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

コンパスと加速度センサーで単純な 2Dミニゲームを作りましょう~

- プロジェクト仕組みの理解
- 回転の操作
- 移動の操作

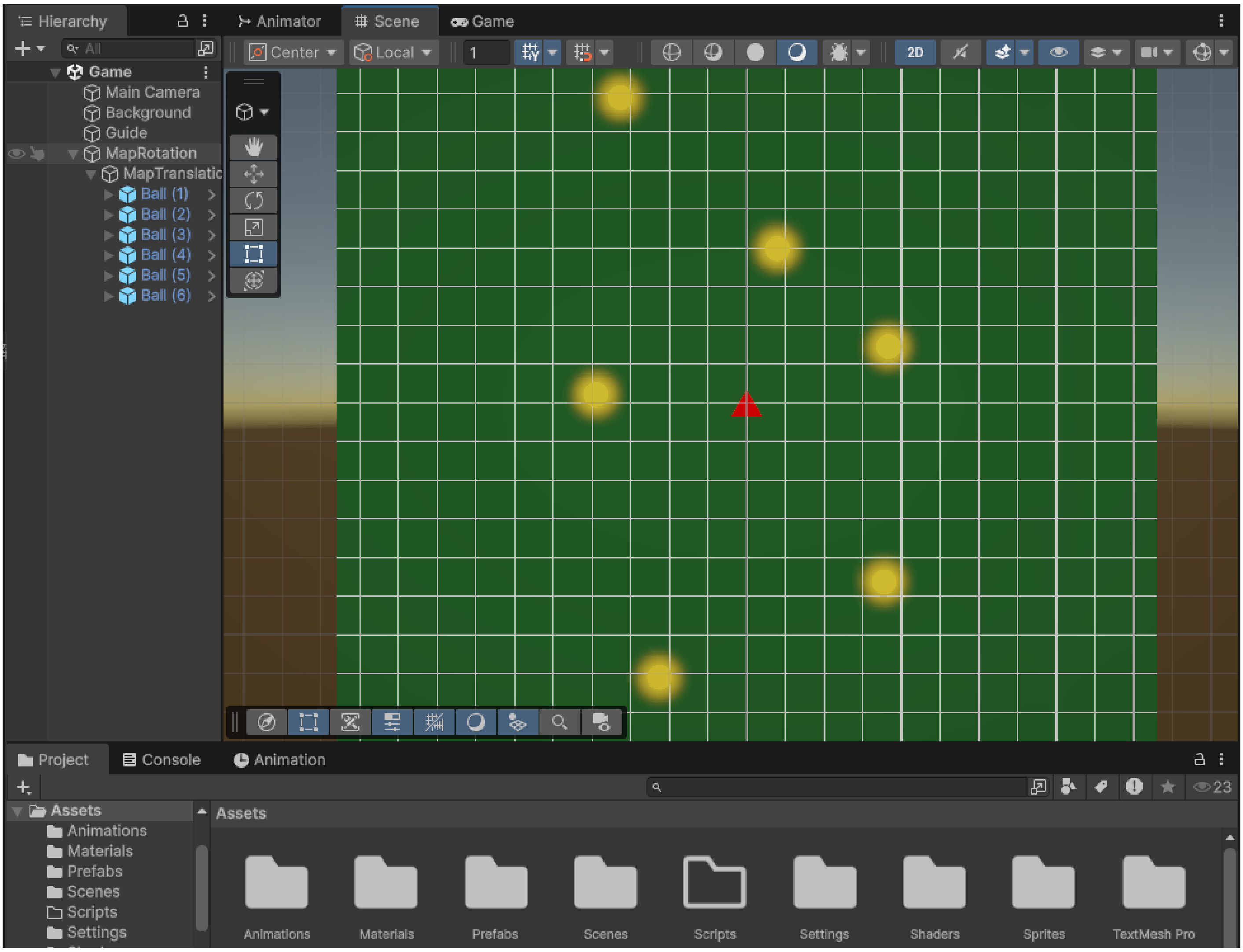
????????????

?????

□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
□□□□□□□□□□□□□□□□□□□□□□□□□□□

?????

