Exploitation et développement du Cube VR pour de la réalité virtuelle

Mémoire de Projet de Recherche Technologique

Luca JOSEPH

Encadré par : Jade-Emmanuelle HEITZ, Mathieu KOEHL

Présenté le 22 janvier 2025
INSA Strasbourg
Spécialité Topographie
France

ABSTRACT

This project focuses on the enhancement and exploitation of the Cube VR, a virtual reality environment aimed at creating immersive and interactive experiences, particularly for the preservation and promotion of cultural heritage. Key developments include the creation of Non-Player Characters (NPCs) with custom textures, animations, and intelligent navigation, as well as the integration of interactive dialogues and modular menus, such as a toolbox and a tutorial scene, to improve user engagement and accessibility. These features, designed for modularity and reusability, enrich the virtual environment and facilitate future development. Comprehensive documentation ensures the project's sustainability, enabling seamless integration into new contexts. This work highlights the potential of VR to enhance interactivity and immersion in educational and cultural applications, paving the way for further innovations.

RÉSUMÉ

Ce projet porte sur le développement et l'exploitation du Cube VR, un environnement de réalité virtuelle destiné à créer des expériences immersives et interactives, notamment pour la préservation et la valorisation du patrimoine culturel. Les principaux développements incluent la création de Personnages Non Joueurs (PNJ) avec des textures, animations et une navigation intelligente, ainsi que l'intégration de dialogues interactifs et de menus modulaires, tels qu'une boîte à outils et une scène tutoriel, pour améliorer l'accessibilité et l'engagement des utilisateurs. Ces fonctionnalités, conçues pour être modulaires et réutilisables, enrichissent l'environnement virtuel et facilitent les développements futurs. Une documentation complète garantit la pérennité du projet, permettant une intégration fluide dans de nouveaux contextes. Ce travail met en lumière le potentiel de la réalité virtuelle pour améliorer l'interactivité et l'immersion dans des applications éducatives et culturelles, ouvrant la voie à de nouvelles innovations.

RERMERCIEMENTS

Je tiens à remercier mes encadrant.es Jade-Emmanuelle HEITZ et Mathieu KOEHL du soutien constant et du suivi de mon projet durant ce semestre chargé. Merci également à Philippe SEITIER pour sa disponibilité et des précisions sur l'environnement du Cube.

TABLE DES MATIÈRES

RERMERCIEMENTS

Je tiens à remercier mes encadrant.es Jade-Emmanuelle HEITZ et Mathieu KOEHL du soutien constant et du suivi de mon projet durant ce semestre chargé. Merci également à Philippe SEITIER pour sa disponibilité et des précisions sur l'environnement du Cube.

INTRODUCTION

0.1 Contexte du projet

Le projet Châteaux Rhénans - Burgen am Oberrhein, financé par le programme européen INTERREG VI, a pour but de préserver et de promouvoir le patrimoine castral rhénan, notamment les châteaux situés le long du Rhin, en Alsace et en Allemagne. Ce projet inclut l'action 4.6, gérée par l'INSA de Strasbourg, qui se concentre sur la modélisation 3D des sites historiques, et l'action 5.4, gérée par Tourismus Südliche Weinstraße, qui vise à créer des supports de valorisation 3D. Le projet utilise des modèles 3D et 4D pour mettre en valeur les châteaux, souvent en ruine, à travers des visites virtuelles. Ces châteaux en ruine, font l'objet de restitutions 4D, en utilisant notamment la lasergrammétrie et la photogrammétrie. Le modèle 4D du château est mis en valeur grâce à une visite virtuelle interactive dans le Cube VR[1], une technologie développée par Virtual Concept. L'objectif est de rendre le site accessible à un large public et de définir des méthodes pour faciliter la restitution de châteaux en ruine pour les projets futurs.

Les objectifs du projet sont de développer de nouvelles fonctionnalités interactives pour enrichir l'immersion dans le Cube VR, tout en assurant leur documentation détaillée afin de garantir la pérennité et la facilité de reprise du projet par de futurs développeurs. Cette démarche vise à offrir non seulement une immersion améliorée, mais aussi des outils clairs et structurés pour simplifier les évolutions futures et le partage des connaissances.

0.2 Travaux précédents

Avant de discuter d'environnements de visite en Réalité Virtuelle ou même de texturage d'un modèle 3D, il est nécessaire de réaliser des travaux topographiques et d'acquérir des données. Dans le cadre de la préservation du patrimoine

1. Virtual Reality

castral du Rhin supérieur, mes prédécesseurs ont employé un certain nombre d'instruments et de méthodes que nous allons présenter.

Des observations GNSS (Leica AS10 (Benazzi, 2018), Trimble R10 (Heitz, 2023)) RTK (TERIA) ont été réalisées pour le géoréférencement du chantier au référentiel national (RGF93 CC48 IGN 69) et pour relever des cibles de de photogrammétrie (Heitz, 2023). Les points GNSS ont ensuite été reliés par cheminement polygonal fermé au tachéomètre (Trimble M3 et Leica TS02 (Benazzi, 2018)). La tachéométrie est également utilisée pour relever des points d'appui sous forme de cibles photogrammétriques (cibles codées, au sol) ou lasergrammétriques (sphères) (Rigaud, 2023).

Pour l'acquisition de nuages de points denses, les sites ont été principalement relevés au Scanner Laser Terrestre (Faro Focus 3D X330) puis complétés par photogrammétrie aérienne (DJI Phantom 3 et 3 Pro) et photogrammétrie terrestre (Canon EOS 700D (Benazzi, 2018), Samsung NX1000 (Bruna, 2014) et Canon EOS R5 (Rigaud, 2023))

Les données brutes sont traitées dans des logiciels spécialisés tels que Trimble Realworks, Faro SCENE, Metashape, 3DReshaper, MeshLab, et CloudCompare. Ces logiciels permettent de consolider, géoréférencer, nettoyer et segmenter les nuages de points. Désormais, les modèles des édifices et les MNT peuvent être générés.

Les travaux précédents ont permis de poser les bases essentielles, notamment en réalisant des relevés topographiques précis et en modélisant les sites patrimoniaux en 3D. Depuis ces fondations, le développement VR s'inscrit comme une étape suivante, visant à exploiter ces données pour créer des expériences immersives et ludiques destinées au grand public.

CHAPITRE 1

ÉTAT DE L'ART

1.1 Introduction de l'état de l'art

La préservation et la valorisation du patrimoine historique nécessitent aujourd'hui de combiner les techniques traditionnelles d'acquisition de données avec les nouvelles technologies de visualisation et d'interaction. Les relevés topographiques fournissent les données brutes essentielles qui doivent ensuite être traitées et exploitées pour répondre aux enjeux de la conservation et de la transmission du patrimoine. Dans ce contexte, cet état de l'art examine la chaîne complète de création d'une expérience immersive de visite virtuelle d'un site patrimonial, en se concentrant particulièrement sur les châteaux.

Nous aborderons d'abord les différentes méthodes de préparation des données, depuis la modélisation 3D par primitives géométriques, maillage ou approche hybride, jusqu'au texturage et à l'intégration de la dimension temporelle (4D). Nous explorerons ensuite des techniques de communication pour maximiser l'impact des modèles créés, ainsi que les possibilités offertes par la Réalité Virtuelle, ses différentes applications et ses divers environnements immersifs, qu'ils soient mobiles (casques VR) ou fixes (systèmes CAVE). Nous examinerons également comment la réalité virtuelle peut être spécifiquement mise au service de la préservation du patrimoine, à travers notamment la création de musées virtuels et l'utilisation de la *gamification* pour enrichir l'expérience utilisateur. Enfin, nous nous pencherons sur les deux principales méthodes de navigation, le graphe de *waypoints* et le *NavMesh*, puis sur les fondements du *pathfinding*, en détaillant les algorithmes classiques comme Dijkstra, Best-First et A*. Ces solutions sont au cœur des systèmes de déplacement en VR, permettant une gestion efficace des mouvements et une interaction réaliste avec les environnements virtuels.

1.2 Préparation des données

Dans cette section, nous examinerons les étapes clés de la préparation des données, en commençant par la modélisation 3D et le texturage, avant d'aborder la modélisation 4D et les méthodes de communication visant à valoriser et partager les modèles créés.

1.2.1 Modélisation 3D

La modélisation par primitives géométriques consiste à construire des éléments en utilisant des formes géométriques simples (plans, cylindres, cubes). Elle est adaptée pour restituer des éléments disparus, comme l'ont fait Rocha (2022) et Benazzi (2018). Le gros avantage de cette approche est qu'elle est peu coûteuse en termes de stockage. Pour le PFE de Heitz (2023), la modélisation a été réalisée par primitives géométriques en structure quad sous Autodesk Maya, car les phases historiques du site avaient presque toutes disparues.

La modélisation par maillage utilise des algorithmes pour représenter des éléments 3D à partir de nuages de points obtenus par différentes techniques, telles que la lasergrammétrie ou la photogrammétrie (Rigaud, 2023) (Heitz, 2023). La triangulation est une technique courante utilisée pour concevoir le Modèle Numérique de Terrain (MNT) ou les éléments 3D existants (Rigaud, 2023). Heitz (2023) a utilisé cette méthode pour la sauvegarde numérique et l'analyse du site.

La modélisation hybride s'agit d'une combinaison des deux méthodes précédentes. Elle permet de compenser la surcharge de stockage tout en conservant une représentation fidèle des éléments précis. Bruna (2014) a utilisé cette méthode pour la chapelle Saint-Laurent. Rigaud (2023) précise que cette méthode nécessite un effort chronophage pour isoler les parties à modéliser avec une méthode plutôt qu'une autre. Heitz (2023) utilise aussi le modèle hybride pour la modélisation de bâtiments du patrimoine, car il est plus léger, conserve certains détails, et est sémantiquement interprétable.

Un certain nombre de logiciels ont été utilisés afin d'appliquer les différentes méthodes de modélisation 3D citées précédemment. Ils sont brièvement présentés dans le tableau 1.1.

Une fois la modélisation 3D réalisée, le texturage permet d'enrichir ces modèles en leur attribuant des détails visuels et réalistes, essentiels pour une représentation fidèle et immersive.

Logiciel	Utilisation
Metashape	Créer et éditer des maillages de manière automatisée. Permet de choisir la qualité du maillage, puis de l'éditer avec des outils comme le bouchage de trous ou l'affinage (Heitz, 2023). Rigaud (2023) a utilisé ce logiciel pour la modélisation par maillage.
3DReshaper	Vérifier l'intégrité des maillages et pour des outils de texturage par projection d'image (Heitz, 2023)
MeshLab	Vérifier la cohérence de la topologie (Heitz, 2023)
Autodesk Maya	Modélisation par primitives géométriques (Heitz, 2023)
Blender	Modélisation 4D et de rambardes métalliques de château (Rigaud, 2023)

TABLE 1.1 – Logiciels utilisés pour la modélisation 3D et leur utilisation

1.2.2 Texturage

Le texturage est le processus qui consiste à donner un rendu à un modèle 3D en lui attribuant des couleurs et des détails (rugosité, relief). Il s'agit de la projection d'une image ou de plusieurs sur une surface (Heitz, 2023).

La cartographie UV permet de projeter une image 2D sur la surface d'un modèle 3D. Le logiciel Blender permet d'importer différentes cartes UV sur différents paramètres du Principed BSDF afin d'appliquer des textures réalistes au modèle (Rigaud, 2023).

