2年 情報実習⑦,⑧

～～

**Ｓ２　　　　　　番**

**名前：**

**提出期限**

**平成　　　年　　　月　　　日**

1. 理論

前回までの実習では，*x*軸上での運動を*xy*平面での運動に拡張するために，*x*軸と*y*軸それぞれ別々に計算を行う方法を使った。これをさらに3次元空間での運動に拡張することを考えたとき，*z*軸を加えた*xyz*空間となるが，*x*座標*y*座標*z*座標を別々に計算するとプログラムの長さが*x*座標だけのものと比べて単純に3倍になってしまい，可読性（プログラムの読みやすさ）が著しく低下する。これを防ぐためにベクトル演算を利用する。

1. ベクトルとは

ベクトルは，高校物理では「向きと大きさをもつ量」として考えるが，計算上は*n*次元ベクトルであれば*n*個の実数の集まりとして扱う。例えば，2次元ベクトルは2つの実数の集まりとして表され，2つの実数を*x*，*y*とすると(*x*, *y*)と表すことができる。これが3次元ベクトルとなれば*xyz*を使って(*x*, *y*, *z*)と表せる。

1. ベクトルの和

2次元ベクトルの場合，ベクトルの和（合成ベクトル）は，幾何学的（図として考える）には平行四辺形の対角線と考えることができる。しかし，計算上はベクトルの各要素の和をそれぞれ求めたに過ぎないため，3次元ベクトルでは次のような式で表せる。

1. ベクトルの差

ベクトルにマイナスが付いた（－1をかけた）場合，大きさは同じで原点対称なベクトルとなる。この性質を使って下図のようにベクトルの差をベクトルの和を使って求めることができる。



図で表すとやや複雑にみえるベクトルの差の求め方であるが，計算上は各要素の差を求めたに過ぎないため，3次元ベクトルでは次のような式で表せる。

（問1）　3次元ベクトル，について，次の各問いの答えをコンピュータを使用して求めよ。なお，Unity C#において3次元ベクトルはVector3型として扱うことができる。

(1) 　　(2) 　　(3)

1. ベクトルの大きさ

ベクトルの大きさは絶対値の記号を使って表し，次の図のように（　三平方　）の定理を使って求めることができる。



また，3次元ベクトルでは次のような式で求めることができる。

大きさ1のベクトルは特別に「（　単位　）ベクトル」と呼び，向きだけを計算に使いたい場合に便利である。

（問2）　3次元ベクトル，について，次の各問いの答えをコンピュータを使用して求めよ。なお，「ベクトルの大きさ」は英語で「magnitude of a vector」となり，Unity C#ではVector3クラスのmagnitudeパラメータでベクトルの大きさを取得できる。

(1) 　　　　　　(2) 　　　　　　(3)

1. ベクトルの積

2つのベクトルの積（掛け算）をベクトルの和と同様に各要素の積として求めた場合，結果は何の意味も持たないベクトルとなってしまう。これは，ベクトルに積の演算が存在しないことを意味する。そこで，ベクトル演算で積の代わりに利用される実数倍，内積，外積の３種類の演算手法を紹介する。

* 1. ベクトルの実数倍

ベクトルに対して，ベクトルではないただの実数（スカラーという）を掛けた場合，掛けた実数の分だけ大きさのみが変化し，向きは変わらない（右図）。

計算上はベクトルの各要素に実数をそれぞれ掛けたに過ぎないため，3次元ベクトルでは次のような式で表せる。

※実数倍は別名「スカラー倍」とも呼ばれる。後述の内積の別名である「スカラー積」と混同しやすいので注意すること。

* 1. 内積

内積はベクトルの各要素の積をすべて合計する計算方法のことである。演算記号として「・」が使用され，計算結果はスカラーとなるため，別名「（　ドット　）積」，「スカラー積」と呼ばれる。3次元ベクトルでは次のような式で表せる。

|  |  |
| --- | --- |
|  | ・・・・・・・・① |

また，内積は2つのベクトルのなす角をθとすると，次のような式でも表すことができる。

|  |  |
| --- | --- |
|  | ・・・・・・・・② |

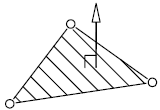
①，②式を使うと，2つのベクトルの相対的な角度を簡単に求めることができるため，あらゆる工業分野において内積の演算は多用される。

