#### 

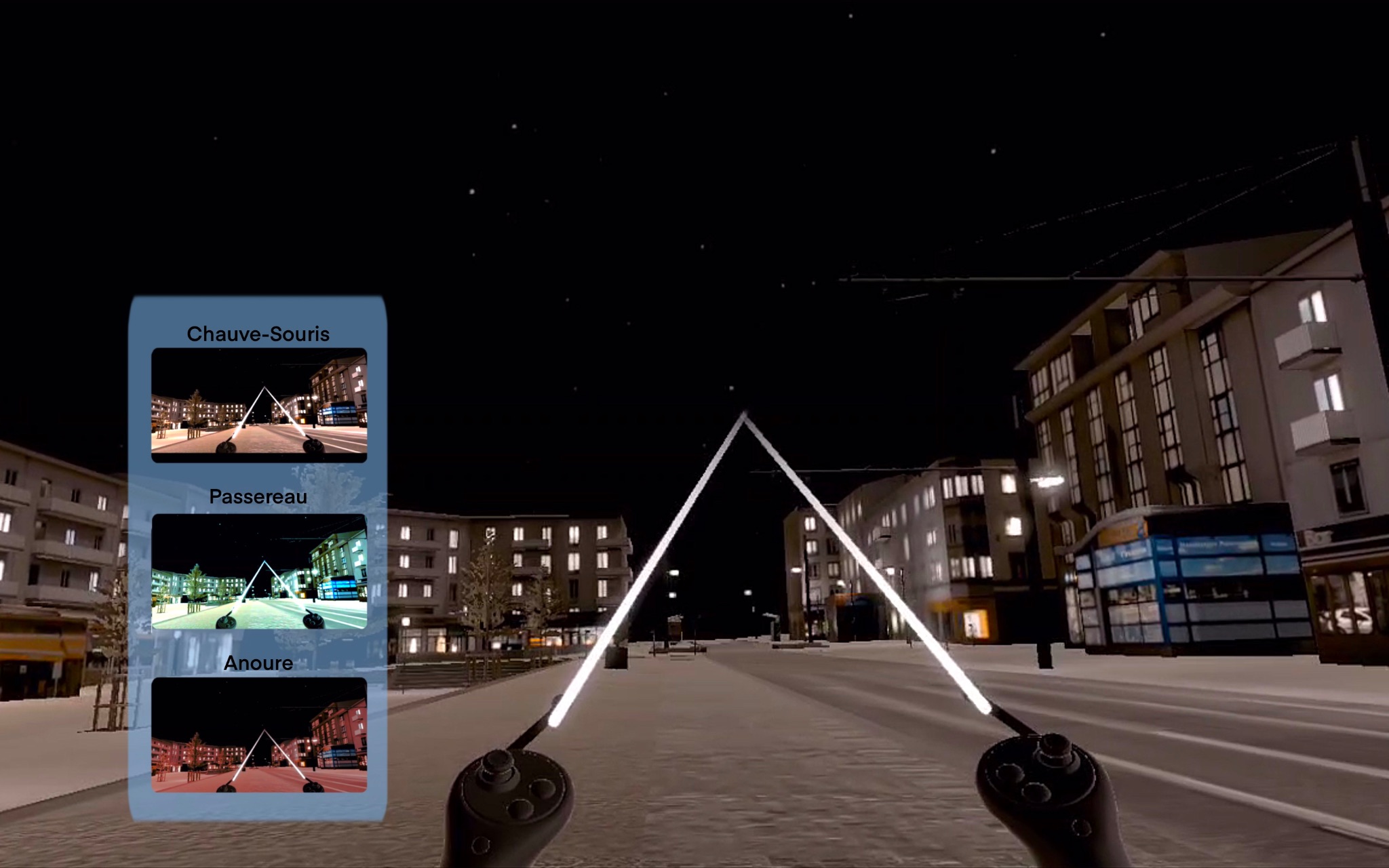
#### 

#### 

#### 

#### 

DOCUMENTATION TECHNIQUE



#### 

## **TABLE DES MATIÈRES**

#### 

#### 

[**TABLE DES MATIÈRES 2**](#_ffoacvq1hjrf)

[I. INTRODUCTION 3](#_4zvxon924dk9)