Après avoir appliqué les textures, l'utilisation des *shaders* permet d'ajouter des effets visuels avancés, tels que la gestion de la lumière, des reflets ou des transparences, pour renforcer le réalisme des modèles. Les *shaders* sont des programmes informatiques qui permettent de générer ou de paramétrer des matériaux, des transformations ou des effets au sein du matériel graphique. Ils permettent de personnaliser les textures (Heitz, 2023).

Une texture peut être appliquée sur un modèle avec une image sous 3DReshaper, c'est le texturage par projection d'image (Heitz, 2023).

Maintenant que la 3D est terminée, il est intéressant de se pencher sur la dimension du temps, enrichissant les modèles et apportant un autre aspect réaliste aux simulations.

Logiciel	Utilisation
Metashape	Texture sous forme d'Atlas, permettant de gagner de l'espace (Heitz, 2023)
Blender	Utilisé pour l'application de textures réalistes (Rigaud, 2023)
Autodesk Maya	Texturage des modèles à l'aide de textures majoritairement créées sur la base de photographies prises sur site (Heitz, 2023)

TABLE 1.2 – Logiciels utilisés pour le texturage et leur utilisation

1.2.3 Modélisation 4D

La modélisation 4D s'ajoute à la 3D, visant à représenter une évolution temporelle. Elle permet de retracer l'histoire d'un site en mettant en correspondance différents modèles 3D du site à différentes époques. Souvent, les états antérieurs ont été perdus et c'est dans ce cadre que l'on fait appel à des primitives géométriques sur la base des ruines actuelles afin de modéliser une époque précédente, comme mentionné dans la partie 1.2.1. Cependant, ces objets sont extrapolés de la structure actuelle incomplète et il est intéressant d'évaluer l'incertitude de la restitution 4D. Heitz (2023) et Rigaud (2023) ont évalué les primitives géométriques générées à partir d'une échelle de couleurs qui représente des niveaux d'incertitude (*Level of Uncertainty*) proposé par Landes et al. (2019). Ce LoU est basé sur des critères d'indice de fiabilité, d'importance et d'incertitude (Hermon et al., 2006).

Une fois la 4D intégrée, la phase de communication permet de valoriser ces créations en les intégrant dans des présentations, des visualisations interactives ou des supports de partage adaptés aux différents publics cibles.

1.2.4 Communication, mise en valeur

La 4D est souvent représentée sous forme de vidéos puisqu'elles permettent de visualiser l'évolution d'un site à travers différentes phases historiques. Des vidéos ont notamment été produites au cours des PFE de Benazzi (2018), Rocha (2022) et de Heitz (2023). Cependant, il existe d'autres média qui permettent de mettre en valeur une maquette 3D et de la transmettre à un public plus élargi. Heitz (2023) utilise le moteur de rendu Arnold d'Autodesk Maya, mais aussi Lumion pour des rendus photoréalistes et des vidéos des restitutions des phases historiques. Elle utilise également V-Ray qui est un moteur de rendu basé sur le lancer de rayons (*ay tracing* 1.3.1) et l'illumination globale, simulant précisément l'interaction de la lumière avec les surfaces pour générer des images réalistes. Ainsi, tandis que les

Etudiant	Fonctionnalité ajoutée	Description
Rigaud (2023)	UI Dialogues	Boîtes de dialogue pour améliorer la description des histoires du château
	Affichage du château en 4D	Possibilité d'afficher le château à différentes périodes historiques
Kourouma (2024)	Menu principal	Création d'un menu principal pour faciliter l'interaction avec le château
	Simulation de destruction et de reconstruction du château	Permet de simuler la destruction du château pour revenir à l'état actuel ainsi que la reconstruction pour revenir à l'état historique
	Outils de mesure	Ajout d'une tablette permettant de mesurer les distances et les surfaces
	Audio clip pour le défilement du texte	Utilisation de l'audio pour le défilement du texte des boites de dialogues
	Interactions et déplacement des objets	Ouvrir et fermer la porte du château et déplacement de briques

TABLE 1.3 – Tableau récapitulatif des différentes fonctionnalités développées dans le jeu VR, tiré de Kourouma (2024)

vidéos offrent une approche narrative et linéaire pour représenter l'évolution d'un site, d'autres technologies immersives, comme la Réalité Virtuelle, permettent d'engager les utilisateurs dans des expériences interactives et participatives.

Rigaud (2023) a exploité le Cube VR et a développé un jeu VR avec le moteur de Unity. Ce jeu permet non seulement de visualiser les différentes époques d'un modèle 3D, mais aussi d'interagir avec cette maquette et d'y être acteur offrant une véritable visite virtuelle, contrairement à la vidéo où l'on est que spectateur. Kourouma (2024) a ajouté des fonctionnalités au jeu qui utilise le langage C#. Toutes les fonctionnalités ajoutées sont listées dans le tableau 1.3.

Afin de compléter la modélisation et d'enrichir les données, il devient essentiel de s'intéresser aux outils immersifs, tels que la Réalité Virtuelle, qui permettent une exploration dynamique et interactive des modèles 3D, ouvrant ainsi de nouvelles perspectives pour la mise en valeur du patrimoine.

1.3 Réalité Virtuelle

1.3.1 Intérêt et applications

Les moteurs de jeux (*Game Engines*) fournissent l'infrastructure de base pour créer et contrôler les environnements virtuels. Ils incluent des moteurs de rendu, des moteurs audio, des moteurs physiques et des moteurs d'animation (Anderson et al., 2010). Ils offrent une base solide pour le développement de jeux et d'applications VR[1]. Ils permettent la création d'environnements virtuels complexes, la gestion des ressources et des interactions utilisateur. Les moteurs de jeux modernes prennent en charge le rendu en temps réel et des graphismes de haute qualité. Cependant, ces moteurs commerciaux peuvent être coûteux et les moteurs open source peuvent manquer de certaines fonctionnalités. Une des techniques essentielles pour le rendu graphique en temps réel dans les moteurs de jeux est la rastérisation, qui constitue la méthode principale utilisée pour transformer les modèles 3D en images visibles.

La rastérisation est une technique de rendu graphique où les objets sont dessinés à l'aide de triangles qui sont ensuite convertis en pixels. Elle consiste à appliquer des transformations de visualisation aux sommets des triangles, puis à remplir les pixels en utilisant la couleur et la texture. C'est la technique standard pour les jeux et les environnements interactifs puisqu'elle est rapide et efficace. Cependant, elle est moins exacte que la méthode de *ray tracing* quand il s'agit de simuler la réflexion ou la réfraction de la lumière (Anderson et al., 2010). Le *ray tracing* calcule la couleur de chaque pixel en simulant le trajet d'un rayon de lumière partant de l'œil du spectateur, traversant la scène et interagissant avec les objets. Toutefois, cette dernière technique est intensive en calculs et gourmand en ressources, mais les progrès récents des GPU et des CPU permettent de l'utiliser de plus en plus dans des applications en temps réel.

Après avoir abordé les techniques de rendu, il est pertinent de s'intéresser à l'application de la *gamification*, qui permet d'intégrer des dynamiques ludiques dans des contextes non-jeux, comme la réalité virtuelle, pour enrichir l'interaction et l'engagement des utilisateurs.

Le terme de *gamification* est l'introduction de stratégies et de composants de jeux dans des contextes qui ne sont pas des jeux. Theodoropoulos and Antoniou (2022) suggèrent que la *gamification* accroît la motivation et l'engagement dans une discipline. C'est un atout lorsque l'on souhaite enseigner quelque chose, ou simplement distribuer une information à un certain public. En effet, la VR permet une formation rapide et sans risques, l'acquisition de compétences pratiques, tout

1. Virtual Reality

en réduisant les coûts. Notamment dans le corps médical infirmier, elle est en plus envisagée comme une solution à la pénurie dans ce métier en Allemagne (Weiss et al., 2018). Cependant, les recherches actuelles sont limités et Weiss et al. (2018) appellent à réaliser des études plus approfondies sur des échantillons plus larges, afin de confirmer les bénéfices de la VR et de généraliser son utilisation.

Une fois que nous avons exploré les avantages et les diverses applications de la réalité virtuelle, il est essentiel de se pencher plus spécifiquement sur les environnements immersifs mobiles et les simulations virtuelles, qui permettent d'étendre ces possibilités dans des contextes dynamiques et interactifs.

1.3.2 Les environnements immersifs mobiles et autres simulations virtuelles

Weiss et al. (2018) soulignent que les casques VR (HMD[2]) sont actuellement les seuls dispositifs capables d'offrir une immersion complète, puisqu'ils reposent sur leur aptitude à isoler les utilisateurs du monde réel, combinée à des interactions visuelles, auditives, et parfois haptiques. Ces caractéristiques renforceraient le réalisme et l'engagement. Milgram et al. (1995) classent les environnements de réalité mixte (combinant VR et AR[3]) en 7 catégories. Les casques VR et le Cube VR appartiendraient à la classe 5. En plus de la VR, il existe d'autres types d'environnements immersifs comme l'AR ou la MR.

La réalité augmentée (AR[4]) consiste à superposer des éléments virtuels au monde réel. Cette technologie utilise des smartphones ou des lunettes spéciales pour ajouter des informations contextuelles et des détails supplémentaires dans le champ de vision de l'utilisateur. En AR, le monde réel reste prédominant. La réalité mixte (MR[5]) combine des éléments du monde réel et du monde virtuel pour créer un environnement hybride où les objets réels et virtuels peuvent interagir en temps réel. Elle offre une expérience plus immersive et réaliste que la VR ou l'AR seules et tend à brouiller les frontières entre le monde réel et le monde virtuel (Siddiqui et al., 2022). Maintenant, citons quelques exemples de casques mobiles.

Les HMD mobiles qualifient généralement de simples boîtiers avec des lentilles où il faut insérer un smartphone pour être utilisé comme casque VR. Le premier appareil de ce genre est le Google Cardboard, qui ne permet même pas de l'attacher autour de la tête, mais a l'avantage d'être très peu coûteux et d'être fonctionnel

2. *Head Mounted Display*
3. *Augmented Reality*
4. *Augmented Reality*
5. *Mixed Reality*

FIGURE 1.1 – Les HMD fixes populaires : l'Oculus Rift (gauche), le HTC Vive (centre) et le PlayStation VR (droite)

pour des applications VR simples. Samsung et Oculus ont développé un HMD mobile appelé le GearVR. Une manette sans fils peut être utilisé avec le GearVR et permet une expérience VR interactive mobile. Ces appareils dépendent totalement de la qualité du smartphone employé. Des systèmes comme le Gameface Mark IV ou le Auravisor sont construits avec un environnement Android, ce qui permet une expérience indépendante et toujours mobile. Ces HMD mobiles ont l'avantage d'être peu coûteux, mais peuvent manquer de puissance de calcul pour des applications plus poussées (Anthes et al., 2016).

Alors que les HMD mobiles favorisent une expérience portable et accessible, les environnements immersifs fixes permettent une immersion plus poussée et des interactions plus complexes, souvent grâce à des dispositifs plus puissants et adaptés à des usages spécialisés.

