ゲーム作り2

敵を作ろう

no4

あなたが正義を説くのならそれは誰かの悪でもあるわけです。というわけで敵を 作りましょう。

今回は敵を作りにあたってprefabという機能を使います。敵は基本的に複数いま す。これらを1体1体作っていたら作業量が膨大になってしまいます。これを回避 するためにそのオブジェクトの原型を作って、それらをコピーしてゲームシーン 上に出す機能がprefabです。

今までと同じようにヒエラルキーで右クリックから2D object > sprite >



 Main Camera 	
💬 Player	

これをprefabにします。やり方としてはヒエラルキー上のオブジェクトをプロジ ェクトウィンドウにドラッグアンドドロップするだけです。

+-	• • All	집	
	🔻 🔂 MainScene*	:	
	 ☆ Main Camera ◇ Grid ◇ Player ◇ enemy 	>	÷ ⇒ ↔ 5 1 3 3 3 3 3 3 3 3 3 3 3 3 3
P	Project E Console		
+• - •	, Foundation		
· -	Q All Materials Q All Models Q All Prefabs		
× 🖻	Assets kenney_pico-8-platformer enemy prefabs Scenes		
▶∎	■ Script Packages		

no4

画像のようにenemyが青っぽくなればできています。これから敵を編集するとき はヒエラルキーにあるenemyではなくプロジェクトウィンドウの方のenemyを変更 します。また、インスペクターウィンドウからSpriteをtilemap_packed_114とし て、BoxCollider2DとRigidbody2Dをつけましょう。

	🥥 🔻 2D 🌻 🎼 📚 🔻 🕫	≶ ≡ ₹ €	F F	Rotation Scale	ß	хо <u>үо</u> х1 Ү1	Z 0 Z 1		F
			▼ 1	🛃 🖌 Sprite Renderer					
						:•:tilemap_packed_114		0	
				Color				200	Į.
				Jraw Mode		Simple			
						None			
						Center		•	
				Material Additional Settings		Sprites-Default		0	
			- 1°	Sorting Layer					
			v I	🔳 🗹 Box Collider 2D			0		
						None (Physics Material 2D)		٥	
				s Trigger					
				Jsed By Effector Ised By Composite					
			4	Auto Tiling					
			S			X 1 Y 1			
			E ► II	age Radius nfo					
				Rigidbody 2D			0		
		a		Body Type					1
٩		* * *	72>5 ∧			None (Physics Material 2D)		٥	
				Jse Auto Mass					

プレイヤーと敵の死亡判定

プレイヤーが敵に当たったら死ぬようにしましょう。当たり判定と接触したとき にそれが地面なのか敵なのかを判定しなければなりません。今回は、tagという 機能を使って判定します。まず、敵にtagをつけましょう。enemyを選択してイン スペクターウィンドウから上部のtagを選んでAdd tagを選択して+からEnemyと入 力してsaveを押してください。



```
private void OnCollisionEnter2D(Collision2D collision)
{
    is_field = true;
}
```

ここでOnCollisionEnter2D(Collision2D collision)の部分に引っ付いてい るCollision2D collisionを活用します。これはこの関数が実行されるとき、引 数として接触した当たり判定を取得できます。つまり、敵にぶつかったときは、 このcollisionという変数を敵の当たり判定として使用できます。ので、この collisionがつけられているオブジェクトのtagがEnemyの場合死亡するという処 理を作ればいいわけです。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Enemy")
    {
        //死ぬ
    }
    is_field = true;
}
```

このようにすればいいです。次は死んだときの処理を作りましょう。今回は 簡単にプレイヤーを消すだけにしましょう。オブジェクトを破壊するときは Destroyという関数を使います。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Enemy")
    {
        Destroy(gameObject);
    }
    is_field = true;
}
```

この状態で敵にプレイヤーがぶつかると消滅するはずです。では次にプレイ ヤーが敵を倒す処理を作りましょう。ゲームの敵というものはこぞって頭上から の攻撃に弱いものなので上から踏んだら倒せるようにしましょう。 上から踏んだら倒せるということは、プレイヤーが敵と接触したときに上から踏 んだなら敵を消し、それ以外ならプレイヤーを消せばいいわけです。つまり次の ようになります。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Enemy")
    {
        if (/*上から踏んだかどうか*/)
        {
            Destroy(collision.gameObject);
        } else
        {
            Destroy(gameObject);
        }
    }
    is_field = true;
}
```

上から踏んだかどうかの判定は法線での判定を使いましょう。法線というの は当たり判定の接触面からプレイヤーに線を引いたときにできる線のことで上向 きであるほど1に近づき、下向きであるほど-1に近づきます。



この黒線が法線で、画像の場合だいたい0.5くらいです。今回はプレイヤーと敵の接触でできた法線が0.7以上なら敵を踏んで倒せたことにしましょう。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Enemy")
    {
        ContactPoint2D contact_face = collision.contacts[0];
        Debug.Log(contact_face.normal.y);
        if (contact_face.normal.y >= 0.7f)
        {
            Destroy(collision.gameObject);
        } else
        {
            Destroy(gameObject);
        }
    }
}
```

}

is_field = true;

こう書き直してプレビューを確認しましょう。だいたい思った通りの挙動に なったと思います。

敵の移動とバグ修正

次に敵が動くようにしましょう。今回はクリボーに習ってひたすら前に進むよう にしましょう。これは簡単ですね。Update関数に前に進む機能を実装すればいい だけです。EnemyControllerを開いてください。

```
public class EnemyController : MonoBehaviour
{
    Rigidbody2D rb;
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
    }
    void FixedUpdate()
    {
        if(rb.velocity.x > -2)
        {
            rb.AddForce(new Vector2(-50, 0));
        }
    }
}
```

PlayerControllerの復習にもなると思います。始めにRigitbody2Dを取得し

て、Update関数で速度制限を設けながらAddforceで力を加えています。

Enemy(Prefabの元の方)にEnemyControllerをつけて実行してみてください。

Interpolate	None				
Constraints					
► Info					
🗯 🗹 Enemy Controller (Script)					
	EnemyController				
Sprites-Default (Material)					
Shader Sprit	tes/Default				

そしてここで忘れていましたがたまに敵やプレイヤーが転んだり、プレイヤーと敵がぶつかったときに押されて移動するときがあると思います。これを修正しましょう。まず、転ぶのに関してはRigitbody2Dの設定で治せます。z軸での回転を無効化しましょう。

. .

