

TP - Réalité Augmentée avec Vuforia : Ajout de Son et Vidéo

Introduction

Dans ce TP, vous allez enrichir votre application de réalité augmentée Vuforia existante. Vous avez déjà une **Image Target** qui affiche un **modèle 3D** quand la cible est détectée. Maintenant, vous allez ajouter :

- Un **son** qui joue quand on appuie sur un bouton
 - Une **vidéo** qui s'affiche quand on appuie sur un autre bouton
 - Une **interface utilisateur (UI)** avec des boutons
-

Partie 1 : Comprendre l'Architecture

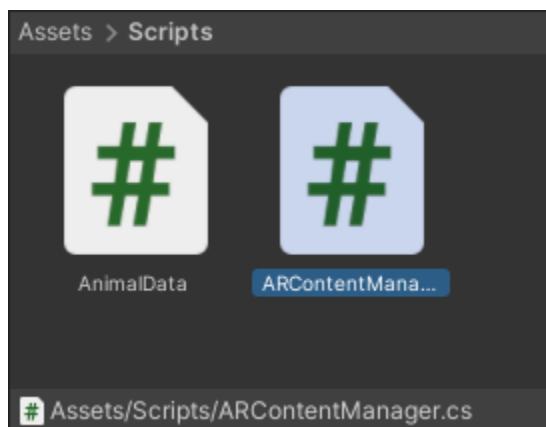
Les Deux Scripts Principaux

Votre application utilise **deux scripts C#** qui travaillent ensemble :

1. **AnimalData.cs** → Attaché à chaque modèle 3D (l'animal)
2. **ARContentManager.cs** → Attaché à un GameObject vide dans la scène (gère l'UI)

Comment ça fonctionne ?

Image Target détectée → AnimalData → ARContentManager → Affiche les boutons
(sur l'animal) (gestionnaire)



Partie 2 : Le Script AnimalData.cs

⌚ Rôle

Ce script est attaché à **chaque modèle 3D** et contient les médias spécifiques à cet animal.

📝 Explication Ligne par Ligne

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Video;
using Vuforia;
// using are a way to include namespaces that contain useful classes and functions.

public AudioClip soundClip;
public VideoClip videoClip;
```

Ce que ça fait : Deux variables publiques pour stocker le son et la vidéo de cet animal. Vous les assignerez dans l'Inspector Unity.

```
private ARContentManager manager;
```

Ce que ça fait : Une référence privée vers le gestionnaire principal. Comme une “adresse” pour communiquer avec l'autre script.

```
manager = FindObjectOfType<ARContentManager>();
```

Ce que ça fait : Au démarrage, trouve automatiquement le gestionnaire dans la scène.

```
public void TargetFound()
{
    if (manager != null)
    {
        manager.OnTargetFound(soundClip, videoClip);
    }
}
```

Ce que ça fait : Quand Vuforia détecte l'image, cette fonction envoie le son et la vidéo au gestionnaire.

```
public void TargetLost()
{
    if (manager != null)
    {
        manager.OnTargetLost();
    }
}
```

Ce que ça fait : Quand l'image disparaît, cette fonction cache les boutons et arrête les médias.

Partie 3 : Le Script ARContentManager.cs

⌚ Rôle

Ce script gère l'interface utilisateur (boutons) et les lecteurs de médias (son et vidéo).

📝 Explication des Variables

```
public GameObject buttonsCanvas;  
public GameObject vidHolder;
```

Ce que ça fait : Références vers les objets UI : - buttonsCanvas : Le Canvas contenant les boutons Son et Vidéo - vidHolder : Le panneau (Panel) qui affiche la vidéo

```
private VideoPlayer videoPlayer;  
private AudioSource audioSource;  
private AudioClip currentSound;
```

Ce que ça fait : Composants pour lire la vidéo et le son. currentSound stocke le son de l'animal actuel.

📝 Explication des Fonctions Principales

OnTargetFound (Cible Détectée)

```
public void OnTargetFound(AudioClip sound, VideoClip video)  
{  
    buttonsCanvas.SetActive(true); // Affiche Les boutons  
    currentSound = sound; // Stocke Le son  
    videoPlayer.clip = video; // Prépare La vidéo  
    // Arrête tout média en cours  
    videoPlayer.Stop();  
    vidHolder.SetActive(false);  
    audioSource.Stop();  
}
```

OnTargetLost (Cible Perdue)

```
public void OnTargetLost()  
{  
    buttonsCanvas.SetActive(false); // Cache Les boutons  
    vidHolder.SetActive(false); // Cache La vidéo  
    audioSource.Stop(); // Arrête Le son  
    videoPlayer.Stop(); // Arrête La vidéo  
}
```

```
ToggleSound (Bouton Son)
public void ToggleSound()
{
    if (audioSource.isPlaying && audioSource.clip == currentSound)
    {
        audioSource.Stop(); // Si le son joue → l'arrêter
    }
    else
    {
        audioSource.clip = currentSound;
        audioSource.loop = true; // Répéter en boucle
        audioSource.Play(); // Jouer le son
    }
}
```