1.3.3 Les environnements immersifs fixes

Les HMD fixes sont généralement plus performants que les HMD mobiles et sont équipés de capteurs supplémentaires en plus de leur suivi optique. Ces capteurs incluent des accéléromètres, des magnétomètres et des gyroscopes. Les trois principaux concurrents (figure 1.1) sont l'Oculus Rift qui est largement utilisé à la fois pour le développement amateur et la recherche, le PlayStation VR qui utilise des lentilles asphériques pour réduire la distorsion au centre, et le HTC Vive qui est conçu pour une utilisation à l'échelle de la pièce (5x5 mètres), ce qui ouvre la voie à différentes applications. Il existe d'autres casques avec des fonctionnalités qui leurs sont propres, comme le StarVR qui offre un grand champ de vue (FOV[6] avec des écrans de 2560x1440 pixels par œil et donc un champ de vue de 210x130 degrés. Des casques comme le VRvana Totem ou le Meta Quest sont équipés de caméras pour pouvoir visualiser son environnement, de tels appareils sont donc qualifiés de casques MR. L'Avegant Glyph est équipée d'une centrale inertielle 6DoF et de micro-miroirs pour une projection des images dans l'œil (Anthes et al., 2016).

6. *Field Of View*

Tredinnick et al. (2017) ont développé le plugin Uni-CAVE pour Unity3D, qui simplifie le développement d'applications pour des systèmes immersifs CAVE (*Cave Automatic Virtual Environment*) comme le Cube VR, en automatisant la configuration multi-écrans (projection stéréo). Il fournit des outils prêts à l'emploi pour synchroniser les perspectives des utilisateurs, intégrer des interactions immersives, et coordonner les affichages projetés. L'extension Unity développée par Sigitov et al. (2015) permet de modéliser la disposition des écrans dans un espace 3D via un outil visuel, facilitant l'alignement et les perspectives. Elle est surtout adaptée pour les écrans haute résolution et multi affichages (LHRD[7]), testée sur leur système à 35 écrans HORNET.

Une fois que nous avons exploré les applications de la réalité virtuelle dans différents domaines, il est intéressant de se concentrer sur son rôle spécifique dans la préservation du patrimoine, où cette technologie offre des possibilités uniques pour la conservation et la transmission de sites historiques.

1.4 La VR au service de la préservation du patrimoine

Cette section présente l'impact de la réalité virtuelle sur la préservation du patrimoine, en soulignant l'importance de l'immersion pour la recréation, la restauration et la diffusion des sites historiques de manière innovante et interactive.

1.4.1 Préservation du patrimoine

Les musées virtuels utilisent la VR pour créer des expositions en ligne ou sur site permettant aux visiteurs d'explorer des artefacts et des sites du patrimoine de manière interactive (Anderson et al., 2010). Les avantages majeurs sont la préservation du patrimoine historique et l'accès à la culture par le monde entier. Le jeu *Gates of Horus* (figure 1.2) présente un temple égyptien hypothétique où le but est d'explorer la culture de l'Égypte antique en répondant à des questions posées par un prêtre. Ce jeu développé sur le moteur Unreal Engine est également adapté pour un environnement CAVE (Jacobson et al., 2010).

D'autres jeux sont basés sur des sites réels et permettent de les visiter virtuellement. Dhanda et al. (2019) ont modélisé l'intégralité du temple du XIe siècle Myin-pya-gu, à Bagan en Birmanie. Ils ont créé une chaîne de traitement utilisant la photogrammétrie (*Structure from Motion*) et ont ensuite développé un jeu avec Unreal Engine.

7. *Large High-Resolution Displays*

FIGURE 1.2 – Extrait du jeu *Gates of Horus* de Jacobson (2011), source : publicvr.info

Ainsi, en abordant la préservation du patrimoine, il devient essentiel de souligner le rôle primordial de l'immersion, qui permet une expérience plus profonde et réaliste des sites et objets historiques.

1.4.2 L'importance de l'immersion

Le jeu de Jacobson et al. (2010) illustrent la *gamification* en intégrant un système de récompenses et un moyen de gagner en répondant correctement aux questions posées. La narration introduite permet à l'utilisateur d'aborder et d'apprendre la culture de l'Égypte antique de manière ludique.

Dhanda et al. (2019) soulignent les différents degrés d'immersion. Un système à 6 degrés de libertés (6DoF) où les mouvements de la tête et du corps sont suivis offre une meilleure immersion comparé à un système à 3 degrés de libertés (3DoF) où seuls les mouvements de la tête sont suivis. Ils utilisent également une méthode de rendu basé sur la physique (PBR[8]) qui simule de manière réaliste la manière dont la lumière interagit avec les matériaux et les surfaces. Cette méthode de rendu créé un éclairage plus précis, réaliste et donc plus immersif.

Hutson (2024) souligne le rôle important des LLM[9] pour la préservation du patrimoine. Ils peuvent aider à restaurer et préserver des langues perdues ou en danger de disparition. Des projets comme l'*Endangered Languages Project* protègent le patrimoine linguistique grâce à la participation du grand public.

8. *Physically Based Rendering*
9. *Large Language Models*

Ces enjeux de préservation et d'immersion posent alors la question des solutions techniques mobilisées pour concevoir et naviguer dans ces univers virtuels, thème que nous abordons à présent dans la section suivante.

1.5 Différents outils pour le développement VR

Dans cette section, nous présenterons d'abord les principales solutions de navigation en VR (graphe de *waypoints* et *NavMesh*), puis nous aborderons le concept plus large de *pathfinding* et ses principaux algorithmes.

1.5.1 Navigation en VR : *NavMesh* et graphe de *waypoints*

Le graphe de *waypoints* relie des points clés pour guider les déplacements dans des environnements VR ciblés. Au lieu de couvrir toute la zone, il se concentre sur des passages pertinents et simplifie la gestion du parcours. Cette méthode reste toutefois moins flexible qu'un *NavMesh*, car les entités suivent surtout les connexions entre *waypoints*. Elle se révèle néanmoins pratique pour des niveaux linéaires ou des espaces limités où la simplicité prime (Brewer, 2019).

Le *NavMesh*, plus complexe et coûteux, segmente les espaces VR en polygones triangulés pour une navigation précise. Il réduit le nombre de points à explorer et peut inclure des annotations tactiques, mais sa génération et sa mise à jour sont délicates, surtout dans un environnement dynamique. Cette complexité est justifiée lorsque la fluidité et la précision du déplacement sont prioritaires. De plus, il s'adapte mieux aux scénarios exigeant un réalisme poussé ou des IA très autonomes (Brewer, 2019).

Ces deux approches de navigation s'inscrivent dans le cadre plus général du *pathfinding*, un domaine incontournable de l'intelligence artificielle qui comprend notamment les algorithmes de Dijkstra, *Best-First* et A*.

1.5.2 Concept de *Pathfinding*

La recherche de chemin (*pathfinding*) est un problème de l'intelligence artificielle et de recherche de solution. Elle consiste à trouver comment se déplacer dans un environnement entre un point de départ et un point d'arrivée en prenant en compte différentes contraintes comme des obstacles. Une multitude d'algorithmes ont été développé et ont tous leurs avantages et inconvénients, nous n'en présenterons que trois principaux.

L'algorithme de Dijkstra est un algorithme de plus court chemin, développé par Edsger Dijkstra en 1959. Il explore un graphe en partant du point initial, en examinant à chaque étape le nœud le plus proche qui n'a pas encore été traité. Il s'étend progressivement jusqu'à atteindre l'objectif. L'algorithme garantit toujours un chemin le plus court en explorant , mais il est lent car il examine systématiquement chaque option (Cormen et al., 2022).

L'algorithme de recherche *Best-First* utilise une estimation "prometteuse" (heuristique) pour évaluer la distance de chaque nœud à l'objectif. Il privilégie les nœuds les plus proches de la destination plutôt que ceux proches du point de départ. Cet algorithme est rapide, mais ne garantit pas de trouver le chemin le plus optimal. Certains auteurs utilisent le terme *Greedy Best-First* (Glouton) pour désigner spécifiquement une recherche qui vise à explorer en priorité le chemin le plus proche de la solution (Pearl, 1984).

L'algorithme A* combine les forces de Dijkstra (exploration systématique) et de *Greedy Best-First-Search* (utilisation d'heuristiques). Il garantit un chemin optimal tout en étant plus rapide que Dijkstra dans de nombreux cas (Amit, 2024). A* utilise une fonction de coût :

$$f(n) = g(n) + h(n) \tag{1.1}$$

où pour un nœud n :

- g(n) est le coût exact pour atteindre un nœud donné (Dijkstra).
- h(n) est l'estimation du coût heuristique pour aller de ce nœud à la destination (*Best-First*).

En somme, cette diversité d'outils et d'approches pour la navigation et la conception d'environnements virtuels souligne l'importance de choisir des solutions adaptées aux besoins spécifiques de chaque projet, qu'il s'agisse de réalisme, d'immersion ou de performances.

1.6 Conclusion de l'état de l'art

Cet état de l'art a permis de dresser un panorama complet des technologies et méthodologies impliquées dans la création d'expériences de visite virtuelle patrimoniale. La chaîne de traitement débute par l'acquisition précise de données, constituant ainsi une base solide pour la modélisation numérique.

La modélisation 3D s'appuie sur trois approches principales : la modélisation par primitives géométriques, particulièrement adaptée pour la reconstitution d'éléments disparus ; la modélisation par maillage, privilégiée pour représenter

fidèlement l'existant ; et l'approche hybride qui combine les avantages des deux méthodes précédentes. Le texturage, à travers la cartographie UV et l'utilisation de *shaders*, permet ensuite d'enrichir ces modèles avec des détails visuels réalistes. L'intégration de la dimension temporelle (4D) apporte une profondeur historique essentielle, permettant de visualiser l'évolution des sites à travers différentes époques, tout en prenant en compte les niveaux d'incertitude des restitutions.
La Réalité Virtuelle, qu'elle soit mise en œuvre via des casques mobiles ou des systèmes fixes, offre de nouvelles possibilités pour la préservation et la transmission du patrimoine. Les moteurs de jeux comme Unity fournissent l'infrastructure nécessaire pour créer des expériences interactives, tandis que les techniques de rendu comme la rastérisation et le *ray tracing* permettent d'obtenir un rendu visuel de qualité. La *gamification* enrichit ces expériences en introduisant des éléments ludiques qui favorisent l'engagement et l'apprentissage des utilisateurs. Les outils de navigation, tels que le graphe de *waypoints* et le *NavMesh*, et des algorithmes de *pathfinding* permettent de créer des déplacements intelligents et immersifs dans les environnements virtuels. Chaque méthode répond à des besoins spécifiques en matière de simplicité ou de précision. L'intégration de ces technologies dans les systèmes VR garantit une expérience utilisateur engageante, immersive et optimisée, tout en ouvrant la voie à des applications plus avancées dans le futur.

L'ensemble de ces technologies et méthodologies ouvre ainsi de nouvelles perspectives pour la conservation et la médiation du patrimoine historique. Elles permettent non seulement de préserver numériquement les sites historiques, mais aussi de les rendre plus accessibles et interactifs pour différents publics, tout en offrant des possibilités innovantes pour l'étude et la compréhension de notre histoire. Les développements futurs dans ces domaines, notamment l'amélioration des techniques de rendu et l'intégration de l'intelligence artificielle, promettent d'enrichir encore davantage ces expériences de visite virtuelle.

