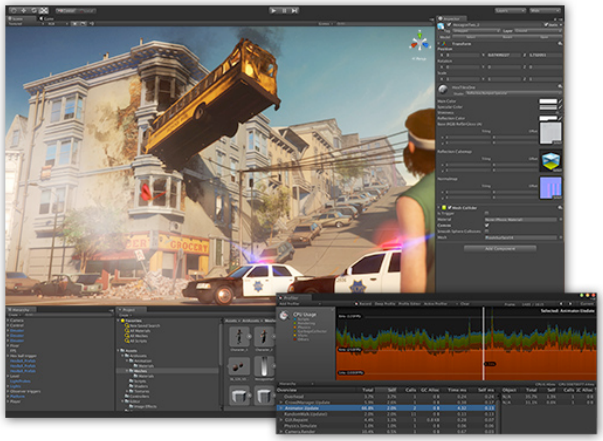

図 5 Unity のシーンエディタ（www.unity3d.com）

3.3 Havok

Havok の統合的な開発環境では、シミュレーションを再現できる Havok Animation Studio と呼ばれるソフトウェアや、リアルタイムでレンダリング・シェーディングを行なう Havok Vision Engine、ゲーム開発で経路探索などを行なうための AI ツール、スクリプト言語 Lua を用いたプログラム環境が用意されている。また、Visual Studio と連携して、C++を用いても開発が行なえる。アカデミックおよび学生用には、Windows 版の無償の開発環境が用意されている。

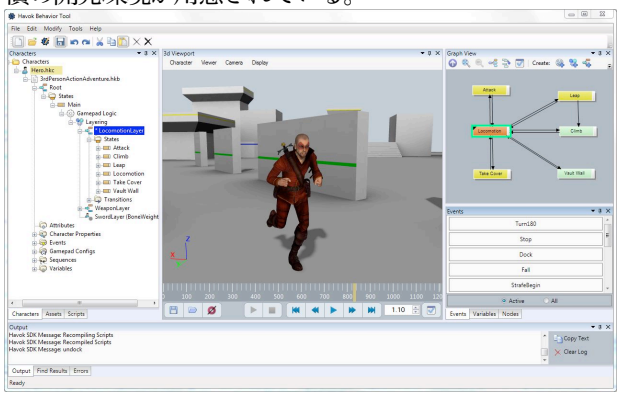

図 6 Animation Studio での挙動設定（www.havok.com）

これらの環境を用いて、表 1 のように各物理エンジンを使用してみた。他に、よく使われているものとして、

Unreal 4[1]などの開発環境もあるが、今回の試用では、ライセンス料金が掛かるので見送ることにした。

表 1 評価を行なった物理エンジン

物理エンジン	評価環境	使用言語
Bullet	Panda3D	Python
PhysX	Panda3D	Python
PhysX	Unity	JavaScript/C#/Boo
Havok	Havok	Lua

4. 評価

評価は、機能的な差はほとんどなく、どれもが衝突判定、レイキャスト（Ray cast）、剛体の挙動、布のようなソフトボディの挙動の再現、人体のような関節のあるような動き、機械のような制約のある挙動の再現などを実現している。しかしながら、性能的には GPU の恩恵を受けられる PhysX に軍配が上がる。ただ、Havok のパイプライン処理も、実際に Havok が使われているソフトウェアの速度から考えると、かなりの高速感がある。また、教材の開発能力的なものは、やはりシーンのスタジオから直接物体等の配置を行なえる PhysX（Unity を利用して）や Havok は、3DCG 作成ソフトウェア Maya, 3ds Max や Blender を利用するか、スクリプトで全部記述するかしなければならない Bullet と比べると、一日の長がある。

4.1 機能

機能的には、Bullet は PhysX や Havok に比べて遜色があるとは思えない。しかしながら、サポートやプロモーションにおいてはオープンソースのボランティアに頼っている分、商用でも活躍している後者の２者に対して、派手さには欠ける。Google に開発者が移ったようなので、今後の展開に期待したい。機能で一番の差があるのが PhysX の流体シミュレーションと考えられる。以下の図は、NVIDIA が用意した（YouTube にアップロードされている）動画の一場面だが、これは Havok に対して突き出た技術になっている。

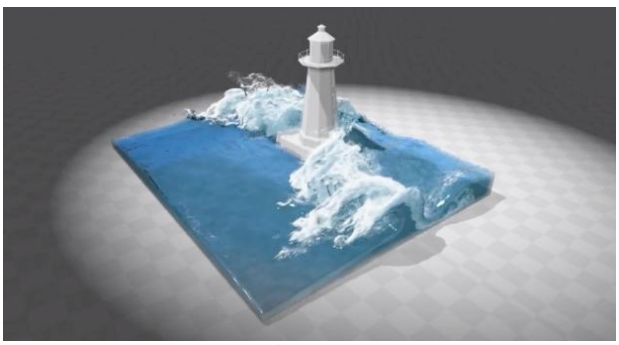

図 7 PhysX の流体シミュレーション
(www.youtube.com/watch?v=teD6vq8DVVE)

また、Havok の AI 機能を使えば、物理空間の中で様々な自律的なオブジェクトを運動させるのが非常に簡単になるだろう。プログラミングを必要とせず、このような自律的な挙動が記述できるのは、大量のオブジェクトが動くシミュレーションには便利であると思われる。

図 8 自律的なオブジェクトを多数持つことができる Havok AI (www.havok.com/ai-screenshots)