[II. ARCHITECTURE DU PROJET 3](#_7t2u1ljoes2i)

[III. DEMO SCENE 4](#_9vd7qislkcx4)

[IV. BUT DE LA SCÈNE 6](#_vichqhy77yu8)

[V. PROCESSUS ET COMPOSANTS CLÉS 7](#_lls1npd5nel0)

[1. Changer d'espèces 7](#_pgpvmmtti26j)

[2. Changer l’éclairage 7](#_pcg3t6t6fzen)

[Bake 7](#_9h763qdesxdd)

[C# Lightmap Switcher 8](#_iqbid4pa6jdl)

[C# Environment Switcher 9](#_dhilh4y7waoe)

[3. Déplacement / Rotation 9](#_55wz1qgkczmt)

[4. Interaction avec le menu 10](#_ia5klyxl6x84)

[5. Changement de langue 11](#_rjug4oitnlcp)

#### 

#### 

## **I. INTRODUCTION**

Le projet Sensilum a été initié en collaboration avec la Chaire NOZ Breizh, avec pour objectif principal de sensibiliser le public à l'impact de la pollution lumineuse sur la biodiversité. Ce projet utilise Unity pour créer une simulation immersive permettant d'explorer différentes scènes nocturnes et d'observer l'effet de l'éclairage sur diverses espèces.

Ce document vise à fournir une vue d'ensemble de l'architecture du projet Unity Sensilum. Il expliquera les principaux composants et fonctionnalités de la scène, ainsi que les processus et scripts utilisés pour les implémenter.

## **II. ARCHITECTURE DU PROJET**

Le projet Unity Sensilum est structuré de manière à faciliter la gestion des ressources et des scripts. Les dossiers principaux incluent :

* **Assets** : Contient les ressources 3D, les scripts, les scènes, les textures, les matériaux et les packages.
  + **3D** : Contient les *materials* et les *textures* de la scène (rue de siam).
  + **Plugins** : contient le fichier AndroidManifest.xml, essentiel pour la configuration et le bon fonctionnement de l’application Unity sur la plateforme Android.
  + **Prefabs** : Contient les *prefabs*, qui sont des modèles réutilisables d'objets de jeu. Il contient les *controllers* gauche et droit, les zones de téléportations, les questionnaires et les commandes pour pivoter et se téléporter.
  + **Resources** : Regroupe les ressources utilisées par les scripts pendant l'exécution. Il contient des images (les différentes espèces et éclairages pour l’UI), des vidéos (animations des *controllers*) ainsi que des *lightmaps.*
  + **Samples** : Exemples de configurations et d'utilisations de certains packages.
  + **Scenes** : Comprend les différentes scènes du projet comme les scènes *TutorialScene* et *ExperienceScene*.
  + **Scripts** : Organisation des scripts par fonctionnalité, comme les contrôleurs de l'utilisateur, la gestion de l'éclairage, et les interactions avec le menu.
    - **CSVReader** : permet de charger les données csv du dossier *StreamingAssets.*
    - **EnvironmentSwitcher** : permet de changer d'environnement lumineux et d'appliquer des post-traitements spécifiques en fonction des actions de l'utilisateur.
    - **LightmapSwitcher** : permet de gérer et de changer les lightmaps et les paramètres de lumière dans une scène Unity. Il comprend des fonctionnalités pour sauvegarder et charger les lightmaps, les light probes, et les paramètres de matériaux pour différentes configurations d'éclairage, et appliquer ces configurations au moment de l'exécution.
    - **LookAt** : permet à un objet (comme une UI) de suivre et de faire face à la caméra VR en utilisant un mouvement lisse.
    - **SpeciesFileReader** : permet de lire les données de spectre des animaux depuis un fichier CSV, génère et applique des textures 3D pour chaque espèce, et ajuste les paramètres de post-traitement basés sur ces données pour des effets visuels spécifiques.
    - **SpeciesMenu** : gère l'affichage d'un menu de sélection d'espèces animales, permettant de naviguer entre les animaux précédents, actuels et suivants en mettant à jour les textes correspondants.
    - **SpeciesSwitcher** : permet de changer les textures et les paramètres de post-traitement (bloom et ajustements de couleur) pour représenter différentes espèces animales, en fonction des entrées de l'utilisateur et en mettant à jour l'interface utilisateur en conséquence.
    - **Spectrum** : permet de gérer des spectres lumineux, permettant de manipuler et convertir des données de longueurs d'onde en valeurs de couleurs, avec des fonctionnalités de clonage, de ré-échantillonnage et d'opérations arithmétiques sur les spectres.
    - **UIController** : implémente un contrôleur UI, réagissant à des actions d'entrée pour afficher ou masquer un GameObject spécifié. Il utilise les événements d'InputSystem pour détecter les changements de périphérique, activant ou désactivant les actions d'entrée en conséquence.
    - **WavelengthColor** : permet de générer des couleurs RGB réalistes à partir de longueurs d'onde visibles.
  + **StreamingAssets** : Ressources regroupant les données de vision des animaux et de l’homme (configuration des ambiances et des espèces).

