DOI：10.20776/S24326291-6-P185

Processingを用いた物理シミュレーション教材の制作 ―斜方投射を例にして―

吉田　賢二[1)]・三野　弘文[2)]

[1)]千葉大学教育学部　　[2)]千葉大学大学院国際学術研究院

Creation of physics simulation materials using Processing: Taking projectile motion as an example

YOSHIDA Kenji and MINO Hirofumi

要旨

近年、高等学校等の教育現場では、物理実験が積極的に実施されていない一方で、タブレットやノートPCを用いたICT教育が進められ、パラメータを変化させることで物理現象をシミュレーションする教材に注目が集まりつつある。本研究では、映像表現に特化したプログラミング言語であるProcessingを用いて物理現象をシミュレーションする教材を制作し、斜方投射を例にした物理シミュレーションによって、コンピュータ画面内でも動きのある映像を通じた実験に近い形での学習の可能性について考察した。制作にあたっては、放物運動をしている物体の座標や速度などの変化を視覚的に観測しながら、それらの物理量データを取得し、定量的な評価ができることを重視した。更に学校現場で教員が使用する際に、必要に応じて物理シミュレーション教材をカスタマイズできるように、その制作方法を段階的に示した。パラメータを自在に変化させて実行できるシミュレーションの容易さや、パラメータの違いによる現象の比較を画面上で明確に示すことができる点では、シミュレーション教材でしか得ることができない教育効果が期待される。

キーワード
物理教育、シミュレーション教材、ICT教育、プログラミング

1．序論

近年の教育現場（特に高等学校）では、実験が積極的に実施されていない。その原因として、教員の実験の経験不足や実験で使用する器具や設備のサポート体制が整っていないこと[1]、そもそも実験を行う時間がないこと[2]等が挙げられる。その一方で、タブレットやノートPCを用いたICT教育が積極的に進められている。令和３年の３月に文部科学省にて行われた「令和２年度学校における教育の情報化の実態等に関する調査結果（概要）」[3]によると、公立学校（小学校、中学校、義務教育学校、高等学校、中等教育学校及び特別支援学校）における教育用のコンピュータ１台当たりの児童生徒数は年々減少しており、平成21年３月に比べて、およそ80％減少している（図１）。このことからICT教育の環境の整備状況は改善されつつあるといえる。このような状況で、ICT機器を用いて実験シミュレーションを体験させる試みに注目が集まっている。先行研究では、図２のような特定の数種類の値でパラメータを変化させ、シミュレーションを行う教材が存在する[4]。それらの教材は、その使いやすさと親しみやすい見た目から、現象の概要を説明する際には大いに役立つ。また、パラメータの系統的変化によって生じる現象を観測することで、一部の条件における実験をイメージすることも可能である。しかし、それらの教材を構成するプログラムは、コードが使用者に見えないようになっているか、プログラム自体が難解であるという理由から、入力する値を自由に変えたり、実験する条件をカスタマイズしたりすることが難しい。

そこで本研究では、次に示す二つの機能を持つ物理シミュレーション教材を制作し、そ

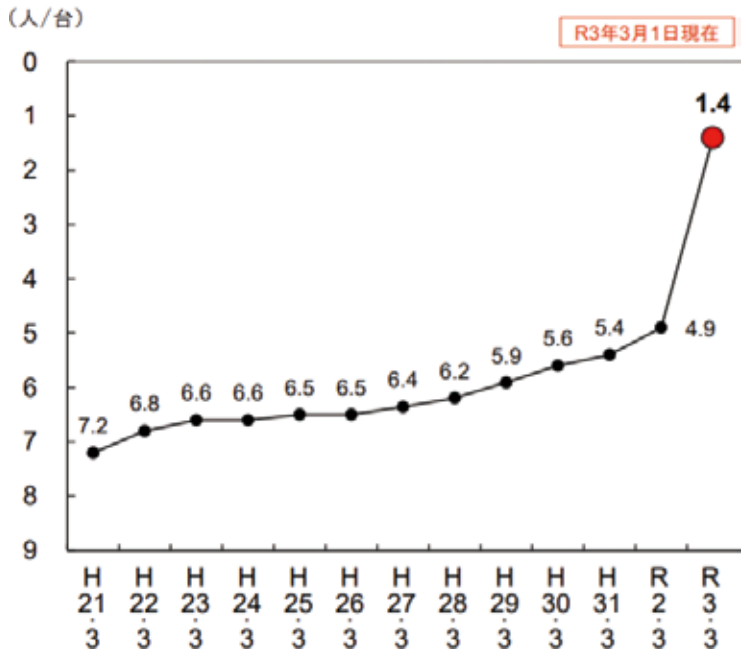


図１　教育用コンピュータ１台当たりの児童生徒数の経年変化[3]

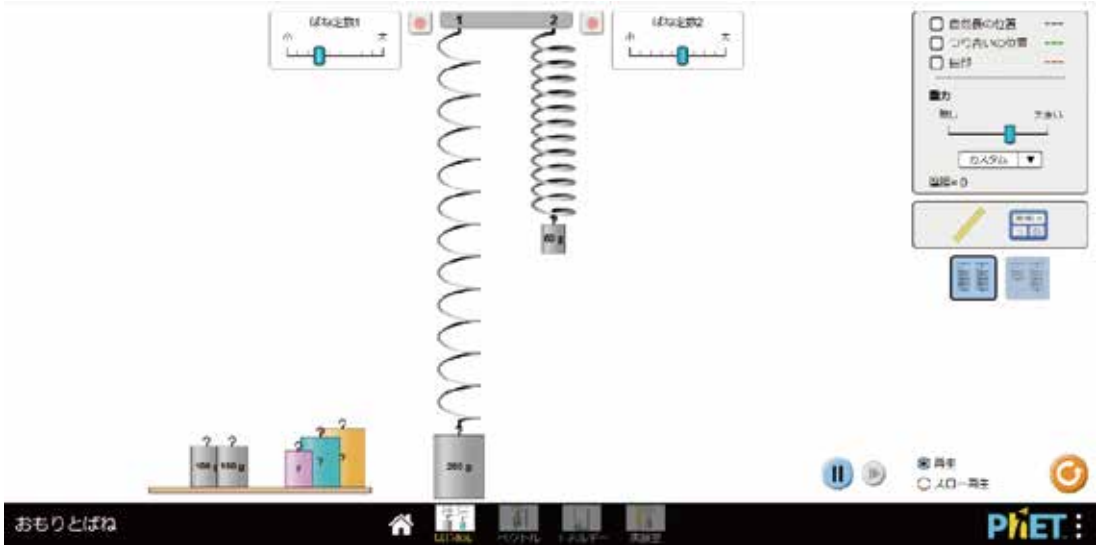


図２　コロラド大学のシミュレーション教材[4]

の活用方法について検討する。一つ目は、物理シミュレーションを自由にカスタマイズすることができる機能である。先にも述べたように、既に存在している物理シミュレーション教材の殆どが、そもそも構成されているプログラミングのコードが見えない状態になっているか、プログラミング言語自体が難解であることを理由として、カスタマイズを行うことが難しい。ICT教材を使用する場合には、教員が教えたい内容に合うようにカスタマイズができた方が、学校現場で取り入れやすく、教科書の静止画による学習よりも動きのある現象を見せることで生徒により深い理解に導く事が期待される。二つ目は、シミュレーションを通して得られた物理現象に関わる物理量を、数値データとしてcsvファイルで出力する機能である。この機能を付けることによって、取り上げている現象を画面上で表示される直感的な理解だけではなく、数値として捉え、データを解析し、グラフに表示するなど、定量的な評価についての学びにもつなげる事が期待される。

