Unity 講習資料２

１．変数

プログラムには「変数」という概念があります。その解説をします。

変数とは、データを入れる箱です。その中には数値だったり文字だったり、Unity ではオブジェクトなんかも入れられます。しかし、中に入れられるデータの種類は決まっており、数値を入れる箱に文字を入れることはできません。この中に入れられるデータの種類は「型」と呼ばれます。

変数にできることは主に①宣言、②代入、③参照の３つです。

① 宣言

まず初めに、「こんなデータが入れられるこんな名前の箱をつくります！」ということ。変数は宣言しないと使えません。

具体的には

int A ;

のように書きます。これは、int 型（整数を入れられる型）の変数 A を作る、という意味です。

② 代入

宣言した変数のなかにデータを入れ、書き換えること。

具体的には

A = 2;

のように書きます。ここでの「＝」は数学的な＝とは違い、左のデータを右の変数に代入する、という意味です。そのため、

A = 2;

A = 3;

のように書いても問題ありません。（このプログラムは、A の中身を２にした後、３にする、という意味です。そのため、現在の A の値は３です。２というデータは消えます）

また、宣言と代入は同時に行えます。

int A = 3;

と書けば初めから３が入った変数 A がつくれます。

③ 参照

変数は中身のデータと同じようにふるまいます。

例えば、

int A = 2;

int B;

B = A + 3;
と書いたとき、B の値は 5 になります。

2．変数を使う

変数を使ってみましょう。

① プログラムを以下のように書き換え、保存します。（背景が変わったのは気にしないでください）

```
Player.cs
Assets > Player.cs > Player > Update()
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    0 references
5   public class Player : MonoBehaviour
6   {
        5 references
7       public float Speed;
8       // Start is called before the first frame update
        0 references
9       void Start()
10      {
11          Speed = 0.5f;
12      }
13
14      // Update is called once per frame
        0 references
15      void Update()
16      {
17          if (Input.GetKey(KeyCode.RightArrow))
18          {
19              GetComponent<Rigidbody>().AddForce(Speed, 0f, 0f);
20          }
21          if (Input.GetKey(KeyCode.LeftArrow))
22          {
23              GetComponent<Rigidbody>().AddForce(-Speed, 0f, 0f);
24          }
25          if (Input.GetKey(KeyCode.UpArrow))
26          {
27              GetComponent<Rigidbody>().AddForce(0f, 0f, Speed);
28          }
29          if (Input.GetKey(KeyCode.DownArrow))
30          {
31              GetComponent<Rigidbody>().AddForce(0f, 0f, -Speed);
32          }
33
34      }
35  }
```

float：小数を扱う型。この型に代入する数値の最後には「f」がつきます。

public：宣言の前に public とつけると、このスクリプトの外（別のスクリプトやインスペクターなど）からの操作を受け付ける変数ができます。逆に、操作を受け付けない変数は private と書きます。省略すると private になります。

② 再生ボタンを押すと、先ほどより遅いスピードで球が動きます。

これは与える力が 1 から Speed（ここでは 0.5）になったからです。

③ Speed = 0.5f;

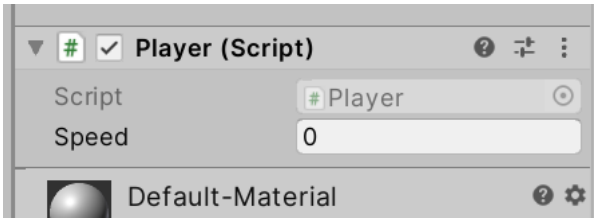
を

Speed = 10.0f;