CHAPITRE 2

DOCUMENTATION ET PÉRENNISATION DES PROJETS DU CUBE VR

Le premier objectif de ce projet est l'implémentation de nouvelles fonctionnalités afin de rendre les visites virtuelles d'avantage immersives et ludiques. Toutefois, ce PRT ne signe pas la fin du projet Château Rhénans, bien au contraire. Afin de rendre le développement du jeu VR par d'autres éditeurs plus rapide et le plus complet possible et également avec la volonté du partage des connaissances plus généralement, le second objectif se présente comme la documentation et l'explication des fonctionnalités ajoutés et des outils utilisés.
Ce chapitre explore les actions menées afin de documenter le projet et le rendre plus facile à reprendre. Nous commencerons par la rédaction de protocoles et de guides pour utiliser et gérer le Cube VR. Ensuite, nous aborderons le projet INSA 2025 et son rôle dans le partage des connaissances en VR. Enfin, nous nous pencherons sur l'environnement spécifique du Cube VR de Virtuel Concept, avec un focus sur les outils fournis et les difficultés rencontrées.

2.1 Protocole d'utilisation et guides pratiques pour le Cube VR

Kourouma (2024) a rédigé un *Protocole d'allumage , de calibration et d'extinction du cube de réalité virtuelle de l'INSA-Strasbourg* décrivant exhaustivement chaque étape dans le but d'utiliser le Cube et qui est très utile pour un utilisateur non initié. Cependant, le guide est très protocolaire pour arriver au lancement du Cube et il manque quelques explications sur le fonctionnement de l'environnement. En effet, j'ai rencontré certaines difficultés lors des premières utilisations malgré la lecture du protocole. Les principaux points d'ombre sont l'importance des quatre trackers disposés dans les coins afin de suivre les mouvements des lunettes et des manettes, ainsi que des problèmes de configuration une fois un jeu lancé. En effet,

arrivé dans le jeu, le mode de déplacement (téléportation, vol ou marche) peut être changé et la direction des joysticks peut être inversé, ce qui est perturbant si c'est le cas. Nous avons donc complété ce protocole pour répondre à d'avantage de problèmes qu'il est possible de rencontrer en utilisant le Cube.

De manière similaire, j'ai rédigé des protocoles d'utilisation ou d'explications en lien avec le Cube VR ou le développement du jeu Unity, dont certaines sont disponible en annexes.
Toujours dans le cadre de la rédaction de protocoles et guides, nous verrons dans la partie suivante comment le projet INSA 2025 participe au partage des connaissances, qui est un enjeu de ce projet.

2.2 Collaboration via le projet INSA 2025

Pendant ce projet, nous avons été contacté par du personnel chargé de travailler sur le projet INSA 2025 qui est une initiative du Groupe INSA visant à transformer numériquement le parcours des apprenants, au travers du service OpenINSA. Il s'agit d'un service inter-établissements qui vise à promouvoir une innovation pédagogique en accompagnant les enseignants dans la réalisation d'outils et de modules numériques de formation. Un des objectifs de ce projet est le partage des connaissances au sein des INSA, notamment dans le domaine de la VR. En effet, l'INSA Hauts-de-France et l'INSA Toulouse disposent également de cubes immersifs et l'INSA Lyon a aménagé une pièce équipée de casques VR. Le partage des informations sur la VR est donc un enjeu de l'INSA afin de faire évoluer les formations de manière numérique. Dans ce cadre, nous nous sommes réunis quelques fois pour discuter de ce PRT, et notamment de l'outil Scenari Dokiel. C'est une plateforme qui permet de concevoir, stocker et partager des documentations et des supports de formation. Dans notre cas, il permet de rédiger les protocoles liées au Cube. Nous avons jugé l'objectif d'alimenter cette plateforme comme un objectif secondaire par manque de temps (puisque les documents sur Dokiel suivent une symbologie très particulière et nécessite un certain temps d'apprentissage, un peu comme LaTeX) et nous n'avons pas jugé la portée de ce PRT de la même envergure que le projet INSA 2025 puisque la documentation générée a un but pratique en interne. Dans le cas où il est totalement pertinent de transmettre cette documentation, il sera toujours possible de les traduire sur Dokiel.

Toutefois, nous avons pu jouer et nous inspirer des jeux de démonstration du Cube VR créés par l'INSA Toulouse que nous remercions (élaboré dans la partie 3.2). Revenons à présent aux particularités qui caractérisent le Cube VR et les fonctionnalités développées qui lui sont propres.

2.3 Outils et spécificités du Cube VR (Virtuel Concept)

La lecture attentive du *Protocole d'allumage , de calibration et d'extinction du cube de réalité virtuelle de l'INSA-Strasbourg* s'avère très efficace pour comprendre l'environnement physique du Cube. Toutefois, un nouvel éditeur du jeu doit se familiariser, si ce n'est pas déjà le cas, avec Unity et en particulier avec l'environnement spécial de DEC, développé uniquement pour le Cube. Dans cette partie, nous passerons en revue quelques outils (*assets*) principaux fournis par DEC, ce qui les rend particuliers et les problèmes rencontrés. DEC est le SDK[1] développé par Virtuel Concept (distributeur du Cube) qui s'apparente à une version modifiée de l'éditeur Unity.

2.3.1 Prefabs essentiels du SDK DEC

L'arborescence de Unity est générale mais peut porter à confusion puisqu'il implique une certaine nomenclature (voir figure 2.1). Le SDK DEC est un *template* de Unity qui est en quelque sorte une version modifiée de Unity. Il contient des *packages*, contenant des *assets*. Ces *assets* peuvent être des scènes, des *prefabs*, par exemple. Une scène est l'environnement dans lequel un utilisateur joue et un *prefab* est un groupe d'objets du jeu. Chaque objet contient des composants paramétrable qui définit la nature et le comportement de l'objet. Notons qu'il est possible d'associer un objet dans un autre objet, et un *asset* (script) dans comme un composant. Nous nous intéresseront plus particulièrement aux *prefabs*.

Un *prefab* est un *asset* contenant des objets (*GameObjects*) entièrement configurés avec des composants (comme un modèle 3D, une texture, un fichier audio ou une animation) paramétrés par le créateur du *prefab* et possiblement modifiable par toute personne qui l'utilisera. Ces *assets* peuvent ensuite être partagés entre les scènes ou même d'autres projets, sans avoir à être configurés de nouveau. Chacun peut créer un *prefab* en exportant un ou plusieurs objets dans les *assets* du jeu.

Nous avons utilisé certains *prefabs* disponibles par DEC (présentés dans le tableau 2.1) très utiles puisqu'ils sont déjà adaptés à l'environnement de Virtuel Concept et quasiment prêt à l'emploi. Par exemple, le *XRPlayer* est l'objet incarné par l'utilisateur dans le jeu. Il met en relation un réseau d'objets complexe, tels que l'avatar (enveloppe physique du personnage), la caméra (sans elle, le joueur de voit rien), le menu et qui est personnalisable avec des boutons cliquables et il fait appel à de nombreux scripts dans chacun des éléments. Tout est développé et

1. *Software Development Kit*

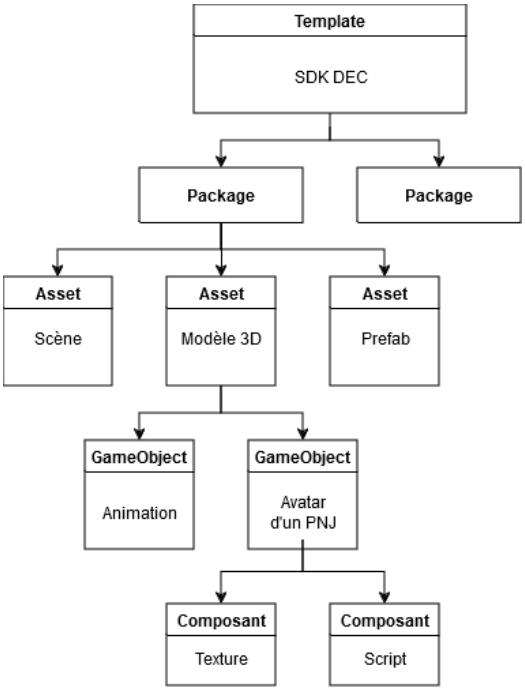

FIGURE 2.1 – Arborescence des différents éléments d'un jeu Unity

configuré au sein de ce *prefab* pour fonctionner dans le Cube.
Un UIDialog est une boîte de dialogues interactive et visible dans le jeu. Ce *prefab* est moins complexe que celui du *XRPlayer*, mais permet tout de même d'ajouter de l'information textuelle plus rapidement qu'en ajoutant du texte en partant d'un canva vierge.
Rigaud (2023) parle de ces *assets*, mais également des scripts *Floor* et *Wall*. Ceux-ci sont des composants à ajouter aux modèles 3D afin de les rendre tangibles vis-à-vis du *XRPlayer*. Toutefois, la documentation manque pour décrire ces scripts en détail (explications en section 2.3.2).
Nous avons également reprit certaines fonctionnalités du *prefab Tablet* (implémenté par Kourouma (2024)) afin de créer une nouvelle version du menu. Cette tablette est un objet physique contenant des boutons avec lequel on peut interagir pour utiliser quelques fonctions telles que mesurer des distance, ou prendre des photos.

Maintenant que nous avons introduit le SDK de DEC et ses outils mis à disposition, discutons aux difficultés rencontrées liées à cet environnement.

Asset DEC utilisé	Description
XRPlayer	Avatar incarné par le joueur. Contient la caméra de vue du jeu, le menu et permet les interactions avec les manettes.
UIDialog	Boîte de dialogue permettant d'afficher de l'information textuelle, de manière interactive avec le joueur.
Script Floor	Script permettant au joueur de se déplacer sur le sol d'un modèle 3D tangible.
Script Wall	Script ne permettant pas au joueur de traverser un modèle 3D qui s'apparente à un mur.
Tablet	Objet apportant de nombreuses fonctionnalités personnalisable, de manière élégante (Kourouma, 2024).

TABLE 2.1 – *Assets* DEC utilisés dans le projet

2.3.2 Difficultés techniques et limites de l'environnement DEC

DEC fournis une multitude de *prefabs* complexes et prêt à l'emploi, ce qui rend le développement de jeu bien plus facile, mais il est donc difficile de les modifier, ou de n'utiliser qu'une partie d'un *prefab*, puisqu'il faut l'examiner entièrement et ces *prefabs* font souvent appel à d'autres objets ou scripts. De plus, l'intégralité des scripts développé par DEC sont compilés. C'est-à-dire que le langage C# est converti en un langage ordinateur que seul la machine peut lire. L'avantage est que l'exécution est plus rapide et plus compacte, mais il n'est plus lisible pour l'Humain. Cette particularité fait que le développement de fonctionnalités et scripts doit généralement être indépendant des fonctionnalités de DEC. En effet, si nous voulons faire appel aux fonctions ou aux variables existantes, il faut connaître leur nom exacte, mais cette information est généralement indisponible. Dans le cas échéant, le script que nous souhaitons ajouter peut entrer en conflit avec d'autres scripts existants. Comme dit Rigaud (2023), aucune documentation à ce sujet n'existe, hormis des vidéos d'instruction pour implémenter des *prefabs* DEC, mais ces explications restent incomplètes.

