【目標】ダメージコントロール。

前回で、敵がプレイヤーに接触した場合ダメージを受け、HUD に反映されるところまで完成しました。

しかし、実行すると、ダメージを受けるのは、敵と接触した瞬間 1 回のみとなります。プレイヤーを操作して、いったん敵と離れることで再度接触し、ダメージを受けますが、再接触するまでやはりダメージを受けることがありません。

そこで、連続してダメージを受けることを考えますが、その際一定時間無敵状態を設定し、ダメージコントロールを行います。

ここでの重要キーワード： レイヤーの仕組みと利用方法、コルーチンの仕組みと利用方法、非同期処理

1. レイヤーの作成

今回の無敵状態を実装する際にレイヤーを設定することで、オブジェクトとオブジェクトの衝突を有効にしたり無効にしたりできます。そのためまずはレイヤーを設定します。

Edit > Project Settings > Tags and Layers

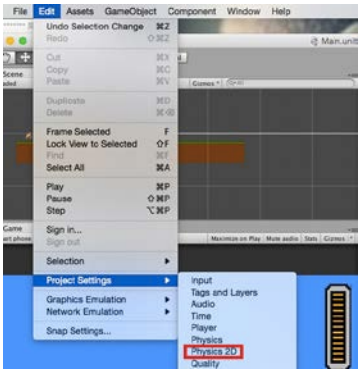
として、Tags＆Layer 設定を開き、Player、PlayerDamage、Enemy、EnemyDamage レイヤーを新規に作成します。

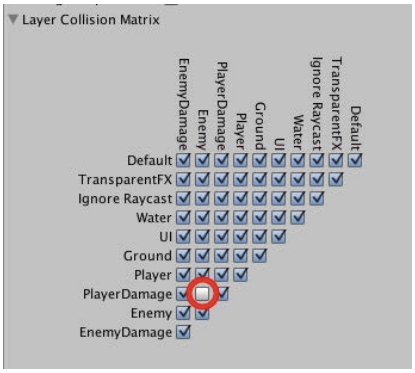
2. Layer Collision Matrix の設定

PlayerDamage レイヤーと Enemy レイヤーの衝突判定を無効にするために、Layer Collision Matrix を設定します。

Edit > Project Settings > Physics 2D

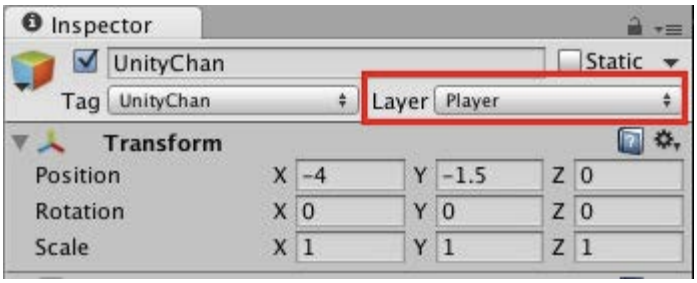
を選択し、Inspector ビュー一番下にある Layer Collision Matrix の、PlayerDamage と Enemy がぶつかっている部分のチェックを外します。



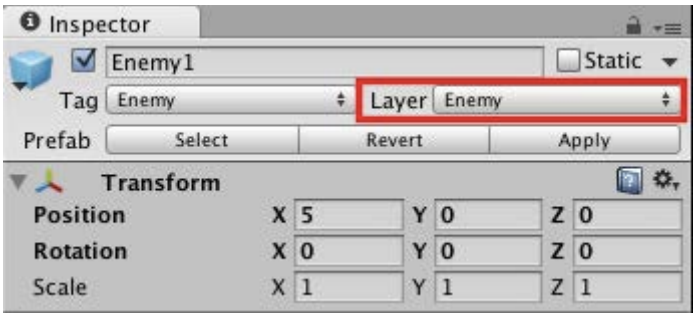


3. 無敵状態を実装する

作成したレイヤーを、ユニティちゃんと Enemy1 に設定します。



UnityChan の Inspector ビューにてレイヤー設定



Enemy1 の Inspector ビューにてレイヤー設定

ダメージを受けた際に、スクリプトを利用してプレイヤーのレイヤーを “Player” から “PlayerDamage” に一定時間変更し、敵との当たり判定を回避するようプログラムに追記します。

その制御を行っているのが、72 行目以降で、OnCollisionEnter2D で敵との当たり判定を行い、その中でダメージ処理”Damage”を呼び出しています。
この処理は、ゲーム進行に非同期して、ダメージ発生時のみの動作となるため、コルーチン呼び出しを行っているところがポイントです。