に書き換えてみましょう。

すると、動く速度がとても速くなります。
変数を使うメリットの一つは、このように書き換えが１回で済むところにあります。

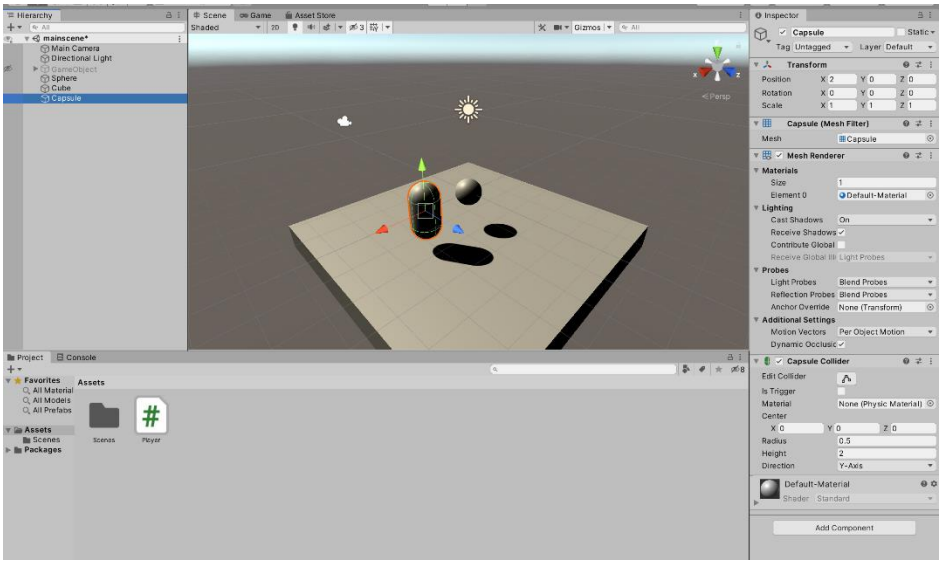
④ Speed =10.0f;
を削除してみましょう。
ここで球のコンポーネントを見ると、Player の下に Speed という欄ができています。これを１００に書き換えてみましょう。



球の動きがとても速くなりました。このように、public 変数は Unity エディター上からも編集できます。

⑤ Speed は５くらいに戻してください。

３．アイテムを作る
このゲームはアイテムを集めるゲームにしようと思います。そのため、アイテムを作ります。
① 3D Object の Capsule を作ります。
② Capsule の x 座標を２にします。



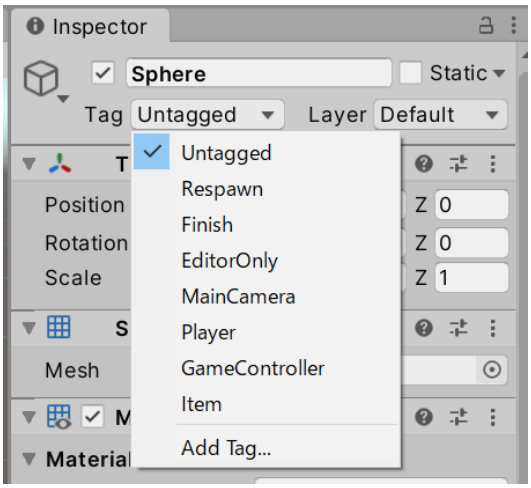
この時点では Capsule は空中に浮かぶ障害物です。

③ 触ったら消えるようにします。Player スクリプトを以下のように加筆して保存してください。

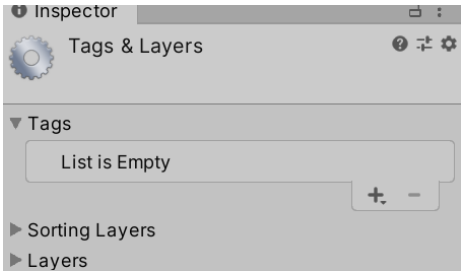
```
29          if (Input.GetKey(KeyCode.DownArrow))
30          {
31              GetComponent<Rigidbody>().AddForce(0f, 0f, -Speed);
32          }
33
34      }
35
        0 references
36      void OnCollisionEnter(Collision Hit)
37      {
38          GameObject obj = Hit.gameObject;
39          if(obj.CompareTag("Item")){
40              Destroy(obj);
41          }
42      }
43  }
44
```