Par exemple, pour la création d'un meilleur menu (évoquée plus tard dans la section 3.2), nous avons essayé de rédiger un script permettant au menu d'apparaître et de disparaître en appuyant sur un bouton de la manette, ou encore de le rendre accessible en le faisant suivre le joueur. Il existe un fichier lisible (pas de la documentation) qui récapitule les appels aux différents boutons, puisque l'envi-

ronnement est particulier (différent du clavier et de la souris ou de manettes HTC Vive, par exemple). Toutefois, nous n'avons pas réussi à faire appel à ces variables. De plus, nous avions l'idée de modifier le système de déplacement du joueur. Dans les jeux vidéos, il est courant d'utiliser des déplacements latéraux (visuellement comme un crabe), ce qui a l'avantage de réduire l'impression de nausée que peut donner un jeu VR lorsqu'on tourne la tête. Cependant, il semble que nos essais de scripts entrent en conflit avec le composant *RigidBody* du *XRPlayer*. Pour les raisons citées précédemment, nous n'avons pas pu étudier l'erreur et la corriger.

En plus des difficultés logiciels, j'ai rencontré des difficultés liées au Cube en lui même. Il s'agit d'une technologie innovante qui a ses avantages intéressants, comme la possibilité de partager une expérience VR avec plusieurs utilisateurs en même temps, mais c'est aussi un environnement fixe non-transportable. J'ai tout de même réussi à installer la même version d'Unity sur mon ordinateur personnel et à utiliser le template de DEC, mais sans les manettes, impossible de tester toutes les fonctionnalités.

La documentation mise en place dans ce projet vise à faciliter la prise en main et la poursuite du développement du Cube VR. Les protocoles détaillés, le partage de connaissances avec le projet INSA 2025, et l'analyse de l'environnement DEC offrent une base solide pour les futurs utilisateurs et développeurs. Malgré quelques limitations et difficultés rencontrées, ces efforts renforcent la pérennité et l'accessibilité du projet. Nous proposons de contacter Virtuel Concept afin d'avoir plus de détails sur le fonctionnement de leur produit, si celui-ci n'est pas confidentiel. Dans le prochain chapitre, nous explorerons l'étendue des fonctionnalités et interactions ajoutés dans le cadre de ce PRT, dans le but d'améliorer l'immersion au sein du jeu VR.

CHAPITRE 3

AJOUT DE FONCTIONNALITÉS INTERACTIVES POUR ENRICHIR L'IMMERSION

Ce chapitre se concentre sur les fonctionnalités et interactions développées pour enrichir l'immersion au sein du Cube VR. L'objectif principal est de rendre l'expérience plus interactive et engageante pour les utilisateurs, en combinant des éléments visuels, des animations, et des interactions dynamiques.
Nous débuterons par la création et la gestion des Personnages Non Joueurs (PNJ), ensuite, nous présenterons la conception d'un menu boîte à outils amélioré et enfin, nous aborderons les modifications apportées à la scène du menu principal.

3.1 Création et gestion des PNJ

La création et gestion des Personnages Non Joueurs (PNJ) vise à ajouter du dynamisme et de l'interaction dans le Cube VR. Cette partie aborde leur personnalisation visuelle et animée, leur navigation intelligente dans les environnements 3D, ainsi que l'intégration de dialogues interactifs pour enrichir l'expérience utilisateur.

3.1.1 Personnalisation des textures et animations des PNJ

Unity propose déjà une librairie de modèles de personnages, mais nous avons décidé d'utiliser une autre plateforme pour plus de diversité. Mixamo est un site developpé par Adobe qui propose un vaste choix de modèles et d'animations, le tout gratuitement. Il est très facile d'y télécharger les bases visuelles pour créer un PNJ. C'est donc sur Mixamo que provient toutes les textures des PNJ que nous avons utilisé dans ce projet.

En plus des modèles de personnages, chaque animation est adaptée et utilisable pour tout modèle. Nous avons également utilisé leurs animations telles que les animations de marche et stationnaires. Il est possible de télécharger les fichiers

avec diverses extensions, celle qui nous intéresse est le FBX. Un fichier FBX contient le modèle, l'autre l'animation. Un tel fichier est lu par Unity comme un *prefab* contenant les différentes parties d'un personnage. Pour importer un personnage sur une maquette, il faut importer tous les fichiers dans le projet Unity, modifier le *prefab* du modèle dans l'éditeur en reliant toutes les textures du fichier. Pour l'animation, il faut créer un *AnimationController* qui lui, relie le personnage à une ou plusieurs animations, puis lui associer cet *AnimationController*.

Maintenant, l'animation de marche est un ajout purement visuel, c'est-à-dire qu'un PNJ avec l'animation de marche va marcher sur place, il faut développer son déplacement.

3.1.2 Navigation intelligente des *Agents*

Il est possible de rendre une scène encore plus vivante en ajoutant du mouvement. Les personnages mobiles, aussi appelés *Agents*, peuvent se déplacer de manière intelligente au sein d'une maquette. C'est-à-dire qu'ils utilisent des algorithmes de recherches de chemins et des algorithmes de déplacement tout en évitant les collisions avec des obstacles. Cette partie explore la mise en place de la navigation intelligente des *Agents*, en commençant par l'utilisation des surfaces de navigation telles que le *NavMesh* et le *HeightMesh*, pour ensuite aborder l'automatisation des déplacements et la gestion des interactions dynamiques.

3.1.2.1 Déplacements réalistes : *Navigation Mesh* et *Height Mesh*

Tout d'abord, un *Agent* ne peut utiliser la même surface de navigation que le joueur (script *Floor* 2.1), il a besoin de sa propre zone praticable. Cette zone réunit les endroits sur lesquels un *Agent* est capable de se tenir debout et se déplacer. Elle est construite automatiquement en testant si la géométrie de la scène est praticable, en fonction de la taille de l'*Agent*. Une nouvelle surface est générée en sortie et est couramment appelée *Navigation Mesh* (*NavMesh*). Nous avons choisi d'utiliser la méthode de *NavMesh* plutôt que celle de graphe de *waypoints* (voir partie 1.5.1) puisque nous disposons d'un MNT qui reflète la réalité et nous ne souhaitons pas que les *Agents* se déplacent de manière préméditée et systématique. Le *NavMesh* permet une navigation fluide et réaliste qui couvre l'ensemble du modèle 3D de surface.

Cette nouvelle surface est un ensemble de polygones convexes et son algorithme de création utilise la voxélisation. Il commence par rastériser (voir partie 1.3.1) la scène en voxels, puis il extrait les surfaces praticables et enfin, les transforme en *NavMesh*. Les voxels peuvent être perçus comme des pixels 3D (volumetric pixel).

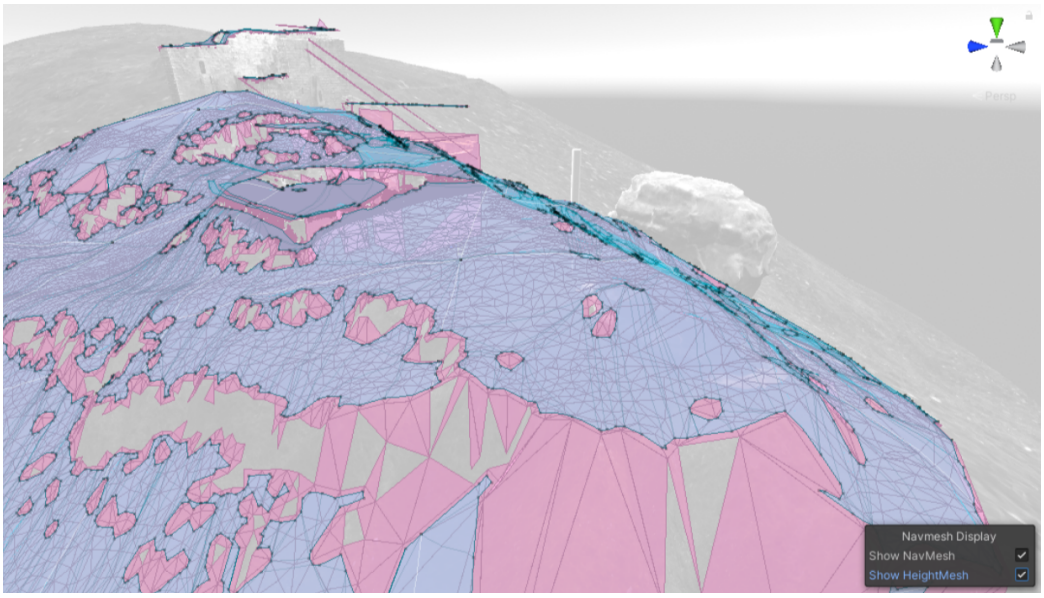

FIGURE 3.1 – Rendu final des *NavMesh* (bleu) et *HeightMesh* (rose) sur le château du Wassenbourg

Il est possible de choisir la taille des voxels voulus, plus ils sont petits, plus ils vont épouser la surface d'origine. Par défaut, la taille du voxel est de 0,167 m, ce qui correspond au tiers du rayon d'un *Agent* (0,5 m), on dit alors que la taille est de 3 voxels par rayon d'*Agent*. D'après la documentation de Unity, plus de 8 voxels par rayon n'apportent généralement pas beaucoup d'avantages supplémentaires (Unity, 2024). Nous verrons par la suite le paramétrage du *NavMesh* généré.

En affinant les paramètres et surtout en réduisant la taille des voxels, nous pouvons rencontrer des problèmes comme la formation de trous dans le *NavMesh* (voir figure 3.2). C'est pour cela que pour le château du Wassenbourg, après de nombreux tests, nous trouvons les paramètres optimaux suivant :

- *Max Slope* = 45
- *Step height* = 1,2m
- *Voxel size* = 0,125m = 4 voxels par rayon d'*Agent*
- Les autres paramètres sont par défauts.

Puisque les mailles du *NavMesh* ne sont pas infiniment petites, ce dernier est une approximation de la surface du MNT ou du château. Par exemple, des escaliers peuvent ressembler à une pente pour un *Agent*. Pour plus d'exactitude, nous avons ajouté un *HeightMesh* qui est une surface qui gère mieux les pentes et les marches. C'est-à-dire que l'*Agent* gravira toutes les marches d'un escalier, au lieu de "glisser" jusqu'en haut (voir figure 3.1.

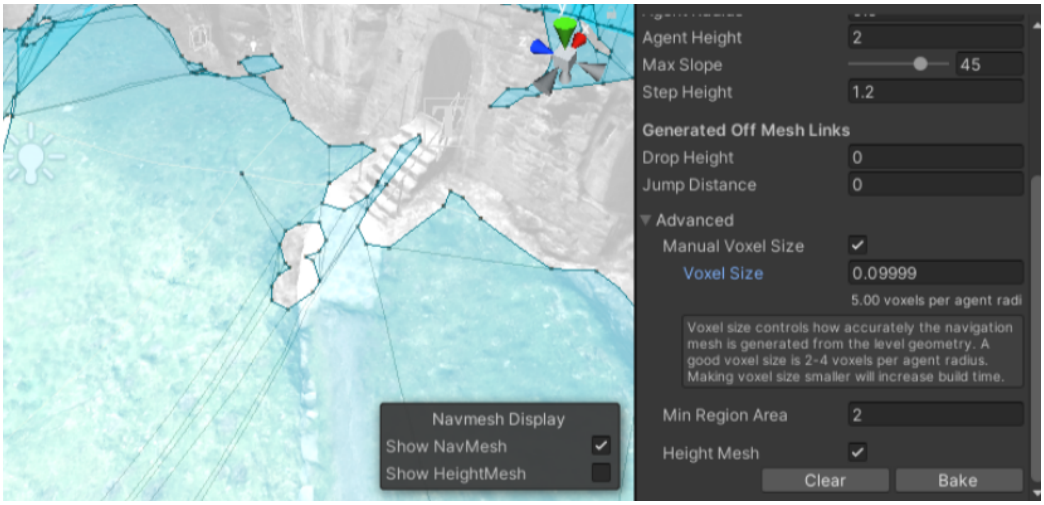

FIGURE 3.2 – *NavMesh* troué, avec 5 voxels par rayon d'*Agent*

Après avoir configuré le *NavMesh* et le *HeightMesh*, il est nécessaire de définir des mécanismes permettant aux agents de se déplacer de manière autonome et aléatoire dans l'environnement.