物理シミュレーション教材の制作においては、Processingを用いる事とした。Processing[5]は、デザイナーやアーティストなどの視覚的な表現をしたいプログラミング初学者に向けて作られたプログラミング言語である。そのため、構成されている言語の文法が単純なものになっており、更に、プログラミングを行った結果を実行ボタン一つで、スムーズに画面上で表示させる機能も初学者にとって扱い易いものと言える。2001年にリリースされてから、インターネット上に学習するためのヒントが多く存在しており、その開発環境は無料で提供されているなど、比較的容易にProcessingに関する技術を習得することができる。なお、2021年８月９日に大幅なアップデートがされており、最近でも開発が活発に進められている。

２．斜方投射の理論[6]

ここでは､物理シミュレーション教材として扱う斜方投射の理論について簡単に説明する。

斜方投射において、空気抵抗がない場合を考える。条件を図３のように、地面と投射方向がなす角をθ[rad]、初速度をv_0[m/s]、投げてからの経過時間をt[s]、重力加速度をg[m/s^2]、投射してからt秒後の水平方向、垂直方向それぞれの変位をx[m]、y[m]、初期位置をそれぞれx_0[m]、y_0[m] とすると、それぞれの関係は次のように表すことができる。

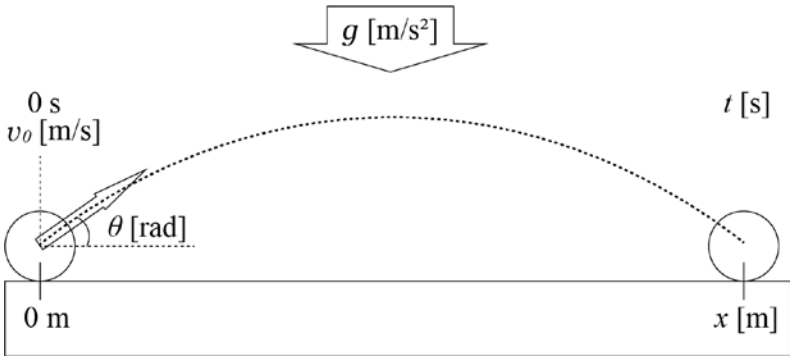


図３　空気抵抗のない斜方投射

$$x = v_0 t\cos\theta + x_0 \cdots (1)$$

$$y = v_0 t\sin\theta - \frac{1}{2}gt^2 + y_0 \cdots (2)$$

次に斜方投射において、速度に比例する空気抵抗がある場合を考える。条件を図４のように､地面と投射方向がなす角をθ[rad]､初速度をv_0[m/s]､投げてからの経過時間をt[s]､重力加速度をg[m/s^2]、投射してからt秒後の水平方向、垂直方向それぞれの変位をx[m]、y[m] とし、空気抵抗がかかる方向は図５のように速度の逆向きの方向とし、その大きさは速度に比例するとした。図５中のv[m/s] は球の進行方向の速度である。

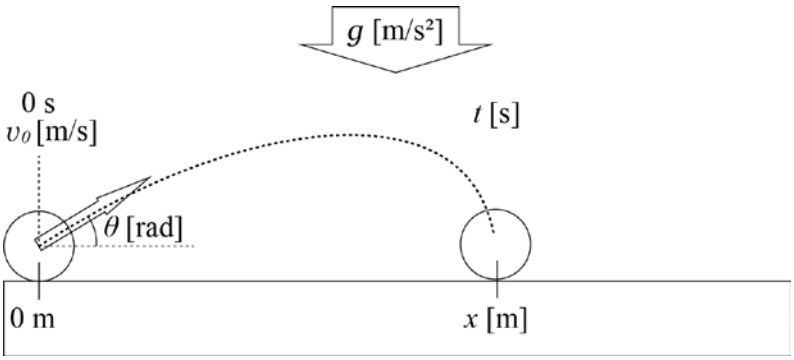


図４　速度に比例する空気抵抗がある場合の斜方投射

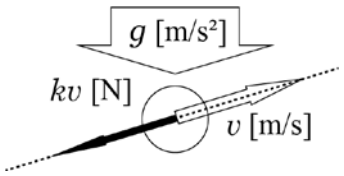


図５　空気抵抗と速度の関係

水平方向、垂直方向の運動方程式は物体の質量をm[kg]、重力加速度をg[m/s^2]、空気抵抗係数をk[kg/s]、物体の水平方向、垂直方向の速度をv_x[m/s]、v_y[m/s] として、それぞれ式（３)、（４）と表すことができる。

$$m\frac{dv_x}{dt} = -kv_x \cdots (3)$$

$$m\frac{dv_y}{dt} = mg - kv_y \cdots (4)$$

また、投射してからt秒後の水平方向・垂直方向それぞれの変位x[m]、y[m] は、式（３)、（４）の微分方程式について、初期値（初速度v_0、初期位置x_0、y_0）を用いて解くことで、次のように表すことができる。

$$x = \frac{m}{k}v_0\cos\theta\left(-e^{-\frac{k}{m}t} + 1\right) + x_0 \cdots (5)$$

$$y = \frac{m}{k}\left(v_0\sin\theta - \frac{mg}{k}\right) \times \left(-e^{-\frac{k}{m}t} + 1\right) + \frac{mg}{k}t + y_0 \cdots (6)$$

３．Processingのコードについて

物理シミュレーション教材を制作するために必要なProcessingを用いたプログラミング及びコードについて説明する。

3-1　物理シミュレーションの画面上での表示設定

物理シミュレーションのPC画面上における見た目の設定について、図６で示すprogram1を例として説明する。Processingにおける描画では、２つのコード（setupとdraw）を基本として使用する。①のsetupはシミュレーション実行後一度だけ読み込まれる関数であり、主にシミュレーションの基本設定や後述する変数の初期値の挿入などを行う。②のdrawはシミュレーション実行後、停止するまで繰り返し読み込まれる関数であり、具体的なシミュレーションの内容、つまり表現を決める部分の設定を行う。２行目のsizeは、実行するシミュレーションのウィ

ンドウサイズを設定する関数であり、第一引数が*x*方向の長さ［px（ピクセル）］、第二引数が*y*方向の長さ［px］である。program1ではsize(100,100);なので、縦横それぞれ100 pxのウィンドウが生成される。また、sizeはsetupの１行目に書く必要がある特殊な関数である。３行目のframeRateは、実行するシミュレーションのフレームレートを設定する関数であり、引数には１秒間に何フレーム実行するかの数を挿入する。ここではframeRate(60);なので、フレームレートが60 frames/sのシミュレーションが実行される。９行目のellipseは楕円を描画する関数であり、第一引数が*x*方向の中心座標、第二引数が*y*方向の中心座標、第三引数が*x*方向の直径、第四引数が*y*方向の直径である。ここでは、ellipse(50,50,25,25);なので、中心*x*座標が50 px、中心y座標が50 px、縦横それぞれの直径が25 pxの円が描画される。７行目のbackgroundはシミュレーションの背景色を設定する関数である。８行目のfillは図形の塗色を設定する関数である。Processingでの塗色の設定は共通して０～255までのRGB値と透明度で行う。第一引数がRGBの赤の値、第二引数が緑の値、第三引数が青の値、第四引数が透明度である。また、第四引数は入力しなければ、不透明な255が適用される。ここではbackground（255,255,255）；なので、背景色は白になり、fill(255,0,0,100);なので、半透明な赤の円が描画される。図７に、program1の実行画面を、表１にprogram1のコード表を示す。