```
ToggleVideo (Bouton Vidéo)
public void ToggleVideo()
{
    if (videoPlayer.isPlaying)
    {
        videoPlayer.Stop(); // Si vidéo joue → l'arrêter
        vidHolder.SetActive(false); // Cacher le panneau
    }
    else
    {
        vidHolder.SetActive(true); // Afficher le panneau
        videoPlayer.Play(); // Jouer la vidéo
    }
}
```

Partie 4 : Implémentation Étape par Étape

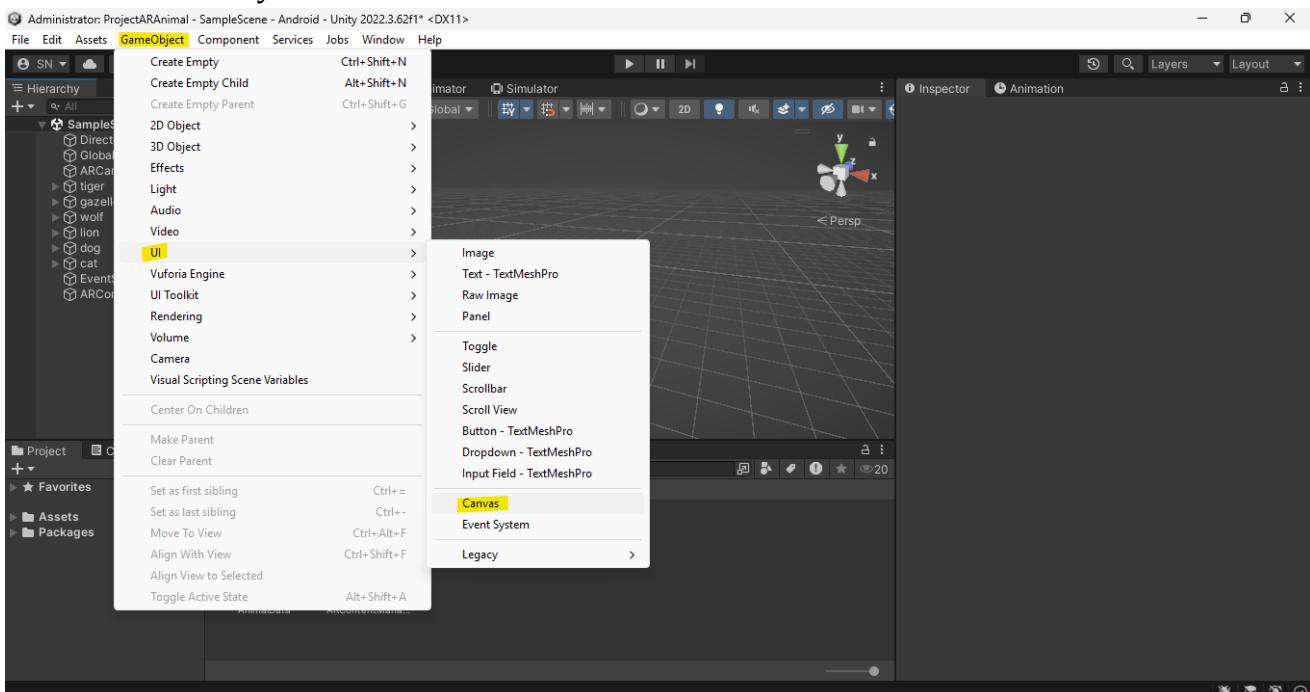
ÉTAPE 1 : Créer le Gestionnaire (ARContentManager)

1. Clic droit dans la Hierarchy → Create Empty // Ctrl+Shift+N
 2. Renommez-le “**ARContentManager**”
 3. Cliquez dessus → Inspector → Add Component