3.1.2.2 Automatisation des déplacements aléatoires

Le déplacement d'un *Agent* se déroule en deux grandes étapes : trouver la destination et un chemin qui concerne la maquette globale de manière statique, et comment s'y déplacer qui est une étape locale et dynamique. La recherche de chemins optimaux (*pathfinding*) permet de planifier les déplacements dans un environnement tout en évitant des obstacles. Sans *pathfinding*, les algorithmes de mouvement simples peuvent bloquer un *Agent* dans des impasses, puisque ces algorithmes ne considèrent pas les obstacles.

Pour le *pathfinding*, Unity utilise un algorithme A* qui a l'avantage d'être rapide et de trouver le chemin le plus court, à la manière des algorithmes *Best-First* et de Dijkstra, explicités dans la partie 1.5.2. Dans le cas de Unity, le chemin en sortie est une suite de polygones du *NavMesh*, appelée *Corridor*. L'*Agent* atteint la destination en se dirigeant toujours vers la maille visible suivant du corridor. Avec plusieurs *Agents*, il faut tenir compte des écarts par rapport au chemin pour éviter les collisions, ce qui nécessite une correction locale du *Corridor*. Unity utilise un algorithme d'obstacles de vitesse réciproques (RVO) pour prévenir les collisions. C'est-à-dire qu'une nouvelle vitesse et une nouvelle direction sont calculées pour atteindre la prochaine maille, tout en évitant la collision avec un *Agent* (Unity, 2024).

La navigation globale est utilisée pour trouver le *Corridor* dans une maquette 3D.

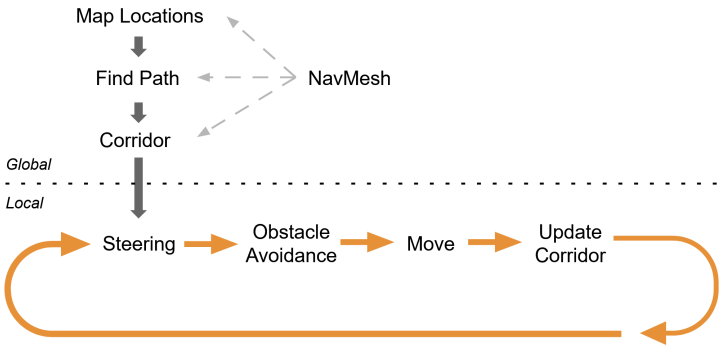

FIGURE 3.3 – Diagramme de la navigation des *Agents* dans Unity (Unity, 2024)

Ce dernier est une structure flexible et peut être ajustée localement pour éviter les collisions avec d'autres *Agents* ou objets en mouvement (Unity, 2024). La différence entre navigation globale et locale est décrite visuellement sur la figure 3.3. Enfin, il est possible de créer des raccourcis dans la navigation en autorisant les sauts entre les obstacles grâce au *NavMesh Link*. Toutefois, nous n'avons pas jugé cette option comme pertinente pour le projet puisqu'elle aurait complexifié la navigation des *Agents* sans offrir un avantage significatif dans un environnement où les déplacements restent principalement sur le MNT ou quelques surface du château.

Une fois la navigation possible, il faut définir une destination vers laquelle un PNJ se rendra. Il est possible de créer un point défini comme destination, ou même de créer une ligne de déplacement suivie par un *Agent* en boucle, un peu comme une ronde de garde (utilisé par la méthode du graphe de *waypoints*). Toutefois, le processus est chronophage et unique pour chaque *Agent*. La solution choisie est d'écrire un script ordonnant au PNJ de se déplacer en créant des points de destinations de manière aléatoire et infini. Ce procédé mettra en mouvement l'*Agent* dans la maquette et y apportera de la vie, en quelque sorte. Pour cela, nous avons reprit le script C # de "JonDevTutorial" (2022) depuis GitHub, qui est explicité par la figure 3.4.

La navigation intelligente des agents, grâce au *NavMesh* et à l'automatisation des déplacements, apporte une dimension de réalisme et de fluidité aux interactions dans l'environnement 3D, rendant les scènes plus vivantes et immersives. Pour compléter cette dynamique, l'ajout de dialogues interactifs permet d'enrichir encore davantage l'interaction entre les utilisateurs et les PNJ, renforçant l'engagement et l'immersion dans l'expérience VR.

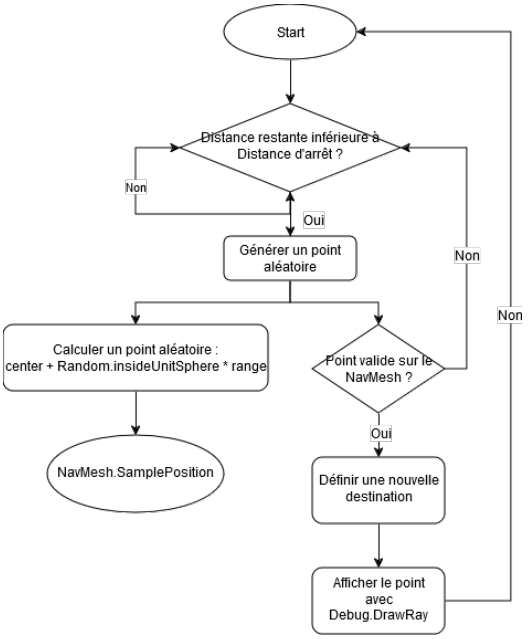

FIGURE 3.4 – Diagramme du script C# *RandomMovement* de "JonDevTutorial" (2022)

3.1.3 Ajout de dialogues interactifs

Les PNJ, une fois associées à des dialogues, créé une dimension d'interaction avec eux en leur "parlant". Tout comme l'ont fait Rigaud (2023) et Kourouma (2024), nous avons utilisé le *prefab UIDialog* développé par DEC. Un *UIDialog* est simplement lié à un *Agent* grâce à la variable *Anchor* du *prefab*. Enfin, une voix lui permet de parler avec une piste audio du texte lu. Pour cela, nous avons utilisé la foncitonnalité *Text-to-Speech* de Google Cloud en insérant le dialogue au préalable. Un PNJ a été associé à un dialogue implémenté par Kourouma (2024) à l'entrée de la cour du château du Wassenbourg, qui apporte des informations sur l'utilisation de cette cour en écurie à une époque antérieure. On peut facilement imaginer que tous les dialogues soient associés à des personnages, ce qui rendrait la visite bien plus engageante, comme pour *Gates of Horus* de Jacobson (2011).

La création et gestion des PNJ, combinant personnalisation visuelle, navigation intelligente et dialogues interactifs, permet d'ajouter une profondeur immersive et des interactions riches, rendant l'environnement du Cube VR plus vivant et engageant pour les utilisateurs.

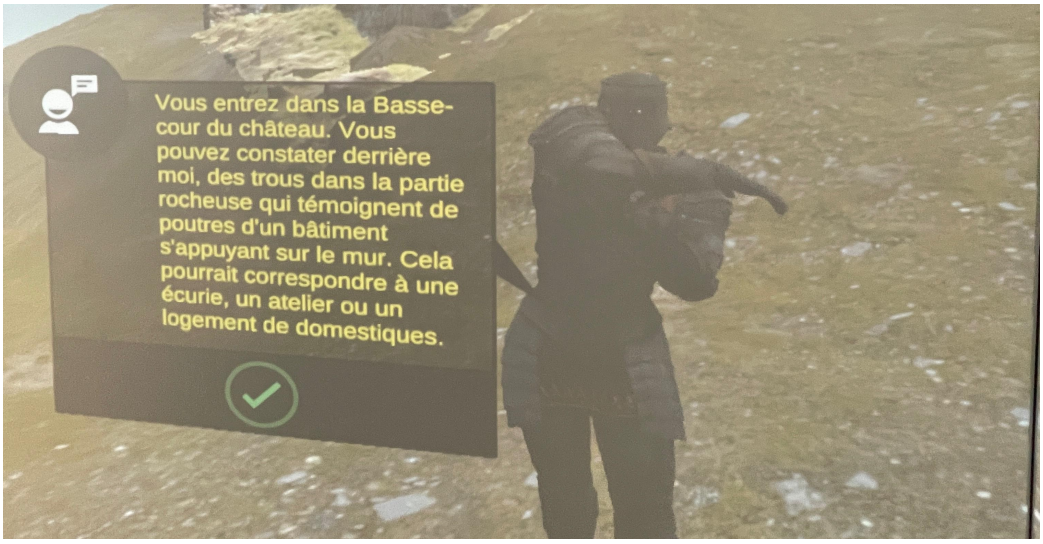

FIGURE 3.5 – Illustration dans le Cube d'un PNJ avec un *UIDialog* associé

3.2 Création du *prefab* boîte à outils

Le *prefab* *UserVRInventory* intégré au *XRPlayer* déjà existant est très pratique et dans ce projet, il sert notamment à visualiser la 4D, en choisissant quelle époque du château à afficher Rigaud (2023). Cependant, il n'est pas toujours évident de l'utiliser dans le Cube, surtout pour un joueur débutant. Nous avons donc eu l'idée de créer un menu externe qui n'est pas attaché aux manettes, mais qui suit le joueur, tout en gardant les mêmes fonctionnalités.

Ce nouveau menu conserve la même nature *UI Canvas*, mais il a un mode de rendu physique (*World Space*). Les boutons de Rigaud (2023) sont implémentés sur la partie haute du menu, permettant de modifier l'époque du château affiché (voir figure 3.6. De manière similaire, les boutons du prefab *Tablet* ajoutés par Kourouma (2024) sont présents sur la partie inférieure. Les boutons jugés inutiles au jeu tels que Documents ou Heure n'y apparaissent pas, ne conservant au final que les boutons Aménagement, Mon Equipement, App. Photos et Vidéos du *prefab* originel. Le bouton Vidéos renvoit à la vidéo introduite par Kourouma (2024). Enfin, un nouveau bouton *Menu Principal* apparaît en bas, renvoyant au menu principal créé par Kourouma (2024), reprenant son script *MenuController* qui permet de changer de scènes. Toujours dans une démarche de pérennité, nous avons exporté ce menu en *prefab*.

