Timmy??????

???????????????

主人公のキャラクターの移動を実装しましょう。まず、空のゲームオブジェクトを作成し、「 Player 」という名前を付けてください。このゲームオブジェクトはプレーヤーの役割にする。

3D??????

Characters/Timmy/Models から Timmy のモデルを追加してください。 Player の子オブジェクトとして追加しましょう:

よく見ると、色が多少おかしい。これはノーマルマップ（ ※）の問題であるので、解決しましょう。

Characters/Timmy/Textures の中にあるノーマルテクスチャ（青い画像）を選択し、ノーマルマップであることを指定し、最後に「 Apply 」を押してください:

（ ※）ノーマルマップについては、また「ゲームエンジン

」で勉強する予定

???????

キャラクターを物理的に移動するべきなので、コライダーとリジッドボディが必要。

Player に Rigidbody と CapsuleCollider を追加してください。なお、コライダーの形がある程度3Dモデルの体に合わせましょう:

?????

Inputクラスを使用し、キャラクターを動かしましょう。後で改善するべきですが、とりあえず、 WASD のキーで前後左右を移動しましょう:

AとDでX軸（赤い軸）で移動し、W とSでZ軸（青い軸）で移動する。なお、各入力は:

- `Input.GetAxis("Horizontal")`: 入力の「水平軸」（つまり Aと D）の入力をの取得ができる
  - Aの場合は -1 を返す
  - Dの場合は1を返す
  - 入力のない場合は0を返す
- `Input.GetAxis("Vertical")`: 入力の「垂直軸」（つまりWとS）の入力をの取得ができる
  - Wの場合は1 を返す
  - Dの場合は -1を返す
  - 入力のない場合は0を返す

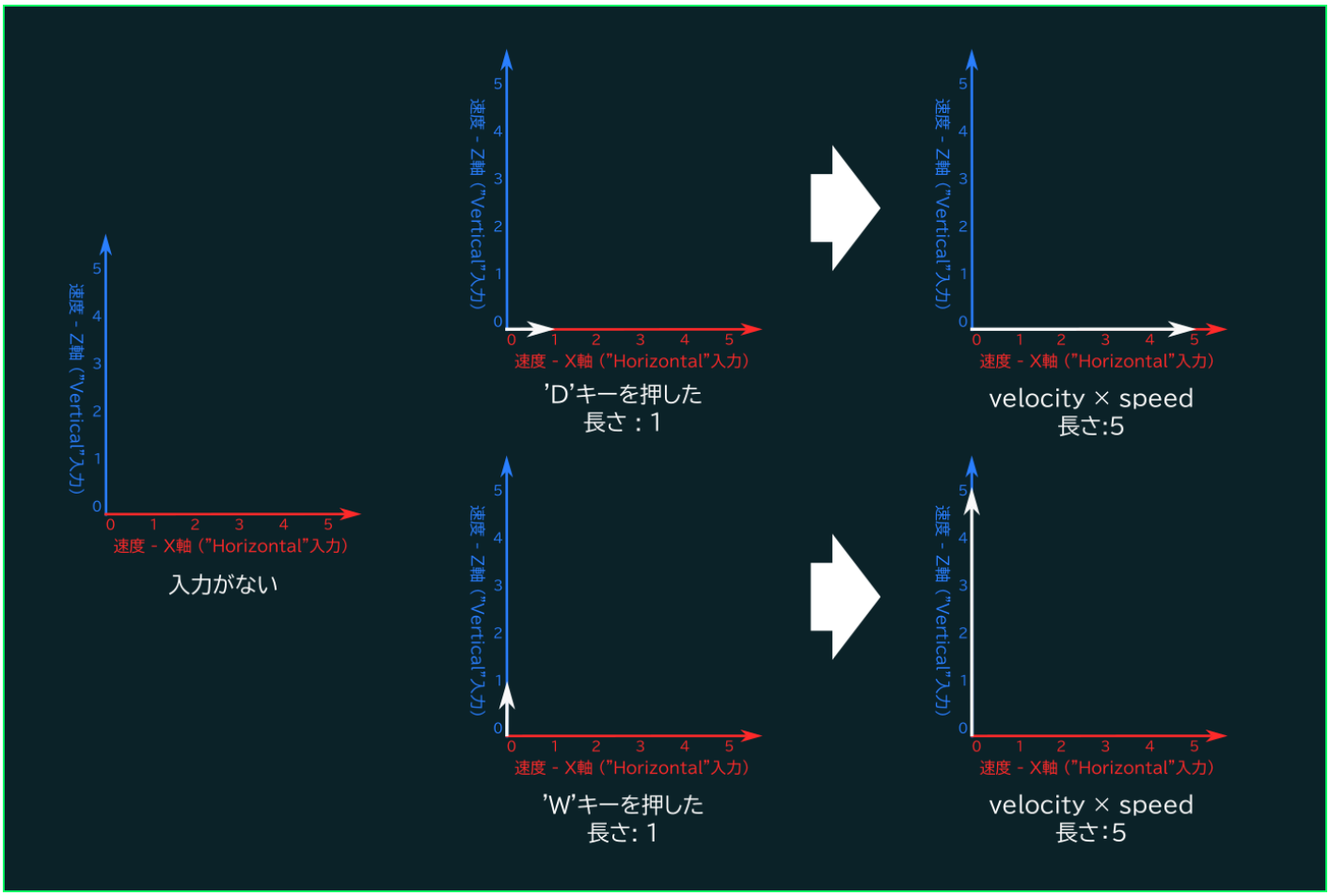
また、移動速度を Unityで変えられるようにしましょう。この知識を活用し、移動のスクリプト「PlayerMove」を実装:

```
// プレーヤーの移動を処理する
public class PlayerMove : MonoBehaviour
{
    // 速度をInspectorで設定
    [SerializeField]
    private float speed = 5.0f;
```

```
    // 物理的に移動するので、リジッドボディが必要
    private Rigidbody rbody;

    // 3D空間の速度ベクトル
    private Vector3 velocity;

    private void Start()
    {
        // 参照を取っておく
        rbody = GetComponent<Rigidbody>();
    }

    private void Update()
    {
        // 何も入力がない前提で、速度をゼロにする
        velocity = Vector3.zero;

        // A/S（水平入力）を取得し、3D空間のX軸に割り当てる
        velocity.x = Input.GetAxis("Horizontal");

        // W/D （垂直入力)を取得し、3D空間のZ軸に割り当てる
        velocity.z = Input.GetAxis("Vertical");

        // 単位ベクトルにする
        velocity.Normalize();

        // 速度を乗算する
        velocity *= speed;
    }

    private void FixedUpdate()
    {
        // 速度を設定
        rbody.linearVelocity = velocity;
    }
}
```