 4. Cherchez “**Audio Source**” → Ajoutez-le
 5. Add Component → New Script → Nommez-le “**ARContentManager**” // Drag & Drop It if already exists
 6. Double-cliquez sur le script → Copiez-collez le code fourni
-

ÉTAPE 2 : Créer l’Interface Utilisateur

A. Crée le Canvas

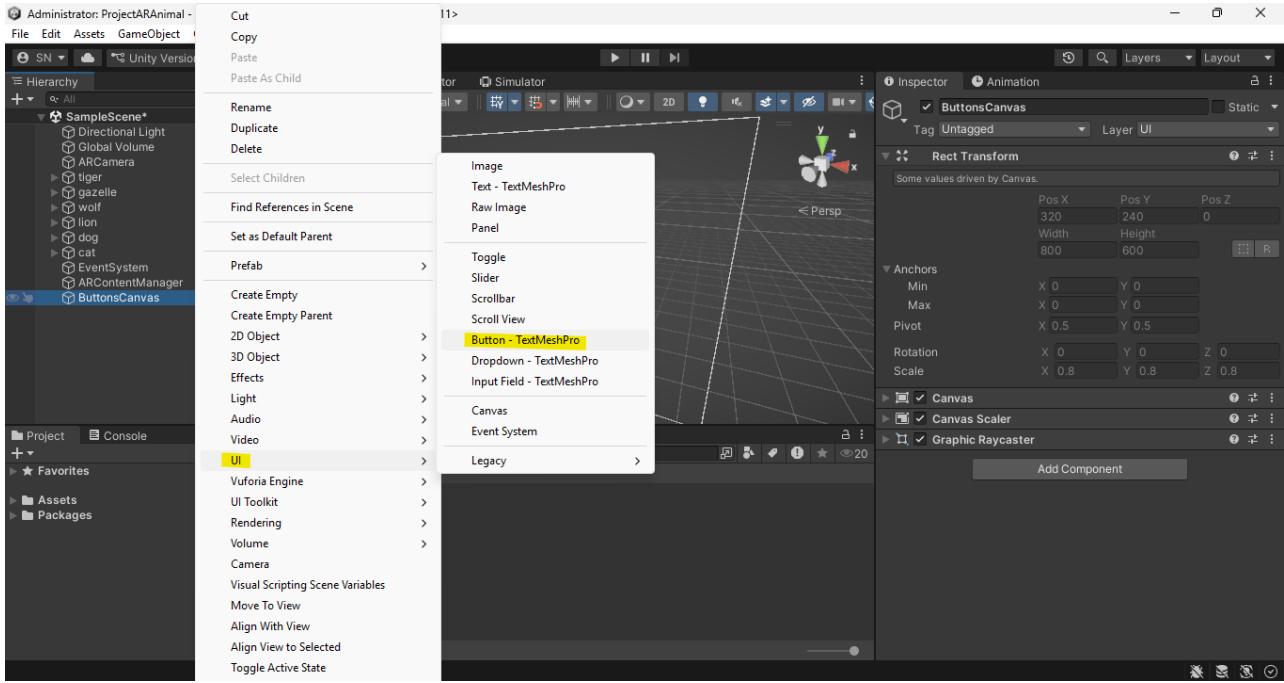
1. Clic droit Hierarchy → UI → **Canvas**



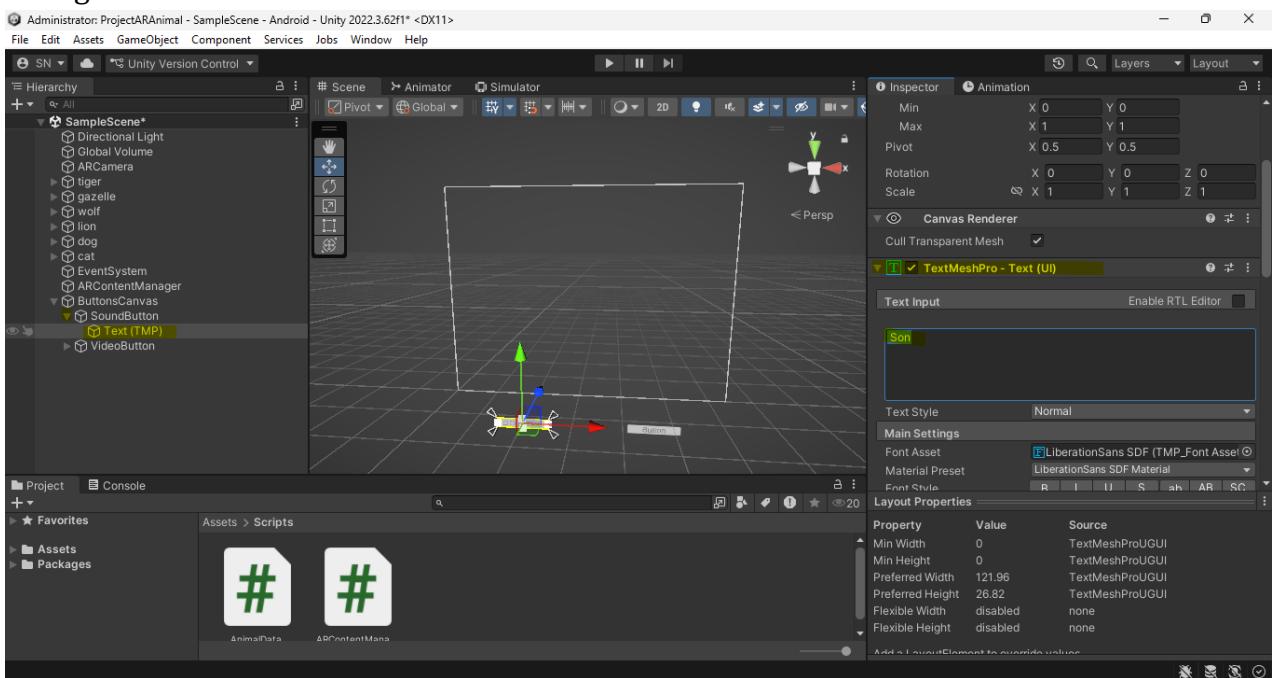
- 2.
3. Renommez-le “**ButtonsCanvas**”
4. Dans l’Inspector :
 - **Render Mode** : Screen Space - Overlay
 - **Canvas Scaler** → UI Scale Mode : Scale With Screen Size

B. Crée les Boutons

1. Clic droit sur ButtonsCanvas → UI → Button - TextMeshPro



