



- **LE STAGIAIRE**

Nom : ZENATI

Prénom : Racim

CURSUS SUIVI : GRENOBLE - L1 INFORMATIQUE PARCOURS INFORMATIQUE, MATHÉMATIQUES ET APPLICATIONS (IMA)

- **TUTEUR DE STAGE :**

Nom : FOUCAULT

Prénom : Gilles

Fonction : Enseignant-Chercheur

- **Intitulé du stage :**

Développement d'un plugin Rhino pour la modélisation directe avec une IHM en réalité virtuelle

Explorez toutes les simulations VR :

[Cliquez ici pour accéder aux simulations](#)

Table des matières

1. Laboratoire d'accueil : G-SCOP	3
1.1 Plateforme Vision-R	3
2. Intitulé du projet	4
2.1 Contexte du projet	4
2.2 Objectifs spécifiques du projet	4
2.3 Description du flux de travail	5
3. Architecture générale du projet	6
3.1 Exploration des commandes Rhino 8	7
3.2 Développement du serveur	8
3.3 Déploiement du client Python	9
3.3.1 Avantage	9
3.3.1 Inconvénient	9
3.4 Développement des Scripts Unity et des Classes d'Action	11
3.4.1 Scripts d'interaction :	11
3.4.2 Communication avec le serveur C#	12
3.4.3 Contrôle des scripts par les contrôleurs VR	12
3.5 Configuration de la Scène Unity	13
3.5.1 Interface basée sur des GameObjects Représentant des Actions	13
3.5.2 Préfabriqués Alignés avec Matériaux Spécifiques et XR Rig	13
3.5.3 Interaction avec la Scène via les Contrôleurs VR	13
4. Liens vers les Dossiers de Simulation et Sources	14

1. Laboratoire d'accueil : G-SCOP



Le laboratoire **G-SCOP** (Sciences pour la Conception, l'Optimisation et la Production de Grenoble) est une unité mixte de recherche (UMR 5272) affiliée à Grenoble INP, à l'Université Grenoble Alpes et au CNRS. G-SCOP se spécialise dans les sciences de l'ingénierie, avec un accent particulier sur la conception, l'optimisation et la production de systèmes complexes. Le laboratoire intègre diverses disciplines telles que la mécanique, l'informatique, la robotique, et les mathématiques appliquées pour relever des défis industriels et sociétaux.

Dans le cadre de ce projet, le laboratoire G-SCOP est impliqué dans le développement de nouvelles méthodes de modélisation et d'interaction en réalité virtuelle, notamment grâce à l'utilisation de la plateforme Vision-R. Ce laboratoire dispose d'expertises reconnues dans le domaine de la conception assistée par ordinateur (CAO), de la réalité virtuelle, et des interfaces homme-machine, ce qui en fait un cadre idéal pour la réalisation du projet de modélisation 3D en VR.

1.1 Plateforme Vision-R

La plateforme Vision-R, située au sein du laboratoire G-SCOP, est une infrastructure dédiée à la recherche et au développement dans le domaine de la réalité virtuelle et augmentée. Cette plateforme offre un environnement technologique avancé, comprenant des systèmes de visualisation 3D immersifs comme les caves à réalité virtuelle, les casques VR, et des dispositifs de capture de mouvements.

Vision-R est utilisée pour explorer de nouvelles méthodes d'interaction et de visualisation dans des environnements virtuels, avec des applications dans des domaines variés tels que l'industrie, l'architecture, et la formation. Dans le cadre de ce projet, la plateforme Vision-R joue un rôle crucial en fournissant les outils nécessaires pour développer et tester les nouvelles métaphores d'interaction en réalité virtuelle, en particulier pour la modélisation 3D directe et intuitive en VR. Cette infrastructure permet de mettre en œuvre et de valider les concepts développés, en offrant un retour immédiat sur les interactions et les modifications apportées aux modèles 3D en temps réel.