Le projet inclut une maquette détaillée de la rue de Siam, comprenant des éléments urbains tels que des bâtiments, des lampadaires, et des points de téléportation. La scène est construite autour de cette maquette, offrant un environnement réaliste pour l'expérience utilisateur.

## **III. DEMO SCENE**

Notre Demo Scene est la scène principale de notre application. Elle regroupe les fonctionnalités essentielles pour permettre une bonne démonstration devant le grand public.



* **Architecture de la scène** :

**WorldState**   
 └── Affichage de l'espèce (Image)   
 └── Affichage de l'éclairage (Texte)   
**Global Volume**   
**Lights**  
**MyReflectionProbe**   
**XR Interaction Setup**   
 └── XR InteractionManager  
 └── [...]  
 └── XR Origin  
 └── Offset  
 └── Camera  
 └── Left Controller   
 └── Right Controller  
 └── SelectMenu  
 └── Locomotion System  
**Siam\_Global**   
 └── Modèles 3D  
 └── Limites de téléportation  
**LightingSwitcher**   
 └── Canvas\_Animals\_Names  
 └── Species Switcher

* **Composants de la scène** :
  + WorldState :   
    Contient l'interface utilisateur qui affiche l'espèce actuellement sélectionnée (sous forme d'image) ainsi que les informations sur l'éclairage actuel (sous forme de texte).
  + Global Volume :   
    Gère les paramètres globaux de post-traitement pour la scène, tels que les effets visuels globaux, plus spécialement le bloom dans notre cas.
  + Lights :   
    Regroupe toutes les lumières, et leurs variantes en température, utilisées dans la scène.
  + MyReflectionProbe :   
    Utilisé pour capturer les réflexions dans la scène et améliorer le réalisme visuel en appliquant des réflexions cubemaps.
  + XR Interaction Setup :   
    Setup de base pour gérer les interactions VR dans la scène.
    - Left Controller :   
      définir les différents types d’intéraction (Poke/Direct/Ray/Teleport) possibles avec la manette gauche
    - Right Controller :  
      définir les différents types d’intéraction (Poke/Direct/Ray/Teleport) possibles avec la manette droite
  + Siam\_Global :   
    Maquette de la rue de Siam.
    - Limite de téléportation :   
      mesh retravaillé du sol de Siam pour définir uniquement les zones possibles où se déplacer
  + LightingSwitcher :   
    Élément qui regroupe les différents scripts qui gèrent le changement d’éclairage
    - Canvas\_Animals\_Env :   
      liste des espèces, utilisée par les scripts pour modifier l’état du monde.
    - Species Switcher :   
      Élément qui regroupe les différents scripts qui gèrent le changement d’espèce.

## **IV. BUT DE LA SCÈNE**

La scène vise à offrir une expérience réaliste et immersive permettant d'évaluer l'impact de l'éclairage public nocturne sur différentes espèces. Les utilisateurs peuvent interagir avec divers éléments pour observer les effets de la lumière sur la faune et le sentiment de sécurité la nuit.

Les utilisateurs peuvent :

* **Changer d’espèces** : Passer d'une perspective à l'autre entre un oiseau, une chauve-souris, une grenouille, et un humain pour voir comment chaque espèce est affectée par la lumière.
* **Modifier l’éclairage** : Changer la couleur, l’intensité, et le type d’éclairage pour observer les différentes conditions lumineuses.
* **Déplacement et rotation** : Utiliser des points de téléportation pour se déplacer dans la scène et explorer différents points d'intérêt.
* **Interaction avec le menu** : Utiliser des pop-ups et une interface utilisateur pour accéder à des informations supplémentaires et contrôler les paramètres de la scène.