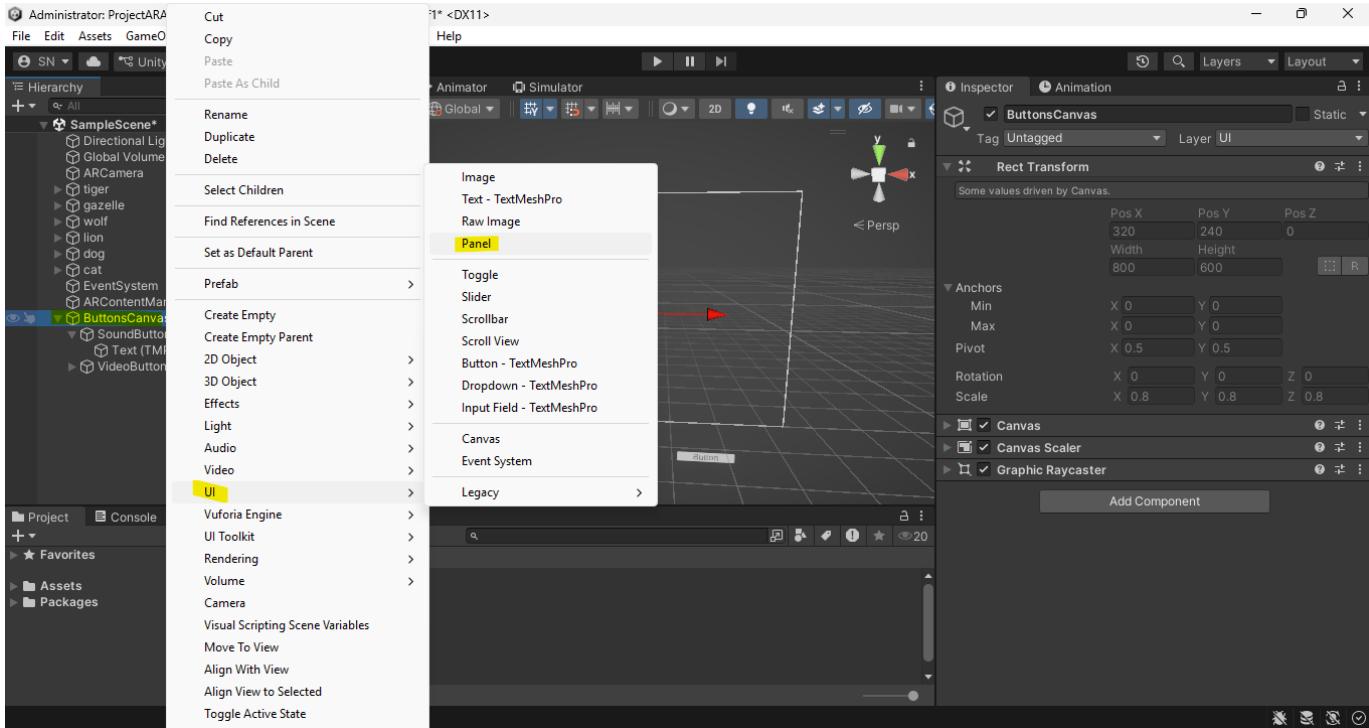
- Nommez-le “**SoundButton**”
- Positionez-le
- Changez le texte en “ **Son**”



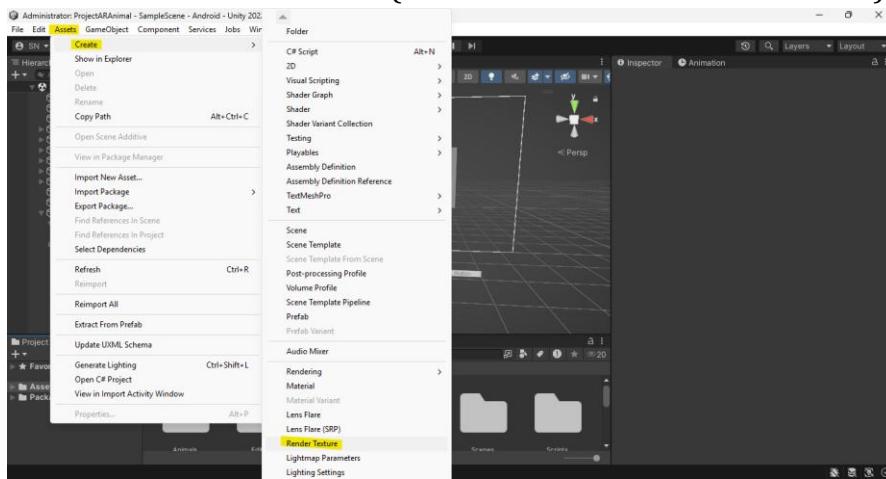
2. Clic droit sur ButtonsCanvas → UI → Button - TextMeshPro

- Nommez-le “**VideoButton**”
- Positionez-le
- Changez le texte en “ **Vidéo**”

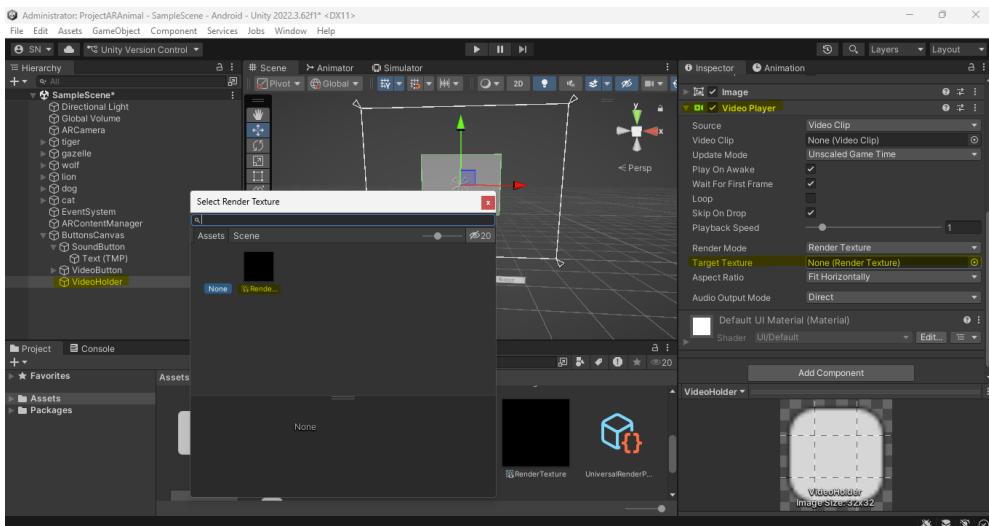
C. Créer le Panneau Vidéo



1. Clic droit sur ButtonsCanvas → UI → Panel
2. Renommez-le “VideoHolder”
3. Ajustez sa taille
4. Cliquez dessus → Add Component → Video Player
5. Dans Video Player :
 - Render Mode : Render Texture
 - Créez une Render Texture (Assets → Create → Render Texture)

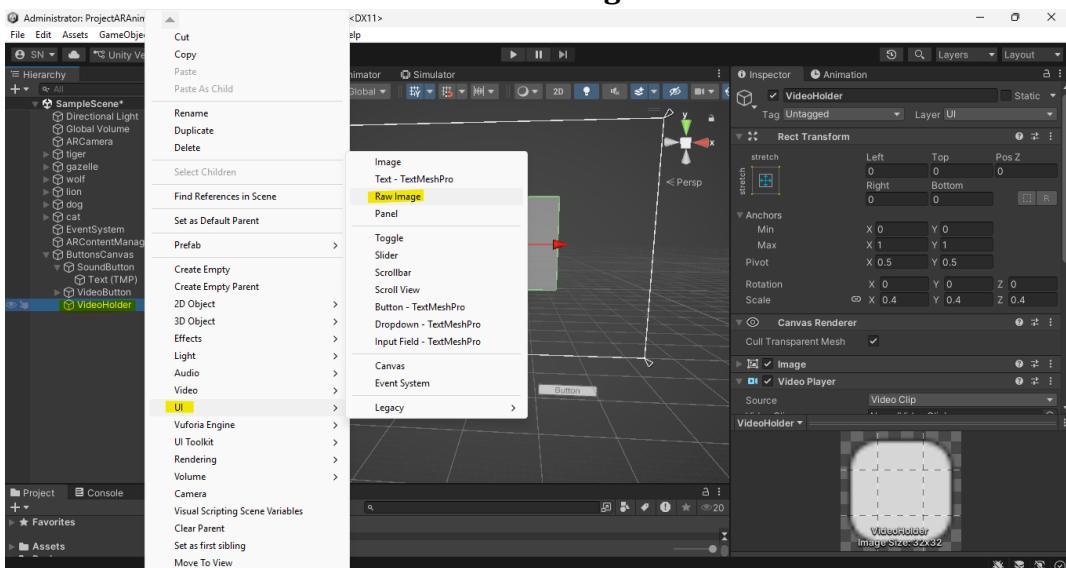


