2年 情報実習③,④

～物理演算の基礎～

**Ｓ２　　　　　　番**

**名前：**

**提出期限**

**年　　　月　　　日**

1. 理論
2. 無次元化

数値計算を行う際には，**無次元化**を行う必要がある。無次元化とは物理量（単位を持つ量）を数値（単位の無い量）に変換することをいう。数学の計算において一般に行われる作業でもある。

＜無次元化のポイント＞

|  |
| --- |
| 物理量　→　無次元化　→　数値 |

量の値は**数値と単位の積**で表され，単位に乗ぜられる数値は，量のその単位における量的な大きさを表す。数値と単位の間には通常1字分の空白（スペース）を挿入する。無次元の量の単位記号は1であるが，この記号は省かれる。

＜物理量，数値，単位の関係＞

|  |
| --- |
| 物理量　＝　数値　×　単位 |

無次元化は物理量を基準となる物理量によって除する（割り算する）ことで行われる。

＜無次元化の公式＞

|  |
| --- |
|  |

例）たとえば，1.23 mm を無次元化する場合，1 mmを基準とすれば，

となるが，1 mを基準とすれば，

となる。

したがって，何を基準となる物理量として設定するのが適切であるかは状況に応じて考える必要がある。例えば，「惑星の動き」のような巨大な現象をメートル基準で扱うのは不適切である。

Unityにおいて扱われる数値は無次元であるが，無次元化の基準となる物理量が何であるかを理解しておく必要がある。デフォルトの単位スケールは以下の通り。時間以外についてはRigidbodyコンポーネントが持つ重力加速度プロパティに起因する。なお，重力加速度は変更できる。

|  |  |  |
| --- | --- | --- |
| 物理量 | スクリプト上の用途 | 基準となる物理量 |
| 変位 | Transform.positionなど | 1 m |
| 力 | Rigidbody.AddForceなど | 1 N |
| 質量 | Rigidbody.mass | 1 kg |
| 速度 | Rigidbody.velocityなど | 1 m / s |
| 時間 | Time.time など | 1 s |

1. 物体の位置を変える

物体の位置の変更方法には，Rigidbodyコンポーネントを使わない方法と使う方法の２種類がある。

1. Rigidbodyコンポーネントを使わない方法

（例1）物体のpositionを(0, 10, 0)に変更する。

|  |
| --- |
| this.transform.position = new Vector3(0f, 10f, 0f); |

（例2）物体をx座標正の向きに1 m動かす

|  |
| --- |
| this.transform.Translate(1f, 0f, 0f); |

（例3）物体をx座標正の向きに1 m/sの速さで動かす

|  |
| --- |
| void Update () {  this.transform.Translate(1f \* Time.deltaTime, 0f, 0f);  } |

※Time.deltaTimeには直前のフレームの実行にかかった時間が入っており，これに値を掛けることで1秒あたりの変化量として取り扱うことができる。

1. Rigidbodyコンポーネントを使う方法

（例4）物体へx座標正の向きに10 Nの力を加える。

|  |
| --- |
| void FixedUpdate () {  this.GetComponent<Rigidbody>().AddForce(10f, 0f, 0f);  } |

※Rigidbodyに関するメソッドは，FixedUpdateメソッド内で実行する必要がある。

（例5）物体をx座標正の向きに3 m/sの速さで動かす。

|  |
| --- |
| void FixedUpdate () {  this.GetComponent<Rigidbody>().velocity = new Vector3(3f, 0f, 0f);  } |

（例6）物体を正面の向き(ローカル空間のz軸正の向き）に3 m/sの速さで動かす。

|  |
| --- |
| void FixedUpdate () {  this.GetComponent<Rigidbody>().velocity = this.transform.forward \* 3f;  } |

1. ローカル空間とグローバル空間

画面左上のLocal, Globalボタンを切り替えることで，選択している物体の向きに基づくローカル空間と，シーン全体で固定されているグローバル空間の２つを切り替えることができる。

Unityではスクリプト上でローカル空間での物体の向きが決められており，ローカル空間のz軸正の向きが前( forward )，x軸正の向きが右( right )，y軸正の向きが上( up )と定められている。

1. 軌跡の表示

Unityで物体が移動した軌跡を表示するには（　Trail Renderer　）コンポーネントを使用する。Timeプロパティは軌跡の表示時間，Widthプロパティは軌跡の線の太さを表す。

1. 角度の単位

角度の表し方には1周を360°とする度数法(Degree)と，1周を半径1の円の円周である2πとする弧度法(Radian)の二種類がある。度数法から弧度法へ変換するには値を360で割ってから2πを掛ければよい。また逆に弧度法から度数法へは値を2πで割ってから360を掛ければよい。  
なお，， となる。