36 行目：このオブジェクトが何かと衝突したときにカッコ内の処理を実行し、またその衝突判定（Collition）を Hit と名付ける、という意味。
38 行目：GameObject 型の変数 obj を宣言し、Hit の（当たった）オブジェクトを代入する、という意味。Hit.gameObject の「.」は「複数の情報をもつもののうち、これ」という感じの意味です。今後も出てきます。
GameObject 型：オブジェクトを入れる型。
39 行目：obj（当たったオブジェクト）のタグが Item ならカッコ内の処理を実行。タグについてはあとで解説します。
40 行目：obj を消滅させる、という意味。
Destroy(オブジェクト)：オブジェクトをゲーム内から消すプログラム。

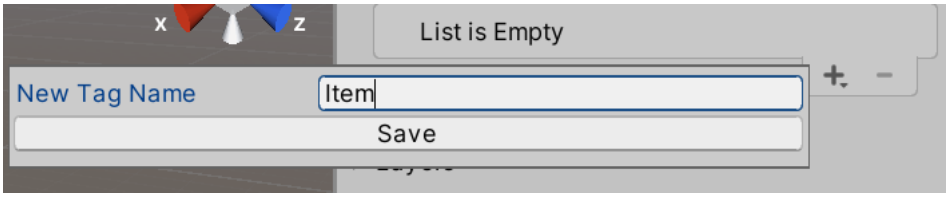
④ このままだと動かないので、Unity でタグに関する編集をします。
タグ：コンポーネントの一種で、オブジェクトの識別によく使われます。インスペクターの一番上から編集できます。



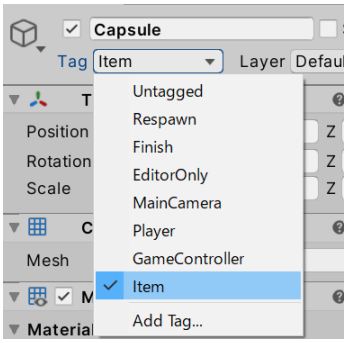
最初は Item というタグはないので、Add Tag を押して追加します。



＋を押して、Item と入力して save ボタンを押します。



Item タグが追加されたので、Capsule のタグを Item に変更します。クリックでできます。



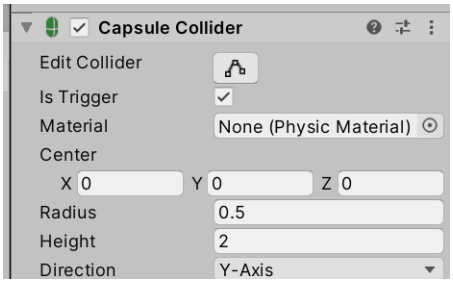
⑤ 実行すると球が衝突したときにアイテムが消滅します。

4．アイテムの挙動を変更する

このままだとアイテムに衝突したときに一瞬止まってしまいます。別にこれでもいい

のですが、プレイのスムーズさを考えて止まらないようにします。

① Capsule の Capsule Collider にある Is Trigger にチェックをいれます。



こうすると、球が衝突せずにすり抜けるようになります。実行すると、アイテムが消えずに球をすり抜けているのがわかります。

② プレイヤースクリプトを以下のように書き換え、保存してください。

```
0 references
void OnTriggerEnter(Collider Hit)
{
    GameObject obj = Hit.gameObject;
    if(obj.CompareTag("Item")){
```

OnTriggerEnter は OnCollisionEnter の Is Trigger 版です。Is Trigger なオブジェクトに触れるとカッコ内の処理が実行されます。また、()の中身が Collision から Collider に変更されていますが、Collision が衝突の情報を表すのに対し、Collider はコライダーという当たり判定のコンポーネントを表します。(この辺は複雑なのでいつか改めてやります)