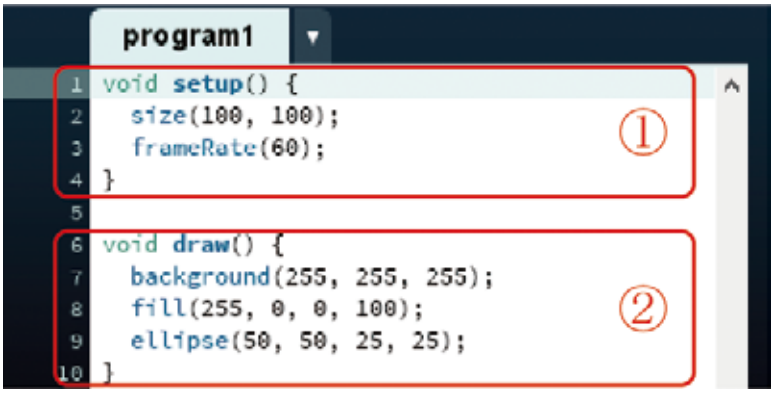


図６　program1

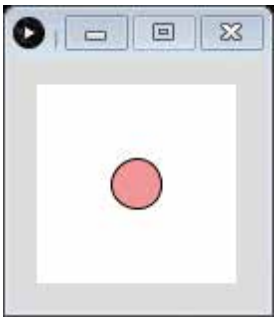


図７　program1の実行画面

表１　program1のコード表

分類	文法	機能
構造	void setup()　{処理}	シミュレーション実行後一度だけ読み込まれる
環境	size(*x*方向の大きさ, *y*方向の大きさ);	ウィンドウサイズを設定する
環境	frameRate(フレームレート数);	フレームレートを設定する
構造	void draw() {処理}	シミュレーションが停止するまで繰り返し読み込まれる
図形	ellipse(*x*方向の中心座標, *y*方向の中心座標, *x*方向の直径, *y*方向の直径);	楕円を描画する
色	background(RGB値);	背景色を設定する
色	fill(RGB値);	図形の塗色を設定する

3-2　変数の導入方法

変数の導入方法について、図８のprogram2を例として説明する。Processingでは、シミュレーション実行中に変化する数値や、複数のオブジェクトが同じ数値を持つ場合にそれらの数値を同時に管理できるように、いくつかの変数が用意されている。①で、シミュレーション全体で使用する変数を定義する。１行目のfloatは小数の変数型であり、続くposxは球の*x*座標を示す変数名である。変数名は自由に決めることができ、position_xなどでも良い。②では変数の初期値を設定している。③では１フレーム毎の変数の変化を表している。ここではposx+=0.1;としているので、１フレーム毎に球は*x*方向に0.1 px移動する運動をする。図９にprogram2の実行画面を示すが、実際には実行ボタンを押すと同時に赤い円が中心の位置から右側に移動する様子が見られる。表２にprogram2のコード表を示す。

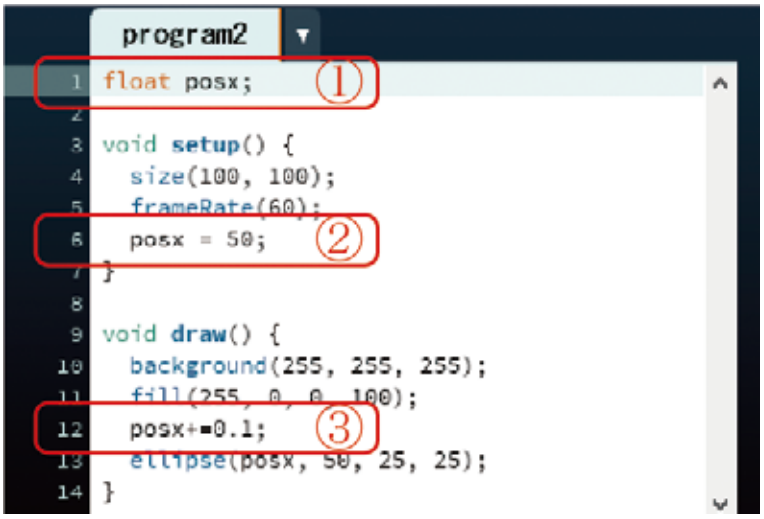


図８　program2

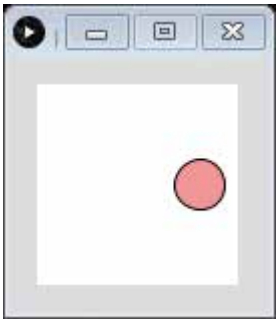


図９　program2の実行画面

表２　program2 のコード表

分類	文法	機能
データ	float変数名	小数の変数型

3-3　繰り返し文の導入方法

繰り返し文の導入方法について、図10のprogram3を例として取りあげる。Processingでは繰り返し同じ処理を行う際に、コーディング量を減らすための機能が備わっており、代表的なものがfor文である。for()内の三つの引数は、変数の初期設定、繰り返しをするか判断するための設定、変数の増減の設定を、それぞれ示している。その後の{}内には、実際に繰り返す処理を入力する。８行目のfor文の設定は、変数はint型のiで初期値が０、変数の最大値がwidth、変数は10ずつの増加であり、処理はline関数という線を引く関数なので、10 px毎に縦線を引いている。intは整数の変数型である。widthはディスプレイウィ

ンドウの幅を格納している変数であり、size関数の第一引数が格納されている。heightはディスプレイウィンドウの高さを格納している変数であり、size関数の第二引数が格納されている。line関数は二点間を結ぶ線を描画する関数であり、第一引数は一つ目の点のx座標、第二引数は一つ目の点のy座標、第三引数は二つ目の点のx座標、第四引数は二つ目の点のy座標である。11行目のfor文も殆ど同じで10 px毎に横線を引いている。これらを組み合わせることによって、グリッド線を引くことができる。program3の実行画面が図11である。また、program3のコード表が表３である。本機能をシミュレーションに導入することによって、任意の時間における物体の座標を画面上から読み取ることができるため、物体がどのような運動をしているのかを確認しやすくなる。

```
program3
1  void setup() {
2    size(100, 100);
3    frameRate(60);
4  }
5
6  void draw() {
7    background(255, 255, 255);
8    for (int i = 0; i < width; i+=10) {
9      line(i, 0, i, height);
10   }
11   for (int i = 0; i < height; i+=10) {
12     line(0, i, width, i);
13   }
14 }
```

図10　program3

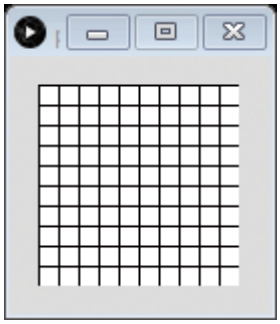


図11　program3の実行画面

表３　program3のコード表

分類	文法	機能
制御	for(変数の設定;繰り返しをするか判断するための設定;変数の増減の設定) {繰り返す処理}	繰り返し処理をする
データ	int変数名	整数の変数型
図形	line(一つ目の点のx座標、一つ目の点のy座標、二つ目の点のx座標、二つ目の点のy座標)	二点間を結ぶ線を描画する
環境	width	ディスプレイウィンドウの幅を格納している変数
環境	height	ディスプレイウィンドウの高さを格納している変数