L'objectif de ce menu est qu'il remplace le menu *UserVRInventory* d'origine, c'est-à-dire qu'il reprenne au moins les mêmes caractéristiques que lui. Celles-ci

FIGURE 3.6 – Visuel de la boîte à outils dans le jeu

comprennent le fait que le menu est accessible partout et à tout instant en étant attaché au joueur. Malheureusement, nous avons tenté de rédiger un script *OnOff*, permettant en théorie de l'activer et de le désactiver avec un bouton de la manette. Pour les raisons évoquées dans la partie 2.3.2, nous n'avons pas réussi à récupérer les variables associées aux boutons de la manette. De plus, un second script *FaceAuXRPlayer* permettrait au menu de faire face au joueur pour le rendre accessible à tout moment. Enfin dans le cas général, lorsqu'on associe un objet à un autre, il se déplace aussi avec mais ici ce n'est pas le cas avec le *XRPlayer* de DEC. Nous pensons qu'il réside des conflits avec des scripts ou des appels à certaines fonctions dont ne connaissent pas la nature. Toutefois, le menu reste fonctionnel et le joueur peut interagir avec.

Nous parlions précédemment du menu principal du jeu, décrivons dans la partie suivante, les modifications qui ont été apportées.

3.3 Scène du menu principal

Nous avons repris le menu principal introduit par Kourouma (2024) pour y ajouter des fonctionnalités. Il a pour but d'aider les joueurs novices à l'environnement à se familiariser avec le Cube et les manettes de jeu en particulier (voir figure 3.7. Un PNJ a été ajouté qui sert d'interlocuteur pour le tutoriel. Celui-ci est texturé et animé comme vu précédemment dans la partie 3.1. La génération d'un

FIGURE 3.7 – Scène du menu principal

NavMesh n'est pas nécessaire ici, puisque l'*Agent* n'a pas vocation à se déplacer dans cette scène. De plus, une boîte de dialogue et une piste audio lui sont attaché. Le dialogue porte sur l'utilisation des manettes pour se mouvoir dans le jeu. En guise d'illustration, nous avons repris des images d'un jeu VR d'exposition créé par l'INSA de Toulouse intitulé *TP-ElectroVRLab*. Celui-ci met en valeur les possibilités qu'offrent le *prefab* UIDialog de DEC, en évoluant dans un jeu de questions-réponses sur le thème de l'électronique, avec des QCM interactifs. D'autres modifications mineures de traduction ont été apportées, le bouton *Aides* devient *Vidéo* puisqu'il renvoi à la vidéo contextuelle sur les châteaux rhénans. Enfin, cette scène est exportable dans d'autres projets sous forme de *prefab* et modifiable librement.

Ces fonctionnalités et interactions, développées pour enrichir l'immersion au sein du Cube VR, rendent l'expérience plus interactive et engageante pour les utilisateurs. En combinant des éléments visuels, des animations, et des interactions dynamiques, elles offrent une base solide pour des améliorations futures et renforcent la qualité immersive de l'environnement virtuel.

CONCLUSION

Ce projet a permis d'exploiter et de développer le potentiel du Cube VR, en associant des outils numériques avancés et des méthodologies innovantes pour la valorisation du patrimoine historique. La création de PNJ dotés de textures, d'animations, et de capacités de navigation intelligente a permis de dynamiser les environnements virtuels et de rendre les interactions plus immersives. L'ajout de dialogues interactifs, combinés à des éléments audio et visuels, renforce l'engagement des utilisateurs tout en offrant une meilleure compréhension des environnements explorés. Par ailleurs, la conception et l'optimisation des menus, notamment la boîte à outils et du menu principal, apportent une dimension pratique et intuitive, facilitant la prise en main pour les nouveaux utilisateurs.
Ce projet illustre comment les technologies immersives peuvent transformer la manière de préserver et de transmettre le patrimoine, tout en offrant des opportunités d'apprentissage et de collaboration au sein de la communauté scientifique et éducative. Les résultats obtenus ouvrent la voie à de nouvelles perspectives, que ce soit pour la valorisation de nouveaux sites ou pour l'amélioration continue des outils numériques en réalité virtuelle.

En guise d'ouverture, nous pourrions citer des améliorations aux fonctionnalités ajoutées, comme modifier le script de déplacement aléatoire des PNJ pour qu'ils s'arrêtent lorsque le joueur a démarré u ne discussion avec lui, ou encore contacter Virtuel Concept afin d'étoffer nos scripts et d'être en mesure de modifier l'environnement du SDK DEC en cas de besoin. Sur ce dernier point, il serait également possible de décompiler les scripts, comme l'a mentionné Rigaud (2023). Nous pouvons même imaginer avoir de réelles discussions avec des intelligences artificielles (LLM), associées aux PNJ au travers des dialogues, dans le but d'offir une immersion complète au joueur.

TABLE DES FIGURES

LISTE DES TABLEAUX

BIBLIOGRAPHIE

Amit, P. (2024). Introduction to A*. Available at : https ://theory.stanford.edu/ amitp/GameProgramming/AStarComparison.html. (Accessed : 15-01-2025).

Anderson, E. F., Mcloughlin, L., Liarokapis, F., Petridis, P., and Freitas, S. D. (2010). Developing serious games for cultural heritage : a state-of-the-art review. *Virtual Reality*, 14(4) :255–275.

Anthes, C., García-Hernández, R. J., Wiedemann, M., and Kranzlmüller, D. (2016). State of the art of virtual reality technology. In *2016 IEEE Aerospace Conference*, pages 1–19.

Benazzi, T. (2018). Restitution 4D du Château du Kagenfels par combinaison de l'existant et d'hypothèses archéologiques pour une visite virtuelle du site. Master's thesis, INSA Strasbourg.

Brewer, D. (2019). Tactical pathfinding on a navmesh. In *Game AI Pro 360 : Guide to Tactics and Strategy*, pages 25–32. CRC Press.

Bruna, R. (2014). Modélisation 3D de la chapelle Saint-Laurent et de la place du Château (secteur 3) pour extraction de données archéologiques et visite virtuelle. Master's thesis, INSA Strasbourg.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2022). *Introduction to algorithms*, pages 595–601. MIT press.

Dhanda, A., Reina Ortiz, M., Weigert, A., Paladini, A., Min, A., Gyi, M., Su, S., Fai, S., and Santana Quintero, M. (2019). Recreating Cultural Heritage Environments for VR Using Photogrammetry. *The International Archives*

of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-2-W9 :305–310.

Heitz, J.-E. (2023). Modélisation 4D du château de l'Oedenbourg (Petit Koenigsbourg) par combinaison de relevés de terrain et d'hypothèses archéologiques. Master's thesis, INSA Strasbourg.

Hermon, S., Nikodem, J., and Perlingieri, C. (2006). Deconstructing the VR - data transparency, quantified uncertainty and reliability of 3D models. pages 123–129.

Hutson, J. (2024). Technoculture and Language Models in Archaeology : Reconstructing and Preserving Cultural Narratives Through Digital Humanities. *Journal of Anthropological and Archaeological Sciences*, 10(1).

Jacobson, J. (2011). Digital Dome Versus Desktop Display in an Educational Game : Gates of Horus.

Jacobson, J., Handron, K., and Holden, L. (2010). Narrative and content combine in a learning game for virtual heritage.

"JonDevTutorial" (2022). Randomnavmeshmovement. Available at : https ://github.com/JonDevTutorial/RandomNavMeshMovement (Accessed : 15-01-2025).

Kourouma, M. (2024). Transfert et intégration de données 4D du projet Châteaux rhénans dans le cube de réalité virtuelle. Master's Research Project, INSA Strasbourg.

Landes, T., Heissler, M., Koehl, M., Benazzi, T., and Nivola, T. (2019). Uncertainty Visualization Approaches for 3D Models of Castles Restituted from Archeological Knowledge. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2-W9 :409–416.

Milgram, P., Takemura, H., Utsumi, A., and Kishino, F. (1995). Augmented reality : A class of displays on the reality-virtuality continuum. 2351 :282–292.

Pearl, J. (1984). *Heuristics : intelligent search strategies for computer problem solving*, page 48. Addison-Wesley Longman Publishing Co., Inc., USA.

Rigaud, D. (2023). Restitution 4D du château de Wasenbourg dans le cadre du projet INTERREG VI. Master's thesis, INSA Strasbourg.

Rocha, M. (2022). Levé et numérisation du château de Lichtenberg en vue d'une proposition de visite virtuelle du site à des périodes remarquables. Master's thesis, INSA Strasbourg.

Siddiqui, M. S., Syed, T., Nadeem Al Hassan, A., Nawaz, W., and Alkhodre, A. (2022). Virtual Tourism and Digital Heritage : An Analysis of VR/AR Technologies and Applications. *International Journal of Advanced Computer Science and Applications*, 13.

Sigitov, A., Scherfgen, D., Hinkenjann, A., and Staadt, O. (2015). Adopting a Game Engine for Large, High-Resolution Displays. *Procedia Computer Science*, 75 :257–266.

Theodoropoulos, A. and Antoniou, A. (2022). VR Games in Cultural Heritage : A Systematic Review of the Emerging Fields of Virtual Reality and Culture Games. *Applied Sciences*, 12(17) :8476.

Tredinnick, R., Boettcher, B., Smith, S., Solovy, S., and Ponto, K. (2017). Uni-CAVE : A Unity3D plugin for non-head mounted VR display systems. In *2017 IEEE Virtual Reality (VR)*, pages 393–394.

Unity (2024). Inner Workings of the Navigation System | AI Navigation | 2.0.5. Available at : https ://docs.unity3d.com/Packages/com.unity.ai.navigation@2.0/manual/NavInnerWorkings.html (Accessed : 15-01-2025).

Weiss, S., Bongartz, H., Boll, S., and Heuten, W. (2018). Applications of immersive vr in nursing education. In *Conference : Clusterkonferenz Zukunft der Pflege—Innovative Technologien für die Praxis*.

ANNEXES

PRT Fiche : Boîte à outils

Luca Joseph

January 2025

1 Descriptions générales

On peut voir à quoi ressemble la boîte à outils sur la figure 1. Elle est composée de 2 parties "Epoques du château" et "Outils divers", ainsi que du bouton "Menu Principal". Il s'agit simplement d'un objet "UI Canvas" qui est rattaché au prefab "XRPlayer" de DEC (figure 2). La particularité de ce UI Canvas

Figure 1: Visuel de la boîte à outils dans le jeu

est qu'il a un mode de rendu dit "World Space", c'est-à-dire qu'il aura une enveloppe physique dans le jeu (figure 3). De plus, il faut lui associer une Event Camera nommée "EventCamera (Camera), que l'on peut récupérer dans l'architecture du XRPlayer.