2. Intitulé du projet

2.1 Contexte du projet

Le projet consiste à développer un outil avancé de modélisation 3D directe destiné à être utilisé dans un environnement de réalité virtuelle (VR). L'objectif à long terme est de permettre aux utilisateurs d'interagir avec des modèles 3D de manière intuitive, en utilisant des dispositifs VR tels que des contrôleurs 6D ou des systèmes de capture de gestes, au sein d'une cave immersive à trois murs tactiles. Ce projet s'inscrit dans le cadre de l'exploration des possibilités offertes par la réalité virtuelle pour améliorer et simplifier la modélisation 3D, en s'inspirant d'outils comme Google SketchUp et les fonctionnalités de modélisation directe de Rhino 8.

2.2 Objectifs spécifiques du projet

L'objectif principal du projet est de développer une méthode pour exécuter des opérations de modélisation directe, telles que le Push-Pull, le MoveFace, et d'autres commandes similaires, en intégrant les interactions VR avec les capacités de modélisation de Rhino 8. Plus précisément, le projet vise à :

- **Intégrer des commandes de modélisation directe** : Développer et implémenter des commandes comme Push-Pull (pousser-tirer), MoveVertex (déplacement d'un sommet), MoveEdge (déplacement d'arête), et les fonctions de création de formes géométriques telles que les cubes, cônes, sphères ... dans un contexte VR.

Explorez toutes les simulations VR : [Cliquez ici pour accéder aux GIF](#)

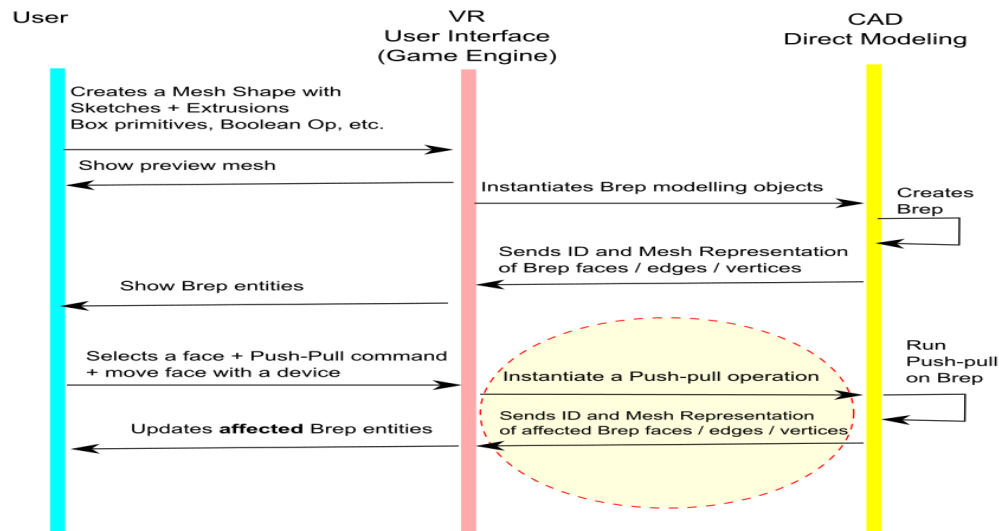
- **Remplacer l'interface traditionnelle de Rhino 8** : Mettre en place une interface utilisateur simplifiée et adaptée à la VR pour interagir avec les commandes de modélisation, remplaçant ainsi l'interface fenêtrée classique de Rhino. Cette interface devra permettre la sélection intuitive des surfaces, la définition de la direction et de la distance des manipulations, ainsi que l'utilisation du "snapping" pour s'aligner sur la géométrie existante.

Le **snapping** est une fonctionnalité dans les logiciels de modélisation 3D qui permet d'aligner ou de faire "coller" des objets ou des points à des éléments spécifiques

