7
Collisions et gestion de la vie

Dans ce chapitre, nous allons voir comment détecter les collisions par le code. Cela nous permettra de savoir si une balle touche le joueur, s'il tombe dans un piège, s'il ramasse un objet ou s'il tire sur un ennemi. Nous allons programmer tout cela ici.

7.1. Perdre des points de vie

La première chose à faire c'est créer un script de collision qui sera placé sur le joueur. Créez donc un script `PlayerCollision` et placez-le sur le FPSController. Ouvrez ensuite ce script. Vous pouvez supprimer les fonctions `Start` et `Update`. Ce script détectera si un ennemi a touché le joueur et lui fera perdre de la vie. Perdre de la vie signifie également mettre à jour l'interface utilisateur. Pour pouvoir utiliser des fonctionnalités du système UI de Unity, nous devons le déclarer en préliminaire au début du programme à l'aide du mot clé `using` : `using UnityEngine.UI;`. Au niveau des variables, il nous faudra une variable de type `Text` qui stockera le texte affichant la vie : `public Text healthTxt;`. Notez que sans le `using` précédent, il nous serait impossible d'utiliser le type `Text`. Nous avons également besoin d'une variable de type `int` qui stockera la valeur de la vie : `public int health = 100;`.

Nous pouvons maintenant coder la détection de collision. Au chapitre précédent, nous avons créé un projectile pour tourelles. Ce projectile a un Collider et un Tag. Pour ma part, ce tag est `"Tourelle"`. Nous voulons donc détecter la collision avec un objet tagué `Tourelle`. Pour détecter une collision, Unity propose la fonction `OnCollisionEnter` qui prend en paramètre une collision :

```
void OnCollisionEnter(Collision collision)
{

}
```

Dans cette fonction, nous pouvons écrire une condition afin de tester si le tag de la collision correspond à celui que nous souhaitons tester :

```
// Détecter les collisions
void OnCollisionEnter(Collision collision)
```

```
{
  // Si l'objet qui touche le joueur a le Tag Tourelle
  if(collision.gameObject.tag == "Tourelle")
  {

  }
}
```

À l'intérieur de cette condition nous pouvons décrire ce qu'il se passe lorsque le personnage joueur est touché par un projectile. Dans notre exemple, il perdra de la vie, disons 20% : `health = health - 20;`. Puis nous devons mettre à jour le texte affiché à l'écran : `healthTxt.text = health + "%";`. Pensez également à détruire le projectile une fois qu'il a touché le joueur avec `Destroy(collision.gameObject);` car il n'est pas nécessaire de le conserver actif. Voilà le code complet :

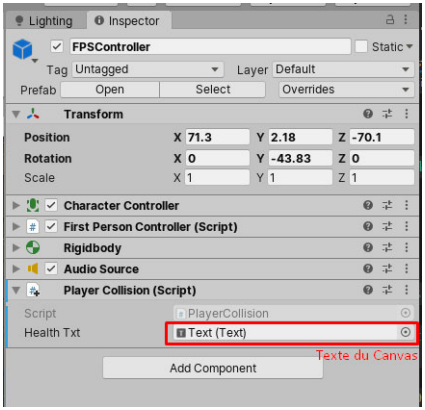
```
using UnityEngine;
using UnityEngine.UI;

public class PlayerCollision : MonoBehaviour
{
  public Text healthTxt;
  public int health = 100;

  // Détecter les collisions
  void OnCollisionEnter(Collision collision)
  {
    // Si l'objet qui touche le joueur a le Tag Tourelle
    if(collision.gameObject.tag == "Tourelle")
    {
      health = health - 20; // Le joueur perd 20% de vie
      healthTxt.text = health + "%"; // Mise à jour du texte
      Destroy(collision.gameObject); // Détruire le projectile
    }
  }
}
```

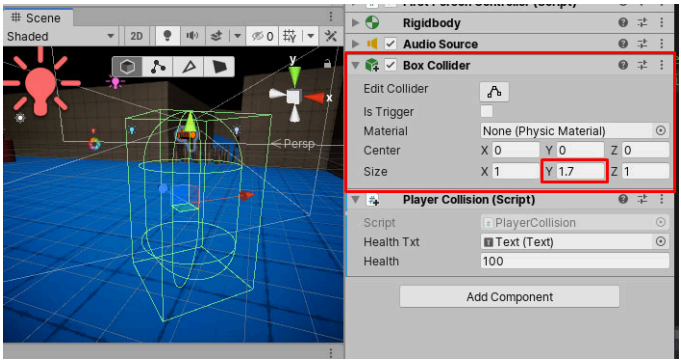
Avant de tester votre programme, renseignez la variable du script :