- Assignez cette Render Texture

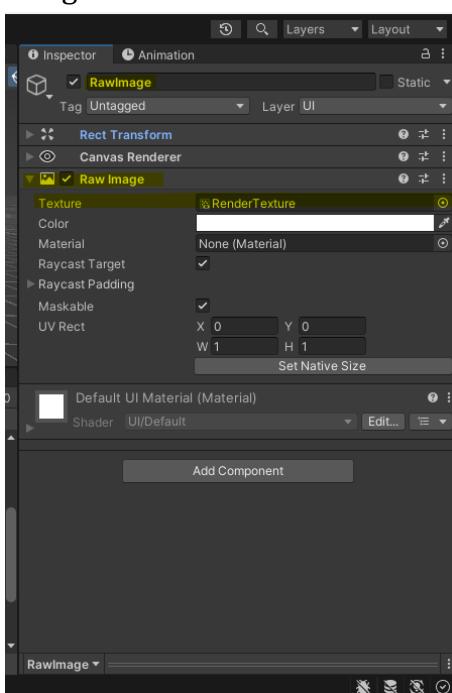


6. Ajoutez un Raw Image au panneau :

- Clic droit sur VideoHolder → UI → Raw Image

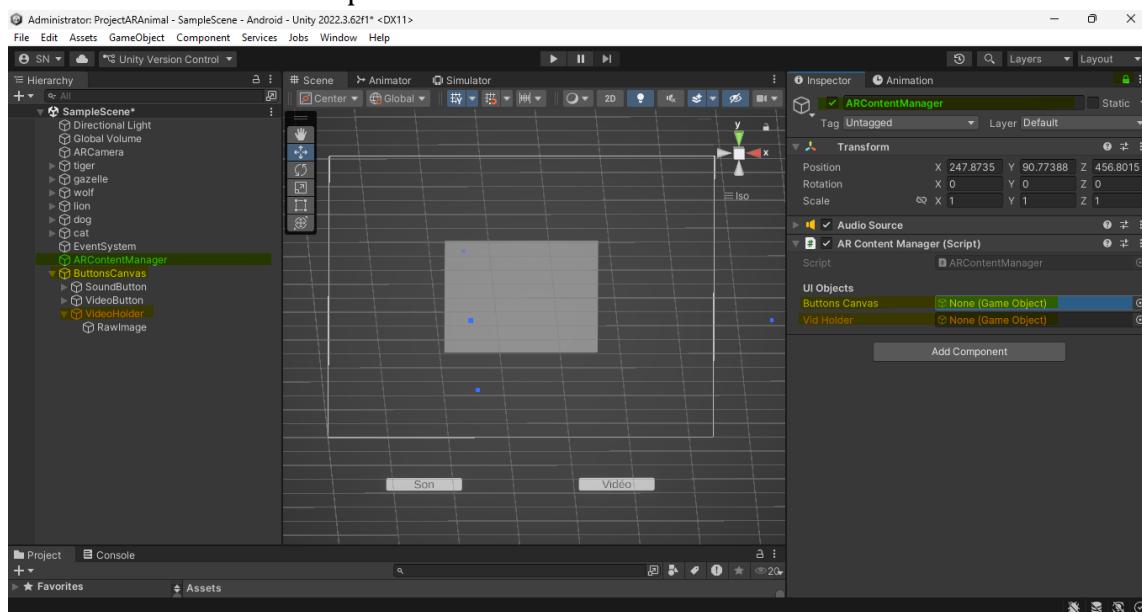


- Assignez la même Render Texture dans Texture



ÉTAPE 3 : Connecter ARContentManager

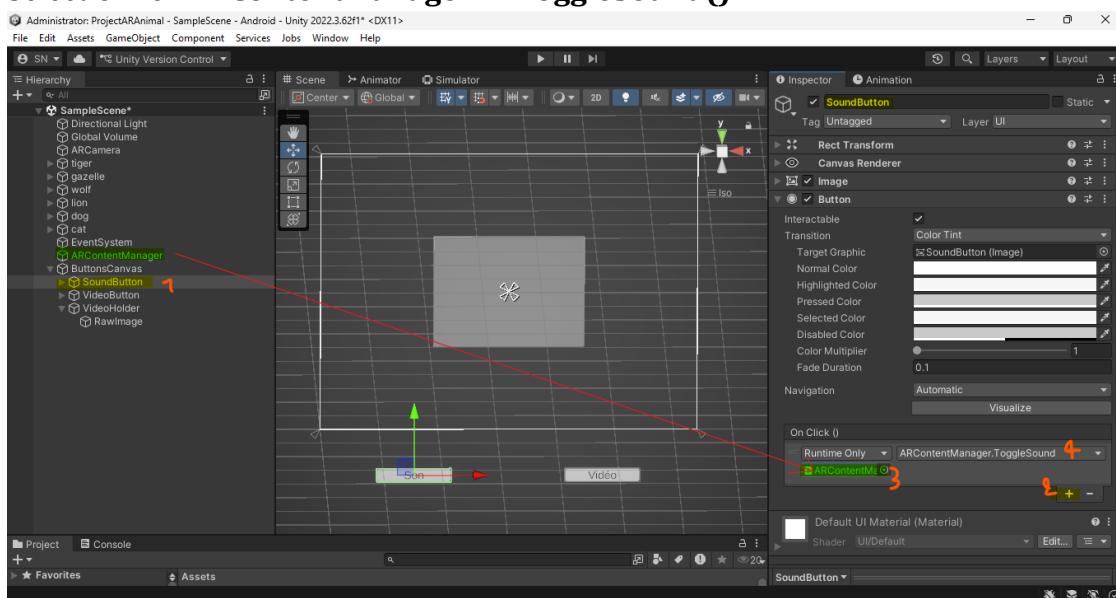
1. Sélectionnez **ARContentManager** dans la Hierarchy
2. Dans l'Inspector, dans le script **ARContentManager** :
 - **Buttons Canvas** → Glissez-déposez **ButtonsCanvas**
 - **Vid Holder** → Glissez-déposez **VideoHolder**



ÉTAPE 4 : Configurer les Boutons

Bouton Son :

1. Sélectionnez **SoundButton**