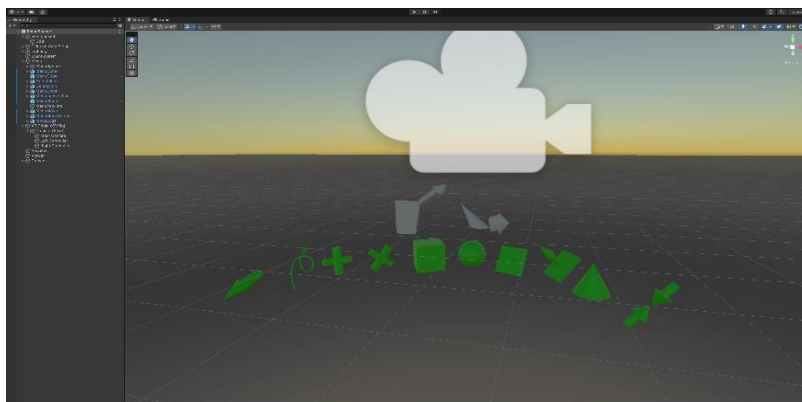
- Assurer la synchronisation entre Rhino et l'interface VR : Développer un serveur Rhino capable de recevoir et d'exécuter des commandes depuis l'interface VR, en mettant à jour les entités géométriques en temps réel.

2.3 Description du flux de travail

Le flux de travail du projet, tel qu'illustré dans la figure, se décompose en plusieurs étapes cruciales :



- Création et Prévisualisation de la Forme de Maillage :** L'utilisateur commence par créer une forme de maillage dans l'interface VR en utilisant des outils de base comme des croquis, des extrusions, des primitives de boîte, de sphères ...etc, ou des opérations booléennes. Cette forme est prévisualisée dans l'interface utilisateur.

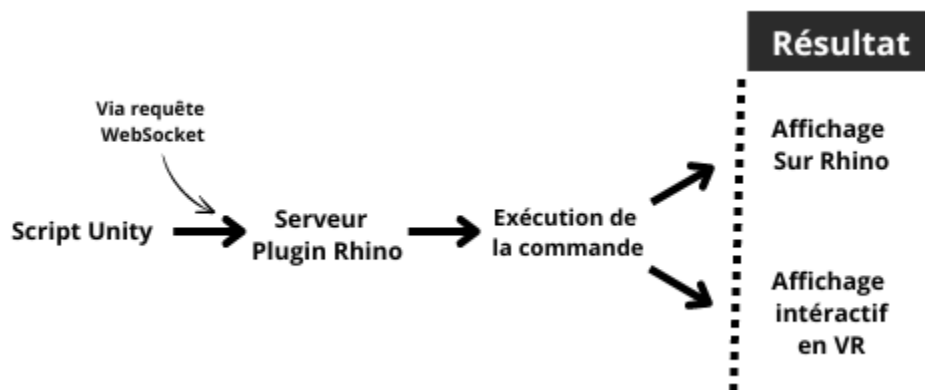


Ce **menu VR** en arc dans **Unity** présente des icônes vertes représentant divers outils de modélisation, disposées dans un environnement **3D**. Les éléments du menu sont conçus pour être utilisés avec des contrôleurs **VR**, permettant une interaction immersive dans l'espace virtuel.

2. **Envoi des Objets BRep vers Rhino** : Une fois la forme de maillage créée, le serveur instancie les objets de modélisation BRep correspondants et envoie l'ID et la représentation en maillage des faces, arêtes, et sommets BRep à Rhino via une communication **WebSocket** ([The websocket protocol](#)).

Un **BRep** (Boundary Representation) est une méthode de modélisation géométrique qui représente des objets 3D en représentant leur surface frontière par un ensemble de face délimités par des arêtes et des sommets ([Brep Data Structure](#)).

3. **Exécution des Commandes de Modélisation** : Lorsque l'utilisateur sélectionne une face et choisit une commande (par exemple, Push-Pull), cette action est transmise à Rhino qui exécute l'opération sur l'objet BRep. Le serveur Rhino traite cette commande et met à jour les entités BRep affectées.
4. **Mise à jour et Affichage en Temps Réel** : Après l'exécution de la commande, les entités BRep modifiées sont renvoyées à l'interface VR, où elles sont immédiatement mises à jour et affichées, permettant à l'utilisateur de visualiser les changements en temps réel.



