📘 Tutoriel AR Foundation avec Unity : Détection de Plans, Reconnaissance d'Images et Utilisation du Gyroscope

🧰 Prérequis : Configuration du Projet Unity avec AR Foundation

1. Créer un Nouveau Projet Unity

- Utilisez Unity Hub pour créer un projet en 3D (URP ou 3D classique).

2. Installer les Packages Nécessaires

- Ouvrez le **Package Manager** (`Window > Package Manager`).
- Ajoutez les packages suivants :
    - **AR Foundation**
    - **ARKit XR Plugin** (pour iOS)
    - **ARCore XR Plugin** (pour Android)

3. Configurer le Projet pour AR

- Allez dans `Edit > Project Settings > XR Plug-in Management` et activez :
    - **ARKit** pour iOS
    - **ARCore** pour Android
- Dans `Player Settings`, sous **Other Settings** :
    - Définissez le **Minimum API Level** sur **Android 7.0 (API Level 24)** pour Android.

- Ajoutez une description pour l'utilisation de la caméra dans **Camera Usage Description** pour iOS.immersive insiders

---

📐 Partie 1 : Détection de Plans (Plane Detection)

Objectif

Permettre aux étudiants de détecter des surfaces planes dans l'environnement réel et d'y placer des objets virtuels.

Étapes

1. **Configurer la Scène**
    - Ajoutez un **AR Session** et un **AR Session Origin** à la scène.
    - Sous **AR Session Origin**, ajoutez une **AR Camera**.
2. **Ajouter les Composants AR**
    - Ajoutez les composants suivants à **AR Session Origin** :
        - **AR Plane Manager**
        - **AR Raycast Manager**
    - Assurez-vous que **AR Plane Manager** a un prefab de plan assigné pour visualiser les surfaces détectées.
3. **Placer des Objets sur les Plans Détectés**

Créez un script pour instancier un objet lorsque l'utilisateur touche l'écran :

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using System.Collections.Generic;

public class PlaceOnPlane : MonoBehaviour
{
    public GameObject objectToPlace;
    private ARRaycastManager _raycastManager;
    private List<ARRaycastHit> _hits = new List<ARRaycastHit>();

    void Start()
    {
        _raycastManager = GetComponent<ARRaycastManager>();
    }

    void Update()
    {
        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);
            if (_raycastManager.Raycast(touch.position, _hits,
TrackableType.PlaneWithinPolygon))
            {
                Pose hitPose = _hits[0].pose;
                Instantiate(objectToPlace, hitPose.position,
hitPose.rotation);
            }
        }
    }
}
```

-

Ressources Supplémentaires

- Documentation Unity sur AR Plane Manager
- Tutoriel vidéo sur la détection de plans

Image Source: Capture d'écran du tutoriel vidéo sur la détection de plans

---

🖼️ Partie 2 : Reconnaissance d'Images (Image Tracking)

Objectif

Permettre aux étudiants de reconnaître des images spécifiques dans l'environnement réel et d'y associer des objets virtuels.

Étapes

1. **Préparer les Images de Référence**
    - Créez une **Reference Image Library** :
        - Dans le dossier `Assets`, faites un clic droit > `Create > XR > Reference Image Library`.
        - Ajoutez les images que vous souhaitez reconnaître.
2. **Configurer le AR Tracked Image Manager**
    - Ajoutez le composant **AR Tracked Image Manager** à **AR Session Origin**.
    - Assignez la **Reference Image Library** créée précédemment.
    - Définissez un prefab à instancier lors de la détection d'une image.
3. **Gérer la Détection d'Images**

Créez un script pour gérer les événements de détection :

```
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using System.Collections.Generic;

public class ImageTracking : MonoBehaviour
{
    private ARTrackedImageManager _trackedImageManager;

    void Awake()
    {
        _trackedImageManager = GetComponent<ARTrackedImageManager>();
    }

    void OnEnable()
    {
        _trackedImageManager.trackedImagesChanged +=
OnTrackedImagesChanged;
    }

    void OnDisable()
    {
        _trackedImageManager.trackedImagesChanged -=
OnTrackedImagesChanged;
    }

    void OnTrackedImagesChanged(ARTrackedImagesChangedEventArgs
eventArgs)
    {
        foreach (ARTrackedImage trackedImage in eventArgs.added)
        {
            // Instancier un objet ou activer un prefab associé
        }
    }
}
```

🎯 Partie 3 : FPS / Asteroid Shooter avec Gyroscope

Objectif

Créer un jeu de tir en réalité augmentée où le joueur contrôle la visée en déplaçant son téléphone, grâce au gyroscope intégré.

Étapes

1. **Activer le gyroscope**
    - Dans Unity, le gyroscope est accessible via `Input.gyro`.

Activez-le au démarrage de votre script :

```
void Start()
{
    Input.gyro.enabled = true;
}
```

    -
2. **Contrôler la caméra avec le gyroscope**

Utilisez les données du gyroscope pour orienter la caméra :

```
void Update()
{
    Quaternion deviceRotation = Input.gyro.attitude;
    deviceRotation = Quaternion.Euler(90f, 0f, 0f) * new
Quaternion(-deviceRotation.x, -deviceRotation.y, deviceRotation.z,
deviceRotation.w);
    transform.localRotation = deviceRotation;
}
```

Ce code ajuste l'orientation pour correspondre au référentiel de Unity.

3. **Créer le système de tir**

Ajoutez un script pour tirer des projectiles :

```
public GameObject projectilePrefab;
public float shootingForce = 500f;

void Update()
{
    if (Input.touchCount > 0 && Input.GetTouch(0).phase ==
TouchPhase.Began)
    {
        GameObject projectile = Instantiate(projectilePrefab,
transform.position, transform.rotation);
        Rigidbody rb = projectile.GetComponent<Rigidbody>();
        rb.AddForce(transform.forward * shootingForce);
    }
}
```

Ce script instancie un projectile et lui applique une force vers l'avant.

4. **Gérer les ennemis (astéroïdes)**

    - Créez un prefab d'astéroïde avec un Rigidbody et un Collider.

Ajoutez un script pour les faire apparaître et se diriger vers le joueur :

```
public GameObject asteroidPrefab;
public float spawnInterval = 2f;
public float spawnDistance = 10f;

void Start()
{
    InvokeRepeating("SpawnAsteroid", 2f, spawnInterval);
}

void SpawnAsteroid()
{
    Vector3 spawnDirection = Random.onUnitSphere;
    spawnDirection.y = Mathf.Clamp(spawnDirection.y, -0.5f, 0.5f);
    Vector3 spawnPosition = transform.position + spawnDirection *
spawnDistance;
    GameObject asteroid = Instantiate(asteroidPrefab, spawnPosition,
Quaternion.identity);
    Rigidbody rb = asteroid.GetComponent<Rigidbody>();
    rb.velocity = (transform.position - spawnPosition).normalized * 2f;
}
```

Ce script génère des astéroïdes autour du joueur qui se dirigent vers lui.

5. **Détection des collisions**

Ajoutez un script aux projectiles pour détecter les collisions avec les astéroïdes :

```
void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.CompareTag("Asteroid"))
    {
        Destroy(collision.gameObject);
        Destroy(gameObject);
    }
}
```

Assurez-vous que les astéroïdes ont le tag "Asteroid".

Ressources supplémentaires

- Documentation Unity sur l'utilisation du gyroscope
- Tutoriel vidéo sur la création d'un FPS en AR avec Unity