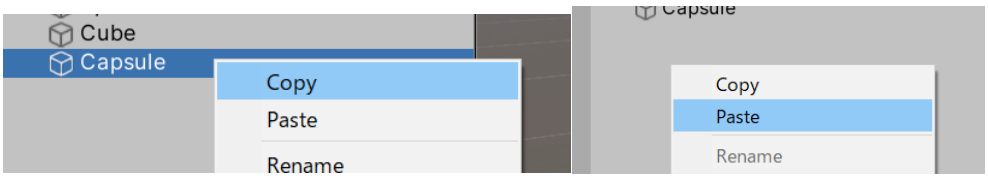
③ 実行すると、衝突時に速度を落とさずに消えるのが分かります。

5．オブジェクトのコピペと削除と命名

アイテムが一個だとつまらないので、増やします。しかし、これと同じものを一から作るのは面倒なので、便利な機能を使います。

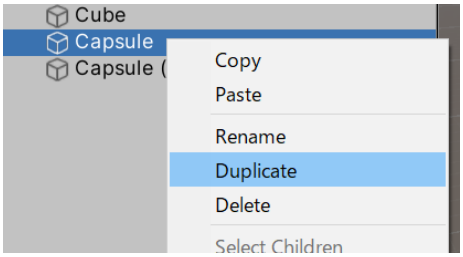
① コピペをします。

ヒエラルキーから Capsule を右クリックし、copy を選択し、ヒエラルキー上で右クリックし、paste を選択します。

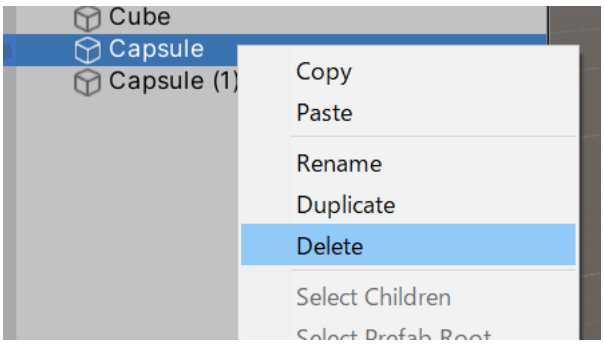


すると Capsule がもう一つできます。Position の X を 2 にするとわかりやすいです。

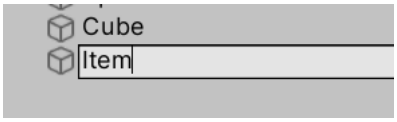
② 別のコピペの方法を教えます。
ヒエラルキーから Capsule を右クリックし、Duplicate を選択します。これだけでもコピペができます。



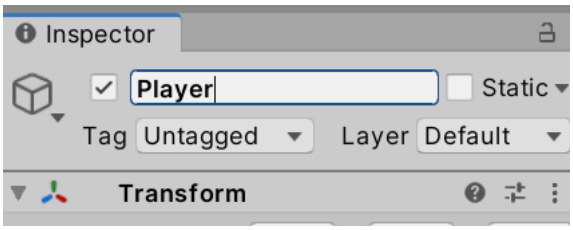
③ これでもいいのですが、もっと便利な方法があるので、そちらの方法で進めます。せっかく作ったオブジェクトですが、削除してください。ヒエラルキーから Capsule(1)を右クリックし、Delete を選択します。これでオブジェクトが削除されます。Capsule(2)も削除してください。Capsule は残しておいてください。



④ さっきから Capsule と言っていますが分かりづらいので、名前を変えます。
Capsule を右クリックし、Remene を選択して Item と名付けてください。