3-4　ボタンの実装方法

ボタンの実装方法について、図12のprogram4を例として説明する。①では、制御判定が格納される変数を設定している。１行目のbooleanは制御判定が格納されている変数で、続くclickedCountは変数名である。clickedCountはボタンが偶数回押されているか、奇数回押されているかを判断する変数としている。②では、シミュレーション内でのテキストの設定および変数の初期値の設定を行っている。６行目のtextSizeはシミュレーション内のテキストフォントの大きさを設定する関数であり、引数には単位を［px］としたテキストの大きさを挿入する。ここではtextSize(15);なので、テキストサイズは15 pxとなる。７行目のtextAlignはテキストの揃え方を設定する関数であり、第一引数には*x*方向においてどこに寄せるか、第二引数には*y*方向においてどこに寄せるかを決めるコマンドを挿入する。program4ではtextAlign（CENTER,CENTER）；なので、*x*方向と*y*方向のどちらも中央寄せをしている。CENTERは文字を描画する際の配置を決定する変数のうちの一つ

```
program4
1  boolean clickedCount;                                    ①
2
3  void setup() {
4    size(100, 100);
5    frameRate(60);
6    textSize(15);
7    textAlign(CENTER, CENTER);                             ②
8    clickedCount = false;
9  }
10
11 void draw() {
12   background(255, 255, 255);
13   if (clickedCount == true) {
14     fill(255, 0, 0);
15   } else {
16     fill(0, 0, 255);                                     ③
17   }
18   ellipse(50, 50, 100, 100);
19   fill(255);
20   text("sample", 50, 50);
21 }
22
23 void mousePressed() {
24   if (dist(mouseX, mouseY, 50, 50)<50) {
25     if (clickedCount == true) {
26       clickedCount = false;
27     } else {                                             ④
28       clickedCount = true;
29     }
30   }
31 }
```

図12　program4

で、文字を中央揃えする際に使用される。配置を設定する変数には他にも、左揃えにするLEFTや右揃えにするRIGHT等がある。８行目は、clickedCountの初期値を設定していて、その値はfalseなので、ボタンが押された数が０回（偶数回）であることを表している。③では、ボタンが何回押されたかによる条件分岐処理および実際に描画するテキストを設定している。13行目のif文はプログラムがどのコードを実行するかの判断を行う制御文であり、小括弧内には、制御する条件を入力し、中括弧内には条件に適合した際に実行するコードを入力する。15行目のelse文はif文と一緒に用いられ、中括弧内にはif文で定めた条件から漏れたときに実行するコードを入力する。ここではclickedCount==trueの場合、つまりはボタンが押された回数が奇数回の場合ボタンが赤くなり、ボタンが押された回数が偶数回の場合ボタンが青くなるように設定している。20行目のtextは描画するテキストとその位置を設定する関数であり、第一引数がテキストの内容、第二引数がx方向の座標、第三引数はy方向の座標、第四引数がx方向のテキスト描画範囲の大きさ、第五引数がy方向のテキストの描画範囲の大きさである。第四引数と第五引数は必須ではない。ここでは、text(“sample”,50,50);なので、sampleという文字が描画される。また、指定する座標の位置はtextAlignに依存していて、program4では縦横中心寄せとなっているため、sampleの中心部分がx方向に50 px、y方向に50 pxの位置になっている。④では、ボタンが押されたかどうかを判断する処理を行っている。23行目のmousePressedはマウスのボタンが押されると一度だけ呼び出される関数である。24行目のdistは二点間の距離を計算する関数であり、第一引数が一つ目の点のx座標、第二引数が一つ目の点のy座標、第三引数が二つ目の点のx座標、第四引数が二つ目の点のy座標である。ここではdist(mouseX,mouseY,50,50)<50となっているので、円上でマウスがクリックされた場合に関数が読み込まれるように設定されている。mouseXはマウスのx方向の座標が格納されている変数であり、mouseYはy方向の座標が格納されている変数である。program4においてボタンが偶数回押された場合の実行画面が図13で、奇数回押された場合の実行画面が図14である。また、program4のコード表が表４である。本機能を導入することによって、実行画面上で物体の設定を決めるパラメータの変更と確認ができ、円滑にパソコン上でのシミュレーションを行うことができるようになる。

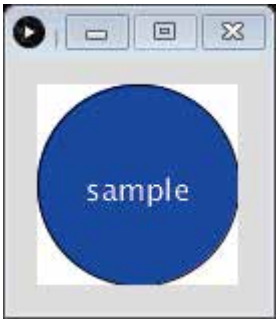


図13　program4の実行画面１

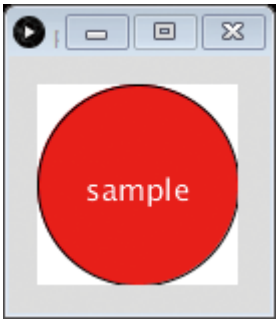


図14　program4の実行画面２

表４　program4 のコード表

分類	文法	機能
データ	boolean変数名	制御判定の変数型
文字	textSize（テキストの大きさ）;	テキストフォントの大きさを設定する
文字	textAlign（x方向のどこに寄せるか, y方向のどこに寄せるか）;	テキストの揃え方を設定する
制御	if（条件）{条件に適合した場合の処理} else {条件に適合しない場合の処理}	プログラムがどのコードを実行するかの判断を行う
文字	text（テキストの内容, テキストのx方向の座標、テキストのy方向の座標, テキストのx方向の描画範囲, テキストのy方向の描画範囲）;	描画するテキストとその位置を設定する
計算	dist（一つ目の点のx座標, 一つ目の点のy座標, 二つ目の点のx座標, 二つ目の点のy座標）;	二点間の距離を計算する

3-5　自作関数の導入方法

自作関数の導入方法について、図15のprogram5を例として説明する。Processingには、機能ごとにコードの管理をするために、自作関数を作ることができる機能が備わっている。①では、自作関数の実行をする。②では、自作した関数の機能を設定する。コードの内容としては、図10（program3）と図15（program5）は同じなので、program5の実行画面は図16になる。表５にprogram5のコード表を示す。本機能を導入することによって、制

図15　program5

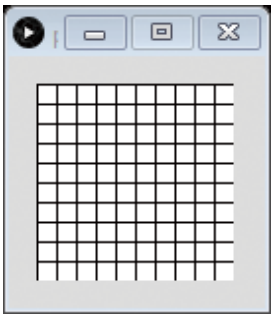


図16　program5の実行画面

作したシミュレーションをプログラム上で管理することができ、カスタマイズがしやすくなる。

表５　program5のコード表

分類	文法	機能
構造	関数名();	自作関数の実行をする
構造	void関数名(){処理}	自作関数の機能を設定する

3-6　画像の挿入方法

画像の導入方法について、図17のprogram6を例として説明する。①では、画像が格納される変数を設定している。１行目のPImageは画像が格納される変数で、続くpanelは変数名である。②では、画像の基本設定を行っている。６行目のloadImageはPImage型の変数に画像をロードする関数で、引数は画像のURLである。７行目のpanel.resizeはPImage画像の大きさを変更する関数で、単位を［px］として、第一引数がx方向の大きさ、第二引数がy方向の大きさである。ここではpanel.resize(width,height);なので、準備したリモコンの画像の大きさはウィンドウサイズと同じとなる。③で示す12行目のimageは画像を描画する関数で、第一引数が描画する画像の変数、第二引数が画像のx方向の座標、第三引数がy方向の座標である。この場合の座標は画像の左上の座標である。ここでは、image(panel,0,0);なので、パネルの画像がウィンドウの左上を基準に表示されている。program6の実行画面が図18である。また、program6のコード表は表６である。本機能を導入することによって、コードを削減することができたり、身近な例（画像）をシミュレーション内に登場させたりすることができるようになる。

図17　program6

図18　program6の実行画面

表６　program6のコード表

分類	文法	機能
データ	PImage変数名	画像の変数型
画像	変数名＝loadImage（“URL”）;	PImage型の変数に画像をロードする
画像	変数名.resize（x方向の大きさ, y方向の大きさ）;	PImage画像の大きさを変更する
画像	image（変数名, x座標, y座標）	画像を描画する