## 

## **V. PROCESSUS ET COMPOSANTS CLÉS**

### **Changer d'espèces**

Les données des espèces sont extraites à partir des fichiers CSV dans le dossier *StreamingAssets*. Ces fichiers contiennent des paramètres prédéfinis pour l'effet de post-traitement "Bloom". Lorsqu'une nouvelle espèce est sélectionnée, les paramètres du Bloom, tels que l'intensité, la dispersion et la teinte, sont automatiquement mis à jour en fonction des valeurs spécifiques de l'espèce choisie. Cela permet de simuler efficacement les variations d'éclairage liées à chaque espèce dans l'environnement virtuel​.

### **Changer l’éclairage**

#### **Bake**

Le "baking" dans Unity est un processus de précalcul des informations d'éclairage pour optimiser les performances de la simulation. Il génère des lightmaps, des textures 2D stockant les informations d'éclairage pour les objets statiques de la scène. Ces lightmaps permettent d'obtenir des effets d'éclairage réalistes sans nécessiter de calculs en temps réel.

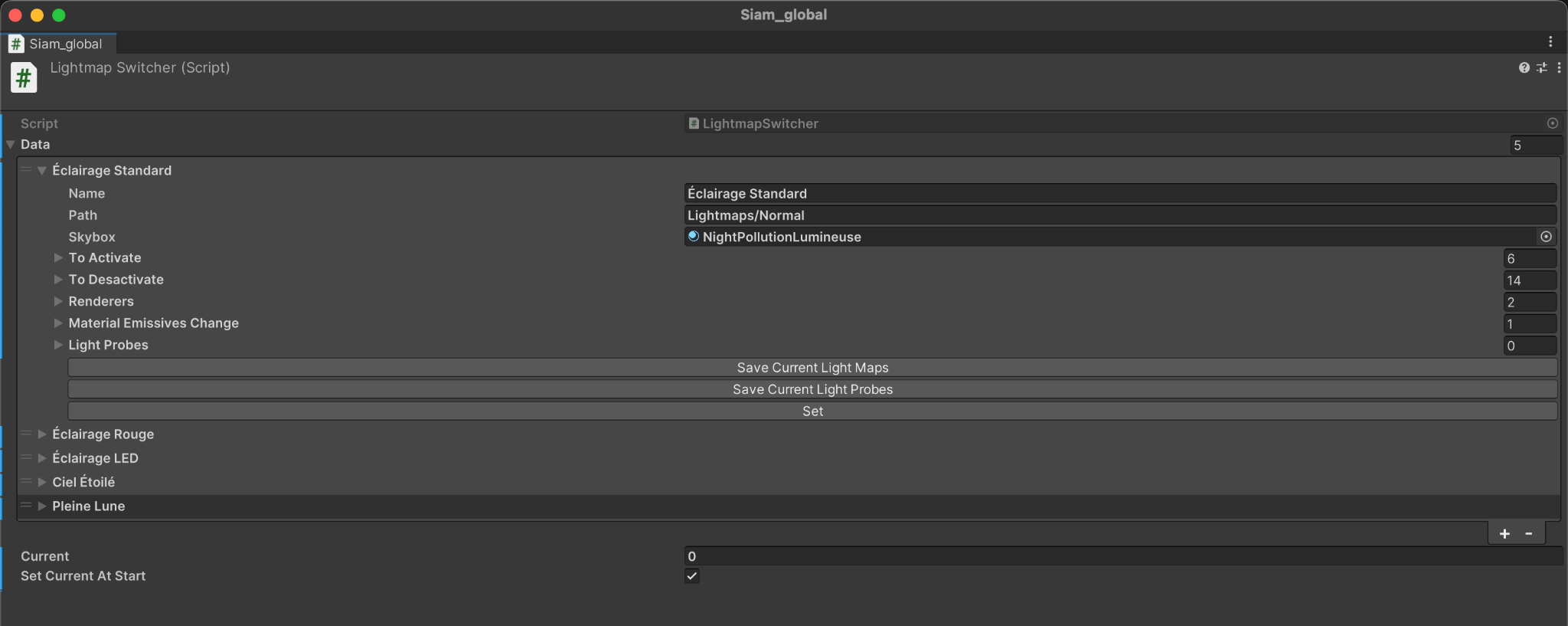
Dans Unity, les paramètres de baking se trouvent dans la fenêtre "Lighting". Vous devez configurer la résolution des lightmaps, le type de compression, et choisir la méthode de baking. Une fois les paramètres définis, lancez le processus de baking en cliquant sur "Generate Lighting". Unity va alors calculer les lightmaps et les probes de lumière et de réflexion, les stockant dans le "Lighting Data Asset".

