Unity を用いた力学実験の 3D エンジン化

熊本県立宇土高等学校２年　科学部情報班 MicrosoUTO　今村　遙　舛田　崇光　安田　陸人

要　旨

物理の授業で、力学分野を学習している。授業では探求的・実践的な問いが設定されており、学習の目的は掴みやすいが、実際の物理現象が公式や理論と結びついている実感が湧きづらいという課題が挙げられた。インターネットに数多く上がっているシミュレーションサイトでは、あらかじめすべてプログラムされているため、一見、物体の運動とグラフの描写が紐づけられているように見えるが、実際は力学法則に則った処理が行われていない。本研究では、力学法則に則った処理ができる Unity を用いることで、現実の物理現象に近い状態でグラフの描写を含めたシミュレーションを行うことが可能になった。また、プログラムを公開することで、力学を学ぶ人がアクセスし、シミュレーションを行うことができるよう、現在準備を進めている。

１：動機

着眼点：教科書➡物理現象の公式やグラフが多様に用いられている

物理現象と公式とが結びついている実感・・・薄

例）水平投射（等速直線運動と自由落下）

↓

結びついている実感を抱き、授業への導入を円滑に行いたい

↓

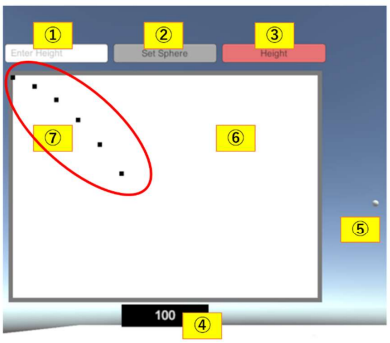
力学法則に則った処理ができるUnity

➡あらかじめプログラムされたシミュレーションサイトよりも現実的に、
グラフの描写を含めたシミュレーションが可能ではないか

２：目的

- 実際の物理現象と公式や理論とが結びついている実感を抱くことで、授業への導入を円滑に行う
- プログラムを公開し、多数の人がシミュレーションを行うことができるようになる

３：シミュレーション概要



資料１：Unity上でのシミュレーションの様子

①Enter Height
数値（高さ）の入力欄　※『100』と入力済み

②Set Sphere
球体を生成するボタン

③Height
球体の落下・グラフの描画開始

④高さ出力
①で入力された数値（高さ）を出力

⑤球体
半径１m/重さ１kg

⑥座標平面（グラフ）
縦軸：高さh(m)/横軸：時間t(s)

⑦座標
経過時間t(s)に対する高さh(m)

Program 1

・GameObject型の“inputf”と“result”を宣言

・inputfには①を、resultには④をアタッチする
➡それぞれのオブジェクトにアクセスできるようにする

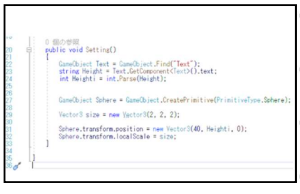
・①に入力された値をstring型の“input”変数に代入

・④のTextコンポーネントにinputを代入し、④に入力された値を出力

・Input Fieldのテキストを空文字に更新し、入力された文字を初期化

上記の処理はGetInput関数のスコープ内にスクリプトする
①で値の入力が完了したら、GetInputが呼び出されるようにする

Program 2



・GameObject型の“Text”を宣言、④を代入
➡④にアクセスできるようにする

・String型の“Height”を宣言、④のTextコンポーネントのTextを代入
➡int型の“Heighti”変数を宣言、Heightをint型に変換したものを代入

・CreatePrimitiveでSphere(球体)のオブジェクトを生成
➡ゲームとして実行する際見やすくなるよう設定。

・Y座標はHeighti
➡①で入力した高さをボールの生成時の高さに反映させる

上記の処理はSetting関数のスコープ内にスクリプトする
②が押されたらSettingが呼び出されるようにする

Program 3

bool型の“isStop”変数を宣言
isStopの真偽でグラフの描画プログラムを制御していく
isStopがTrueの時は描画プログラムは実行される

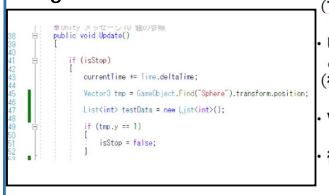
Start関数ではisStopにFalseを代入する

GameObject型の変数“Sphere”を宣言
生成されたSphereを代入する

isStopがFalseであるとき実行されるif文を作る
isStopをTrueにし、SphereにRigidbodyコンポーネントを追加する
➡これによりSphereに重力がはたらき始める

枠内の処理はDrawing関数のスコープ内にスクリプトする
③が押されたらDrawingが呼び出されるようにする

Program 4



・Update関数内にisStopがTrueのときに実行されるif文を作る

・currentTimeにTime.deltaTimeを足す
(Time.deltaTime…直前のフレームと現在のフレーム間で過ぎた時間)

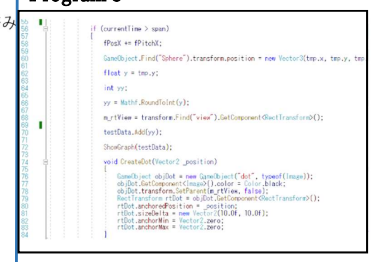
・Update関数は毎フレーム呼び出されるため、currentTimeは処理を始めてからの経過時間を表す
(後ほどcurrentTimeは用いる)

・Vector3型の“tmp”変数を宣言し、Sphereの座標を代入する

・後のプログラムで0.5秒に一度Sphereのy座標を取得するが、その取得した値を保存するためのリストをここで宣言しておく

・tmpのy座標が1になったときに実行されるif文を作る。
➡isStopをFalseにする(=描画処理が停止される)

Program 5



currentTimeがspan以上(=経過時間が0.5秒以上)経ったら処理が実行されるif文を書き、以下の処理をスクリプト

・fPosXにfPitchXを足し、⑦のx座標を更新する

・Sphereの座標を取得し、float型の変数“y”にy座標を代入
int型の変数“yy”にyをint型に変換した値を代入
➡宣言したリストに追加

※CreateDot関数は⑦を生成する関数（主に⑦の細かな設定）

ShowGraph関数を作り、以下の処理をスクリプト

Program 6

・④に出力された値を取得し、string型の変数“Height”に代入
int型に変換しfloat型の変数(int型で宣言するとなぜかエラーになるが、原因不明)“Heighti”に代入

・float型の変数“fGraphHeight”を宣言
これに⑥の高さを代入する

・for文を作る
初期値0のint型変数“i”に1を足していく
「リストに保存されているi個目の値(=現在のSphereのy座標)/y座標の初期値」にfGraphHeightをかけることで適切のグラフ上のy座標を作る
計算結果をfPosYに代入

・CreateDot関数を呼び出し、座標をfPosXとfPosYに更新したうえで⑦を出力

・最後に、currentTimeを0にすることで、「0.5秒に一回処理を実行する」というプロセスを実現させている

４：まとめ

【考察】

今回のシミュレーションで得られた計測値と実際の公式を用いて計算した値を比較

➡Unity上でシミュレーションされた物理法則は理論値に一致することが分かった

【今後の展望】

- 速さや加速度の取得
- 鉛直投げ上げや斜方投射をする物体の運動の値の取得
- コードの公開
- 今回作ったシミュレーションを誰でもアクセスできるようサーバー上で公開