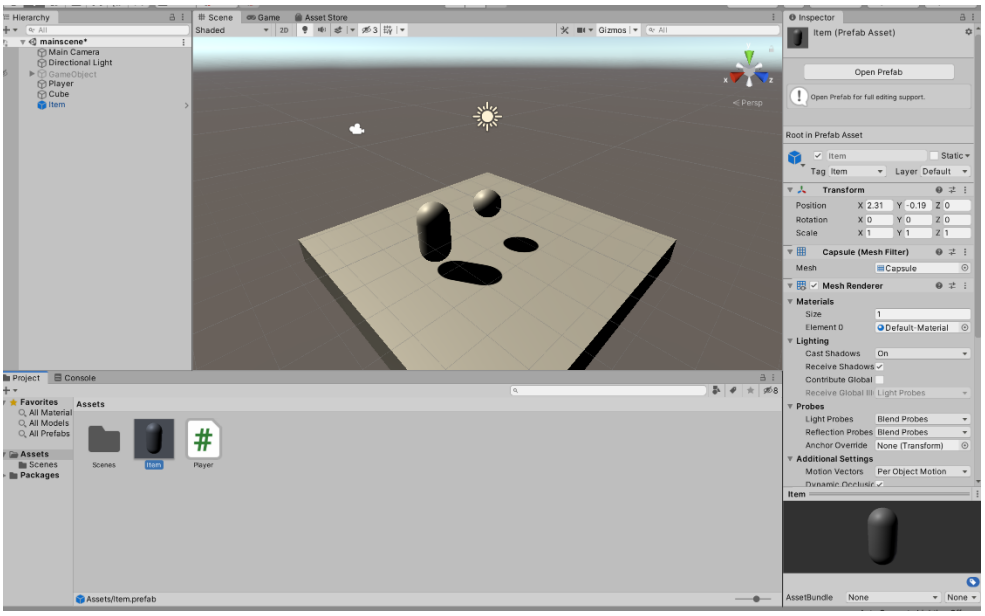
ついでに球の名前も Player に変えましょう。別の方法を教えます。
球を選択した状態で、インスペクターの一番上の Sphere と書かれた部分を選択し、Player に書き換えましょう。



6．プレハブとインスタンス化
Unity にはプレハブという機能があります。これは、オブジェクトの設計図のようなもので、これを使ってスクリプトからオブジェクトを作ることができます。プレハブ

を使ってスクリプトからオブジェクトを作ることをインスタンス化といいます。

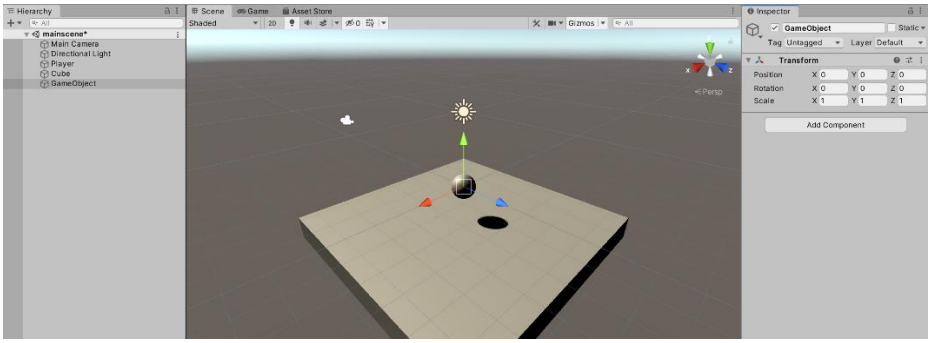
① まずはプレハブを作ります。ヒエラルキーの Item をプロジェクトビューにドラッグアンドドロップします。