3-7　データの出力方法

データの出力方法について、図19のprogram7を例として説明する。①では、フレーム数を計測する変数を設定している。２行目のintは整数型の変数で、続くcountは変数名で

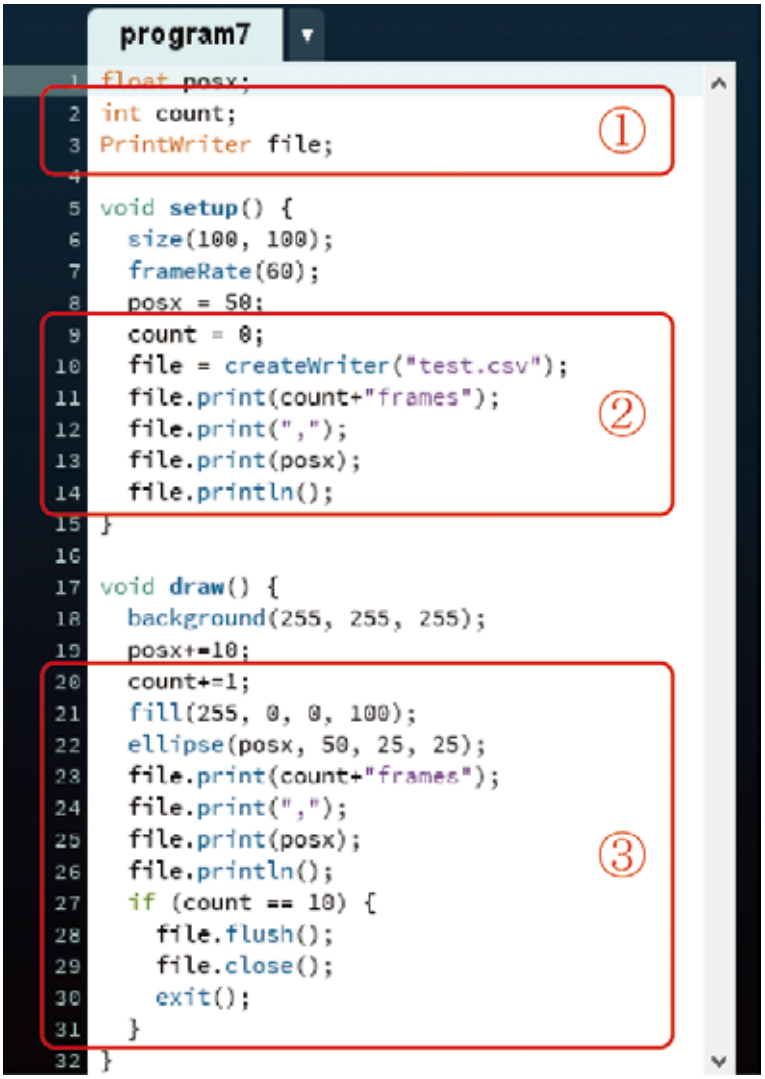


図19　program7

ある。３行目のPrintWriterはテキスト出力オブジェクト名を設定するもので、続くfileはテキスト出力オブジェクト名である。オブジェクトとはデータと処理の集まりのことであり、テキスト出力オブジェクトはテキストを出力するためのデータと処理のことを指す。②では、変数の初期値の設定、テキスト出力オブジェクトの生成、ファイルの１行目の出力値を設定している。10行目のcreateWriterはテキスト出力オブジェクトを生成する関数で、引数は出力するファイルの名前および拡張子である。ここでは、file=createWriter（“file.csv”）;なので、シミュレーションを実行すると、fileという名前のcsvファイルが出力される。11行目のprintは出力するファイルに値を書き込む関数で、引数は書き込みたい値である。また引数を”,”にすると、隣の列に改列することができる。13行目ではfile.print（“posx”）;なので、posxの値がfileというファイルの１行目の２列目に書き込まれる。14行目のprintlnはファイルの改行をする関数である。③では、テキスト出力オブジェクトの設定を行っている。20行目のcount+=１によって、シミュレーションを実行してからのフレーム数を計測している。28行目のflushはファイルにこれまで入力した値を保存する関数である。29行目のcloseはテキスト出力オブジェクトを閉じる関数である。30行目のexitはシミュレーションを終了させる関数である。図20にprogram7の実行画面を、表７にprogram7のコード表を示す。program7の出力値（csvファイルの中身）は図21になる。本機能をシミュレーションに導入することによって、物体の運動を見た目だけでの理解ではなく、数値として定量的な評価をもって理解することができるようになる。

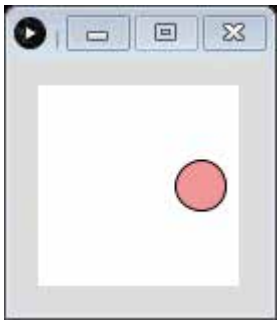


図20　program7の実行画面

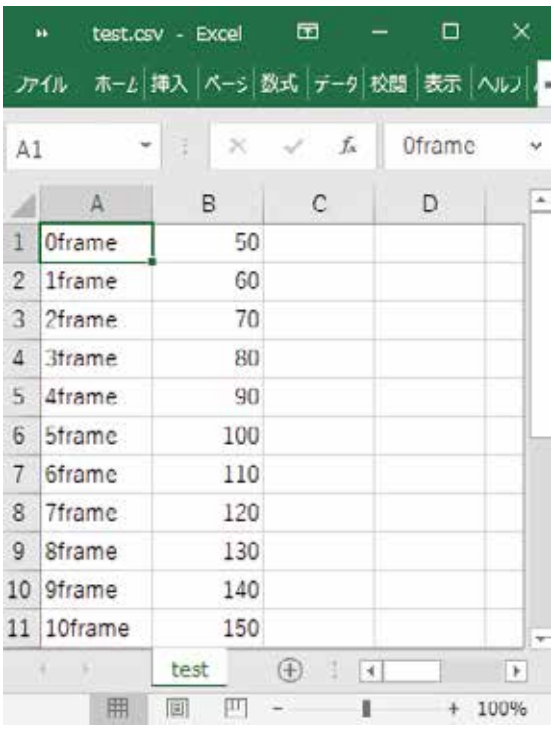


図21　program7の出力結果

表７　program7のコード表

分類	文法	機能
出力	printWriterテキスト出力オブジェクト名	テキスト出力オブジェクト名を設定する
出力	テキスト出力オブジェクト名＝createWriter（出力したいファイル名）	テキスト出力オブジェクトを生成する
出力	テキスト出力オブジェクト名.print（入力する値）	出力するファイルに値を入力する
出力	テキスト出力オブジェクト名.println（）；	出力するファイルの改行をする
出力	テキスト出力オブジェクト名.flush（）；	ファイルにこれまで入力した値を保存する
出力	テキスト出力オブジェクト名.close（）；	テキスト出力オブジェクトを閉じる
構造	exit（）；	プログラムを終了する

3-8　クラスの導入方法

クラスの導入方法について、図22のprogram8を例として説明する。Processingには、同じ機能を持つオブジェクトを簡単に複製できるように、自作でクラスを生成できる機能が備わっている。クラスとは、オブジェクトを生成するための設計図のようなものである。①では、自作のクラスを設定している。１行目のBallは自作したクラス名で、続くb1およびb2はオブジェクト名である。②では、オブジェクトの生成と初期値の代入をしている。③では、オブジェクトを更新している。program8では、b1、b2というオブジェクトが生成されていて、それぞれmoveという自作関数を実行させている。④ではクラスの内容の設定を行っている。16行目のclassはクラスの設定を行うもので、続くBallは自作するクラスの名前である。17行目はクラス内で使用する変数を用意している。18～21行目では、クラスを使用する上で、違いをつける部分の設定を行っている。program8では、形や色は同じであるものの、*x*座標や*y*座標が異なるオブジェクトを生成するので、18行目のようになる。22行目～26行目では、クラスで生成したオブジェクト専用の関数を設定している。図23にprogram8の実行画面を、表８にprogram8のコード表を示す。本機能をシミュレーションに導入することによって、同じルールに従い運動をする物体を簡単に複数用意することができるようになり、異なるシチュエーションの動きを比較して観察することができる。