2. Dans l'Inspector → **Button** component → **OnClick()**
3. Cliquez sur +
4. Glissez **ARContentManager** dans le champ
5. Sélectionnez **ARContentManager** → **ToggleSound()**

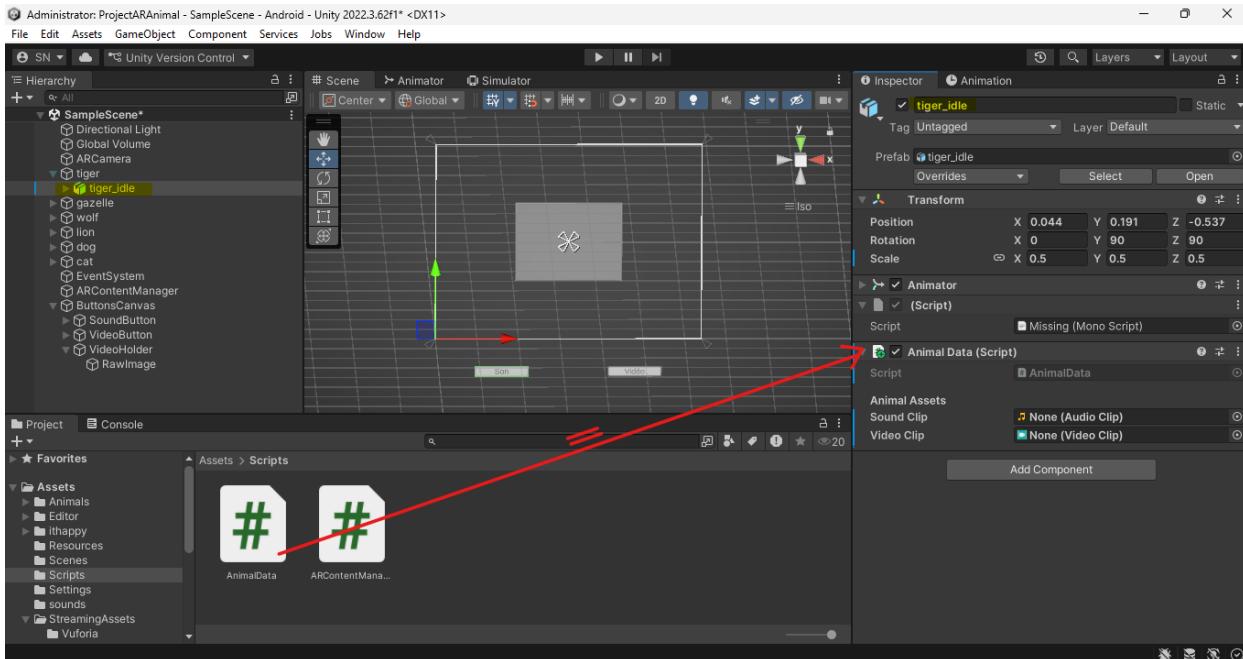


Bouton Vidéo :

1. Sélectionnez **VideoButton**
 2. Dans l'Inspector → **Button** component → **OnClick()**
 3. Cliquez sur +
 4. Glissez **ARContentManager** dans le champ
 5. Sélectionnez **ARContentManager** → **ToggleVideo()**
-

ÉTAPE 5 : Configurer les Modèles 3D (Animals)

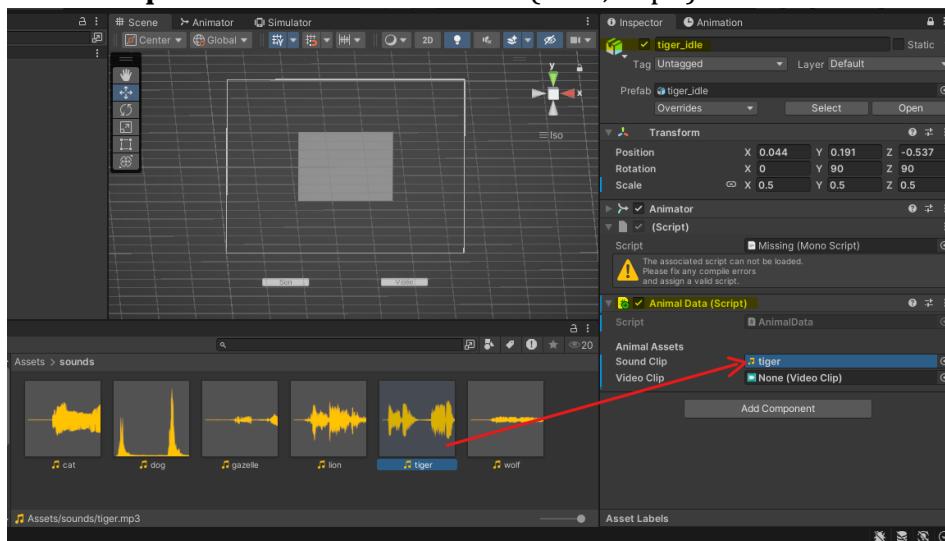
1. Sélectionnez votre **modèle 3D** (sous **ImageTarget**)