Player.cs

```
1   using UnityEngine;
2   using System.Collections;
3
4   public class Player : MonoBehaviour {
5
6       public float speed = 4f;
7       public float jumpPower = 700;
8       public LayerMask groundLayer;
9       public GameObject mainCamera;
10      public GameObject bullet;
11
12      private Rigidbody2D rigidbody2D;
13      private Animator anim;
14      private bool isGrounded;
15      private Renderer renderer;
16
17      void Start () {
18          anim = GetComponent<Animator>();
19          rigidbody2D = GetComponent<Rigidbody2D>();
20          renderer = GetComponent<Renderer>();
21      }
22
23      void Update ()
24      {
25          isGrounded = Physics2D.Linecast (
26              transform.position + transform.up * 1,
27              transform.position - transform.up * 0.05f,
28              groundLayer);
29          if (Input.GetKeyDown ("space")) {
30              if (isGrounded) {
31                  anim.SetTrigger ("Jump");
32                  isGrounded = false;
33                  rigidbody2D.AddForce (Vector2.up * jumpPower);
34              }
35          }
36          float velY = rigidbody2D.velocity.y;
37          bool isJumping = velY > 0.1f ? true : false;
38          bool isFalling = velY < -0.1f ? true : false;
```

```
39                anim.SetBool ("isJumping", isJumping);
40                anim.SetBool ("isFalling", isFalling);
41                if (Input.GetKeyDown ("left ctrl")) {
42                      anim.SetTrigger ("Shot");
43                      Instantiate (bullet, transform.position + new Vector3 (0f, 1.2f, 0f)
44                            ,Quaternion.identity);
45                }
46          }
47
48          void FixedUpdate ()
49          {
50                float x = Input.GetAxisRaw ("Horizontal");
51                if (x != 0) {
52                      rigidbody2D.velocity = new Vector2 (x * speed, rigidbody2D.velocity.y);
53                      Vector2 temp = transform.localScale;
54                      temp.x = x;
55                      transform.localScale = temp;
56                      anim.SetBool ("Dash", true);
57                      if (transform.position.x > mainCamera.transform.position.x - 4) {
58                            Vector3 cameraPos = mainCamera.transform.position;
59                            cameraPos.x = transform.position.x + 4;
60                            mainCamera.transform.position = cameraPos;
61                      }
62                      Vector2 min = Camera.main.ViewportToWorldPoint(new Vector2(0, 0));
63                      Vector2 max = Camera.main.ViewportToWorldPoint(new Vector2(1, 1));
64                      Vector2 pos = transform.position;
65                      pos.x = Mathf.Clamp(pos.x, min.x + 0.5f, max.x);
66                      transform.position = pos;
67                } else {
68                      rigidbody2D.velocity = new Vector2 (0, rigidbody2D.velocity.y);
69                      anim.SetBool ("Dash", false);
70                }
71          }
72          void OnCollisionEnter2D (Collision2D col)
73          {
74                //Enemy とぶつかった時にコルーチンを実行
75                if (col.gameObject.tag == "Enemy") {
76                      StartCoroutine ("Damage");
77                }
78          }
79
80          IEnumerator Damage ()
81          {
82                //レイヤーを PlayerDamage に変更
83                gameObject.layer = LayerMask.NameToLayer("PlayerDamage");
84                //while 文を 10 回ループ：10 回点滅
85                int count = 10;
```

```
86                  while (count > 0){
87                      //透明にする
88                      renderer.material.color = new Color (1,1,1,0);
89                      //0.05 秒待つ
90                      yield return new WaitForSeconds(0.05f);
91                      //元に戻す
92                      renderer.material.color = new Color (1,1,1,1);
93                      //0.05 秒待つ
94                      yield return new WaitForSeconds(0.05f);
95                      count--;
96                  }
97                  //レイヤーを Player に戻す
98                  gameObject.layer = LayerMask.NameToLayer("Player");
99              }
100         }
```

これで、

1. 敵と接触したとき、ダメージを受ける。
2. 一定時間無敵、すり抜け
3. その後復活、接触していれば再度ダメージ受ける

が、実装できました。

4. 動作を検証する

動作確認できたでしょうか？上記の内容を適用した後では下記の通りになっているはずです。

↓ 敵に向かって走り、接触すると点滅しダメージを食らう

↓ レイヤーが異なるためそのまま通り抜ける。

【検証】

敵の動きを停止させ、無敵回復後連続してダメージを受けることを確認しなさい。