



INSA Rouen Normandie
Département Génie Mathématique

**Rapport de projet de fin d'études :
Visite virtuelle de la carrière de Caumont**

Nathan FLAMBARD

24 février 2021

Table des matières

Introduction	3
1 Le cadre du projet	4
1.1 Les Carrières de Caumont et le projet ATP	4
1.2 Objectifs du projet et ressources disponibles	5
2 Version 1 : Création de l'environnement virtuel	5
2.1 Environnement et ambiance	5
2.1.1 Création de l'environnement : maillage et textures	5
2.1.2 Gestion des lumières	7
2.1.3 Autres éléments	8
2.2 Actions de l'utilisateur	9
2.3 Guider l'utilisateur dans l'environnement	10
2.3.1 Carte et points d'intérêt	10
2.3.2 Ajout d'un guide	12
3 Version 2 : Adaptation de l'environnement pour réalité virtuelle	14
3.1 Choix du casque de réalité virtuelle	14
3.2 Actions de base	15
3.2.1 Représentation de l'utilisateur et interactions	15
3.2.2 Ajustement des déplacements	16
3.3 Adaptation des interfaces	17
Bilan et pistes d'améliorations	20
Annexes	21
Manuel d'utilisation	21
3.3.1 Version clavier-souris	21
3.3.2 Version casque de VR (oculus quest)	21
Manuel d'installation	22

Introduction

Ce rapport a pour but de présenter le travail réalisé dans le cadre de mon projet de fin d'études au sein du département Génie Mathématique. Ce projet consistait à développer une application de visite virtuelle des carrières de Caumont. Le sujet a été proposé en partenariat avec l'association normande de spéléologie. Vous trouverez également à la fin de ce rapport des annexes présentant la procédure d'installation de l'application ainsi qu'un rapide manuel d'utilisation pour les différentes versions disponibles. Les différentes versions de l'application et le projet Unity sont trouvables sur le dépôt Gitlab de l'INSA Rouen : <https://gitlab.insa-rouen.fr/nflambard/caumont-vr>

1 Le cadre du projet

1.1 Les Carrières de Caumont et le projet ATP

La ligne directrice de ce projet est de construire un application permettant de visiter une partie des carrières de Caumont. Les carrières de Caumont se trouvent dans la commune de Caumont dans l'Eure (27). On sait que ces carrières ont été exploitées au moins depuis l'époque gallo-romaine et jusqu'au début du XXème siècle. Les pierres extraites dans ces carrières servaient principalement aux constructions et ont été utilisées par exemple pour la construction de la cathédrale de Rouen. Ces carrières ne sont pas restées inoccupées au XXème pour autant, en effet les allemands avaient commencé à y installer une usine pendant la Seconde Guerre Mondiale et une champignonnière avait également essayé de s'y installer dans les années 60. L'accès aux carrières est aujourd'hui détenu par l'association normande de spéléologie.

L'idée initiale de ce projet s'est formée à partir des résultats du projet ATP¹. Dans le cadre de ce projet à l'initiative de l'université de Rouen et de la Région Normandie, un maillage d'une partie de la carrière a été réalisé. L'idée étant ici d'utiliser ce maillage pour réaliser une visite virtuelle de la carrière.

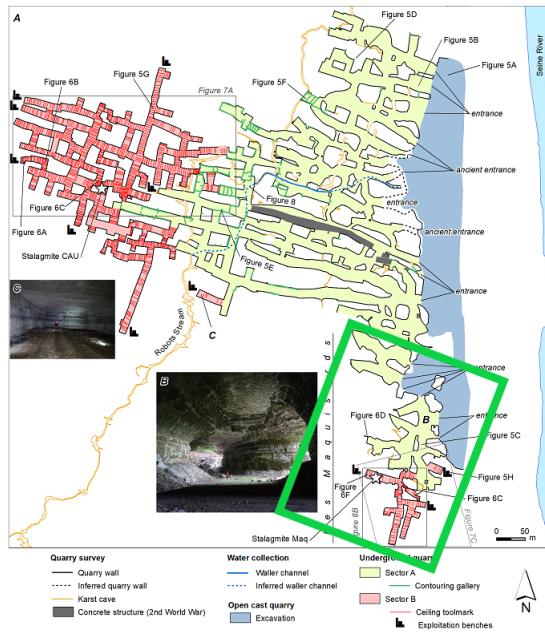


FIGURE 1 – Carte des carrières : la partie maillée est encadrée en vert

1. <https://irihs.univ-rouen.fr/fr/projet/atp-archeomateriaux-territoires-et-patrimoines>