より，

※とはの逆関数であるを表している。と書くこともある。

* 1. 外積

外積は内積と同様にベクトルの演算手法のひとつとしてよく使われる。2つのベクトルに対して垂直なベクトルを求めることができるため，面に対する法線ベクトルや，ねじれた位置に生じる力などを求める際によく使われる。なお，演算記号として「×」が使用され，計算結果はベクトルとなるため，別名「（　クロス　）積」，「ベクトル積」と呼ばれる。



法線ベクトル

（問3）　次の各問いの答えをコンピュータを使用して求めよ。なお，Unity C#で内積はVector3クラスのDotメソッド，外積はVector3クラスのCrossメソッドで求めることができる。

(1) ，

1. ベクトルを使った物体の位置の変更

* 物体を（例？）

1. ベクトルを使った物理シミュレーション

* 自由落下のシミュレーション

*x*軸上を運動する2つの物体の衝突を考える。衝突のシミュレーションを行うにあたって，まず等速運動を行うためのスクリプトを作成する必要がある。

|  |
| --- |
| 1. public class MyScript : MonoBehaviour { 2. public float v; 3. void FixedUpdate () { 4. this.transform.Translate(v \* Time.deltaTime, 0f, 0f); 5. } 6. } |

また，衝突後の速度に関する式③,④は，プログラミング言語（Unity C#）で次のように書ける。

* バネ振り子のシミュレーション

*xy*平面上を運動する2つの物体の衝突を考える。まず等速運動を行うためのスクリプトを*x*, *y*それぞれの向きに対応できるように修正する。

|  |
| --- |
| 1. public class MyScript : MonoBehaviour { 2. public float vx, vy; 3. void FixedUpdate () { 4. this.transform.Translate(vx \* Time.deltaTime, vx \* Time.deltaTime, 0f); 5. } 6. } |

また，衝突に関する演算も同様に修正する。衝突に関する演算は*x*軸方向と*y*軸方向を分けて考え，それぞれ求めればよい。

さらに*z*軸方向を加えると3次元空間での衝突をシミュレートできるが，プログラムが非常に見難くなるため，ベクトルを使って計算するのが一般的である。

* 万有引力のシミュレーション

*xy*平面上を運動する2つの物体の衝突を考える。まず等速運動を行うためのスクリプトを*x*, *y*それぞれの向きに対応できるように修正する。

|  |
| --- |
| 1. public class MyScript : MonoBehaviour { 2. public float vx, vy; 3. void FixedUpdate () { 4. this.transform.Translate(vx \* Time.deltaTime, vx \* Time.deltaTime, 0f); 5. } 6. } |

また，衝突に関する演算も同様に修正する。衝突に関する演算は*x*軸方向と*y*軸方向を分けて考え，それぞれ求めればよい。

さらに*z*軸方向を加えると3次元空間での衝突をシミュレートできるが，プログラムが非常に見難くなるため，ベクトルを使って計算するのが一般的である。

1. 実験環境

|  |  |  |  |
| --- | --- | --- | --- |
| 名称/Name | 種別/category | 会社名/Vendor | ﾊﾞｰｼﾞｮﾝ/Ver. |
| macOS | オペレーティングシステム | Apple | 10.12 (Sierra) |
| Unity | ゲームエンジン | Unity Technologies | 2017.1 |

1. 課題

課題1

課題2

物体Aと原点との中点に物体Bを配置するためのプログラム。

課題3

三辺のうち二辺の座標が与えられた場合の残った辺の長さを求めるプログラム

課題4

△ABC の辺 AB を 2:3 に内分する点を P とするとき，CPをOA，OBをOA, OBを用いて表せ。

課題5

重心の位置に移動するプログラム 重心は(AB+AC+AD)/3で求まる

課題6

４つの物体が同一平面上にあるかを３点の外積の単位ベクトルが等しいかどうかで判定する。

課題7

考察