Completeカンペ

Unityの起動

ShootingGameフォルダの中の**Completeプロジェクト**を開こう。

1. Unityエディタを開く
2. メニューバーのFile>Open Project...を選択する
3. 右上の「Open」を選択する
4. ダウンロードし解凍したファイルのComplete以下を選択する

2D Playgroundで遊ぶ

Unityエディターの設定

手順通りに説明していきます。

手順		
1	レイアウト	2 by 3
2	Projectブラウザ	One Column Layout
3	ビルドターゲット	WebPlayer
4	2Dモード	オン
5	レイアウトモード	Pivot & Local

ゲームを再生してみる

1. Projectと名前の付いたタブを確認する
2. "Assets/2DPlayground/Scenes/2D Playground" をダブルクリックする
3. Unityエディタ上部の▶ボタン（再生ボタン）を押し、ゲームを再生する
4. 左下の「Game」タブを選択すると、ゲームが操作できるようになる。
   上下左右で移動、スペースでジャンプ。特にルールはない

※停止はしない

各ビューの説明

動かしながら各ビューの説明を行う

- Game View:
  ゲーム画面が表示される。カメラが映すものが表示される。（常に左上が x:0,y:0ではない）
- Scene View:
  神様視点。GameViewがカメラが映すものを表示しているなら、SceneViewはカメラが写さないものも全部表示する
- Inspector:
  Sceneビューで選択しているオブジェクトが持つ機能の一覧を表示する
- Projectビュー:
  フォルダ。Unityエディタにインポートしたファイル郡が表示される。

Sceneビューの操作を覚える

1. ホィールで拡大縮小
2. Alt+Ctrl（Alt+Command）もしくはマウスの中央ボタンを押しながらドラッグで平行移動
3. Inspectorで確認したいオブジェクトをダブルクリックすると、オブジェクトが見えるところまでシーンビューが移動

Spriteの操作を覚える

スプライトの操作

1. 操作モードをスプライト操作モードに切り替えるため「T」キーを押す、もしくはスプライト操作モードを選択する。（他のモードはキーボードのQ・W・E・R・Tで切り替える）
2. Hierarchyビューの任意のスプライト（今回はCratePink）を選択しダブルクリックする
3. ドラッグで移動、ドラッグ中にShiftを押して軸固定の移動
4. 青点の少し周りを動かすと回転
5. 青点を引き延ばすと拡大縮小。Shiftを押しながら行うと比率固定で拡大

スプライトの配置

1. ProjectビューのカテゴリからSpriteを選択するか、検索バーに「t:Sprite」と入力する
2. 配置したいスプライトを選択し、シーンへドラッグ＆ドロップする
3. フォルダで絞り込みたい場合、絞り込みたいフォルダを選択した状態で1の操作を行い、検索バーの下の「Search:Assets」を「Search:（先ほど選択していたフォルダ）」に変更する

スプライトの描画順は、SpriteRendererコンポーネントのOrder In Layer（レイヤー内の順番）とSorting Layer（レイヤーでソート）の2つで決定します。

コンポーネントのイメージを掴む

コンポーネントの挙動に関してのイメージを掴みます。

Inspectorが選択中のオブジェクトの内容を示すものであること

1. スプライトを選択する
2. Transformの値が切り替わる

コンポーネントを追加すると機能が追加されること

1. スプライトを選択する
2. Add Componentでrigidbody2dと入力し、出てきた「Rigidbody 2D」の項目を選択する

コンポーネントを削除すると機能が削除されること

1. 先ほどRigidbody 2Dを選択したオブジェクトを再度選択する
2. rigidbody2dのタイトルを右クリック、もしくはRigidbody2Dの右にある歯車マークを選択する
3. Remove Componentでコンポーネントを削除する

総復習

転がるキャラクターを作る

1. スプライト「Explosion_3」をシーンに登録する
2. Explosion_3にAdd ComponentでCircleCollider2Dを登録する（Collider2D = 2D向け当たり判定）
3. Rigidbody2Dを登録する

シーンの再生と停止

再生を停止させる。 全てが元通りに鳴る

prefab

Prefabの概念と動作イメージ

1. もう一度先ほど作成した当たり判定付きで転がるオブジェクトを作る（ゲームを再生していない状態で）
2. Projectビューへ設定したExplosion_3をD&Dし、Prefabとしてファイルに書き出す
3. 先ほど書きだしたExplosion_3（prefab）をシーンへD&Dする