2. **Add Component** → **New Script** → Nommez-le “**AnimalData**” // Drag & Drop It if already exists



3. Double-cliquez → Copiez-collez le code fourni

4. Dans l'Inspector :

- **Sound Clip** → Glissez un fichier audio (.wav, .mp3)

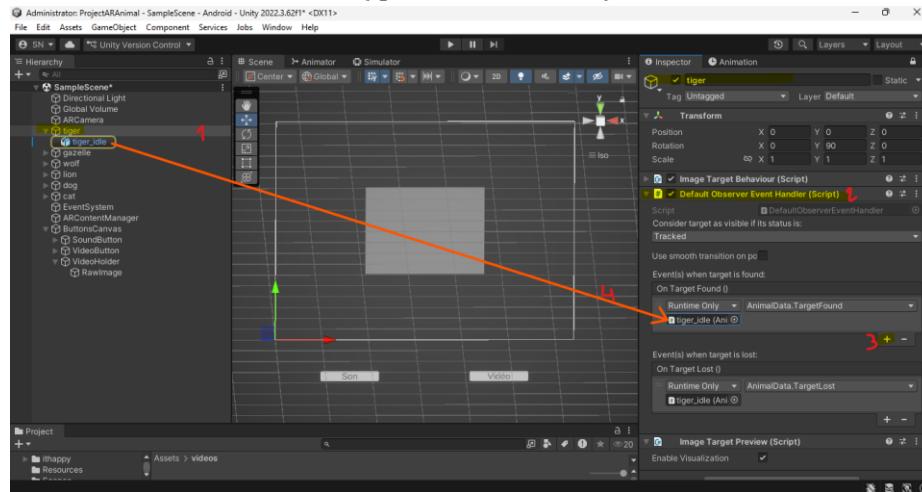


- **Video Clip** → Glissez un fichier vidéo (.mp4)
-

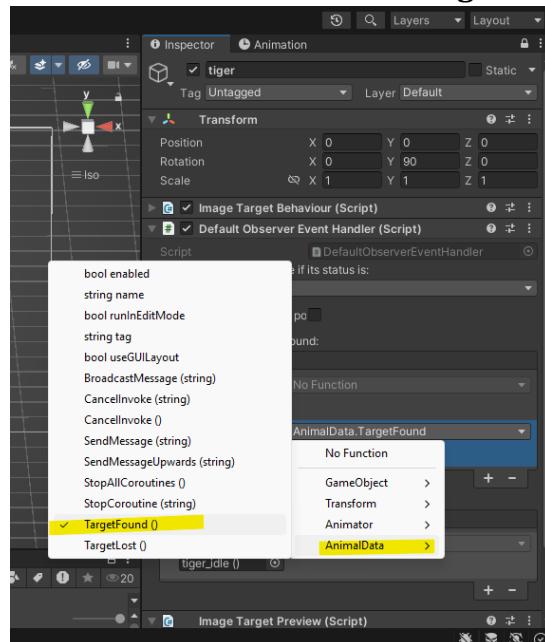
ÉTAPE 6 : Connecter Vuforia aux Scripts

1. Sélectionnez votre **ImageTarget**
2. Dans l'Inspector → **Image Target Behaviour**
3. Faites défiler jusqu'à **Advanced** → **Events**
4. Dans **On Target Found()** :

- Cliquez +
- Glissez votre **modèle 3D** (qui a **AnimalData**)

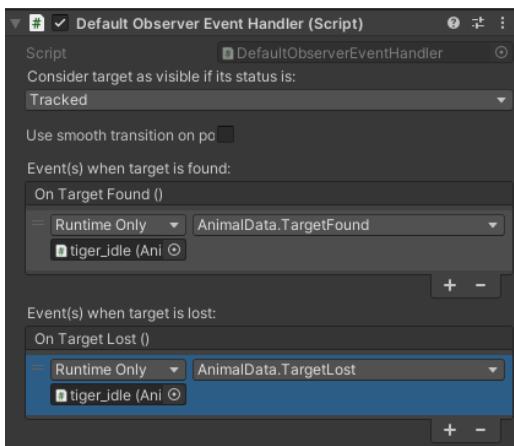


