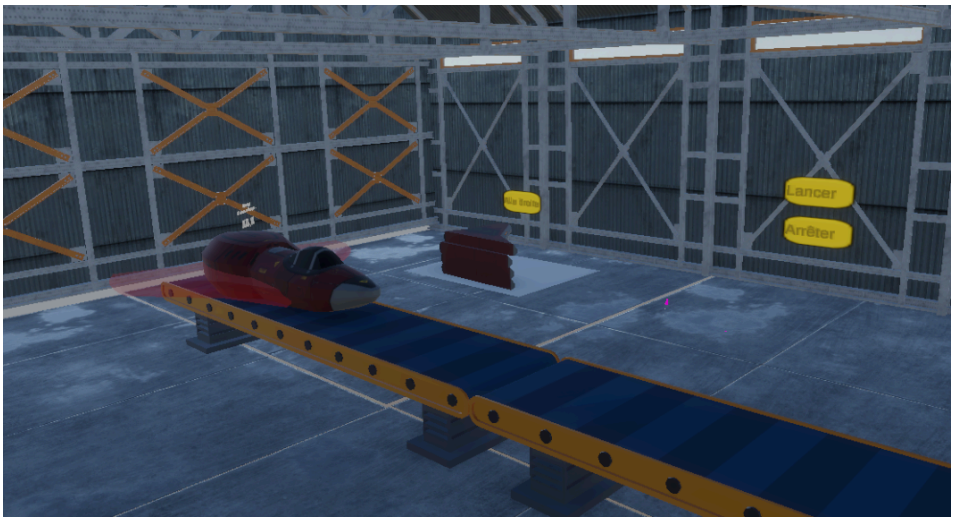
AI38 : Rapport de projet

Spaceship Factory - Virtual Reality



Semestre A24

SIX Maxime - RAFEI Jana

Contexte

Dans le cadre de l'UV AI38, nous avons conçu et développé une application en réalité virtuelle en utilisant Unity et le casque HTC VIVE. Ce projet a pour but de simuler le fonctionnement d'une usine de fabrication de vaisseaux spatiaux, en offrant aux utilisateurs une expérience immersive, interactive et réaliste, mettant en valeur les possibilités qu'offre la réalité virtuelle dans des environnements industriels simulés.

Objectifs

L'objectif principal est d'entraîner l'utilisateur à assembler les différentes composantes des vaisseaux spatiaux de manière efficace et rapide, tout en mettant en avant des mécanismes pédagogiques pour renforcer l'apprentissage. Cette simulation s'inscrit dans une démarche visant à exploiter la réalité virtuelle pour combiner formation et divertissement.

Description

Description générale :

Au démarrage du projet, l'utilisateur se trouve dans une usine d'assemblage et un menu s'affiche dans la scène, permettant de lancer ou d'arrêter le jeu. L'utilisateur, à l'aide de sa manette, peut sélectionner l'option souhaitée.

Une fois que l'utilisateur lance l'application, le chronomètre d'assemblage se déclenche. L'utilisateur doit alors assembler les différentes composantes sur le châssis placé sur un convoyeur en mouvement. Les composantes du vaisseau sont réparties dans différents coins de l'usine, et l'utilisateur doit récupérer chaque pièce pour l'assembler à une position précise sur le châssis. Pour fixer les composantes, l'utilisateur utilise une visseuse pour visser les pièces avec précision. Pendant ce processus, le mouvement continu du convoyeur accroît la difficulté de l'assemblage, car l'utilisateur doit terminer avant que le châssis ne quitte le convoyeur. Cela ajoute un défi supplémentaire à

l’expérience immersive.

À la fin de l’assemblage, l’utilisateur soulève le vaisseau à l’aide de sa manette et le dépose au sol. Cela déclenche l’apparition d’un nouveau châssis sur le convoyeur, marquant ainsi le début d’un nouveau cycle d’assemblage.

Description technique:

Nous avons développé deux interactions principales : l’assemblage et le vissage.