Scenes□□□□□□□□□ Game□□□□□□□□



Hierarchy
Animator
Scene
Game
Center
Local
2D
Game
Main Camera
Background
Guide
MapRotation
MapTranslatic
Ball (1)
Ball (2)
Ball (3)
Ball (4)
Ball (5)
Ball (6)
Project
Console
Animation
Assets
Animations
Materials
Prefabs
Scenes
Scripts
Settings
Assets
Animations
Materials
Prefabs
Scenes
Scripts
Settings
Shaders
Sprites
TextMesh Pro

□□□□□□□□□

- Main Camera: ゲームのカメラで、2Dカメラになっている
- Background: レーダーのグリッド背景
- Guide: 中央にある赤い三角形
- MapRotation: ボールの回転に使うゲームオブジェクト。これを回転すると、ボールが回転する
- MapTranslation: ボールの移動に使うゲームオブジェクト。
- Ball(1)~ (6): いくつかのボール

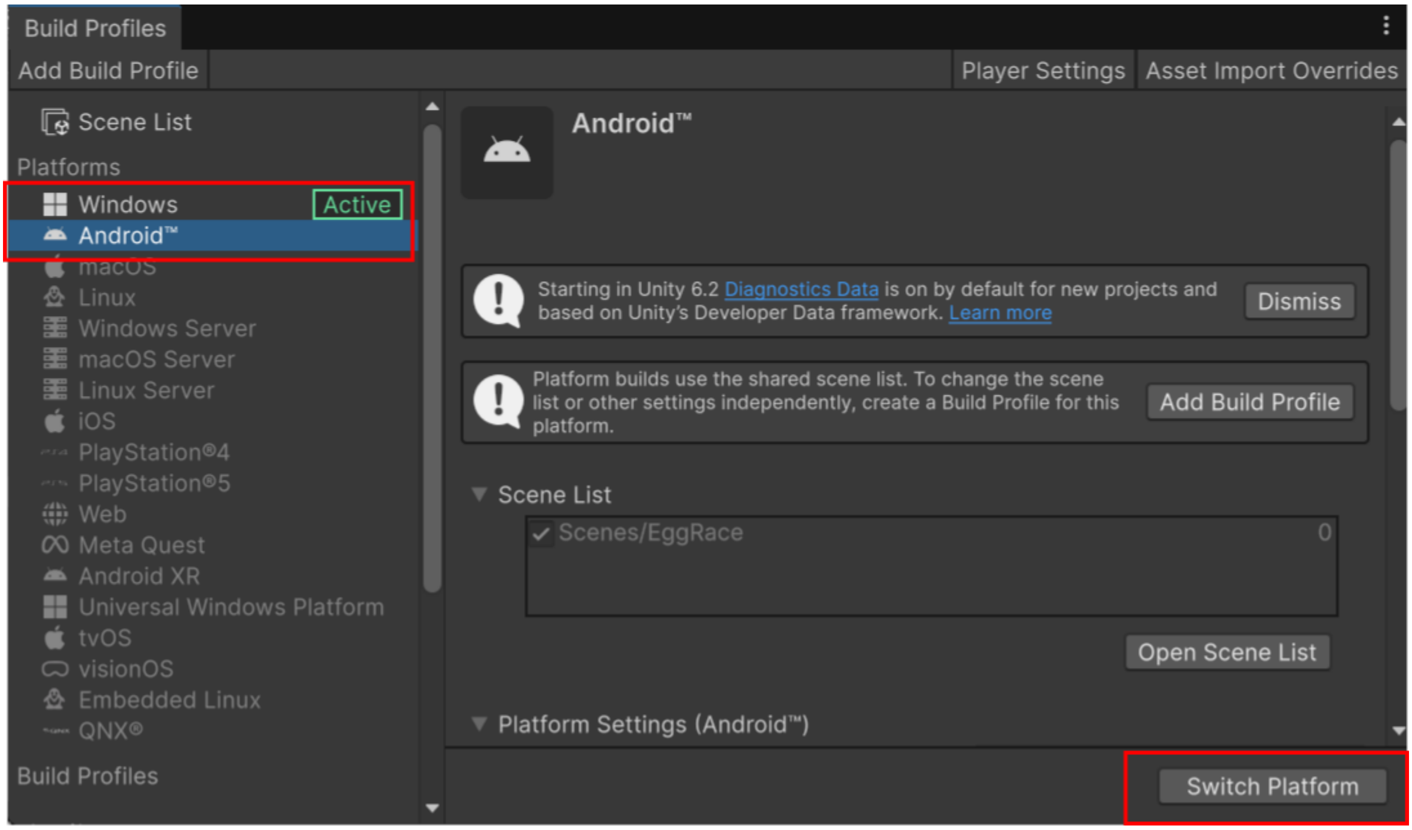
????????