Les lightmaps générées sont ensuite visibles dans l'onglet "Baked Lightmaps" et accessibles dans un dossier généré dans le même répertoire que votre scène.

#### 

#### **C# Lightmap Switcher**

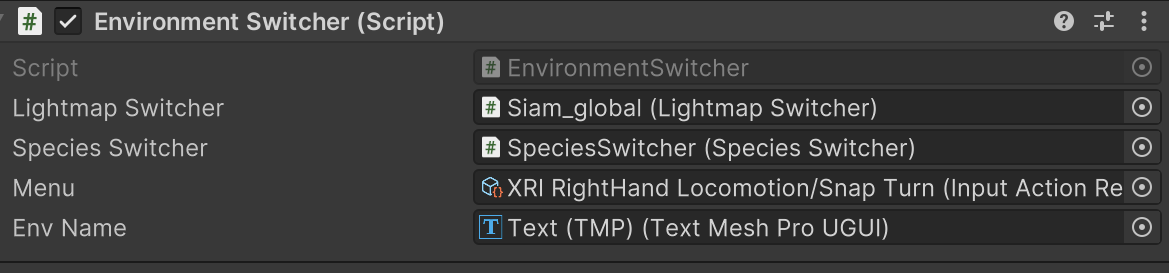
Maintenant qu’on a réussi à générer nos différents scénarios d’éclairages, on va pouvoir les utiliser dans le script “Lightmap Switcher”.Ce composant est utilisé pour gérer et changer les configurations d'éclairage et les lightmaps dans un projet Unity. Il permet de sauvegarder et de restaurer différentes configurations d'éclairage, y compris les lightmaps, les probes de réflexion et les objets de la scène.