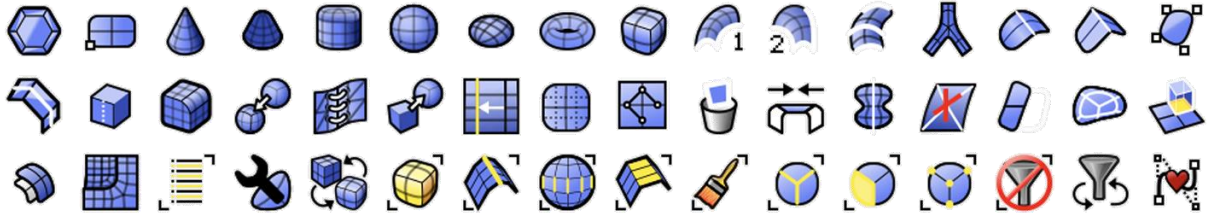
3. Architecture générale du projet

L'architecture du projet intègre plusieurs technologies pour permettre une interaction fluide entre les différents composants. Le **client Unity3D**, développé en **C#**, utilise les bibliothèques Unity3D, Rhino3dm, WebSockets, et XR.Interaction.Toolkit pour afficher et manipuler les modèles BRep de Rhino dans un environnement de réalité virtuelle, visualisé à travers un casque VR. Ce client permet aux utilisateurs de créer et modifier des formes 3D en utilisant des contrôleurs VR. Le **serveur principal**, un plugin Rhino en **C#**, utilise Rhino.Commands, Rhino.Common, WebSockets, et Newtonsoft.Json.Rhino pour gérer les opérations de modélisation, telles que la lecture et la modification des formes Brep, via des requêtes WebSocket. Bien que des alternatives aient été explorées, telles qu'un **serveur Python**, ce projet a été abandonné en raison de défis techniques qui ont limité son efficacité. En somme, l'architecture assure une interaction cohérente et en temps réel entre les commandes de

modélisation Rhino et l'environnement VR, malgré les obstacles rencontrés avec les solutions alternatives.

3.1 Exploration des commandes Rhino 8

- **Inventaire des commandes disponibles** : Identification et documentation des commandes de modélisation directe de **Rhino 8** (Extrude, MoveFace, AddBox, etc.).



- **Tests de compatibilité** : Tester chaque commande dans un environnement de développement pour s'assurer qu'elle fonctionne correctement lorsqu'elle est appelée par des scripts externes.

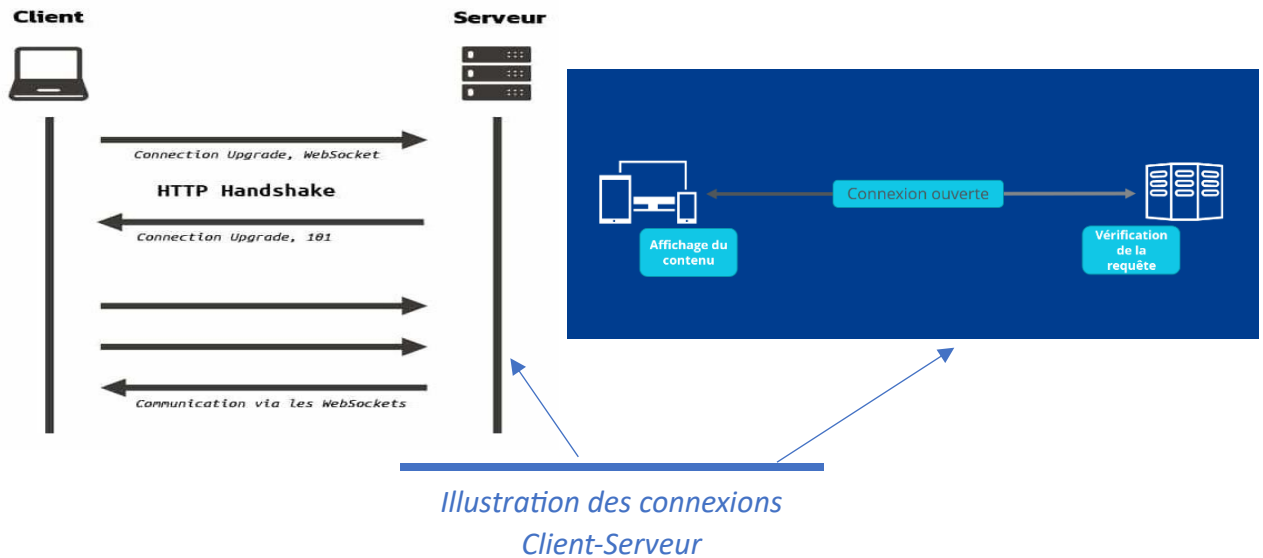
Explorez toutes les simulations Rhino : [Cliquez ici pour accéder aux GIF](#)

- **Adaptation pour la VR** : Adapter les commandes pour qu'elles soient utilisables dans un contexte de réalité virtuelle, en tenant compte des spécificités d'interaction en VR.