🔻 🕂 🛛 Rigidbody 2D		؇∶
Body Type	Dynamic	
Material	None (Physics Material 2D)	
Simulated	~	
Use Auto Mass		
Mass		
Linear Drag	0	
Angular Drag	0.05	
Gravity Scale	1	
Collision Detection	Discrete	
Sleeping Mode	Start Awake	
Interpolate	None	
Freeze Position	XY	
Freeze Rotation	∠ Z	
▶ Info		

画像のようにFreeze Rotation Z にチェックをいれてください。両方のオブジェ クトで行ってください。次に押されるのを直しましょう。これは物理演算によっ て押されているので、重さを変更しましょう。EnemyのRigitbody2Dを見てくださ

• • •		
🔻 ᆉ 🛛 Rigidbody 2D	9	-7 i
Body Type	Dynamic	•
Material	None (Physics Material 2D)	\odot
Simulated	~	
Use Auto Mass		
Mass	3	
Linear Drag	0	
Angular Drag	0.05	
Gravity Scale		
Collision Detection	Discrete	•
Sleeping Mode	Start Awake	•
Interpolate	None	-

また、敵が移動しているときに突然止まることがあるときはBoxCollider2Dの

🔻 🔳 🖌 Box Collider 2D		0	ᅷ
Edit Collider	♪		
Material	None (Physics Material 2D)		
ls Trigger			
Used By Effector			
Used By Composite			
Auto Tiling			
Offset	X 0 Y 0		
Size	Х 1 Ү 1		
Edge Radius	0.01		
▶ Info			

また、最後に敵が壁にぶつかったら逆方向に折り返すようにしましょう。これ は、移動ができなかった場合、反転するというやり方で実装します。

```
Vector3 last_pos;
int vector_pm = -1;
void FixedUpdate()
{
    if (last_pos == transform.position)
    {
```

no4

```
2025/06/16
```

```
this.transform.localScale = new Vector3(this.transform.localScale.x * -1, 1, 1);
    vector_pm *= -1;
}
else
{
    last_pos = this.transform.position;
}
Debug.Log(vector_pm);
if(Math.Abs(rb.velocity.x) < 2)
{
    rb.AddForce(this.transform.right*-50*vector_pm);
}</pre>
```

ここは少しややこしいです。まず、前の状態からかなり変わっていると思い ますが、1つ前の処理後の敵の位置をlast_posに保存し、それを次の処理で今の 位置と比較することで、移動されたかどうかを判別しています。また、 vector_pmは移動するときに係数としてaddforceの値にかけることで移動方向の 逆転を実装しています。

これで最低限の敵は完成です。

ここで、実はバグが残っています。ここで壁に向かってジャンプしてみるとおそ らく何回でもジャンプできちゃうと思います。これはis_fieldが床壁関係なくリ セットされているためです。床かどうかの判別をしなければなりません。今回は 敵を踏んだ時と同じように踏んだ角度を見て床かどうか判断しましょう。次のよ うに書き直してください。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    ContactPoint2D contact_face = collision.contacts[0];
    if (collision.gameObject.tag == "Enemy")
    {
        Debug.Log(contact_face.normal.y);
        if (contact_face.normal.y >= 0.7f)
        {
            this.StepOnEnemy(collision);
        } else
        {
            this.Die();
        }
    } else if(collision.gameObject.tag == "Goal")
    {
        //ゴール検知
        SceneManager.LoadScene("ResultScene");
    }
```

```
}
```

```
private void OnCollisionStay2D(Collision2D collision)
```

```
{
    ContactPoint2D contact_face = collision.contacts[0];
    Debug.Log(contact_face.normal.y);
    if (contact_face.normal.y >= 0.5f)
    {
        is_field = true;
    }
}
```

ゴール作り

no4

今回はゴールしたらそれを検知するまででいいです。今までと同じようにオブジ





また画像はtilemap_packed_146としましょう。今回、このオブジェクトに物理演 算はさせないので、BoxCollider2Dのみつけましょう。



またGoalタグを作成し、つけましょう。

2025/06/16

no4

€	•	4	Tags & Layers			9 -	2:
			Tags				
			Tag O	Enei			
	N		Tag Name	Goal		+	
				Save			

PlayerControllerに戻り、当たり判定に変更を加えましょう。

```
private void OnCollisionEnter2D(Collision2D collision)
{
    if(collision.gameObject.tag == "Enemy")
    {
       ContactPoint2D contact_face = collision.contacts[0];
       Debug.Log(contact_face.normal.y);
       if (contact_face.normal.y >= 0.7f)
        {
           Destroy(collision.gameObject);
       } else
        {
           Destroy(gameObject);
        }
   } else if(collision.gameObject.tag == "Goal")
    {
       //ゴール検知
   }
   is_field = true;
}
```

これで最低限です。次はクリア後の処理とタイトルからの移動、リスタート を作りましょう。