Détection des collisions

A- Avec un objet qui possède un collider sans être trigger

void OnCollisionEnter(Collision infoObjetCollision)

Cette fonction est appelée automatiquement par Unity lorsque le gameObjet qui possède un *Rigidbody* entre en collision avec un autre objet. Cet objet doit avoir un *Collider* et sa propriété *isTrigger* ne doit pas être cochée. Il n'est pas nécessaire que l'objet possède un composant *Rigidbody* . Le script peut être placé sur l'un ou l'autre des objets impliqués dans la collision.

Exemple

```
// détecte les collisions du personnage (l'objet de ce script) s'il touche l'objet "Mur", active l'explosion ….
  public GameObject objetExplosion ;

  void OnCollisionEnter(Collision infoCollision)  // le type de la variable est Collision
  {
    if (infoCollision.gameObject.name == "Mur")
    {
      objetExplosion.SetActive (true);
    }
    else if (infoCollision.gameObject.name == "Eau")
    {
      infoCollision.gameObject.SetActive (false);
    }
  }
```

Rappels:

Détruire un objet : Destroy(GameObject objet, float delais)

Enlève l'objet de la scène. Les objets « parentés » sont aussi détruits (caméra, etc.). S'il joue un son, le son s'arrête. Les scripts de l'objet détruit arrêtent aussi leur exécution.

Activer/Désactiver un objet : gameObject.SetActive(true ou false);

Active ou désactive un objet de la scène. Si les objets enfants sont actifs, ils suivent l'activation ou la désactivation du parent.

B- Avec un objet qui possède un collider qui est trigger

void OnTriggerEnter(Collider infoObjet) -- Attention : type Collider et non Collision

Cette fonction est appelée automatiquement par Unity lorsque le gameObjet qui possède un *Rigidbody* entre en collision avec un autre objet. Cet objet doit avoir un *Collider* et sa propriété *isTrigger* doit être cochée. Il n'est pas nécessaire que l'objet possède un composant *Rigidbody* . Le script peut être placé sur l'un ou l'autre des objets impliqués dans la collision.

Exemple

```
// détecte les collisions du personnage (l'objet de ce script) s'il touche l'objet "Vie" ou "Bombe"
   int nbrVie =  0;

   void OnTriggerEnter(Collider infoObjet)   // le type de la variable est Collider
   {
      if(infoObjet.gameObject.tag == "Vie")
      {
         infoObjet.gameObject.SetActive(false);
         nbrVie ++;
      }
      else if(infoObjet.gameObject.name == "Bombe")
      {
         infoObjet.gameObject.SetActive(false);
         nbrVie --;
      }
   }
```

Relancer la scène de jeu

Important : Pour utiliser les commandes qui suivent, il est nécessaire d'ajouter " **using UnityEngine.SceneManagement;** " dans le haut de votre script

using UnityEngine.SceneManagement;

SceneManager.LoadScene (*IndexDeLaScene*)
SceneManager.LoadScene ("*NomDeLaScene*")

Un deuxième paramètre peut être ajouté, "LoadSceneMode.Single" ou "LoadSceneMode.Additive". Si on ne précise pas de deuxième paramètre, c'est le "LoadSceneMode.Single" qui prévaut. Cette option permet de déterminer si la nouvelle scène est ajoutée à la première (additive) ou si elle est chargée seule (single).

Exemple : SceneManager.LoadScene ("scene2",LoadSceneMode.Additive);

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement; // Nécessaire pour utiliser la classe SceneManager.

public class GestionJeu : MonoBehaviour
{
// Fonction appelée en fin de partie qui permet de charger la scène finale.
   void FinDePartie()
   {
      SceneManager.LoadScene ("SceneFinale");
   }
}
```

Faire une pause dans le script

yield return new WaitForSeconds(secondes:float);

Pour marquer une temporisation en C# à l'aide de *yield WaitForSeconds()* , il est nécessaire d'utiliser une ***coroutine*** avec une fonction de type ***IEnumerator***. Voici un exemple :

Exemple

```
void OnCollisionEnter(Collision infoCollision)   // le type de la variable est Collision
   {
      if (infoCollision.gameObject.name == "Bombe")
      {
         StartCoroutine (Explose(2f, infoCollision.gameObject)); // On déclenche la coroutine Explose en passant
                                                                 // deux valeurs.
      }
   }

   // Fonction de type IEnumerator qui permet l'utilisation de temporisation yield WaitForSeconds.
   IEnumerator Explose(float tempsAttente, GameObject objetAdetruire)
   {
      objetAdetruire.GetComponent<AudioSource> ().Play (); // l'objet touché déclenche un son
      yield return new WaitForSeconds(tempsAttente); // Une pause est marquée. La durée est déterminée par le
                                                     // paramètre tempsAttente
      Destroy (objetAdetruire); // l'objet touché est détruit.
     SceneManager.LoadScene ("SceneFinale");
   }
```

Invoke("nomFonction", temps) et InvokeRepeating("nomFonction", temps, tempsRépétition)

Ces deux commandes permettent aussi de marquer une temporisation.

Invoke appellera une fonction une fois, après un certain temps d'attente. Le premier paramètre (string) précise le nom de la fonction qui sera appelée. Le deuxième paramètre (float) précise le temps d'attente avant l'appel de la fonction.

InvokeRepeating appellera une fonction a répétition. Le premier paramètre (string) précise le nom de la fonction qui sera appelée. Le deuxième paramètre (float) précise le temps d'attente avant l'appel de la fonction pour la première fois. Le troisième paramètre (float) précise le temps d'attente avant chaque répétition.

CancelInvoke() : Cette commande permet de supprimer tous les "Invoke" et "InvokeRepeating" dans un script.

CancelInvoke("nomFonction") : Cette commande permet de supprimer les "Invoke" et "InvokeRepeating" qui appelle la fonction mentionnée dans un script.

Exemples

```
void OnCollisionEnter(Collision infoCollision)   // le type de la variable est Collision
  {
    if (infoCollision.gameObject.name == "Bombe")
    {
        objetAdetruire.GetComponent<AudioSource> ().Play (); // l'objet touché déclenche un son
        Invoke (“Explose”, 2f); // On déclenche la coroutine Explose après 2 secondes d’attente
    }
  }

  // Fonction appelée après 2 secondes.
  void Explose()
  {
      Destroy (objetAdetruire); // l'objet touché est détruit.
      SceneManager.LoadScene ("SceneFinale");
  }
```

```
void Start()
  {
    // Dans 2 secondes, la fonction PousseObjet sera appelée toutes les 0.3 seconde
    InvokeRepeating("PousseObjet", 2, 0.3F);
  }

  void PousseObjet()
  {
    // On pousse l'objet en appliquant une force
    GetComponent<Rigidbody>().AddRelativeForce(0,0,5,ForceMode.Impulse);
  }

  void Update()
  {
    // On annule seulement le Invoke de la fonction PousseObjet
    if (Input.GetKeyDown(KeyCode.Space))
      CancelInvoke("PousseObjet");
  }
```