3.2 Développement du serveur

- **Mise en place du serveur** : Développement d'un serveur en C# pour gérer les communications entre l'interface VR (via **Unity**) et **Rhino**. Le serveur écoute les requêtes **WebSocket**, interprète les messages reçus, et exécute les commandes Rhino correspondantes.



- **Gestion des connexions WebSocket** : Implémentation des WebSocket pour permettre une communication en temps réel. Le serveur doit être capable de gérer les connexions, de recevoir des messages, et de renvoyer des réponses après l'exécution des commandes.

Explorez toutes les simulation VR : [Cliquez ici pour voir tout les GIF](#)

- **Initialisation et gestion des connexions** : Assurer que le serveur démarre correctement et maintient les connexions ouvertes jusqu'à ce que l'opération soit terminée. `{ json }`
- **Interprétation des messages** : Le serveur doit pouvoir décoder les messages JSON reçus, en extraire les commandes Rhino, et gérer les erreurs éventuelles.
- **Exécution des commandes et retour d'information** : Une fois les commandes exécutées dans Rhino, le serveur envoie la nouvelle géométrie **BRep** (sous forme de JSON) au client VR pour mise à jour de la scène 3D.

3.3 Déploiement du client Python

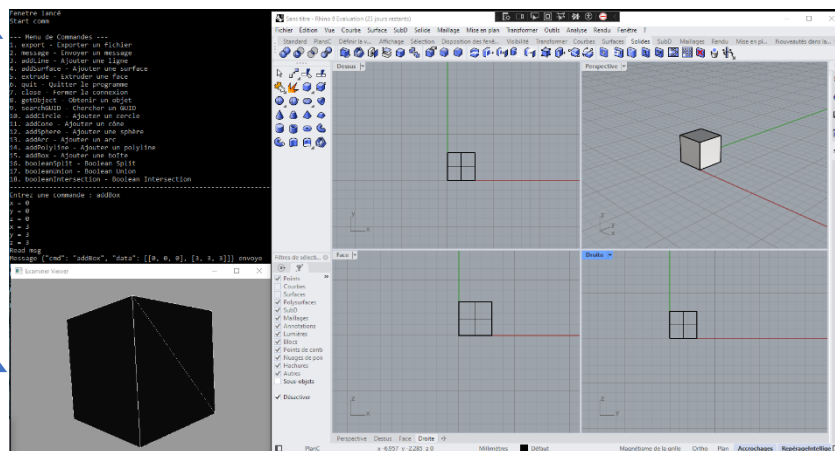
3.3.1 Avantage :

- **Interface avec le serveur C#** : Développement d'un client Python capable de se connecter au serveur C#, d'envoyer des commandes, et de recevoir les réponses. Le client doit s'assurer que les messages sont correctement formatés et que les erreurs sont gérées.
- **Gestion des commandes et des réponses** : Le client envoie les commandes à exécuter dans Rhino et attend la réponse du serveur. Une fois la réponse reçue, elle est analysée et les résultats sont envoyés à Unity pour mise à jour de la scène.
- **Traitement asynchrone** : Le client doit gérer les requêtes et les réponses de manière asynchrone pour éviter tout blocage du programme principal.