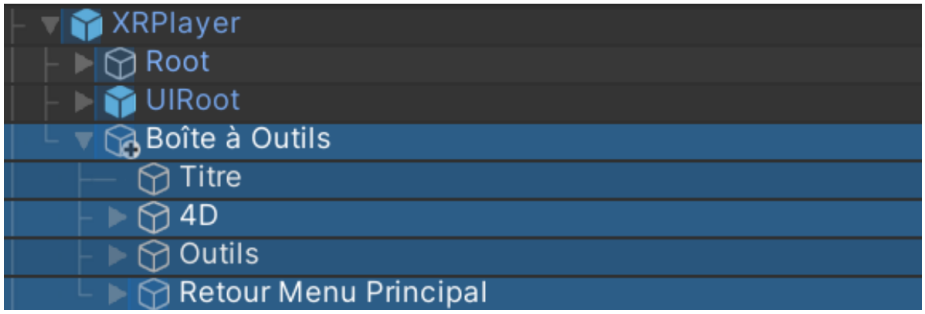

Figure 2: Architecture du GameObject "Boîte à Outils" et ses 4 enfants

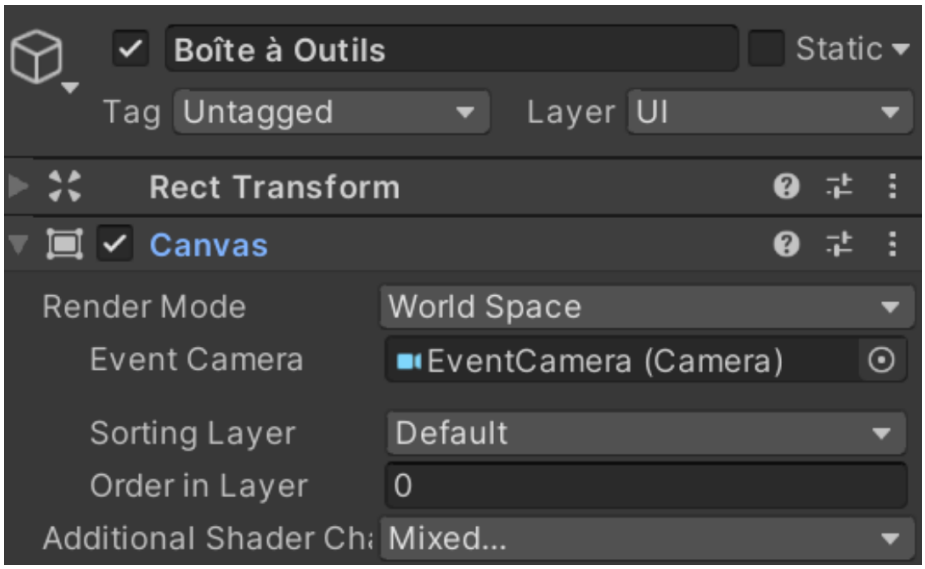

Figure 3: GameObject "Boîte à Outils" et ses composants

2 Description de l'objet "4D"

Le GameObject "4D" est composé de 3 enfants. "Titre 4D", "UserVRInventorySlot01", "UserVRInventorySlot02" et "UserVRInventorySlot03" (figure 4).

"Titre 4D" est simplement le titre de cette partie du menu.

"UserVRInventorySlot01" correspond au 1er bouton permettant d'afficher l'état actuel du château.

"UserVRInventorySlot02" est le 2nd bouton affichant l'état historique.

"UserVRInventorySlot03" permet de visualiser la fiabilité de la reconstruction de l'état historique.

"XRPlayer/Boîte à Outils/4D/UserVRInventorySlot01/Content/TopPart/Picto-background/Picto" contient l'image du bouton que l'on souhaite faire apparaître dans l'interface.

"XRPlayer/Boîte à Outils/4D/UserVRInventorySlot01/Content/Text" contient le texte descriptif du bouton associé.

Les autres objets ont été développé par DEC pour le bon fonctionnement de l'interface initiale dans le Cube.

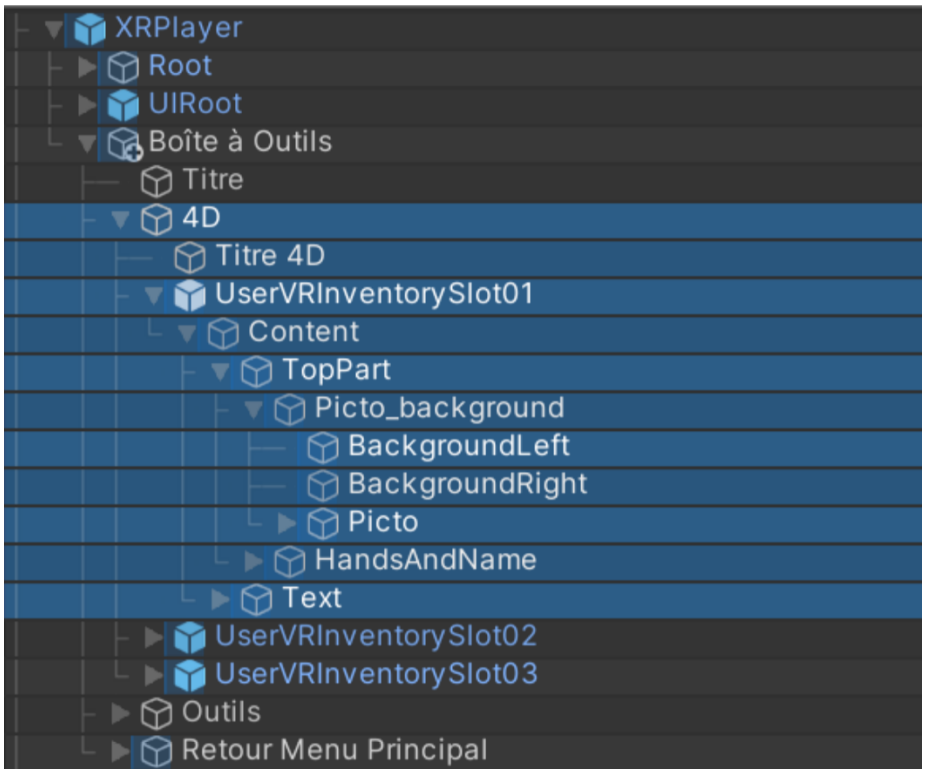

Figure 4: Architecture du GameObject "4D"

3 Description de l'objet "Outils"

Le GameObject "Outils" est composé de 3 enfants. "Titre Outils divers", "HomeMenu" et "Applications" (figure 5). Cet objet vient tout droit du prefab "tablet" de DEC, adapté pour nos besoins. Décrivons ensuite les enfants de "Outils".

"Titre Outils divers" est simplement le titre de cette partie du menu.

"HomeMenu" contient les boutons des divers outils du prefab "tablet". Il y a initialement plus de boutons, mais nous avons décidé de garder les 4 boutons jugés plus importants qui sont : "ObjectsPicker", "ToolsPicker", "Camera" et "Videos".

"Applications" contient les fonctions derrière les 4 boutons vus précédemment.

"XRPlayer/Boîte à Outils/4D/UserVRInventorySlot01/Content/TopPart/Picto-background/Picto" contient l'image du bouton que l'on souhaite faire apparaître dans l'interface.

"XRPlayer/Boîte à Outils/4D/UserVRInventorySlot01/Content/Text" contient le texte descriptif du bouton associé.

Les autres objets ont été développé par DEC pour le bon fonctionnement de l'interface initiale dans le Cube.

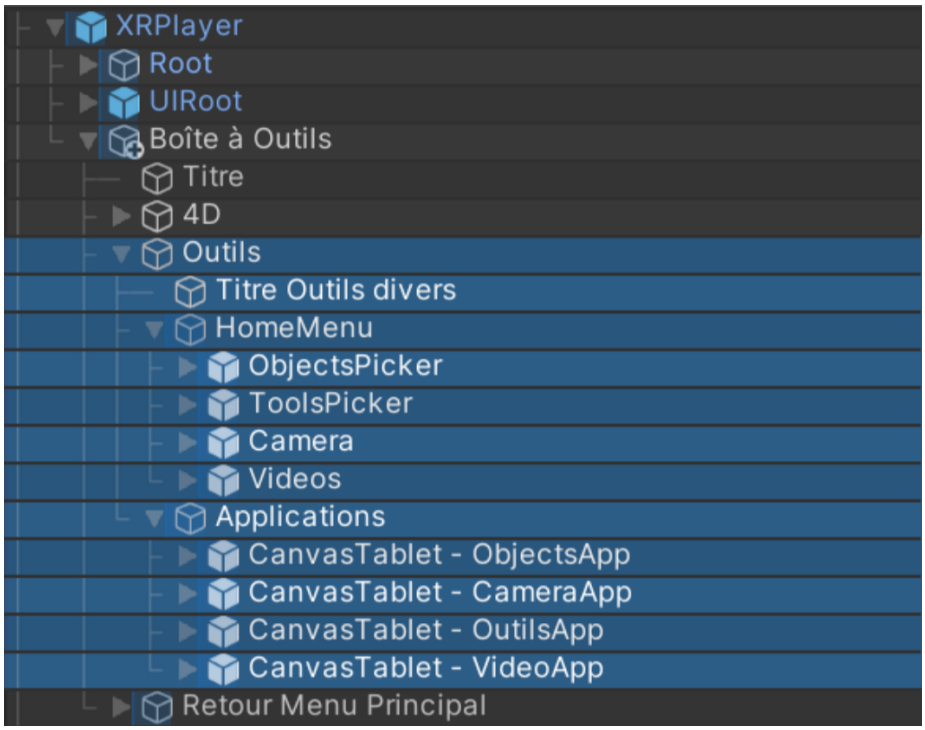

Figure 5: Architecture du GameObject "Outils"

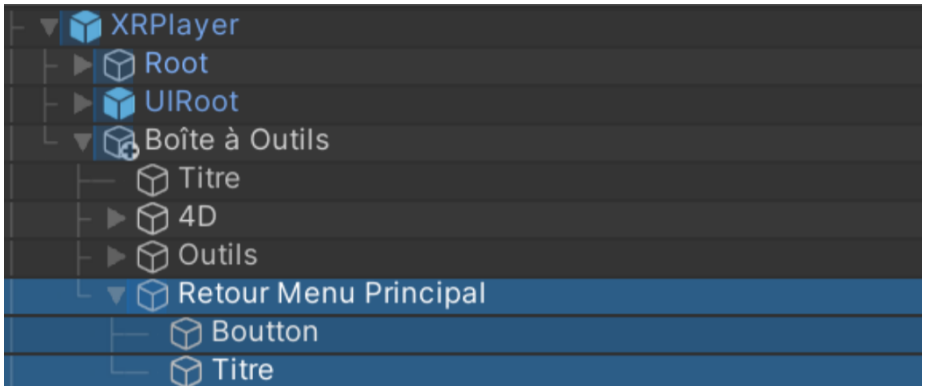

Figure 6: Architecture du GameObject "Retour Menu Principal"

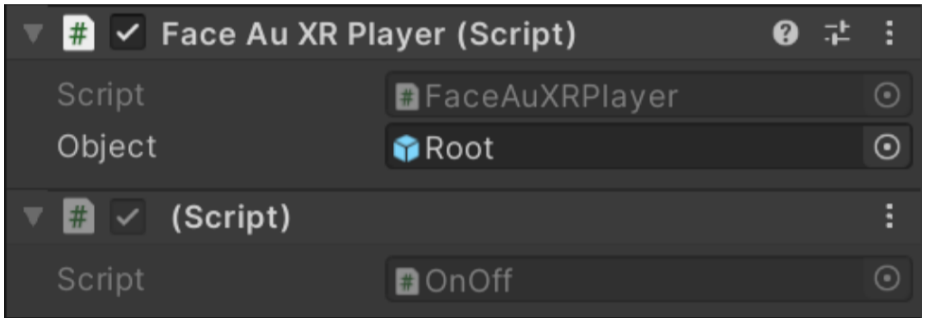

Figure 7: Scripts attachés au GameObject "Boîte à Outils"