休憩！

2D Shootingを作る

実際にゲームをプレイする

とりあえず背景だけ置いてみる

とりあえず背景だけ置きます 背景はPrefabとしてセットアップ済のものを用意しました。

1. ProjectビューのAssets/Shooting/Prefabを開く
2. BackgroundsをInspectorビューへドラッグ＆ドロップする
3. HierarchyビューでBackgroundsを選択する
4. InspectorのBackgroundsのCanvasのRenderCameraへ、HierarchyビューのMain Cameraをドラッグ＆ドロップする

これで再生すると、背景がスクロールします。

この背景は、先ほどの画面のように「カメラが映したものを表示する」のではなく、UIを背景として表示し表示範囲をスクロールさせることで実現しています。

背景が流れる速度を変えたい場合は、以下の操作を行います。

1. HierarchyビューでBackgrounds/Frontを選択する
2. InspectorビューでBackgroundのSpeedを変える（大きいと早い、小さいと遅い）

ちなみに描画順番はUIとして描画しています。 UIの描画順はHierarchyの下から順番です（一番下が一番手前）。

プレイヤーだけ置いてみる

次にPlayerとEnemyを配置してみます。

1. ProjectビューのAssets/Shooting/Prefab/PlayerをSceneビューへドラッグ＆ドロップする

再生してみると、プレイヤーが動きます。

Playerのオブジェクトの機能内訳は以下の様な感じです。

- Transform：座標や回転を管理

- SpriteRendrer：スプライト（2D絵）を描画する。描画順番等もコレが決める
- Animator：アニメーションを行う
- Rigidbody2D：移動した時に当たり判定の収集を行う。ついでに物理演算も行う（IsKinematicで当たり判定の収集のみになる）
- Circle Collider2D：円の当たり判定を行う。rigidbody2dが当たり判定チェックに使う
- Audio Source：オーディオを再生する。弾の発射音が登録されている
- Player:ゲームの実現に足りない機能をC#スクリプト出来術したもの。移動や当たり判定を行う
- Spaceship：スペースシップ全般の挙動について。爆発や発射する弾の制御をコレで行う

敵キャラを配置する

1. ProjectビューのAssets/Shooting/Prefab/EnemyをSceneビューへドラッグ＆ドロップする

敵が表示され、弾を発射してきます。

1. ProjectビューのAssets/Shooting/Prefab/WaveをSceneビューへドラッグ＆ドロップする

複数体の敵がゆっくりと前進してきます。

これをSceneView的に上の方に配置しまくると、ステージっぽくなります。

なおEnemyはEnemyの子として配置しているオブジェクトの方向に弾が発射されます。 弾の発射角度を変更してみます。

1. HierarchyのEnemy/ShotPosition（の一番上のやつ）を選択します。
2. InspectorでRotationのzの値を変更160から100にします

もっと直感的に変更したい場合、シーンビューで「Eキー」を押すかRotation操作モードに変更し、外枠をドラッグして回転させます。 また座標をPosition(x:-0.25, y:-0.07, z:0）とします。より楽に調整したい場合、Sceneビューを選択した状態で「Wキー」を押し、矢印の根本の四角をドラッグします。 （2D"ゲーム"ではz値＝奥行きはほぼ使いません。但し演出上で必要になるケースは多々有ります）

ミュージックスタート

音楽を追加します。

1. ProjectビューのAssets/Shooting/Prefab/BGMをHierarchyへD&D

もし自前の音楽（MP3やWav）を再生したいならば、

1. FinderやExplorerからProjectビューへ音楽をD&D
2. （長いインポート後に）Projectビューに音楽名と同じ内容の波形マークがある事を確認
3. 波形マークをHierarchyビューへD&D
4. HierarchyでD&Dと同じ名前の項目を選択
5. InspectorでAudio SourceのPlay On AwakeとLoopにチェックが入っていることを確認

スコアを設定する

1. ProjectビューのAssets/Shooting/Prefab/Score GUIをシーンビューへ配置
2. HierarchyのScore GUIを選択する
3. InspectorビューのCnavasのRender CameraをMain Cameraに設定する