すると、プロジェクトビューになんかができます。これがプレハブです。
このプレハブは、もとにしたオブジェクト（ここでは Item）と同じデータを持っています。

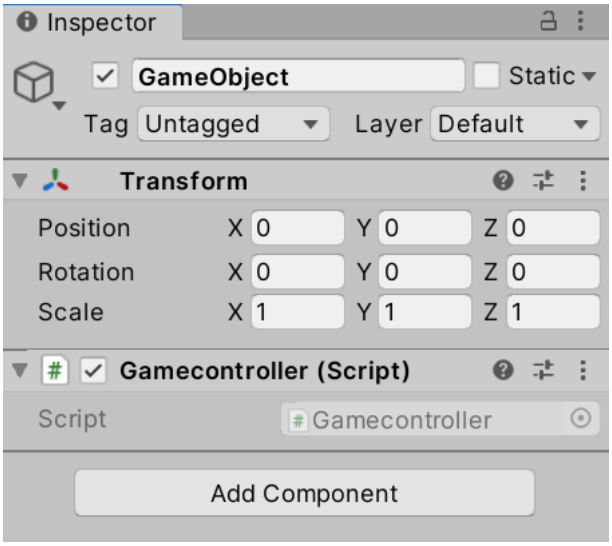
② もとにした Item は削除しましょう。

③ インスタンス化をつかさどるオブジェクトを作ります。ヒエラルキーを右クリックし、Create empty を選択します。すると GameObject ができます。



画面には何もできていないように見えますが、これは GameObject が実体を持っていないからです。

④ Add Component>New Script から GameController という名前でスクリプトを追加します。



⑤ 以下のように GameController を編集してください。

```
Gamecontroller.cs
Assets > Gamecontroller.cs > Gamecontroller > Update()
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
    0 references
5   public class Gamecontroller : MonoBehaviour
6   {
        1 reference
7       public GameObject Item;
        1 reference
8       public Vector3 ItemPosition;
9       // Start is called before the first frame update
        0 references
10      void Start()
11      {
12          Instantiate(Item,ItemPosition,Quaternion.Euler(0,0,0));
13      }
14
15      // Update is called once per frame
        0 references
16      void Update()
17      {
18
19      }
20  }
21
```

８行目：Vector3 型の変数 ItemPosition を宣言します。

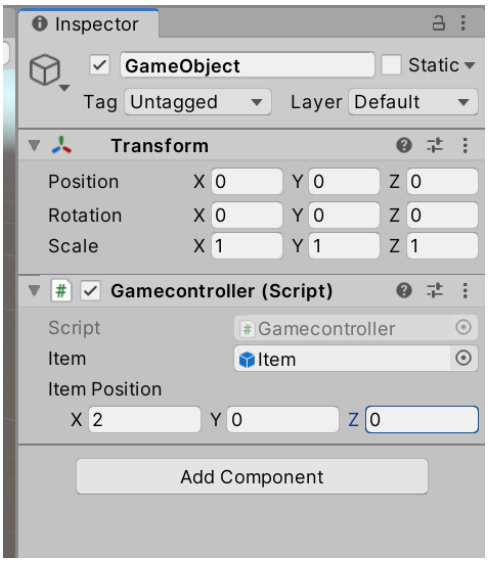
Vector3：数値を三つ持つ変数の型。位置情報などにつかいます。

１２行目：ItemPosition に Item を生成します。

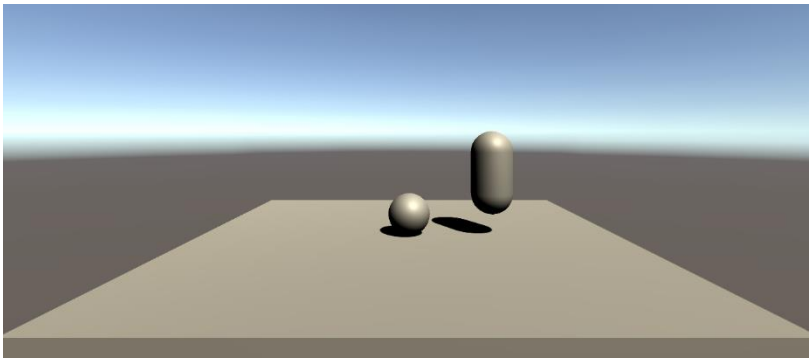
Instantiate(オブジェクト,位置,角度)：オブジェクトを生成する命令。() の中身は「,」（カンマ）で区切られていて、一つ目は作るオブジェクト、二つ目はオブジェ

クトを作る位置、三つ目は作るオブジェクトの向きです。
Quarternion.Euler(0,0,0)については今はいいです。角度がついていないことを表します。

⑥ このスクリプトを動かす準備をします。GameObject を選択し、ヒエラルキーから Item にプレハブをドラッグアンドドロップします。また、ItemPosition の X の値を 2 にします。