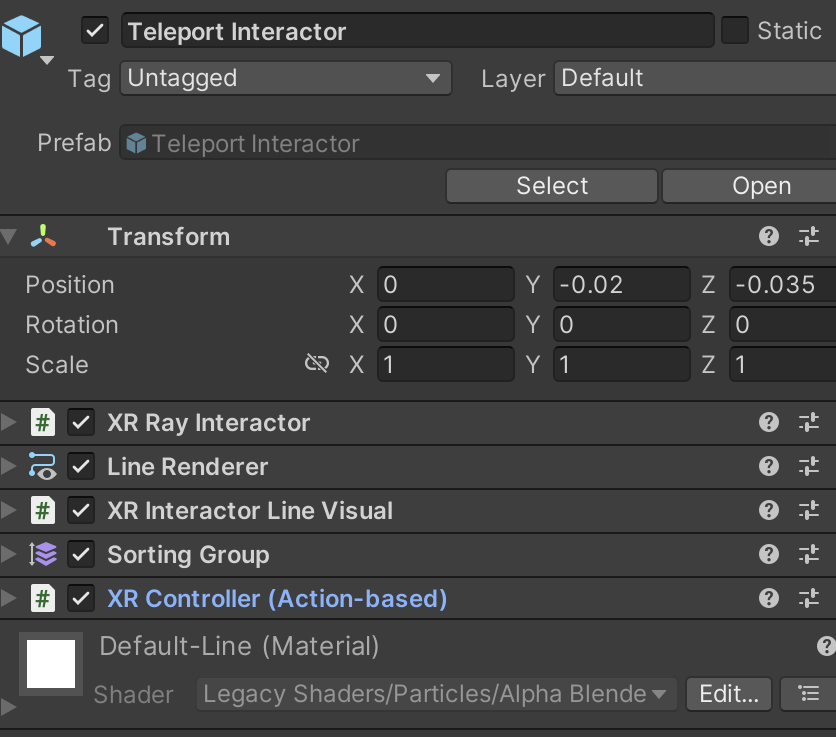
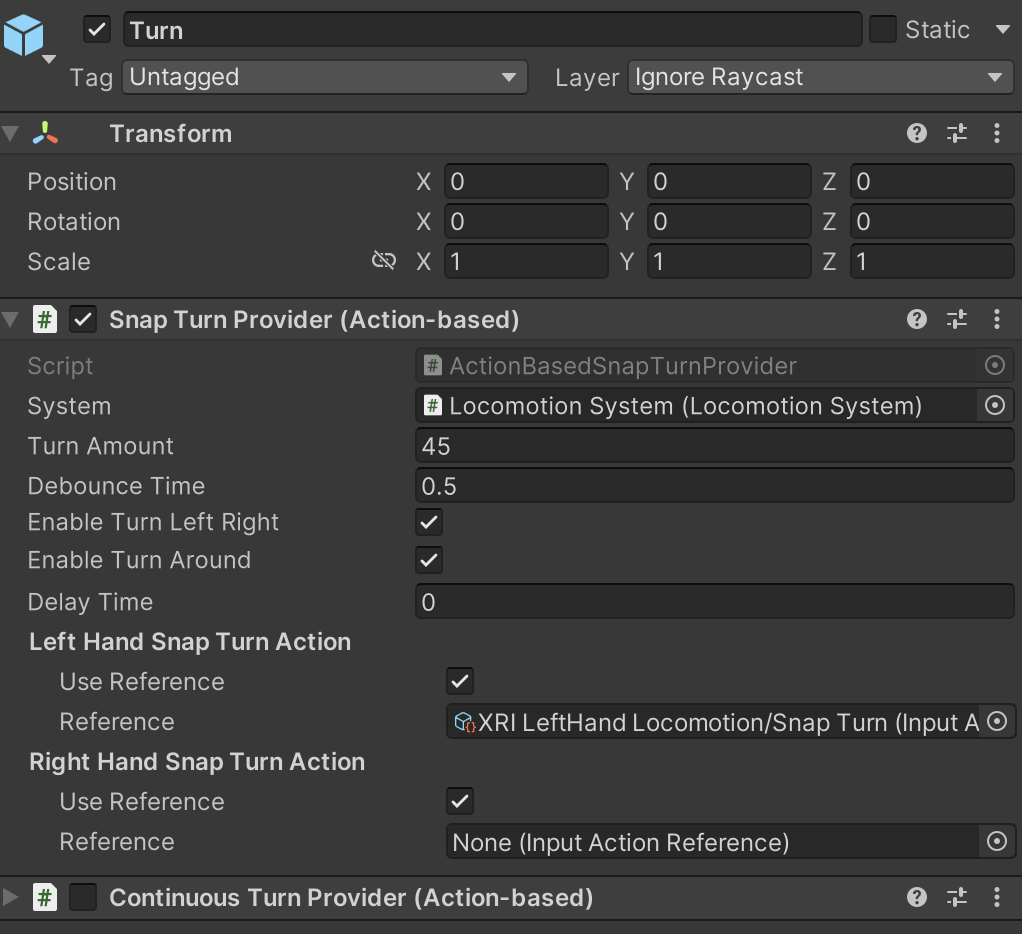
#### **C# Environment Switcher**

Le script EnvironmentSwitcher dans Unity permet de basculer entre différents environnements graphiques en utilisant le LightmapSwitcher pour changer les lightmaps et le SpeciesSwitcher pour appliquer des post-traitements spécifiques. Le LightmapSwitcher joue un rôle crucial en gérant les transitions entre différentes lightmaps, qui sont des textures pré-calculées utilisées pour simuler l'éclairage dans la scène. Ce script facilite le changement de ces lightmaps, améliorant ainsi l'atmosphère visuelle de l'environnement.

Le cœur du script réside dans la méthode Update. Si l'utilisateur fait son action (défini dans la variable “Menu” et que l'entrée est positive, le script appelle lightmapSwitcher.Next() pour passer à la lightmap suivante. Ensuite, il met à jour le texte de envName pour refléter le nom de la nouvelle lightmap. De manière similaire, si l'entrée de menu est négative, le script appelle lightmapSwitcher.Previous() pour revenir à la lightmap précédente.