すでにいくつかのフォルダが準備された:

- Animations: ボールが光るアニメーション
- Materials: グリッドのマテリアル
- Prefab: コライダーなどが付いているプレハブ。
- Script: スクリプト。まだ何も入っていない
- Shaders: グリッド生成のシェーダーグラフ
- Sprites: ボールなどの2Dスプライト

????

Androidスマホで実行できるため、プロジェクト設定でプラットホームを変えないといけない。「File ⇒ Build Profiles」を開き、「Windows」から「Android」へ変換し、「Switch Platform」を押しててください。



Starting in Unity 6.2 Diagnostics Data is on by default for new projects and based on Unity's Developer Data framework. Learn more

Platform builds use the shared scene list. To change the scene list or other settings independently, create a Build Profile for this platform.

✓ Scenes/EggRace 0

?????

スマホの方位に合わせて、マップ（ボールのグループ）を回転しましょう。

?????

まず、PCで確認できるため、キーボードのAとDキーで回転しましょう。スクリプトはとても単純である:

```
// レーダーの回転を制御する
public class RotationController : MonoBehaviour
{
    // 現在の角度
    private float angle;

    private void Update()
    {
        // 回転速度（固定）
        const float rotationSpeed = 180f;

        // 水平入力（-1 ~ 1）
        float input = Input.GetAxis("Horizontal");

        // 角度を更新
        angle += rotationSpeed * Time.deltaTime * input;

        // Transformで回転（2Dゲームなので、Z軸周りの回転）
        transform.rotation = Quaternion.Euler(0, 0, angle);
    }
}
```

スクリプトを「MapRotation」にアタッチし、キーボードで確認してください。

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

つぎ、コンパスを使って、回転してみましょう。ただし、コンパスはスマホの時だ使用するので、実行中にスマホであるかどうかを確認しなければならない。Unityでは:

```
// モバイル（スマホ・タブレット）であるか?
Application.isMobilePlatform
```

で確認できる。なお、コンパスを使うには、まず有効にする必要があるので、以下の機能を使用:

```
// 有効化
Input.compass.enabled = true;

// 無効化
Input.compass.enabled = false;
```

そして、現在の向き（度単位）は:

```
// 方位（0度~360度の実数）
float value = Input.compass.trueHeading;
```

で取得できる。上記の機能を使い、以下のようにスクリプトを編集してください:

```
// レーダーの回転を制御する
public class RotationController : MonoBehaviour
{
    // 現在の角度
    private float angle;

    private void Start()
    {
        // スマホなのか?
        if (Application.isMobilePlatform)
        {
            // コンパスを有効化し、最初の角度を取得
            Input.compass.enabled = true;
            angle = Input.compass.trueHeading;
        }
    }

    private void Update()
    {
        // スマホなのか?
        if (Application.isMobilePlatform)
        {
```

```
            // コンパスから角度を取得
            angle = Input.compass.trueHeading;
        }

        // PCなのか?
        {
            // 回転速度（固定）
            const float rotationSpeed = 180f;

            // 水平入力（-1 ～ 1）
            float input = Input.GetAxis("Horizontal");

            // 角度を更新
            angle += rotationSpeed * Time.deltaTime * input;
        }

        // Transformで回転（2Dゲームなので、Z軸周りの回転）
        transform.rotation = Quaternion.Euler(0, 0, angle);
    }
}
```

ビルドし、実機で確認しましょう。

???????