Assemblage :

Pour réaliser l’assemblage, nous devons simuler l’attache d’un objet sur un autre. Pour cela, le châssis doit détecter l’arrivée de sa composante à l’aide d’un collider placé sur le point d’attache. Un script dédié est associé à chaque point d’attache pour gérer cette interaction.

Lorsqu'une aile entre dans la zone de collision du point d'attache, le script vérifie si elle possède le tag requis pour l'attachement et désactive tous ses scripts et colliders pour éviter toute interaction ou comportement indépendant. Ensuite, l'aile est fixée avec précision à la position et à l'orientation définies par le point d'attache, tout en devenant un enfant de ce dernier pour maintenir son placement. Enfin, l'indicateur visuel du point d'attache est désactivé pour signaler que ce point est désormais occupé.

```
 private void OnTriggerEnter(Collider other){
        if (isOccupied) return;
        if (other.CompareTag(requiredTag)) // Vérifie si l'objet peut être
attaché
{
            Debug.Log($"L'objet {other.name} est entré dans la zone de
l'AttachPoint {name}.");
            AttachObject(other.gameObject);
            HideIndicator();// Désactiver le MeshRenderer de l'indicateur
        }
}
    private void AttachObject(GameObject obj){
       MonoBehaviour[] scripts = obj.GetComponents<MonoBehaviour>(); //
```

```
Désactiver tous les scripts
        foreach (MonoBehaviour script in scripts){script.enabled = false;}
        Collider[] colliders = obj.GetComponentsInChildren<Collider>(); //
Désactiver les colliders
        foreach (Collider col in colliders){
            if (col.name != "Screw_Cross"){col.enabled = false;
            }
        }
        Rigidbody rb = obj.GetComponent<Rigidbody>(); // Rendre l'objet
immobile
        if (rb != null){rb.isKinematic = true;} // Désactive la physique
        obj.transform.SetParent(transform, true);// Faire de l'objet un enfant
de la base
        obj.transform.position = attachTransform.position;// Définir la
position et la rotation
.....
```

Figure 1 : AttachPoint.cs

Vissage :

La deuxième interaction est de faire visser les vis. L'attachement ne sera réalisé que lorsque l'utilisateur prend une perceuse du sol et visse l'endroit à attacher.
Pour simuler cette interaction, chaque vis possède un collider qui détecte l'arrivée de l'embout d'une perceuse dans sa zone et est attaché par un script qui gère cette interaction.

Lorsqu'un tournevis entre en contact avec la vis, le script démarre une coroutine qui déplace progressivement la vis vers une position cible jusqu'à ce qu'elle soit complètement vissée. Pendant ce processus, la distance vissée est suivie et, une fois atteinte, la vis est marquée comme entièrement vissée et sa couleur change en vert. Le contact avec la perceuse est surveillé pour arrêter le processus si elle s'éloigne. Ce lien entre la détection et le vissage simule efficacement l'interaction utilisateur.

```
void OnTriggerEnter(Collider other){
        if (other.CompareTag("Embout")){
            isTouchingScrewdriver = true;
            screwdriverTip = other.transform;
            // Démarre le vissage progressif
            if (screwingCoroutine == null){
                screwingCoroutine = StartCoroutine(Screw());}
            }
        }
private IEnumerator Screw(){
        while (!isFullyScrewed && isTouchingScrewdriver){
            Vector3 direction = (ciblePosition.transform.position -
            transform.position).normalized; // direction vers la position
cible
            float step = screwSpeed * Time.deltaTime; //step de deplacement
            transform.position += direction * step; //Déplace la vis vers
la position cible
            screwedAmount += step; // Met à jour la distance vissée
            If (screwedAmount >= maxScrewDistance)//Vérifie si la vis est
complètement vissée
               {isFullyScrewed = true;}
            yield return null; // Attend la prochaine image
        }
        screwingCoroutine = null;
}
```

Figure 2 : ScrewInteraction.cs

Problèmes rencontrés

Lors de la réalisation du projet, plusieurs problèmes ont été rencontrés, ralentissant parfois le développement. Tout d'abord, des incompatibilités entre les assets importés ont causé des incohérences visuelles, car certains n'avaient pas le même rendu, ce qui a nécessité des ajustements pour harmoniser l'apparence.