Fenêtre Python

Fenêtre Rhino 8

Fenêtre Pivy



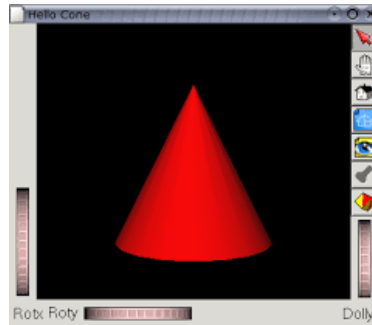
3.3.1 Inconvénient :

L'interface en temps réel du client Python repose sur **Pivy**, une bibliothèque qui sert de pont entre **Python** et **Coin3D** pour les interfaces 3D. Toutefois, cette solution présente certaines limitations.

1. Limites de l'Interface en Temps Réel Pivy :

Pivy a été abandonné parce qu'il était nécessaire de représenter la distance des extrusions sur la fenêtre **2D** en fonction des interactions avec la souris, ce que font les logiciels de **CAO** traditionnels sur desktop. Bien que cela aurait pu être programmé dans **Pivy**, l'objectif du projet était de concevoir un environnement de réalité virtuelle (VR).

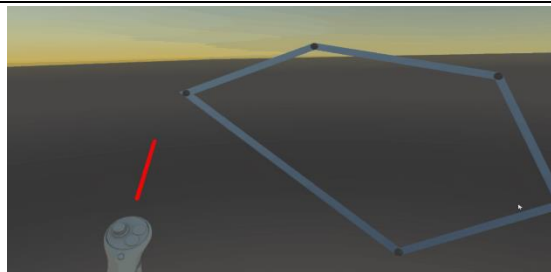
Dans ce contexte, la distance des extrusions pouvait être définie simplement comme la distance parcourue dans la direction normale à la face extrudée. L'interface **Pivy** affiche uniquement l'opération en cours, sans permettre de visualiser les opérations précédentes. Cela signifie que l'historique des modifications n'est pas accessible visuellement, rendant la révision ou l'ajustement des opérations antérieures difficile. Une alternative plus efficace serait l'utilisation d'un moteur de jeu tel que **Unity**, qui offrirait une interface plus robuste et plus intuitive pour gérer les opérations en temps réel.



2. Incapacité de Prévisualiser lors des Opérations :

L'un des principaux inconvénients du déploiement d'un client Python dans ce contexte est l'incapacité à prévisualiser les opérations en cours. Contrairement à d'autres solutions qui permettent une visualisation en temps réel des modifications apportées aux objets, le client Python ne fournit pas cette fonctionnalité de manière native. Cela peut rendre le processus de conception moins intuitif et plus sujet aux erreurs, car il est difficile de juger de l'impact des modifications sans une prévisualisation visuelle.

Prévisualisation lors de la création d'une surface sur un environnement VR (**Unity**)



3. Difficulté des Opérations entre Deux Objets :

Une autre contrainte du déploiement du client **Python** est la complexité des opérations entre deux objets. Actuellement, pour effectuer des opérations entre deux objets, il est nécessaire de récupérer les **GUID** (Global Unique Identifier) de chacun d'entre eux. Cette méthode est peu pratique et peut être source de confusion, surtout dans un environnement de travail complexe avec de nombreux objets. Il serait beaucoup plus simple et ergonomique de pouvoir sélectionner directement les deux objets concernés, une fonctionnalité qui pourrait être plus facilement implémentée avec un moteur de jeu tel qu'**Unity**.



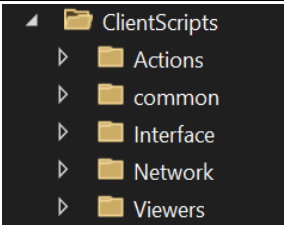
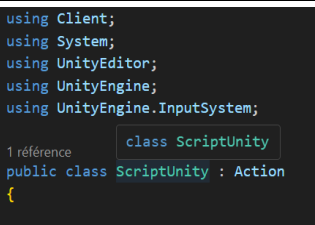
Un **GUID** est un identifiant unique de **128 bits** utilisé pour identifier de manière sûre et unique des objets, des composants ou des ressources dans des systèmes informatiques.