端末によるが、手の動きやセンサーの敏感により、`Input.compass.trueHeading` の出力が振れることがある。スムーズに回転するには、Unityは以下の機能がある:

```
// 回転の振動減衰
// current        : 現在の数値
// target         : 次にしたい数値
// currentVelocity: 自動的に計算される速度。メンバー変数として作成してください
// smoothTime     : 減衰の時間。大きいほどスムーズになるが、遅れが発生する
Mathf.SmoothDampAngle(float current, float target, ref float currentVelocity, float smoothTime);
```

これ機能を使い、スクリプトを編集してください。まず、`currentVelocity` をメンバー変数として追加する。

```
// レーダーの回転を制御する
public class RotationController : MonoBehaviour
{
    // 現在の角度
    private float angle;

    // 振動減衰に必要なデータ
    private float currentVelocity;

    // (省略)
}
```

そして、`Update` では:

```
// スマホなのか?
if (Application.isMobilePlatform)
{
    // コンパスから角度を取得
    // angle = Input.compass.trueHeading;

    // 振動減衰
    angle = Mathf.SmoothDampAngle(
        angle,                      // 現在の角度
        Input.compass.trueHeading,  // 次にしたい角度
        ref currentVelocity,        // 自動的に計算される速度
        0.1f);                      // 減衰の時間
}
```

ビルドし、実機で確認しましょう。

?????

スマホの加速度センサーを使い、強い湯揺れを検知したら「歩」として検知し、移動処理を追加しましょう。

?????

スマホは純粋の万歩計がないが、加速度センサーで動きを検知ができる。「StepDetector」というスクリプトを作成し、歩調を検知しましょう。なお、回転と同様に、まずキーボードを使い、PCで確認できるようにしましょう。

```
// 歩調を検知する
public class StepDetector : MonoBehaviour
{
    // 検知したときのイベント宣言
    public event System.Action Step;

    // 1歩の時間
    [SerializeField] private float stepTime = 0.2f;

    // 次の歩までの時間
    private float waitTime;

    private void Start()
    {
        waitTime = stepTime;
    }

    private void Update()
    {
        // 次の歩までの時間になっていない
        if (waitTime > 0)
        {
            // 待機時間を減らす
            waitTime -= Time.deltaTime;
            return;
        }
```

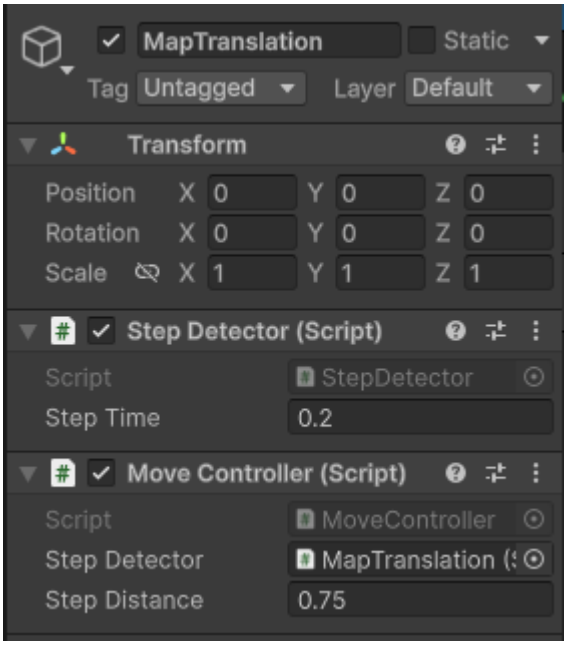
```
        // 入力ある? (0以上) →なければ終わり
        if (Input.GetAxis("Vertical") <= 0)
            return;

        // 1歩進み、時間をリセット
        Step?.Invoke();
        waitTime = stepTime;
    }
}
```

そして、イベントに登録し、前を進む「MoveController」は:

```
// レーダーの移動を制御する
public class MoveController : MonoBehaviour
{
    // 歩調を検知するため
    [SerializeField]
    private StepDetector stepDetector;

    // 1歩進んでら、移動する量
    [SerializeField] private float stepDistance = 0.75f;

    private void Start()
    {
        // 「歩」イベントに登録
        stepDetector.Step += OnStep;
    }

    private void OnDestroy()
    {
        // イベント登録解除
        stepDetector.Step -= OnStep;
    }

    // 1歩を検知した
    private void OnStep()
    {
        // 地図を下の方に移動 (前に進むように見える)
        transform.position += Vector3.down * stepDistance;
```

```
        }
    }
```