これで敵を殺した時にスコアが増えます

画面外に出た敵キャラ・弾を消す

このままではオブジェクトが馬鹿みたいに増えていきます。 なので、オブジェクトが画面外に出たら消すようにします。

1. ProjectビューのAssets/Shooting/Prefab/DestroyAreaをHierarchyへD&D

進行を管理する

ゲームがいきなり始まり、しかも殺された後に何も出来ないのでは面白くありません。 ゲームの進行を作ります。

ゲームの流れとしては、「タイトルが表示される」「ｘキーでゲームスタート」「自機が出現する」「死んだらタイトル」「ｘキーで再開」といった感じです。

とりあえずタイトル配置します。

1. ProjectビューのAssets/Shooting/Prefab/TitleをHIerarchyへ配置
2. HierarchyのTitleを選択
3. InspectorビューのCnavasのRender CameraをMain Cameraに設定する

次にゲームの進行を管理するオブジェクトを配置します。 このオブジェクトはゲームが開始したタイミングでPlayerを生成します。

1. ProjectビューのAssets/Shooting/Prefab/ManagerをHierarchyビューへD&Dします。

再生してみると自機が2機同時に表示されてしまいます。これはManagerがPlayerをゲームスタートのタイミングで生成するからです。

Playerを削除します。

1. HierarchyビューのPlayerを選択
2. 右クリック -> Deleteを選択

(ここで警告が出た場合、誤ってProjectビューのファイルを消しています。消したファイルはゴミ箱に行かず即削除されるので、消さないように注意して下さい)

敵の出現パターンを設定する

敵をひたすら上に置いても良いのですが、ゲームの進行的には「特定の順番で敵を動的に配置する」方が理にかなっているケースがあります。STGとか。

なので、特定のパターンで敵が出現するように変更します。

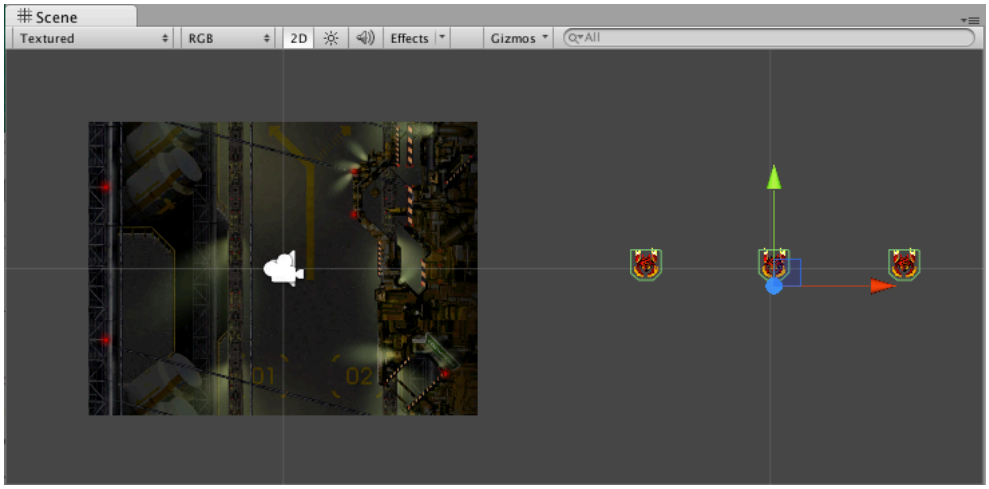
1. HierarchyのEnemyを全て削除する
2. ProjectビューのAssets/Shooting/Prefab/EmitterをHierarcyへD&Dします

新しくWaveのPrefabを作る

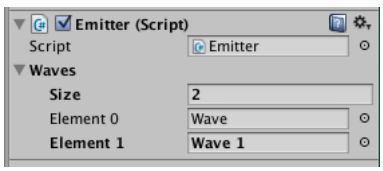
1. 少しだけゲームを面白くしたいと思います。自分で新たなWaveを作成しましょう。
2. Waveのプレハブを複製します。
    - プロジェクトビューのWaveのプレハブを選択し上部メニューの「Edit」から「Duplicate」を選択します。
    - または、cmd(⌘) + D
    - 「Wave 1」というPrefabが作成されました