`Normalize()`??

`Normalize()`はあるベクトルを長さ1（単位ベクトル）にする。今回の処理は以下の図で解析してある：



上下、または左右 のみ で移動する場合は、 GetAxis が -1～ 1 の数値を返すので、移動速度「 speed 」を掛けると、問題なく、長さ５の速度ベクトルができる。一方、同時に入力すると（斜め移動）

掛け算の前に単位ベクトルにすると：

????

このスクリプトを「 Player 」にアタッチし、実行してみてください。移動し、ランプを上り、ジャンプしてみてください。なお、カメラの処理まだ行っていないので、 Game ビューと Scene ビューを同時に表示するのはおすすめ。

??????????????

すべての動きをリジッドボディに任せているため、移動だけではなく、回転も自由になっている。今後、回転はスクリプトで実現するので、リジッドボディが回転ができないようにすれば良いでしょう。制約設定で回転を固定しましょう:

??????????

回転の問題を解決できたが、ランプから飛び出すと、ゆっくりに落ちる。なぜなら、 Updateごとに速度を上書きしている からです! 重力による落下をリジッドボディに任せても、縦軸（ Y 軸）の速度をゼロにしているので、なかなか落ちない!

解決としては、 Y軸の速度を保守し、 Xと Zだけを上書きすれば良い。 `FixedUpdate` を以下のように直しましょう:

```
private void FixedUpdate()
{
    // □□□□□□□□□
    Vector3 v = rbody.linearVelocity;

    // X□Z □□□□
    v.x = velocity.x;
    v.z = velocity.z;

    // □□□□□
    rbody.linearVelocity = v;
}
```

□□□□□□□□□□□

Revision #5
Created 2026-05-04 08:07:30 UTC by Menendez Francisco
Updated 2026-05-21 10:18:43 UTC by Menendez Francisco