この2つのスクリプトを「MapTranslation」にアタッチし、キーボードで確認しましょう:



☑ MapTranslation ☐ Static
Tag Untagged
Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

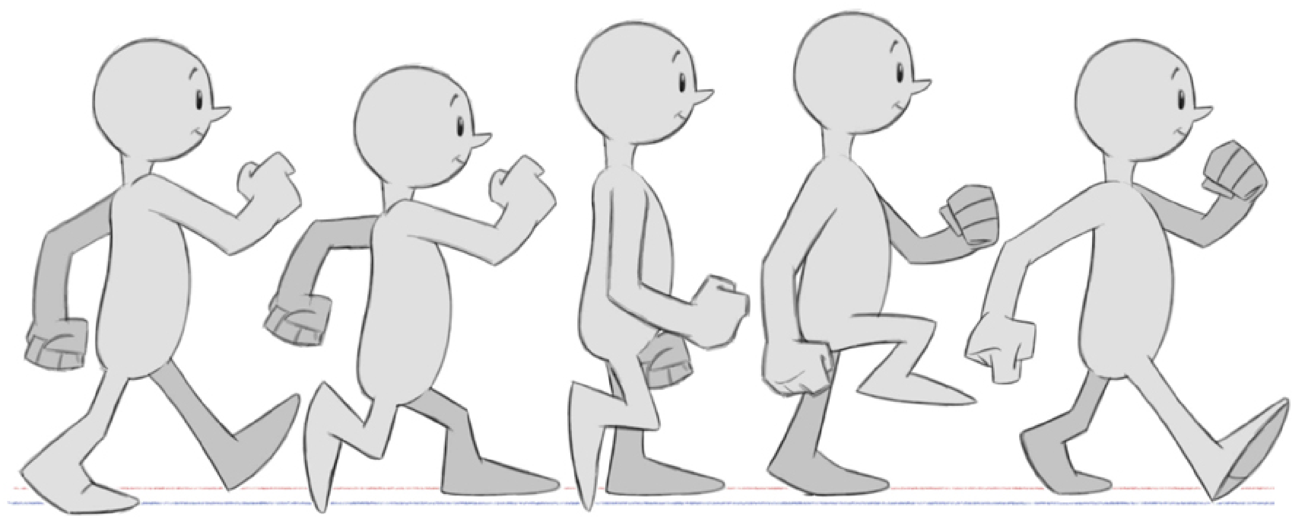
☑ **Step Detector (Script)**
Script StepDetector
Step Time 0.2

☑ **Move Controller (Script)**
Script MoveController
Step Detector MapTranslation (
Step Distance 0.75

