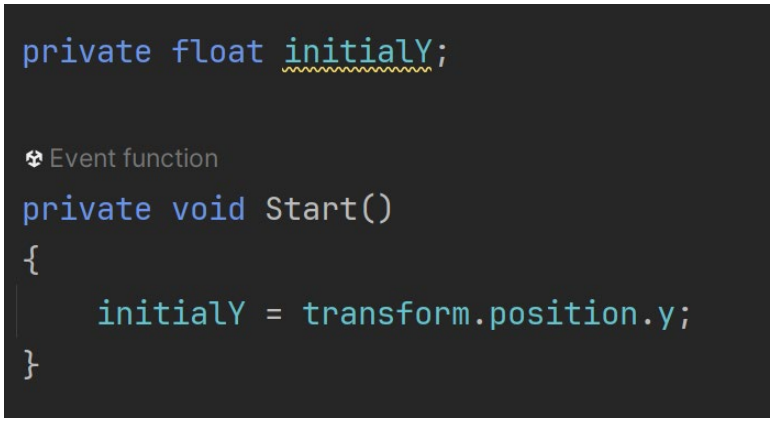
🎮 Projet Unity AR — Angry Bird AR **(Partie 3)**

🎯 Objectif

Faire une application en réalité augmentée, on place une pile d'objets et on lance des balles pour la faire tomber. Un score est calculé.

Pour cette partie 3, on va faire réapparaître une nouvelle table quand toutes les cibles ont étés touchés. On va aussi pouvoir implémenter un timer pour déterminer la durée du jeu. Le jeu s'arrêtera à la fin du temps imparti et on pourra effectuer un calcul des scores.

✅ Bug qui fait que toutes les cibles ne sont pas comptabilisées

1. Avant de faire apparaître une table, régler un bug sur le comptage des cibles touchées
    1. Si les cibles tombent seules (sans la balle ou indirectement parce qu'un autre cylindre les as touché) alors elles ne seront pas comptabilisées
    2. Il faut trouver une manière de savoir si les cibles sont tombées
    3. Une approche consiste à récupérer la position en Y quand la cible apparaît
    4. Et de constamment vérifier si elle descend sous un certain seuil
    5. On peut commencer par créer une variable float pour stocker la position Y et l'assigner dans le Start (dans le script Cible Individuelle)
    6. 

    7. Maintenant que c'est stocké, on utilise la fonction Update pour vérifier constamment

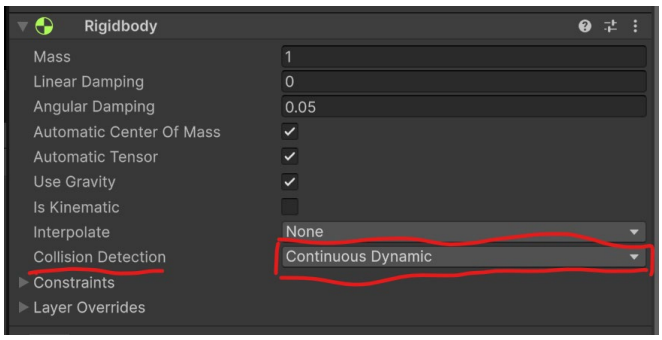
8.
```
private void Update()
{
    if (transform.position.y < initialY - 0.2f)
    {

    }
}
```
9. La valeur 0.2 peut être ajusté en fonction de votre scène
10. Si cette condition est vraie, alors on exécute le code pour valider la cible (exactement comme on fait dans OnCollisionEnter)
11. N'oubliez pas de rajouter une condition dans le update pour que ça se lance qu'une seule fois. Normalement dans le précédent PDF vous avez mis en place une variable bool pour éviter un comptage multiple de points avec une seule cible.

✅ Faire apparaître une nouvelle table

2. Faire apparaître une nouvelle table quand toutes les cibles sont tombées
    1. On va pouvoir appeler une fonction de Object Spawner pour recréer une nouvelle table et détruire celle-ci
    2. La fonction est la suivante :
    3.
```
ObjectSpawner.Instance.TrySpawnObject(transform.position, spawnNormal: transform.up);
```
    4. Mais avant de l'appeler, assurez vous d'avoir réactiver le spawn, puis de le désactiver après l'appel de la fonction du dessus
    5.
```
ObjectSpawner.Instance.enableSpawn = true;
```
    6. Enfin, on va détruire notre table actuelle avec le Destroy
    7.
```
Destroy(gameObject);
```
    8. Bonus : détruire la table avec un léger délai, pour laisser le temps au joueur de voir la dernière cible tomber
    9. Utilisez la fonction Invoke(« NomFonctionALancer », délai) pour exécuter avec un léger délai la commande

✅ Réglage du rigid body pour une meilleure précision

3. Régler le rigid body de la balle pour qu'il soit plus précis
    1. Il existe un mode pour rendre les collisions de la balle plus précise et éviter qu'elle ne traverse trop facilement les objets en allant vite
2. 