3-9　方程式の導入方法

方程式の導入方法について、図24のprogram9を例として取りあげる。program9は自由落下を題材としたシミュレーションである。17行目のcount+=１でシミュレーションの実行後のフレーム数を数えている。18行目は球の垂直方向の座標で、自由落下の垂直方向の

```
program8
1  Ball b1, b2;                                  ①
2
3  void setup() {
4    size(100, 100);
5    frameRate(60);
6    b1 = new Ball(50, 30);                       ②
7    b2 = new Ball(0, 60);
8  }
9
10 void draw() {
11   background(255, 255, 255);
12   b1.move();                                   ③
13   b2.move();
14 }
15
16 class Ball {
17   float posx, posy;
18   Ball(float x, float y) {
19     posx = x;
20     posy = y;
21   }                                            ④
22   void move() {
23     posx+=0.1;
24     fill(255, 0, 0, 100);
25     ellipse(posx, posy, 25, 25);
26   }
27 }
```

図22　program8

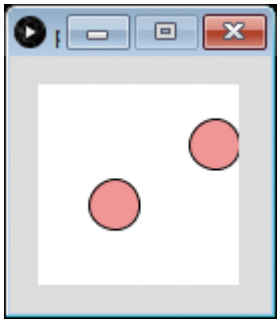


図23　program8の実行画面

表８　program8のコード表

分類	文法	機能
レンダー	PGraphicsオブジェクト名	PGraphicsオブジェクト名を設定する
構造	オブジェクト名＝createGraphics(x方向のレンダーの大きさ、y方向のレンダーの大きさ)；	PGraphicsオブジェクトを生成する
レンダー	オブジェクト名.beginDraw()；	PGraphicsオブジェクトの描画開始の合図をする
レンダー	オブジェクト名.endDraw()；	PGraphicsオブジェクトの描画終了の合図をする

移動距離を示す式である（７）をProcessingでの表現方法に変えたものである。(1.0/2.0)としているのは、(1/2)にしてしまうと整数同士の計算になってしまい、答えも整数で算出されるからである。sqは括弧内の値を二乗する関数である。count/fpsはシミュレーショ

ン実行後のフレーム数をフレームレートで割ったもので、経過時間を示している。program9の実行画面が図25である。本機能を導入することによって、物理分野で学ぶ公式をそのまま代入してシミュレーションを制作することができる。

$$y = \frac{1}{2}gt^2 \cdots (7)$$

```
program9
1  float posy, gravity, count, fps;
2
3  void setup() {
4    size(100, 100);
5    gravity = 9.8;
6    fps = 60;
7    count = 0;
8    frameRate(fps);
9  }
10
11 void draw() {
12   background(255);
13   ball();
14 }
15
16 void ball() {
17   count+=1;
18   posy = (1.0/2.0)*gravity*sq(count/fps);
19   fill(255, 0, 0, 100);
20   ellipse(width/2, posy, 25, 25);
21 }
```

図24　program9

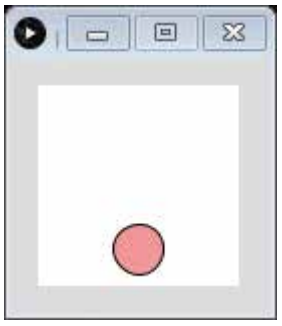


図25　program9の実行画面

3-10　軌跡の実装方法について

軌跡の実装方法について、図26のprogram10を例として説明する。①では、PGraphicsオブジェクトを設定している。PGraphicsとは、Processingで生成した画面の一部を指定した範囲で取り出す機能を持つオブジェクトである。１行目のPGraphicsに続くpgはPGraphicsオブジェクトの変数名である。②では、PGraphicsオブジェクトを生成している。７行目のcreateGraphicsはPGraphicsオブジェクトを生成する関数で、第一引数がPGraphicsオブジェクトのx方向の大きさ、第二引数がy方向の大きさである。ここでは、pg=createGraphics（100,100）;なので、x方向の大きさが100 px、y方向の大きさが100 pxのPGraphicsオブジェクトが生成される。③では、PGraphicsオブジェクトの設定を行っている。14行目のbeginDrawはPGraphicsオブジェクトの描画開始の合図をする関数である。17行目のendDrawはPGraphicsオブジェクトの描画終了の合図をする関数である。PGraphicsオブ

```
program10
1  PGraphics pg;                          ①
2  float posx;
3
4  void setup() {
5    size(100, 100);
6    frameRate(60);
7    pg = createGraphics(100, 100);       ②
8    posx = 50;
9  }
10
11 void draw() {
12   background(255, 255, 255);
13   posx += 10;
14   pg.beginDraw();
15   pg.fill(255, 255, 0, 100);
16   pg.ellipse(posx, 50, 25, 25);        ③
17   pg.endDraw();
18   image(pg, 0, 0);
19   fill(255, 0, 0, 100);
20   ellipse(posx, 50, 25, 25);
21 }
```

図26　program10

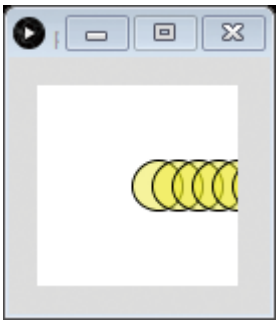


図27　program10の実行画面

表９　program10のコード表

分類	文法	機能
構造	クラス名　オブジェクト名;	自作クラスのオブジェクトを設定する
構造	オブジェクト名＝newクラス名（クラスで設定した値);	自作クラスのオブジェクトの初期値を設定する
構造	オブジェクト名.関数名（);	自作クラス内の自作関数を実行する
構造	classクラス名　{クラスの設定}	自作のクラスの設定をする

ジェクトの設定はPGraphicsオブジェクト名.関数で行い、18行目のようにimageで出力される。図27にprogram10の実行画面を、表９にprogram10のコード表を示す。本機能をアニメーションに導入することによって、物体の運動の軌跡を画面上に残すことができ、運動の概要を掴むことができる。

3-11　斜方投射の実装方法

斜方投射の実装は、２及び3-1から3-10までの内容を用いて行った。空気抵抗なしの斜方投射を考える。本シミュレーションにおいて、投射する球の中心x座標、y座標をそれぞれ、posx[m]、posy[m]、投射する球の初期中心x座標、y座標をそれぞれ、posx0[m]、

posy0[m]、初速をspeed[m/s]、地面と投射する方向がなす角をtheta[rad]、運動を開始してからのフレーム数をcount[frames]、フレームレートをfps[frames/s]、重力加速度をgravity[m/s^2] とする。以上の値をそれぞれ式（１）、（２）に代入すると、Processing上の空気抵抗なしの斜方投射は次のように表すことができる（付録のprojectile_motionフォルダのclassファイル24行目、25行目）。

$$\mathrm{posx} = \mathrm{speed} \times \cos(\mathrm{theta}) \times \left(\frac{\mathrm{count}}{\mathrm{fps}}\right) + \mathrm{posx0}$$

$$\mathrm{posy} = -\mathrm{speed} \times \sin(\mathrm{theta}) \times \left(\frac{\mathrm{count}}{\mathrm{fps}}\right) + \frac{1}{2} \times \mathrm{gravity} \times \left(\frac{\mathrm{count}}{\mathrm{fps}}\right)^2 + \mathrm{posy0}$$