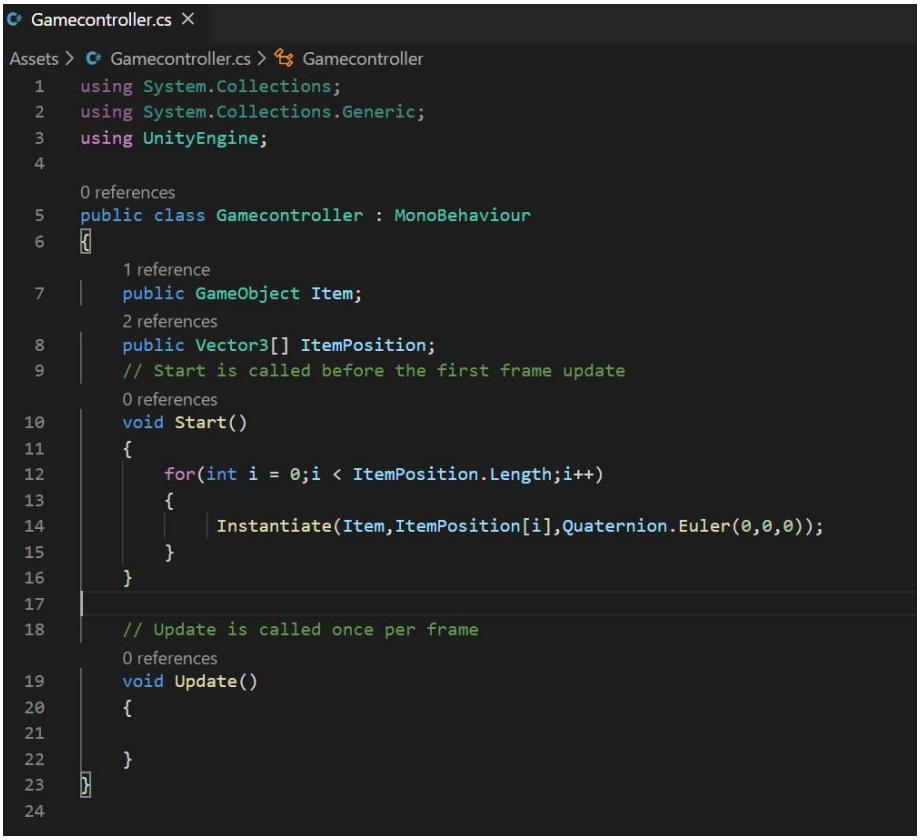
⑦ 実行すると、Item が作られたのが分かります。当たるとちゃんと消滅します。



7．配列と for 文

作る Item の数を増やします。

① 以下のように GameController スクリプトを編集してください。



② ここで新しいことを教えます。

一つ目は「配列」です。配列とは、同じ型の変数が連なったものです。

宣言の際は int[] A;のように、型の後ろに[]をつけます。

このように宣言された配列は、A[0],A[1]のように「変数名[数]」の形で一つの変数として扱うことができます。数は 0 から順番に割り当てられます。例えば

int[] Score;

Score[0] = 50;

Score[1] = 70;

float Average = (Score[0] + Score[1]) / 2;

とすると、変数 Average には 60 が入ります。

また、配列は「いくつの変数を連ねるか」という値を持ちます。これを配列の長さといいます。配列は長さを指定しないと使えません（そのため上の Score の例は、厳密にいうと正しくありません)。今回は public にしたので、Unity エディター上から長さや数値を指定します。長さは「配列名.Length」で取得できます。