Waveの追加

1. シーンビューの適当な位置にドラッグ＆ドロップしましょう。この時に背景と重なってはダメです。この後の操作がしづらいため。



2. シーンビューに表示されたら敵（Enemy）を1機選択します。
3. 敵をDuplicateしてどんどん配置していきましょう。
4. Enemyを選択するとインスペクターに情報が表示されます。そこにはHPや移動スピードを変更してみましょう。
    - ゲームを再生しながら確認すると調整がやりやすいです。パラメーターが戻ってしまうのに注意!
5. 今作ったWaveをPrefabに反映しましょう。「Wave 1」を選択してインスペクター上部の「Apply」を押してください。
6. Applyを押したらシーン上にあるゲームオブジェクトを削除します。
7. シーン上のEmmiterを選択し、今作成したWaveを追加しましょう。
    - **Waves**の▶をクリックして展開します。
    - **Size**を1から2へ変更します。
    - **Element 1**の部分に先程の**Wave 1**のプレハブを格納しましょう。



自分だけのゲーム完成！

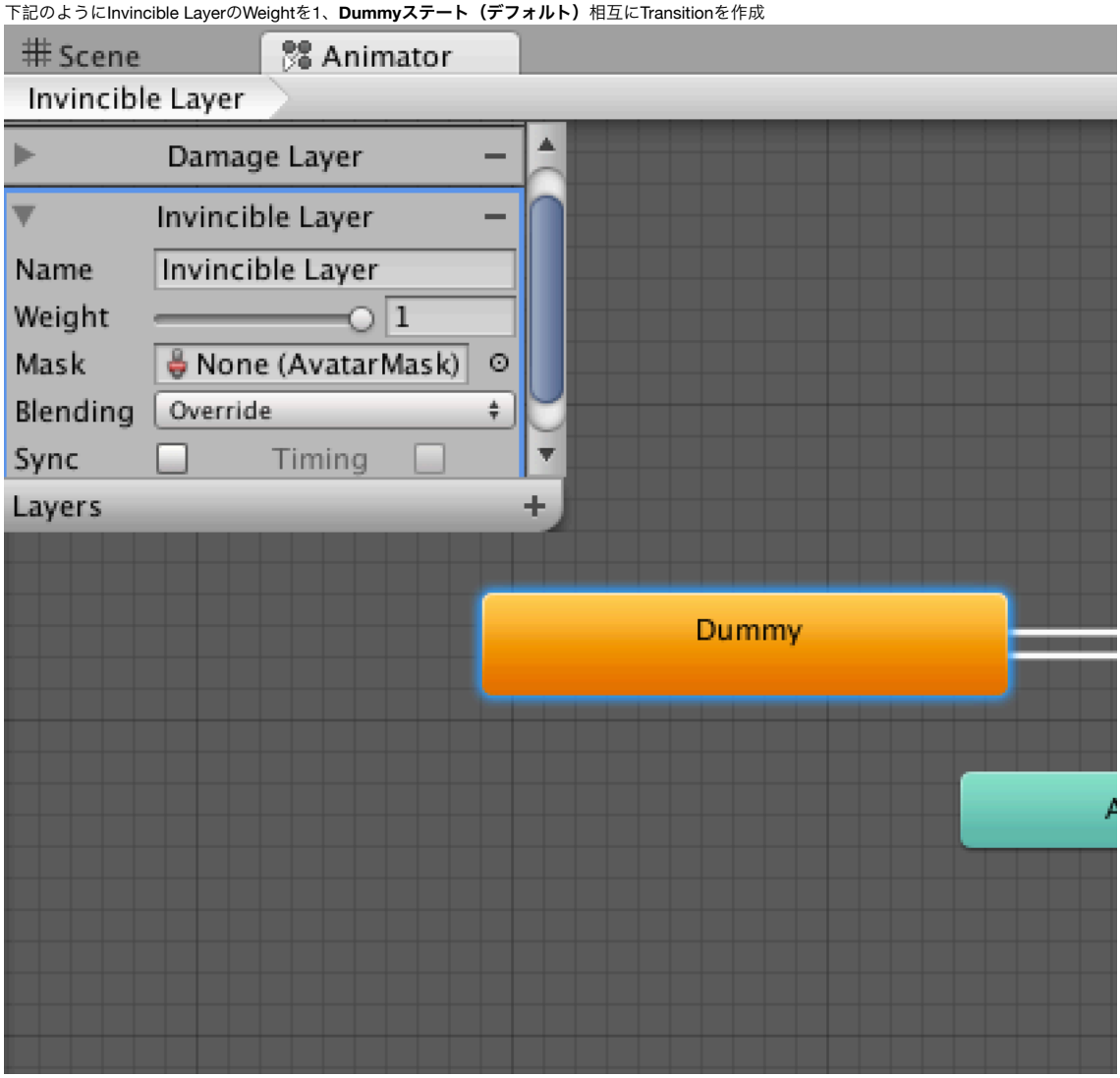
1. オリジナルのWaveが追加され、自分だけのシューティングゲームが出来ました。
2. 「File -> Save Scene」またはcmd(⌘) + S でシーンをアセットとして保存しましょう！