速度に比例する空気抵抗ありの斜方投射について考える。本シミュレーションにおいて、投射する球の中心x座標、y座標をそれぞれposx[m]、posy[m]、投射する球の初期中心x座標、y座標をそれぞれ、posx0[m]、posy0[m]、空気抵抗係数をconstant[kg/s]、球の質量をweight[kg]、初速度をspeed[m/s]、地面と投射する方向がなす角をtheta[rad]、投射してからのフレーム数をcount[frames]、フレームレートをfps[frames/s]、重力加速度をgravity[m/s^2] とする。以上の値をそれぞれ式（５）、（６）に代入すると、Processing上の速度に比例する空気抵抗ありの斜方投射は次のように表すことができる（付録のprojectile_motionフォルダのclassファイル21行目、22行目）。

$$\mathrm{posx} = \frac{\mathrm{weight}}{\mathrm{constant}} \times \mathrm{speed} \times \cos(\mathrm{theta}) \left(-\mathrm{e}^{-\frac{\mathrm{constant}}{\mathrm{weight}} \times \frac{\mathrm{count}}{\mathrm{fps}}} + 1\right) + \mathrm{posx0}$$

$$\mathrm{posy} = \frac{\mathrm{weight}}{\mathrm{constant}} \left(\mathrm{speed} \times \sin(\mathrm{theta}) - \frac{\mathrm{weight} \times \mathrm{gravity}}{\mathrm{constant}}\right) \times \left(-\mathrm{e}^{-\frac{\mathrm{constant}}{\mathrm{weight}} \times \frac{\mathrm{count}}{\mathrm{fps}}} + 1\right)$$

$$+ \frac{\mathrm{weight} \times \mathrm{gravity}}{\mathrm{constant}} \times \frac{\mathrm{count}}{\mathrm{fps}} + \mathrm{posy0}$$

これらの定義式をプログラム画面上で入力し、実行した結果、出力される画面を図28に示す。図28では空気抵抗がある場合における投射角度が30°と45°の運動の比較を行っている。高等学校の物理の授業では空気抵抗がない場合、斜方投射で最も遠くに飛ばすことができる投射角度は45°であることを学ぶが、この図から、空気抵抗がある場合、45°より30°の方がより遠くに飛ぶことが分かる。この知識は、遠投で用いられている。図29のように、画像を挿入するカスタマイズによって、斜方投射を遠投という身近な例を用いて学習することができる。

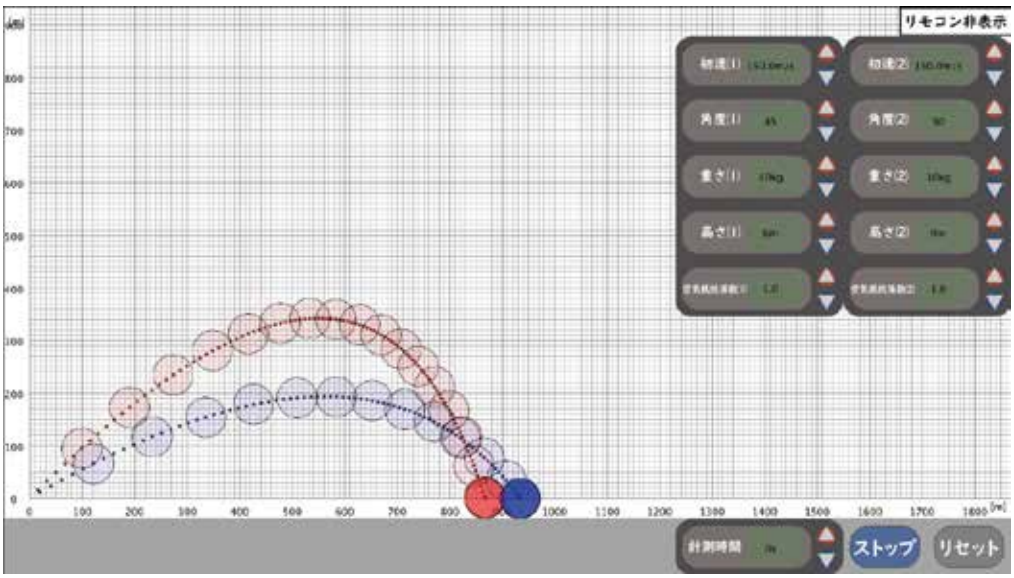


図28　projectile_motionの実行画面

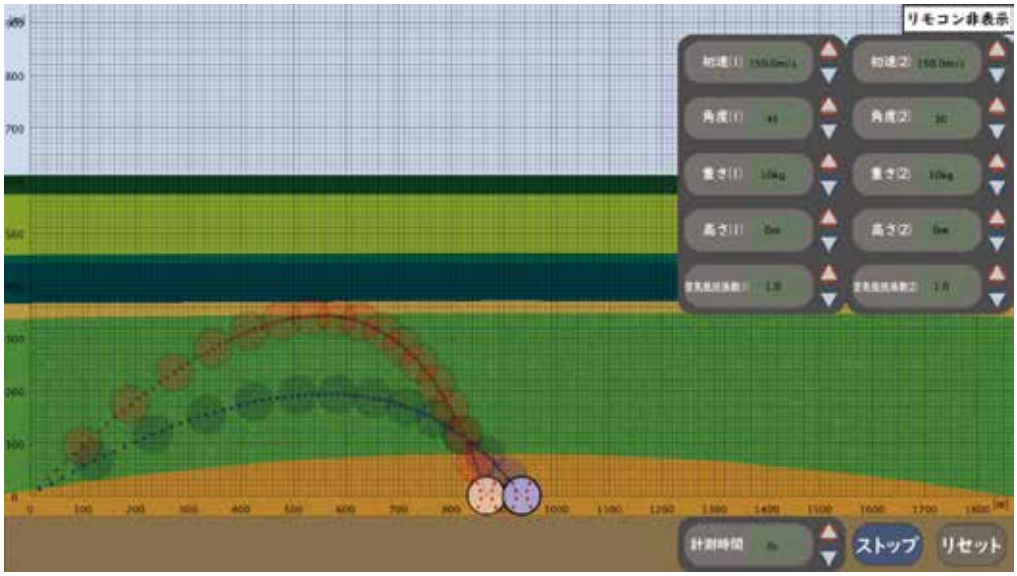


図29　斜方投射のシミュレーションについて、遠投をイメージしてカスタマイズした例

4．運用方法と期待される効果

本研究で制作した物理シミュレーション教材について、その運用方法と期待される効果について考察する。

まず運用方法の一つ目として、本教材をそのまま利用することを考える。ここで想定している対象者は物理を専攻している高校生であり、教育課程における位置づけは力学である。力学の中でも斜方投射の分野を取り上げているため、パラメータの設定を工夫することによって、球の鉛直投げ上げや自由落下の様子も観察することができる。そこで、生徒