3.4 Développement des Scripts Unity et des Classes d'Action



3.4.1 Scripts d'interaction :

Les scripts d'action dans **Unity** sont essentiels pour réaliser des opérations spécifiques dans l'environnement VR. Ils sont responsables de la création de **formes primitives** comme des sphères et des cubes, de la réalisation d'**opérations booléennes** (différence, fusion, intersection) entre objets, ainsi que de la manipulation d'éléments tels que le déplacement de vertices, d'arêtes ou l'extrusion. Chaque script est programmé pour effectuer ces actions précises en fonction des besoins de l'utilisateur, en servant de pont entre l'interface **VR** et le serveur **C#**.

Les Scripts Unity	Création d'une classe Action
	

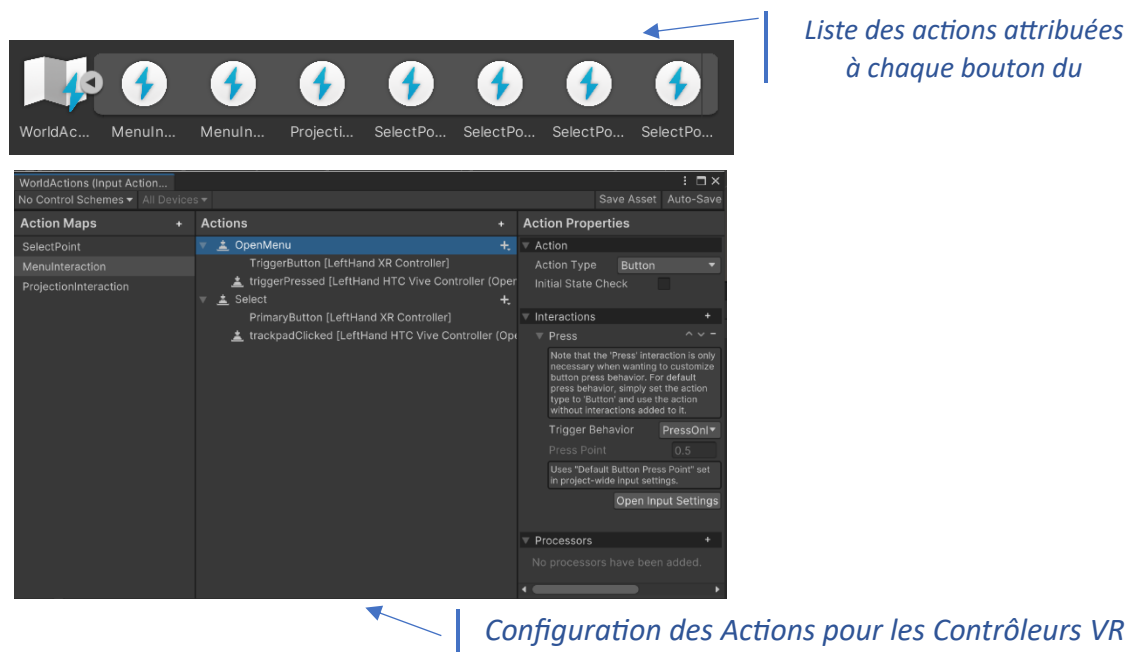
3.4.2 Communication avec le serveur C#

Ces scripts envoient des commandes au serveur C# intégré au plugin de modélisation. Cette communication se fait via des requêtes **WebSocket**, où les commandes sont encapsulées sous forme de messages **JSON**. Une fois que le serveur reçoit ces commandes, il exécute les actions associées dans Rhino, telles que la création de nouvelles formes ou la modification des géométries existantes. Une fois les opérations terminées, le serveur renvoie les résultats sous forme de JSON, que les scripts Unity réceptionnent pour mettre à jour l'affichage dans la scène VR. Cette boucle de communication assure une interaction fluide et synchronisée entre Unity et Rhino.

Explorez toutes les simulation VR : [Cliquez ici pour voir tout les GIF](#)

3.4.3 Contrôle des scripts par les contrôleurs VR :

Les scripts d'action peuvent être directement manipulés à l'aide des contrôleurs **VR**, permettant une interaction intuitive et immersive dans l'environnement virtuel. Chaque bouton ou gâchette sur les contrôleurs VR peut être assigné à une fonction spécifique dans ces scripts. Par exemple, une touche peut déclencher une opération de déplacement, tandis qu'une autre peut initier la création d'une forme primitive. Cette assignation permet à l'utilisateur de contrôler les actions en VR de manière naturelle, en utilisant les contrôleurs pour manipuler les objets et effectuer les opérations de modélisation en temps réel.