- Sélectionnez **AnimalData** → **TargetFound()**



5. Dans **On Target Lost()** :

- Cliquez +
- Glissez votre **modèle 3D**
- Sélectionnez **AnimalData** → **TargetLost()**



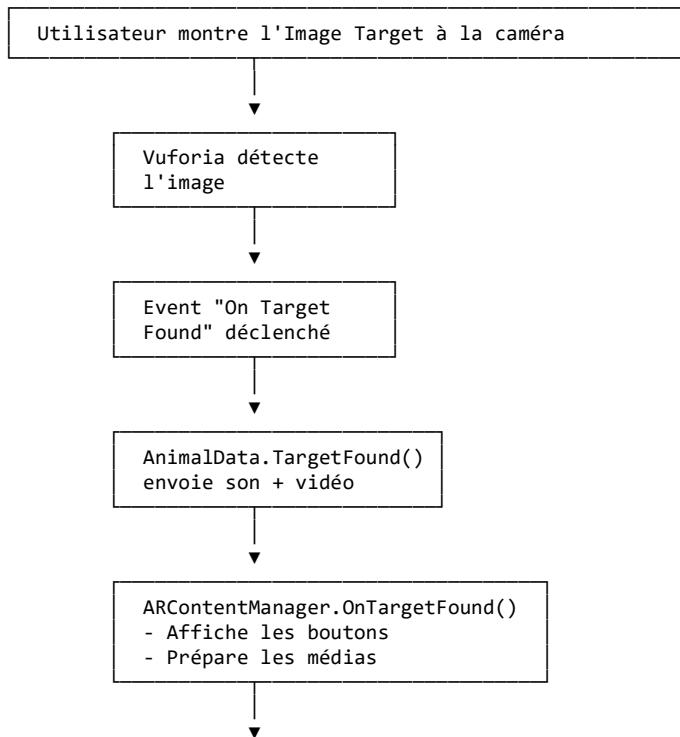
Partie 5 : Tester l'Application

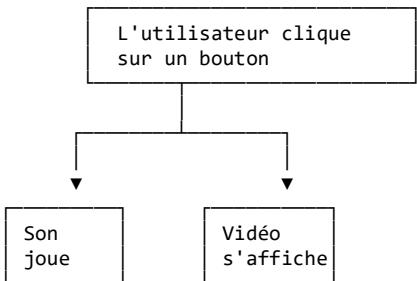
Procédure de Test

1. Play dans Unity
2. Montrez l'**Image Target** à la caméra
3. Vérifications :

- Le modèle 3D apparaît ?
- Les boutons apparaissent en bas de l'écran ?
- Le bouton Son joue/arrête le son ?
- Le bouton Vidéo affiche/cache la vidéo ?
- Tout s'arrête quand l'image disparaît ?

Partie 6 : Schéma de Flux





Bon courage ! 🚀