### **Déplacement / Rotation**

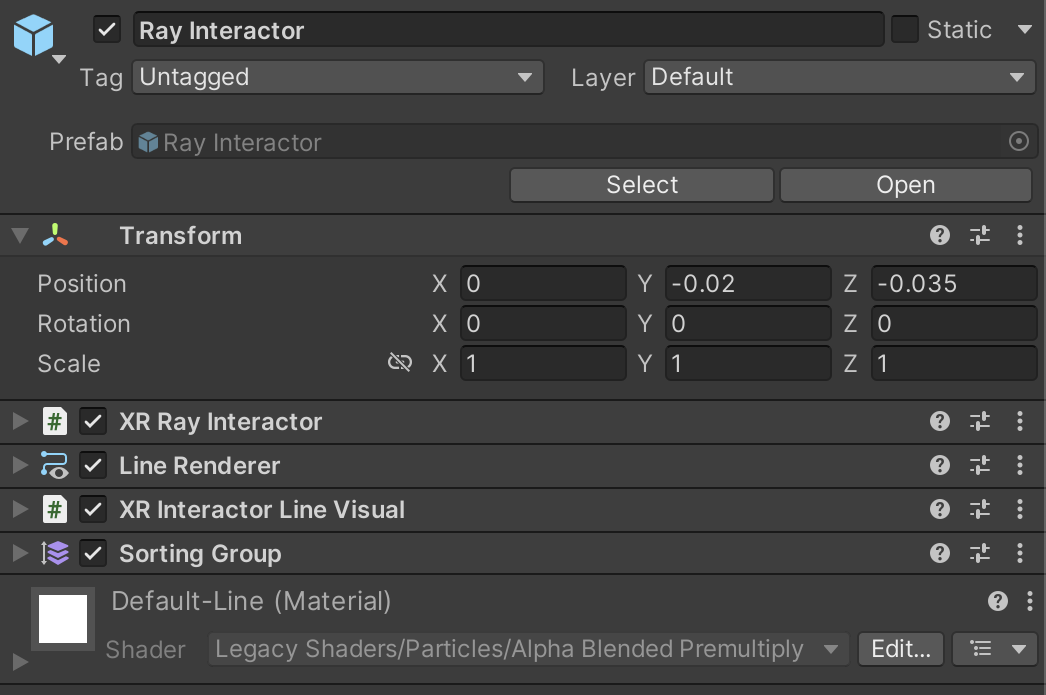


Pour les déplacements, nous utilisons le teleport interactor de la manette gauche.

* **Composants** :
  + XR Ray Interactor / Line Renderer / XR Interactor Line Visual  
     └── gestion de la courbe de téléportation, son affichage et son comportement lorsqu’elle rentre en contact avec un mesh ayant le script #TeleportationArea
  + XR Controller   
     └── gestion des différentes actions en associant les Input Actions désiré.

Pour la rotation, on désactive le continuous Turn Provider, et on active le snap Turn. On ne rentre que la référence de l’input action Snap Turn de la main gauche.

### **Interaction avec le menu**



Pour interagir avec l’UI du menu, nous utilisons le Ray Interactor. Comme nous pouvons le constater, le principe est le même que pour le Teleport interactor, à la différence que cette fois-ci, le ray ne va interagir qu’avec des éléments présents sur le Layer UI.

****

Nous ajoutons à notre menu un script #UI Controller qui récupère un input action (ici l’action d’appuyer sur le primary bouton de la manette droite) pour activer/désactiver le GameObject.  
Pour savoir comment définir les paths des input Actions, nous nous référons à la documentation Unity : XR Input Mappings. Autrement, si nous avions à notre disposition un casque utilisable en runtime, il y avait aussi un detect path automatique.

## 

### **Changement de langue**

L’utilisateur a le choix de la langue utilisée dans les différentes scènes. La première scène (*TutorialScene*) le permet à l’aide d’une dropdown. Chaque scène charge les langues et les phrases contenues dans le fichier *XmlLanguagesScene*. Ce fichier est modifiable sans toucher au code pour permettre l’ajout d’une langue facilement ultérieurement.