3.5 Configuration de la Scène Unity

3.5.1 Interface basée sur des GameObjects Représentant des Actions

L'interface utilisateur dans **Unity** est constituée de plusieurs **GameObjects**, chacun représentant une action spécifique. Par exemple, un cube symbolise la création d'un cube. Ces GameObjects agissent comme des boutons interactifs dans l'environnement VR, offrant un moyen intuitif de lancer différentes opérations.

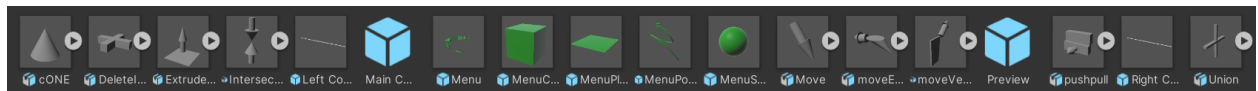
[Voir la démonstration vidéo](#)

3.5.2 Préfabriqués Alignés avec Matériaux Spécifiques et XR Rig

Ces GameObjects sont des **prefabs** soigneusement alignés, chacun doté de matériaux spécifiques pour les distinguer visuellement. Le système **XR Rig** est utilisé pour configurer la scène, incluant la Main Camera et les deux contrôleurs VR (gauche et droite). Le XR Rig permet de positionner et d'orienter correctement la vue de l'utilisateur et les contrôleurs, garantissant une interaction fluide dans l'environnement VR.

Liste des prefabs

[Voir la configuration des contrôleurs](#)

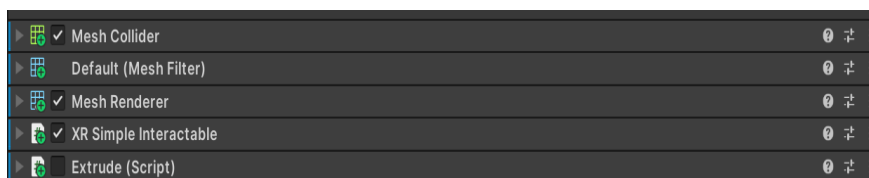


3.5.3 Interaction avec la Scène via les Contrôleurs VR

L'interaction avec cette scène se fait à l'aide des contrôleurs VR et de leur système de **Raycast**. Ce système permet de sélectionner un des GameObjects du menu pour lancer la commande associée. La logique derrière cette interaction repose sur plusieurs éléments :

- Créer un GameObject avec l'icône représentant la fonction souhaitée.
- Ajouter à ce GameObject un **MeshCollider** pour détecter les interactions, un **MeshRenderer** pour le rendre visible, et un **XRSimpleInteractable** pour permettre sa sélection ainsi que son script dérivé de la classe "**Action**".

Composant du
GameObject



4. Liens vers les Dossiers de Simulation et Sources

1. **Simulation Rhino** : [Accéder au dossier Simulation Rhino](#)

Ce dossier contient les simulations effectuées dans Rhino, montrant les différentes opérations et commandes exécutées via les scripts **Unity** et le serveur **C#**. Chaque simulation est documentée pour faciliter la compréhension des étapes suivies.

2. **GIF VR Unity** : [Accéder au dossier GIF VR Unity](#)

Dans ce dossier, vous trouverez une série de **GIFs** capturés lors des interactions dans l'environnement **VR** de **Unity**. Ces GIFs illustrent comment les scripts d'action, les contrôleurs VR, et les commandes interagissent pour créer des objets et manipuler la scène en temps réel.

3. **Sources et Références Utilisées**

Ce dossier regroupe toutes les sources, articles, tutoriels, et autres références qui ont été consultés et utilisés durant le projet. Elles ont été essentielles pour surmonter les défis techniques et pour apporter des solutions innovantes aux problèmes rencontrés.

- [RhinoCommon documentation](#)
- [RhinoScript-API-DOC](#)
- [Unity documentation](#)
- [XR Resources VISION-R](#)