休憩！

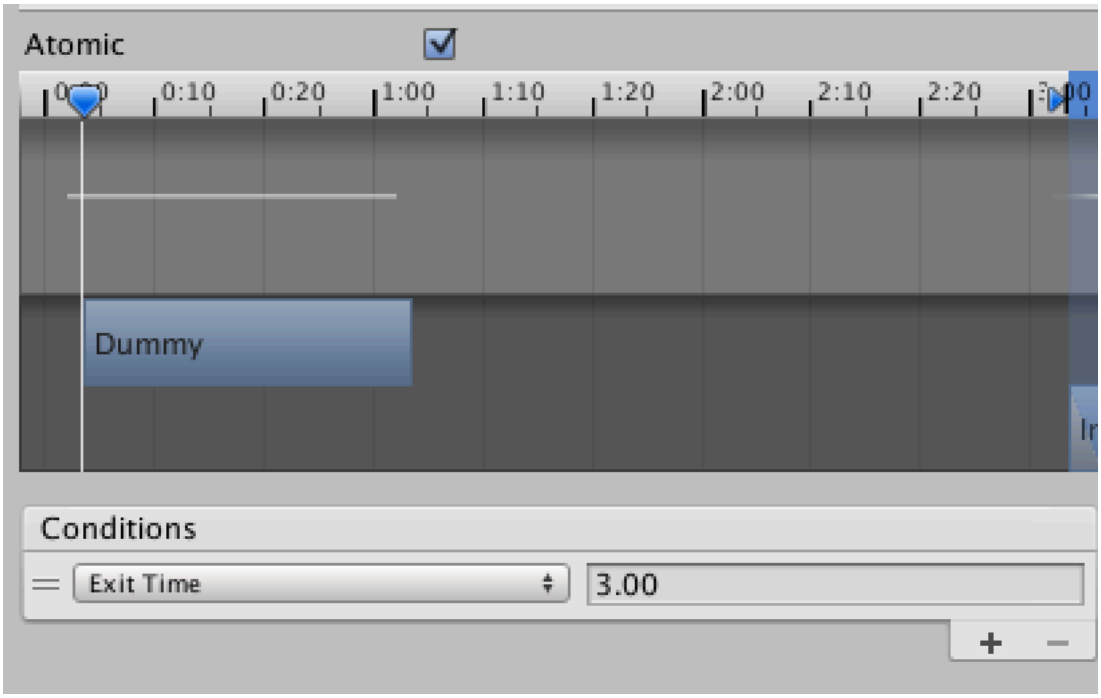
[応用編] Waveを作成

特定のタイミングで無敵になるエネミー

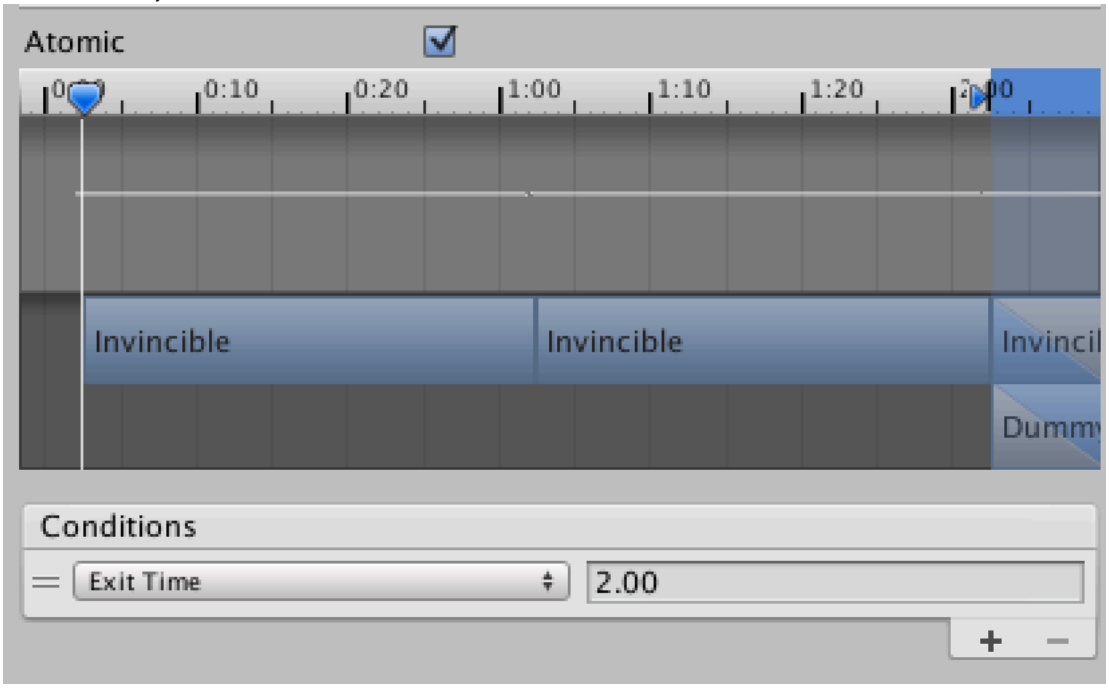
1. 複製してアニメーターコントローラーのEnemy(Invincible)を作成
2. Enemy AnimatorControllerに**「Invincible Layer」**を作成。
3. *Animations/Enemy*に**Invincible**アニメーションを作成
4. 下記のようにInvincible LayerのWeightを1、**Dummyステート（デフォルト）**相互にTransitionを作成



5. Dummy -> Invincibleは**Exit Timeを3**に



6. Invincible -> Dummyは**Exit Timeを2**に



7. 特定のエネミーだけ無敵にしたい場合
8. スクリプトを書く

```cs using UnityEngine; using System.Collections;

public class Enemy : MonoBehaviour { // 略 //
```

```
// 無敵になる間隔
public float invincibleTimeInterval;

void DoInvincible ()
{
    spaceship.GetAnimator ().SetTrigger ("Invincible");
}

IEnumerator Start ()
{
    // Spaceshipコンポーネントを取得
    spaceship = GetComponent<Spaceship> ();

    // 無敵にするメソッドを実行する
    // +2は無敵になっている時間。正確にはAnimator.GetNextAnimatorStateInfoでInvincibleアニメーション時間を取得する
    if (invincibleTimeInterval != 0)
        InvokeRepeating ("DoInvincible", invincibleTimeInterval, invincibleTimeInterval + 2);
```

```
* Invincibleトリガーを作成する。Dummy -> InvincibleのTransitionで、ConditionsをInvincibleトリガーのみにする