Figure 7.1 : Configuration du script de collision



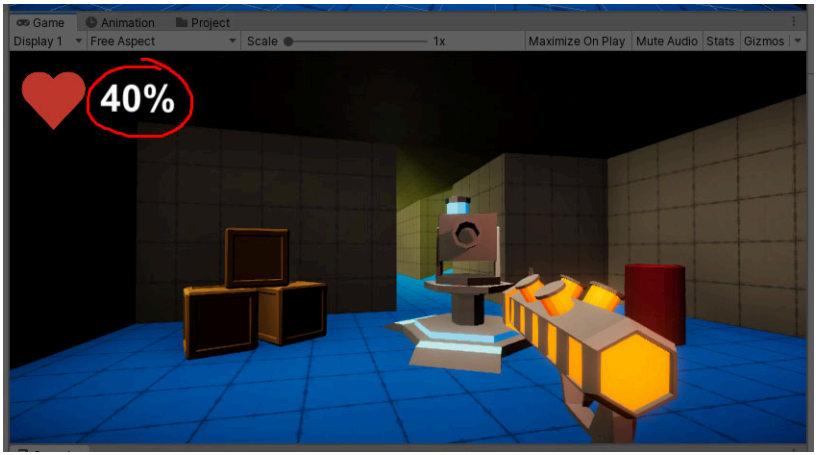
Puis pensez à ajouter un BoxCollider à votre personnage afin qu'il dispose d'une surface de collision (le CHARACTERCONTROLLER ne fonctionnerait pas ici).

Figure 7.2 : Ajout du Collider au personnage



Lancez ensuite votre jeu et faites vous tirer dessus par une tourelle.

Figure 7.3 : Test du script



Comme vous pouvez le voir notre script fonctionne bien et nous perdons de la vie... Enfin il reste un petit bug. Si vous restez sur place, vous verrez que votre vie descendra à l'infini ! Le joueur peut se retrouver avec une valeur négative improbable. Nous allons régler cela avec une condition qui testera si la vie est négative et qui la remettra à zéro si c'est le cas. Pour que le code soit plus propre, je vous propose de créer une fonction indépendante qui sera chargée de diminuer la vie. Voilà le code complet ajusté :

```
using UnityEngine;
using UnityEngine.UI;

public class PlayerCollision : MonoBehaviour
{
  public Text healthTxt;
  public int health = 100;

  // Détecter les collisions
  void OnCollisionEnter(Collision collision)
  {
    // Si l'objet qui touche le joueur a le Tag Tourelle
    if(collision.gameObject.tag == "Tourelle")
    {
      LooseHealth(20); // Perte de 20 points de vie
      Destroy(collision.gameObject); // Détruire le projectile
    }
  }

  // Perdre de la vie
  void LooseHealth(int val) // val = valeur à déduire
```

```
    {
      health = health - val; // Le joueur perd val% de vie
      healthTxt.text = health + "%"; // Mise à jour du texte
      if(health < 0)
      {
        health = 0; // On bloque la vie à 0
        // Le personnage n'a plus de vie, il faut lancer le GameOver ici
      }
    }
  }
```

C'est la fonction `LooseHealth` qui est chargée de diminuer la vie du joueur. Vous remarquerez que si la vie tombe à zéro, il faut lancer le GameOver. Par simplicité, je vous propose de relancer le jeu automatiquement depuis le début. Pour relancer le niveau, vous devez utiliser un `using` qui vous permettra de travailler avec les scènes : `using UnityEngine.SceneManagement;`.

Puis vous devez charger une scène. Notre jeu étant très simple, il ne fonctionne qu'avec une seule scène. Les scènes sont rangées par index en commençant par zéro. Lorsque nous n'avons qu'une scène, elle est à l'index zéro. Pour la charger vous écrirez : `SceneManager.LoadScene(0);`. Cette séquence de code vous permettra donc de relancer le jeu dès que le joueur perd. Si vous souhaitez aller plus loin, vous pourrez créer une interface affichant une image de GameOver avant de relancer le niveau.

7.2. Tomber dans un piège

Nous allons maintenant programmer la perte de vie lorsque le joueur tombe dans le piège. Le code sera très proche de ce que nous venons de faire sauf que le Collider du piège sera en mode TRIGGER. Lorsqu'un objet est en mode TRIGGER, le personnage peut passer au travers comme s'il n'y avait pas de collision mais nous pouvons tout de même détecter la collision par script.

Figure 7.4 : *Le Collider du piège*