二つ目は for 文です。この文は

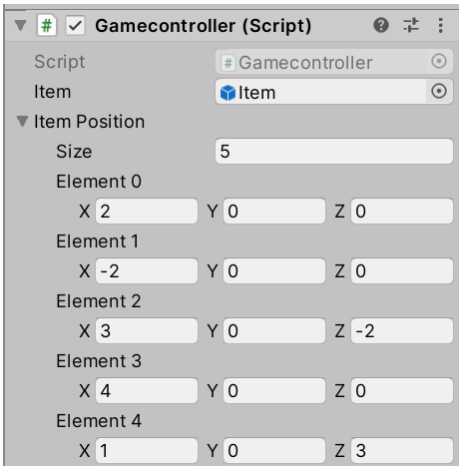
for(int i = 0;i < ～;i++){

　中身

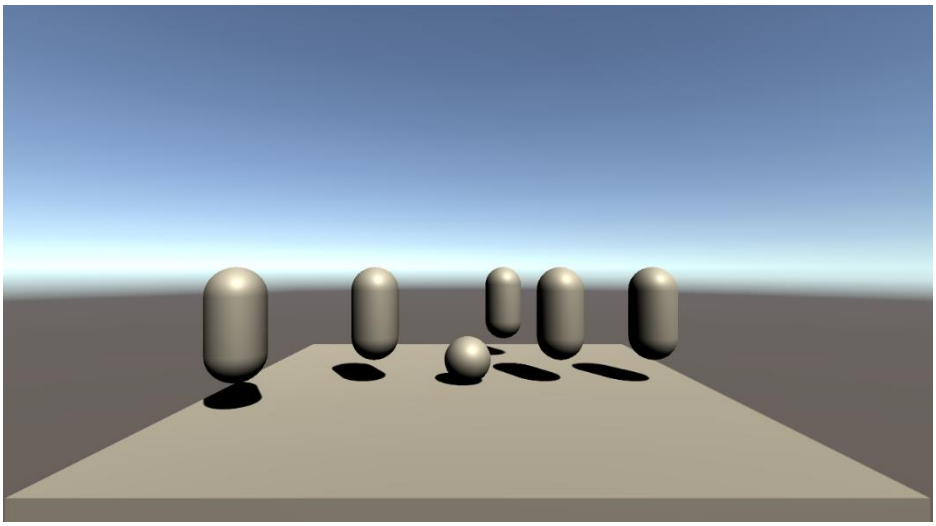
}
の形で使われ、中身の処理を～回繰り返す、という意味です。
具体的にいうと
変数 i を宣言し（int i = 0;）、もし i の値が～未満なら（i < ~;）（～は数値）、中身の処理を実行し、i に 1 を足し（i++）
この時点でもし i の値が～未満なら中身の処理を実行し i に 1 を足し……
という意味です。
また、{}内では変数 i を使うことができます。i は一回目の処理では 0、二回目の処理では 1、三回目の処理では 3、と、 1 ずつ増えていきます。

以上を踏まえると、このプログラムの意味は、ItemPosition の長さ回、ItemPosition[0],ItemPosition[1],ItemPosition[2]……の位置に Item を配置する、という意味になります。

③ このままでは動かないので、Unity から編集をかけます。GameObject の Gamecontroller の ItemPosition の左の三角形を押して、size を 5 に変更してください。これが配列の長さです。すると Element0 とかが出てくると思うので、以下のように入力してください。



④ 実行すると Item が 5 つそれぞれ指定した位置にできます。



8．乱数

ステージが毎回変わるようにします。

① GameController を以下のように加筆してください。

```
void Start()
{
    for(int i = 0;i < ItemPosition.Length;i++)
    {
        ItemPosition[i] = new Vector3(Random.Range(-5,5),0,Random.Range(-5,5));
        Instantiate(Item,ItemPosition[i],Quaternion.Euler(0,0,0));
    }
}
```

new：Vector3 型の変数に値を代入するには、position = new Vector3(x,y,z);のように、new を付けなければいけません。

Random.Range(-5,5)：－5 から 5 までの値をランダムに出力してくれます。

② 実行すると、そのたびごとに Item の位置が変わるようになりました。

第二回講習はこれで終わりです。