例）

Unityにおいて三角関数を扱う場合，MathfクラスのSinメソッドやCosメソッドが利用できる。これらは角度の単位として弧度法を使うため注意が必要である。

例）sin90°はMathf.Sin(90f \* Mathf.Deg2Rad)，

　　cos90°はMathf.Cos(90f \* Mathf.Deg2Rad)と書ける

　　※Deg2Radは0.01745329を表す定数であり， の近似値である。

1. 往復運動

往復運動のシミュレーションには三角関数を用いるのが一般的である。

（例7:MoveBackAndForth）x軸上を振幅5 mで1秒間に1往復(1 Hz)させる

|  |
| --- |
| public class MyScript : MonoBehaviour {  float x;  // Use this for initialization  void Start () {  x = 0f;  }    // Update is called once per frame  void Update () {  x = x + 360f \* Mathf.Deg2Rad \* Time.deltaTime;  this.transform.position = new Vector3 (5f \* Mathf.Sin(x), 0f, 0f);  }  } |

1. 円運動

2つ軸の軸上で往復運動を行い，位相差を90°にすると円運動となる。

（例8:MoveACircle）xy平面上で半径5 mの円運動を1秒間に1周の回転速度で行う

|  |
| --- |
| public class MyScript : MonoBehaviour {  float x, y;  // Use this for initialization  void Start () {  x = 0f;  y = 90f \* Mathf.Deg2Rad;  }    // Update is called once per frame  void Update () {  x = x + 360f \* Mathf.Deg2Rad \* Time.deltaTime;  y = y + 360f \* Mathf.Deg2Rad \* Time.deltaTime;  this.transform.position = new Vector3 (5f \* Mathf.Sin(x), 5f \* Mathf.Sin(y), 0f);  }  } |

1. リサジュー図形

円運動において，2軸の位相差や回転速度（周期）の比率を変化させるだけで，多様な種類の図形を描くことができる。これらはリサジュー図形と呼ばれる。

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 |  |
| Unity | ゲームエンジン | Unity Technologies |  |

1. 課題

課題1

シーンKadai1を開き，Spaceキーが押された回数数え，Spaceキーが押されるたびにその回数をConsole画面に表示するスクリプト(CountSpaces)を作成せよ。

ヒント：aを1増やすには a = a + 1;と書けば良い。また，短縮して a++;と書いても良い。

課題2

シーンKadai2を開き，開始5秒後に，スクリプトが張られた物体を赤色にするスクリプト(TurnRed)を作成せよ。

ヒント：開始してからの経過時間はTime.timeで取得できる。Time.timeが5より大きくなったタイミングで赤にすれば良い。

課題3

Cubeをposition(0, 0, 0)の位置に設置したシーン (Kadai3)について，キーボードのXキーを押すとCubeのpositionを(1, 0, 0)に，Zキーを押すとCubeのpositionを(-1, 0, 0)にするスクリプト(JumpOneMeter)を作成せよ。

課題4

Cubeをposition(0, 0, 0)の位置に設置したシーン (Kadai4)について，キーボードの右矢印(RightArrow)キーを押すとCubeをx軸正の向きに1 m移動し，左矢印(LeftArrow)キーを押すとCubeをx軸負の向きに1 m移動するスクリプト(KeepJumping)を作成せよ。

ヒント：this.transform.Translateを使う

課題5

Cubeをposition(-10, 0, 0)の位置に設置したシーン (Kadai5)について，Cubeを毎秒10 mの速さでx軸正の向きに移動させ，物体のx座標が10を超えた場合は-10に戻し，-10から10へ繰り返し移動するスクリプト(KeepMoving)を作成せよ。

課題6

Plane上にSphereを設置し，SphereにRigidbodyコンポーネントを取り付けたシーン(Kadai6)について，Sphereをキーボードの矢印キーを使って平面上を自由に移動できるようなスクリプト(MoveFreely)を作成せよ。なお，移動速度は問わない。※AddForceを使用する

課題7

Sphereをposition(0, 10, 0)の位置に設置し，Rigidbodyコンポーネントを使って自由落下させるシーン(Kadai7)について，次の機能を満たすスクリプト(Portal)を作成せよ。

①Sphereのy座標が0より小さくなった時に，y座標を10に戻す。

②Spaceキーが押された時，Sphereの速度を0 m/sにする。

課題8

Unityを使って次表の空欄の値を求めよ。なお，小数点以下3桁まで求めよ。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 0° | 30° | 90° | 120° |
| sin |  |  |  |  |
| cos |  |  |  |  |

課題9

Cubeを1つ設置したシーン(Kadai9)について，Cubeに**y軸**上で振幅3 m，3秒間に1往復の頻度で往復運動をさせるスクリプト(MoveBackAndForth2)を作成せよ。

課題10

Cubeを1つ設置したシーン(Kadai10)について，Cubeに**xz平面**上で半径5 m，1秒間に0.5回転の円運動(MoveACircle2)をさせよ。

課題11

リサジュー図形の中で興味深いと感じたものを１つ選び，図形のスケッチとその図形を描くための条件を記録せよ。

課題12

Sphereをy軸正の向き（真上）に10 m/sの速さで打ち出した場合，最高到達点の高さは何mであるか。ただし，空気抵抗は考えない(Drag = 0)ものとする。シミュレーション結果と理論値を比較せよ。

1. 考察

本実験で行った円運動のシミュレーションは厳密には正確な円（真円）を描いていない。正確な円にならない理由と，できるだけ正確な円を描くための方法とその限界について考察せよ。