に教科書に載っているパラメータを入力させ運動の様子を観察させることで、パラメータの変化が物理現象にどのような影響を与えるかについて動きを持ったイメージをもとに理解させることが可能となる。また発展的な運用方法としては、シミュレーションで得られた数値から初速度を求める演習問題を解かせることなども考えられる。二つ目は、本教材をカスタマイズして使用する方法である。本教材は、プログラム上で目的毎に関数を分けていたり、画像を簡単に差し替えることができるため、物体の運動を変化させたり、オリジナルな画像を取り入れたシミュレーションを比較的容易に実現することができる。運動方程式を変更して斜方投射以外の運動に切り替えることも可能であるし、斜方投射のシミュレーションを見せる際に、投射する球を野球ボールに変え、背景を外の風景に変えるなどして身近な現象としてイメージさせることもできる。こうしたカスタマイズは教員がちょっとした手を加えるだけで、状況に応じた活用、またオリジナルな表現で教員独自のシミュレーションを実現できるとして、教員の積極的な利用を促せるものと思われる。

次に、上記のような運用をすることによって期待される効果について考察する。一つ目は、物理現象と数式を関連付けて理解することができるようになることである。本教材には、グリッド線によって座標を読むことができる機能や、様々な数的データをcsvファイルとして出力できる機能が備わっている。これらの機能を使って、値を変えながら物理現象を観察することによって、物理現象と変数の移り変わりの関連性、つまり物理法則についての理解を促進できるのではないかと考えている。二つ目は、物理現象を身近な例と関連付けての理解を促すことである。本教材は前述したように、簡単に画像を差し替えることができ、身近な例を取り上げることができる。これにより、物理現象を教科書上のものに留めるのではなく、身近な現象とも関連付けて理解することができるようになり、興味・関心につなげることも考えられる。

5．まとめ

物理分野における斜方投射を題材として、パラメータを変化させることができ、且つデータを出力することができるシミュレーション教材を、Processingを用いて開発した。本文では、学校現場等で、教員がカスタマイズして利用する事を想定し、シミュレーションにおいて必要となる各機能を実現するProcessingのプログラムについて説明した。その中で取り上げたグリッド線やcsvファイルで物理現象にかかわる物理量を出力できる機能は、物理現象の観察と物理量の取得を並行して行うことを可能とし、実験と近い形での学びにつなげられる。シミュレーションでは実際の実験と違って準備や片付けにかける時間を短縮できる。また異なる数値の物理量を入力した場合の現象の変化について、同一画面上での映像の比較として出力できることも、物理シミュレーション教材の特徴と言える。

6．参考文献等

［1］ 渡辺克己、2011、「日本の理科教育の現状と課題―理科好きな子どもを育てるために―」、『北里大学一般教育紀要』16号、91-106.

［2］ 科学技術振興機構・理数学習支援センター、2013、「平成24年度中学校理科教育実態調査集計結果（速報）」、https://www.jst.go.jp/cpse/risushien/secondary/cpse_report_016.pdf（2021年９月23日）.

［3］ 文部科学省初等中等教育局情報教育・外国語教育課、2021、「令和２年度学校における教育の情報化の実態等に関する調査結果（概要）」、https://www.mext.go.jp/content/20211122-mxt_shuukyo01-000017176_1.pdf（2021年11月６日）.

［4］ PhET Interactive Simulations、https://phet.colorado.edu/ja/（2021年９月27日）.

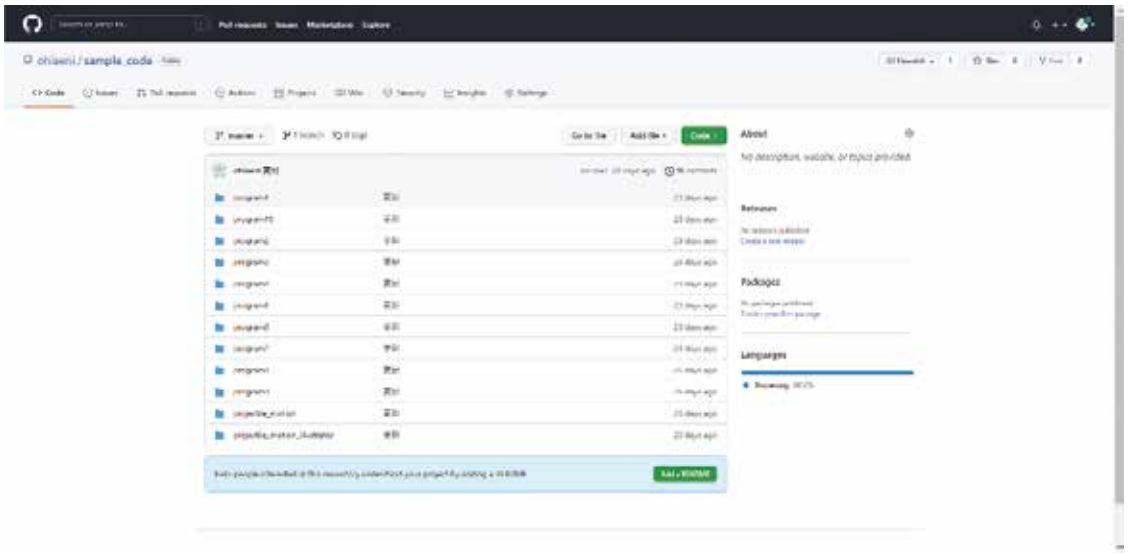
［5］ Processing、https://processing.org/（2021年11月29日）.

［6］ 増井壮太・片岡佐知子・中村元彦・常田琢・松山豊樹、2019、「高校物理における空気抵抗の正しい理解―風洞実験、理論解析、落下実験とデータ解析を通して―」、『次世代教員養成センター研究紀要』５号、37-42.

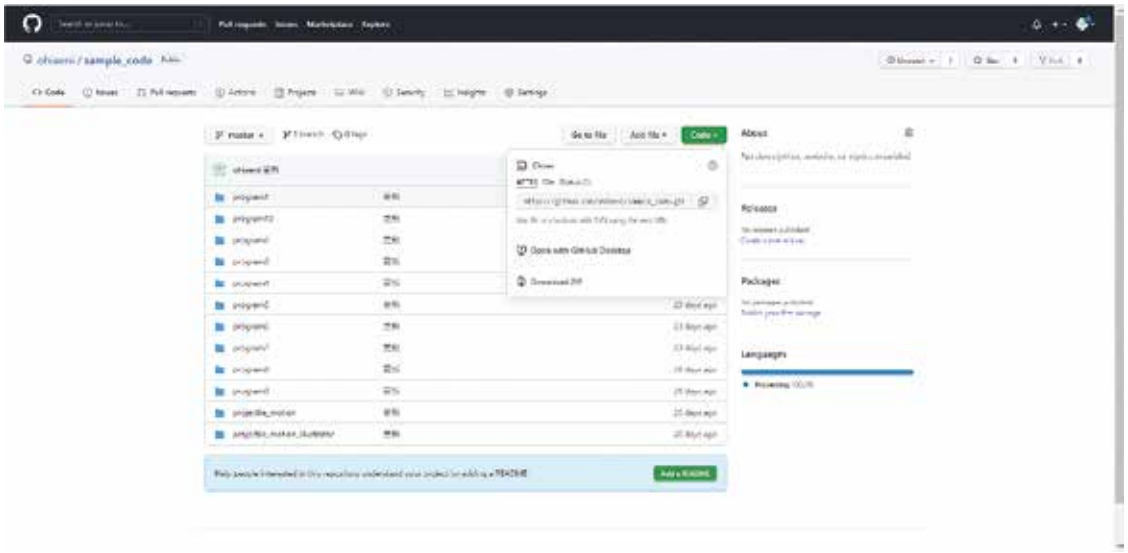
7．コードのダウンロードの手順

サンプルコードの配布用URL＝https://github.com/ohiaeni/sample_code.git

① 配布用URLをクリックし、図のページにアクセスする。



② codeタブを展開し、Download ZIPボタンを押してzipファイルをダウンロードする。



③ ダウンロードしたzipファイルを展開する。