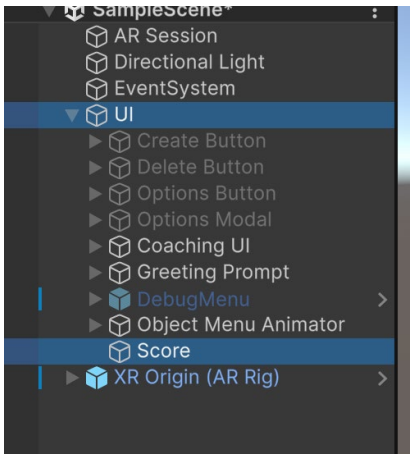
✅ Afficher le score de partie

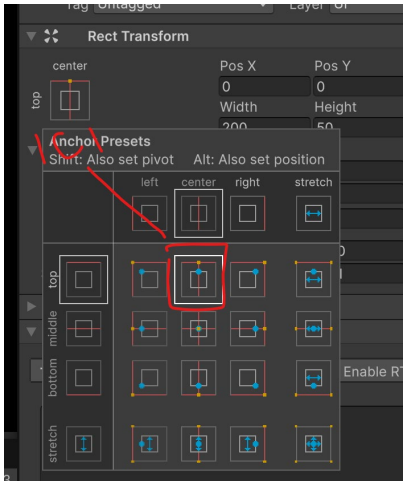
Pour ceux qui aurait utilisé UI Toolkit (ceux qui utilise Canvas passez cette étape), voici le code qui permet de modifier un texte

```
public UIDocument uiDocument;
```

```
uiDocument.rootVisualElement.Q<Label>(name:"NomDuTexte").text = "Score : " + variableDeScore;
```

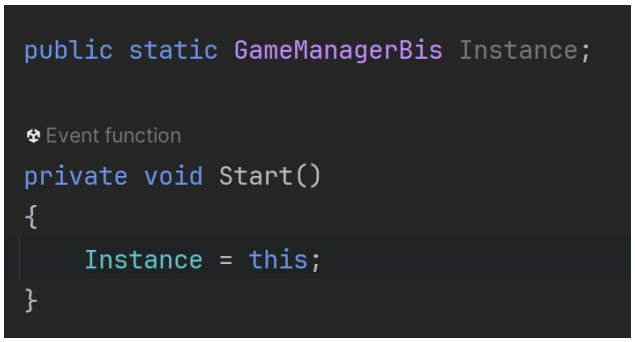
N'oubliez pas de renommer votre texte pour lui donner un nom de classe

4. Afficher un score de partie (Avec un Canvas)
    1. En utilisant le système de Canvas, vous pouvez créer un TextMeshProUGUI ancré au centre haut de l'écran
    2. N'hésitez pas à utiliser le canvas déjà présent dans la scène (qui s'appelle « UI »)
3. 

4. Pour créer un texte, faites un clic droit sur l'objet UI dans la hiérarchie, puis UI > Text – TextMeshPro
5. Ancrez le au centre haut de l'écran, en utilisant le menu d'ancrage sur l'objet Texte (maintenir Alt ou Option et cliquez sur le centre haut)
6. 

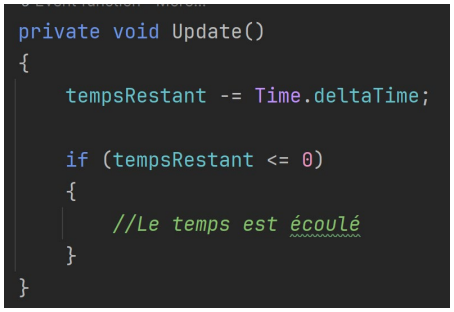
7. Votre texte est maintenant créée, il va falloir faire un script qui le modifie pour afficher le score

5. Créer un script qui calcul le score du jeu ou utiliser un script existant
    1. Ajouter votre script à un objet vide (si c'est un nouveau script)
    2. Créer une variable public de type TMP_Text (using nécessaire)
    3. ```
       using TMPro;
       ```
    4. ```
       public TMP_Text scoreTexte;
       ```
    5. Assignez le texte dans l'inspector
    6. Pour modifier le texte, vous pouvez faire comme ça (exemple)
    7. ```
       scoreTexte.text = "Score : " + valeurDeScore;
       ```
    8. L'idée maintenant, c'est de calculer un score dans votre script, et de l'actualiser a chaque fois que le joueur gagne des points.
    9. Vous pouvez utiliser le singleton pour rendre ce script global et l'appeler facilement de n'importe où
       ```

10. 
```
public static GameManagerBis Instance;

Event function
private void Start()
{
    Instance = this;
}
```
11. Le nom de ma classe est GameManagerBis (donc la variable doit partager le même type)

✅ Afficher un timer pour le temps de partie

6. Affichez un Timer et terminer la partie au bout d'un moment
    1. Faites pareil que l'étape du dessus pour créer un nouveau texte qui affichera le temps restant dans la partie
    2. Vous pouvez contrôler la valeur de temps dans le script GameManager
    3. Faites une nouvelle variable float et donnez-lui un temps en seconde
    4. Une fois que le joueur à placé les cibles, commencez à réduire le temps progressivement :
    5. 
```
private void Update()
{
    tempsRestant -= Time.deltaTime;

    if (tempsRestant <= 0)
    {
        //Le temps est écoulé
    }
}
```
    6. Si le temps est écoulé, on désactive le lancement de balles
    7. On peut soit afficher un bouton « Recommencer » et recharger la scène
    8. Soit supprimer les cibles et autoriser le joueur a en placer une nouvelle (si vous choisissez cette solution n'oubliez pas de remettre le tempsRestant à sa valeur initiale)
    9. N'oubliez pas d'afficher la valeur du tempsRestant sur le texte
7. Bonus pour les motivés :
    1. Arrondir le temps restant pour éviter d'afficher les décimales