    * ![](https://dl.dropboxusercontent.com/u/153254465/screenshot2/ss%202014-10-14%2014.41.05.png)
* 適当なEnemyの「Invincible Time Interval」を0より大きく設定する
        ![](https://dl.dropboxusercontent.com/u/153254465/screenshot2/ss%202014-10-14%2015.13.38.png)

#### 一定時間特定の場所に居続けるエネミー

* Enemyアニメーションコントローラーで作業してもいいがフラグ管理が面倒なので、複製して「Enemy (Standby)」を作成する。
* Standby Layerを作成
    ![](https://dl.dropboxusercontent.com/u/153254465/screenshot2/ss%202014-10-14%2017.44.08.png)

* Standbyアニメーションを作成
    * 下に移動 -> 待機 -> 上に移動
        ![](https://dl.dropboxusercontent.com/u/153254465/screenshot2/ss%202014-10-14%2017.46.33.png)

### [応用] モバイル対応

#### スタート方法変更

* 画面をタップしたらゲームスタートにする
* **Manager.cs**を変更

```cs
void Update ()
{
    // ゲーム中ではなく、タップして離した状態でゲームを開始する。
    if (IsPlaying () == false && Input.GetTouch(0).phase == TouchPhase.Ended) {
        GameStart ();
    }
}
```

- Unity Remoteで確認

インプット対応

- AssetStoreのSmapleAssetsから「Cross Platform Input」をインポート
- Prefabから「Mobile Single Stick Control Rig」を選択し、Jumpボタンを削除
- Player.csを開き、Input.GetAxisRawを**CrossPlatformInput.GetAxisRaw**に変更
```