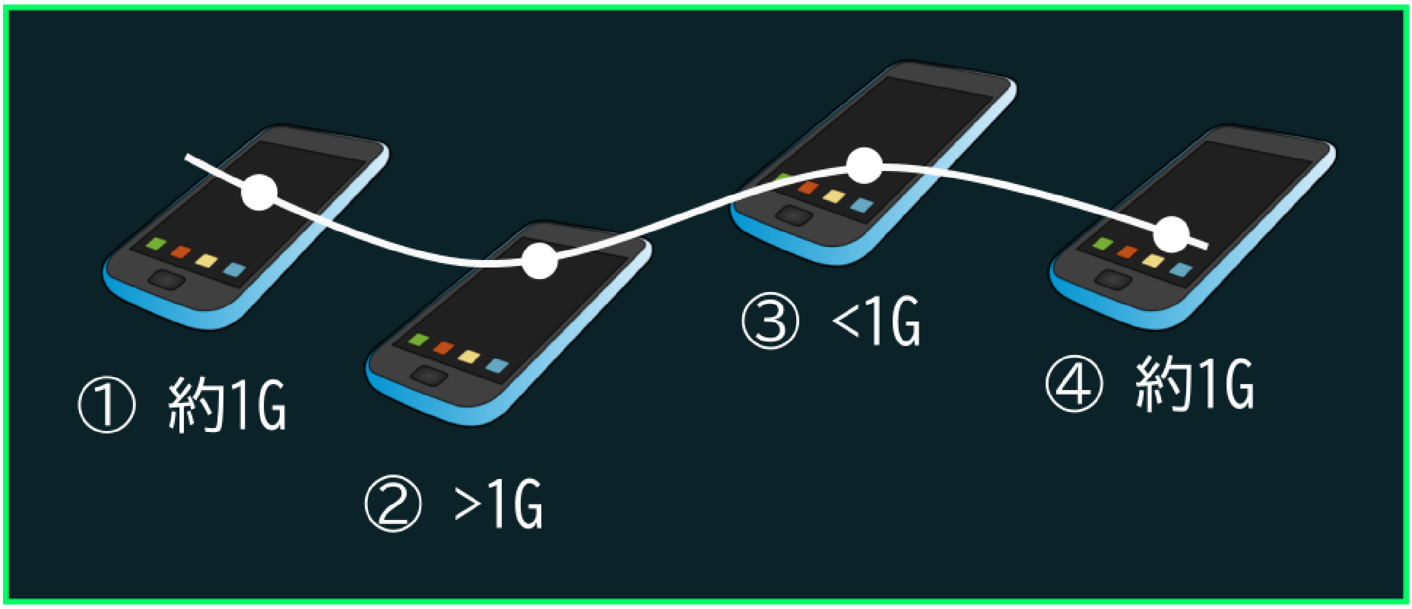
?????????????

加速度センサー「G」単位で加速度を測定する装置である。1G は普通の地球の重力なので、スマホは動かないときは、真下に向かう 1G のベクトルを測定する:

ただし、歩くときは、人間の動きはまっすぐでなく、上がったり、下がったりする：



とすると、加速度センサーの値が変わる：



① 動いていないときは、地球の重力のみで、加速度センサーの強さは約1Gである

② そして歩き始め、体が下がり、スマホが下の方加速するので、加速度センサーの強さは約1Gを超える

③ 次の歩を踏む前、足を上げないといけないので、体が上の方に加速するので、加速度センサーの強さは約1Gよりも小さくなる

④ 最後に止まったら、また1Gに戻る

この現象を使い、単純な歩調を検知を実装できる。加速度センサーの強さが「1G」を超えたら、一歩を進んだことにする。StepDetector を編集し、スマホの対応を追加しましょう:

```
// 歩調を検知する
public class StepDetector : MonoBehaviour
{
    // 検知したときのイベント宣言
    public event System.Action Step;

    // 1歩の時間
    [SerializeField] private float stepTime = 0.2f;

    // 「歩」とする加速度の強さ
    [SerializeField] private float stepForce = 1.05f;

    // 次の歩までの時間
    private float waitTime;

    private void Start()
    {
        // 加速度センサーあったら、有効化する
```

```
        if (Accelerometer.current != null)
            InputSystem.EnableDevice(Accelerometer.current);

        waitTime = stepTime;
    }

    private void OnDestroy()
    {
        // 加速度センサーあったら、無効化する
        if (Accelerometer.current != null)
            InputSystem.DisableDevice(Accelerometer.current);
    }

    private void Update()
    {
        // 次の歩までの時間になっていない
        if (waitTime > 0)
        {
            // 待機時間を減らす
            waitTime -= Time.deltaTime;
            return;
        }

        // PCの場合
        if (Accelerometer.current == null)
        {
            // 入力ある?(0以上)→なければ終わり
            if (Input.GetAxis("Vertical") <= 0)
                return;
        }

        // スマホの場合
        else
        {
            // 加速度ベクトルの長さ（強さ）を求める
            var accel = Accelerometer.current.acceleration.magnitude;

            // 十分に強くない? → なにもしない
            if (accel < stepForce)
                return;
```

```
        }

        // 1歩進み、時間をリセット
        Step?.Invoke();
        waitTime = stepTime;
    }
}
```

これでビルドし、実機で確認してください。