De plus, l'interaction utilisateur manquait de réalisme, l'utilisateur pouvant attraper une composante ou la perceuse même en étant loin de l'objet. Pour résoudre ce problème,

une vérification supplémentaire de la distance entre l'utilisateur et l'objet pourra être ajoutée ou une zone de collision qui détecte l'arrivée de l'utilisateur, limitant l'interaction à une zone définie autour de l'objet.

Un autre frein majeur a été un bug récurrent avec le PC de CI, qui obligeait à recompiler fréquemment, ce qui a consommé un temps précieux. En effet, pour optimiser notre temps de travail, nous avons avancé sur le projet en dehors de la salle de CI. Cependant, lors de la récupération du travail sur le PC de CI, des erreurs de compilation survenaient systématiquement, nécessitant des ajustements et des résolutions, ce qui ralentissait considérablement notre progression.

Nous avons utilisé Gitlab comme un outil de partage de code et de versionnage. Cependant, la migration des avancements s'est avérée complexe, notamment pour les composantes Unity, en raison des dépendances et des différences dans les configurations locales des développeurs. Chaque fonctionnalité ou modification nécessitait souvent que d'autres parties du projet soient finalisées pour éviter des conflits ou des comportements inattendus lors de l'intégration.

Par ailleurs, au début du projet et à cause de notre manque d'expérience, des pertes de travail ont été causées par des oublis de sauvegarde des prefabs, ce qui a nécessité de reprendre certains éléments depuis le début.

Enfin, le manque de temps pour avancer sur le projet a amplifié ces difficultés, rendant leur gestion encore plus complexe.

Proposition d'amélioration

Pour améliorer le projet, plusieurs points peuvent être intégrés afin d'enrichir l'expérience utilisateur et d'optimiser les performances.

- Nous pouvons ajouter un bouton dédié au vissage pour rendre l'interaction plus intuitive et simuler le mouvement de rotation de la vis.
- Nous pouvons intégrer des retours visuels, comme des particules représentant des

éclats de métal, et des retours sonores, comme le bruit du vissage, pour renforcer l'immersion de l'utilisateur.

- Nous pouvons également introduire plusieurs niveaux de difficulté, comme un convoyeur plus rapide ou des assemblages plus complexes, pour offrir un défi progressif et varié.
- Pour motiver davantage les utilisateurs, nous pouvons mettre en place un système de sauvegarde permettant d'enregistrer les meilleurs scores.
- De plus, nous pouvons rendre les interactions plus réalistes en obligeant l'utilisateur à approcher physiquement un objet pour l'attraper.
- Enfin, notre application permet à l'utilisateur de générer plusieurs exemplaires de la même composante, alors qu'en réalité un seul exemplaire est nécessaire par vaisseau. Cette fonctionnalité risque de ralentir l'exécution en raison de la génération simultanée de multiples ressources. Pour éviter cela, nous pouvons limiter la génération des composantes à un seul exemplaire par vaisseau.

Conclusion

La réalisation de ce projet nous a permis de consolider nos connaissances en développement d'applications en réalité virtuelle, tout en nous familiarisant avec les défis techniques et conceptuels qu'implique ce domaine. En intégrant Unity et le casque de réalité virtuelle, nous avons exploré des aspects variés de la VR, tels que l'interactivité utilisateur, les mécanismes de détection et de simulation d'actions physiques, et la gestion des performances dans un environnement immersif. Ce projet nous a également permis de mettre en pratique des notions apprises en cours, notamment l'importance de l'immersion, la conception d'interactions intuitives.

En conclusion, ce projet constitue une étape clé dans notre apprentissage de la réalité virtuelle, en combinant réflexion technique, créativité et méthodologie de gestion de projet, et nous ouvre de nouvelles perspectives pour concevoir des expériences immersives dans des contextes variés.