4.2 性能

Bullet と PhysX については、Physics Engine Evaluation Lab を用いて行なわれた性能比較がある。球体が結合された網構造に対してのシミュレーション時間を示したものである。PhysX の前の版は、最初の方のフレームで、Bullet よりも劣っているが、フレームが進むにつれ、全体として Bullet よりも性能が高いのがこのグラフから伺える（http://physxinfo.com/news/11297/the-evolution-of-physx-sdk-performance-wise/を参照）。

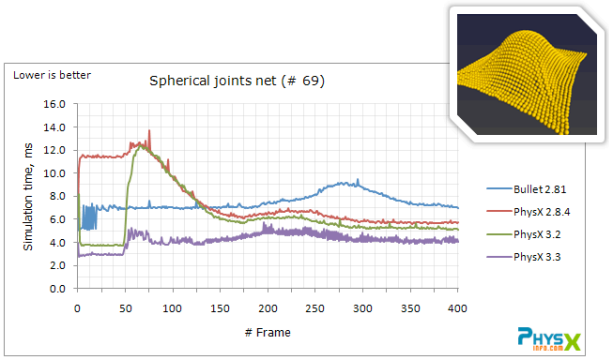

図 9 フレーム数に対するシミュレーション時間

Havok と PhysX についての、このような正規化された形での性能比較がなされていないのは残念である。今回試用した環境では、Havok は Windows 用の無償版を試用した。手元の環境では、Windows は Mac OS X 上の Parallels Desktop を使って稼働しているので、直接 Mac OS X 上で稼働している Unity 上の PhysX と比べることは、Havok にとって明らかに不利となる。そのため、性能の比較は行なわなかった。

4.3 開発能力

Bullet のシーンエディタとして利用することが可能なのは、フリーの 3DCG モデル作成ソフトウェアである Blender や、Autodesk 社の Maya や 3ds Max になっている。それに対して、PhysX は Unity を使えば、シーンエディタ上で直接各オブジェクトに対して、詳細な物理的な設定を行なうことができる。また、Havok の Animation Studio での挙動設定はわかりやすく、物理の

影響を受けた複雑な挙動も直接設定ができる。もちろん、両者はスクリプト言語からも動的に、同じような設定を行なうことが可能になっている。特に、スクリプトを使ってプログラミングしないのであれば、Unity 上の PhysX か Havok の Animation Studio を使うだけで充分であると考えられる。

5. 教育にどのように採り入れるか

採用された論文の分科会への配属がプログラミングだけではなくて、物理系の専門教育の話題も出ている分科会になっていたので、ここではプログラミング教育と物理系の科目への両者において、どのように教材として採り入れていくべきか考察してみる。

5.1 大学でのプログラミング教育において

プログラムとして物理エンジンを利用するのであれば、むしろ Bullet を薦めたい。Unity 上の PhysX を利用するには、まず Unity の開発環境を学ばなければならない。これには、シーンエディタなどのソフトウェアなどの使い方も含まれる。しかし、Panda3D 上の Bullet であれば、純粋に Python のスクリプトを使って、多くの物理的な挙動のための設定を記述することができる。また、Unity 上では PhysX の流体シミュレーションについて、まだサポートされておらず、Unity 5 の登場を待つ状態になっている。

5.2 高校・大学での物理系の科目において

高校・大学での物理系科目においては、逆に Unity 上の PhysX あるいは Havok を使うことを薦めたい。シーンエディタや Animation Studio では、シーンに配置して、物理特性を設定するだけで、後は物理的な挙動は自動的に物理エンジンが行なってくれる。プログラミングをしない前提であれば、Bullet を組み込まれた、別の Maya や Blender などのソフトウェアを使うよりも、直接的に、仮想空間における物理的な環境設定ができるので、Unity や Havok の方が使いやすい。今までの物理実験は、文字通り物理的な制約があるために、主に教員がするのを学生が見ているだけの場合が多くあるが、これらのソフトウェアを使えば、パラメータを変えて学生側も行なうことができる。また、現実には設定しにくい、重力を 0.6 倍や、6 倍にしたような、特殊な物理環境も設定することができる。

6. おわりに

実際に 3 つの物理エンジンを利用してみて、これから教員も学生もこれらのソフトウェアを楽しんで使っていけるのではないかと感じた。特に、PhysX や Havok は、こんなこともできるのかという機能を説明した動画が YouTube 上にアップロードされており、それを観ているだけでも、心逸るものがあるだろう。 現在の学生の物理離れを止めるのに一役買ってくれるのではないかと期待するところが大きい。

参考文献

(1) Bullet Physics Library：Real-Time Physics Simulation," http://bulletphysics.org/wordpress/, Real-Time Physics Simulation（2014）.
(2) Havok："Havok," http://www.havok.com/, Havok.com Inc.（1999-2014）.
(3) Unity："Unity – Game Engine," http://unity3d.com, Unity Technologies（2014）.
(4) Panda3D："Panda3D Manual: Physics," https://www.panda3d.org/manual/index.php/Physics, CMU Entertainment Technology Center（2010）.
(5) PhysX："PhysX Technology | NVIDIA," http://www.nvidia.co.jp/object/physx_new_jp.html, NVIDIA Corporation（2014）.
(6) Unity："Unity – Game Engine," http://unity3d.com, Unity Technologies（2014）.
(7) Unreal 4："Unreal Engine Technology," https://www.unrealengine.com, EPIC Games Inc. (2004-2014).