But du laboratoire

- Continuer l'exploration de l'éditeur Unity et des différents GameObject standards
- Identification des GameObject et des composants dans les scripts
- Distinguer le rôle des principaux événements d'initialisation de la boucle de jeu

Description sommaire du travail pratique

À partir d'une ébauche de projet fournie par vos enseignants, vous devez :

1. Modifier des scènes comportant un ou des GameObject complexes qui se composent d'une hiérarchie de GameObject.
2. Modifier les scripts de comportement pour mettre en place des stratégies d'identification des GameObject et des composantes.

Étapes préliminaires

1. Télécharger la version étudiante du laboratoire 3
2. Décompresser le projet
3. Ouvrir le projet à partir de Unity Hub

Scène 1

1. Assurez-vous que la scène active soit la Scène 1
2. Identification d'un *component* au sein d'un *GameObject*
    a. Ouvrez le script « ComportementCanonAutonome »
    b. On prend le temps de lire le code, en particulier la méthode `TirerProjectile()`

```
public void TirerProjectile()
{
   float vitesseCanon = (transform.position - AnciennePosition).magnitude /
                        Time.deltaTime;
   var projectile = Instantiate(Projectile, BoucheDuCanon.position,
                                BoucheDuCanon.rotation);
   var modeDéplacement = projectile.GetComponent<TrajectoireLineaire>();
   modeDéplacement.AjusterVitesseProjectile(vitesseCanon);
}
```

3. Identification d'un *GameObject* au sein d'une hiérarchie
    a. Effacez ou mettez en commentaire le qualificatif `[SerializeField]` qui au-dessus de la déclaration `Transform BoucheDuCanon`
    b. Ajouter la méthode `void Start()` sous la méthode `void Awake()`
    c. Écrivez le code ci-dessous dans la méthode `Start()`

```
private void Start()
{
   Transform[] mesTransform = GetComponentsInChildren<Transform>();
   BoucheCanon = mesTransform.First<Transform>(X => X.gameObject.name == "Bouche");
}
```

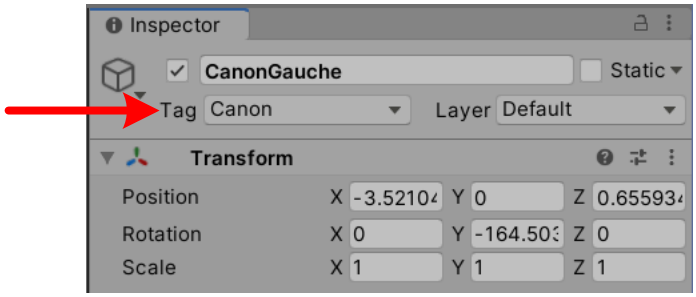
4. À vous de jouer
    a. À partir de la méthode `TirerProjectile()`, activez la trainée du projectile qui vient d'être tiré.
5. N'oubliez pas de sauvegarder la scène.

Scène 2 – Première partie

1. Assurez-vous que la scène active soit la Scène 2
2. Étudiez le script « ComportementCanonBatterie ». Notez que le script ne comporte plus de méthodes de réponse aux événements de la boucle de jeu (fonctions de rappel ou *Callbacks*). Cependant :
    a. Elle contient cependant une méthode publique qui permet de tirer un projectile.
    b. Cette méthode gère le temps de recharge du canon.
3. Exploration de la scène 1 – Rechercher des *GameObjects* en fonction de leur étiquette (*Tag*)
    a. Ouvrez le script « ComportementBatterie »
    b. Déclarer la variable Canons qui est un tableau de `GameObject` destinés à contenir tous les canons de la scène.

```
GameObject[] Canons;
```

    c. Ajouter la méthode `Start()`, puis inscrivez les instructions qui vous permettront de rechercher tous les objets qui partagent l'étiquette « Canon ». L'étiquette d'un `GameObject` est définie dans la zone de l'inspecteur.



    Pour rechercher tous les `GameObject` qui partagent une même étiquette, on utilise la méthode de classe `FindGameObjectsWithTag()` de la classe `GameObject`. Ce qui nous donne le code ci-dessous

```
private void Start()
{
   Canons = GameObject.FindGameObjectsWithTag("Canon");
}
```

    d. Dans la méthode `Update()`, si la variable `tirDemandé` devient égale à vrai, il faut parcourir le tableau Canons et pour chaque canon, identifier son composant « ComportementCanonBatterie », puis appeler la méthode `TirerCanon()`. Ce qui nous donne le code ci-dessous.

```
void Update()
{
   bool tirDemandé = Input.GetKey(ToucheDeTir);
   if (tirDemandé)
   {
      foreach (GameObject canon in Canons)
      {
         canon.GetComponentInChildren<ComportementCanonBatterie>().TirerCanon();
      }
   }
}
```

4. À vous de jouer
    b. Si vous exécutez le programme dans l'éditeur, vous verrez que le script « ComportementBatterie » est fonctionnel. Cependant, l'efficacité du code est perfectible. Modifier ce code de façon à le rendre plus performant, tout en conservant la recherche en fonction de l'étiquette (`FindGameObjectsWithTag`).
6. N'oubliez pas de sauvegarder la scène.

Scène 2 – deuxième partie

1. Assurez-vous que la scène active soit la Scène 2
2. Étudiez le script « DeplacementCanonBatterie ». Notez que le script ne comporte plus de méthodes de réponse aux événements de la boucle de jeu (fonctions de rappel ou *Callbacks*). Cependant :
    a. Elle contient cependant une méthode publique qui permet de déplacer un `GameObject`.
    b. Elle contient cependant une méthode publique qui permet de faire pivoter un `GameObject`.
3. Exploration de la scène 2 – Rechercher des références en fonction de leur type.
    a. Ouvrez le script « DeplacementBatterie »
    b. Déclarer la variable Canons qui est un tableau de `DeplacementCanonBatterie` destinés à contenir les références vers les composants `DeplacementCanonBatterie` de tous les canons de la scène.

```
DeplacementCanonBatterie[] Canons;
```

    c. Ajoutez la méthode `Start()`, puis inscrivez les instructions qui vous permettront de rechercher tous les objets (composant dans notre cas) de type `DeplacementCanonBatterie`. Pour réaliser cette tâche, on utilise la méthode de classe générique `FindObjectsOfType<type>()` de la classe `GameObject` où *type* est le type de l'objet recherché (`DeplacementCanonBatterie` dans notre cas). Ce qui nous donne le code ci-dessous

```
private void Start()
{
   Canons = GameObject.FindObjectsOfType<DeplacementCanonBatterie>();
}
```

    d. Dans les méthodes `Déplacer()` et `Pivoter()`,il faut parcourir le tableau Canons et pour chaque canon, puis appeler la méthode appropriée du composant `DeplacementCanonBatterie`. Ce qui nous donne le code ci-dessous.

```
private void Déplacer(float vitesseLinéaire)
{
   foreach(var canon in Canons)
   {
      canon.Déplacer(vitesseLinéaire);
   }
}

private void Pivoter(float vitesseAngulaire)
{
   foreach (var canon in Canons)
   {
      canon.Pivoter(vitesseAngulaire);
   }
}
```

5. À vous de jouer
    a. Exécuter le programme dans l'éditeur et vérifier le fonctionnement des différentes touches de déplacement.
    b. Que se passe-t-il si l'on ajoute une nouvelle instance du prefab « CanonDeBatterie » sur la scène et que l'on redémarre l'exécution ? (N'oubliez pas de modifier la `position` du composant `transform` 😊)
6. Pensez à sauvegarder la scène.