

Rigidbody

リジッドボディ

- オブジェクトに**物理演算**を適用できるコンポーネント。
- オブジェクトに力がかかると、現実の物体っぽく**リアル**に動く。
- 重力**の影響を与えたい! **力**をかけたい! そんなときによく使う。

①質量

②移動の抵抗

③回転の抵抗

④重力

⑤演算の有無

⑥動きの補間

⑦衝突検知

⑧動きの制限

Rigidbody

Mass1

Drag0

Angular Drag0.05

Use Gravity☒

Is Kinematic☐

InterpolateNone

Collision DetectionDiscrete

Constraints

Freeze Position☐X☐Y☐Z

Freeze Rotation☐X☐Y☐Z

①質量	単位はkg。大きいほど、力の影響を受けにくくなる
②移動の抵抗	空気抵抗。「0」の場合、空気抵抗なし
③回転の抵抗	回転するときの空気抵抗
④重力	<input checked="" type="checkbox"/> がついてると、重力の影響を受ける
⑤演算の有無	<input checked="" type="checkbox"/> がついてると、物理演算を適用しない
⑥動きの補間	物理演算による動きをスムーズにする
⑦衝突検知	高速で動くモノが他のモノをすり抜けるのを防ぐ
⑧動きの制限	<input checked="" type="checkbox"/> をつけた軸の移動や回転を制限する

【Scriptカード】

他のオブジェクトやコンポーネントを取得する

- 変数を宣言するときに「**public**」を付けてから、取得したいものをInspectorウィンドウで**アサイン**。(初級)
- 「**Find**」や「**GetComponent**」を使えば、アサインの操作を省略できる。(中級)

例えば...

Cube

• # CubeScript

Sphere

• Rigidbody

CubeScriptから、SphereのRigidbodyをコントロールしたい...!

方法① (初級)

変数を定義する文に「**public**」を付けて、Inspectorウィンドウからも設定できるようにする。

```
public class CubeScript : MonoBehaviour {  
    public Rigidbody Hoge;  
    void Start()  
    {  
        Hoge.AddForce(0, 10, 0, ForceMode.Impulse);  
    }  
}
```

Inspectorウィンドウで🔍を押して、取得したいものを**アサイン**。

【Designカード】

ボタンを押すと**Text**の得点が増える

設計図

Text

• Text

• # TextChange

変数 text の値が実際に表示される

変数 Score を宣言。Score の値を文字列にして変数 text に代入

Button

• Button

• # AddScore

ボタンが押されたらどのスクリプトを実行するかを設定する

OnClick()に書く。変数 Score を1ずつ増やす

関数

AddForce(**Vector3型**, **ForceMode.★**)

Vector3(x, y, z)軸方向に**力**を加える。

AddTorque(**Vector3型**, **ForceMode.★**)

Vector3(x, y, z)軸回りに**回転力**を加える。

★	力の種類
Force	箱を押すような継続的な力
Impulse	ボールを蹴るような瞬間的な力
Acceleration	質量を無視して、加速させる
VelocityChange	質量を無視して、速度を変える

変数

mass	float型	質量
useGravity	bool型	true→重力が有効。false→無効
isKinematic	bool型	true→物理演算が無効。false→有効
velocity	Vector3型	速度
angularVelocity	Vector3型	回転速度

使用例

```
public class NoteTest : MonoBehaviour {  
    public Rigidbody Hoge;  
    void Start()  
    {  
        Hoge.AddForce(0, 10, 0, ForceMode.Impulse);  
    }  
    void Update()  
    {  
        if(Hoge.velocity.y <= 0)  
        {  
            Hoge.useGravity = false;  
        }  
    }  
}
```

Y方向に、10の力で蹴り上げる!

Y向き速度が0以下なら、重力オフ!

方法② (中級)

```
public class CubeScript : MonoBehaviour {  
    GameObject Piyo;  
    Rigidbody Hoge;  
    void Start()  
    {  
        Piyo = GameObject.Find("Sphere");  
        Hoge = Piyo.GetComponent<Rigidbody>();  
        Hoge.AddForce(0, 10, 0, ForceMode.Impulse);  
    }  
}
```

Sphereという名前のオブジェクトを見つけて、変数Piyoに代入

SphereにアタッチされているコンポーネントRigidbodyを取得して、変数Hogeに代入

方法① (初級) と違って、スクリプトだけでオブジェクトやコンポーネントを取得できる。

関数おさらい

GameObject.Find(**string型**)

指定した名前の**ゲームオブジェクト**を見つける。

GameObject型 .GetComponent<コンポーネント型>()

指定したゲームオブジェクトの、指定した**コンポーネント**を取得する。

自分自身のゲームオブジェクトのコンポーネントを取得したいときは、**GetComponent<コンポーネント型>()** もしくは、**this.gameObject.GetComponent<コンポーネント型>()** と書けばOK!

詳細

Text

• Text

• # TextChange

```
using UnityEngine;  
using UnityEngine.UI;  
public class TextChange : MonoBehaviour {  
    Text Tx;  
    public static int Score = 0;  
    void Start()  
    {  
        Tx = GetComponent<Text>();  
    }  
    void Update()  
    {  
        Tx.text = Score.ToString();  
    }  
}
```

Button

• Button

• # AddScore

OnClick()

Runtime Only

AddScore.OnClick

+ Button (AddScore)

```
public class AddScore : MonoBehaviour {  
    public void OnClick()  
    {  
        TextChange.Score++;  
    }  
}
```