1.2 Objectifs du projet et ressources disponibles

Le but de ce projet est comme dit précédemment de réaliser une application permettant de visiter la carrière. Cette application a principalement pour but d'être utilisée par l'association normande de spéléologie à titre éducatif dans des écoles ou dans des manifestations telles que la fête de la science de l'université de Rouen. L'intérêt de cette application étant qu'elle permet de donner l'accès à la grotte à un public plus large.

Au niveau des ressources disponibles pour la réalisation de ce projet, il y a tout d'abord bien entendu le maillage réalisé pour le projet ATP. Le casque de réalité virtuel a été fourni par l'INSA afin de permettre le développement. Enfin une présentation de l'application à mi-parcours et une visite de la Carrière ont été organisées afin d'avoir un retour sur le travail effectué et de permettre d'orienter certaines décisions concernant l'application.

2 Version 1 : Création de l'environnement virtuel

Dans l'optique de développer notre application par étapes, il a été décidé de créer une première version simple. L'objectif étant d'un côté de prendre en main le moteur Unity tout en ayant un résultat à présenter. Les objectifs de cette première version étaient les suivants :

- * Créer un environnement représentant la carrière
- * Permettre à l'utilisateur de visiter cet environnement avec une interface type clavier-souris
- * Proposer une solution pour ajouter des informations sur la carrière à certains endroits de l'environnement.

2.1 Environnement et ambiance

2.1.1 Crédit de l'environnement : maillage et textures

La première étape dans la création de l'environnement a été de préparer le maillage de la carrière pour pouvoir l'utiliser dans Unity. En effet, le maillage fourni avait ses faces tournées vers l'extérieur. Par conséquent, si on plaçait notre caméra à l'intérieur de la grotte, on ne pouvait pas voir les parois. Afin de résoudre ce problème, il a fallu inverser les normales du maillage en utilisant un logiciel tiers (dans notre cas MeshLab²). Une fois cette opération faite, on peut charger le maillage dans Unity et l'utiliser comme base pour notre environnement. Un second travail nécessaire à fournir concernait les entrées de la grotte. En effet, ces dernières étaient obstruées et il a donc fallu les ouvrir. Ce travail a été fait en utilisant le logiciel Blender³ en se basant sur les cartographies existantes de la carrière. Une rectification de certaines ouvertures a été effectuée

2. <https://www.meshlab.net>

3. <https://www.blender.org/>

après la visite de la carrière car certaines ouvertures pas forcément visibles sur les plans manquaient.

Une fois ces modifications faites, on dispose bien d'un modèle 3D de la grotte compatible avec l'application que l'on souhaite en faire. Afin de rendre notre environnement plus crédible, l'ajout de textures au maillage est une étape importante. Cependant on est confronté ici à un nouveau problème : notre modèle est constitué d'un seul bloc donc si on lui applique un matériau dans Unity, ce matériau recouvrira tout le maillage. Or, dans une grotte réelle, l'aspect du sol n'est en général pas le même que celui des murs. Nous avons donc une nouvelle fois utilisé Blender pour découper le modèle en 3 morceaux (mur, plafond et sol) auxquels on pourra affecter des textures séparément. Bien que la découpe réalisée soit faite "à la main" et donc grossière, elle permet tout de même un premier résultat assez convaincant :



FIGURE 2 – Rendu du maillage avec et sans textures

Si on compare le rendu obtenu avec l'aspect réel de la grotte, on se rend compte que l'on reste loin du compte :



FIGURE 3 – Aspect réel de l'endroit correspondant

Cependant il s'agit tout de même d'une bonne base de départ facile à produire (avoir un rendu ultra réaliste risque de demander beaucoup d'efforts et de

ressources et n'est pas forcément utile pour notre application).

2.1.2 Gestion des lumières

Avoir créé un modèle de la grotte avec des textures n'est cependant pas suffisant. En effet lorsque l'on se trouve dans une grotte la notion d'éclairage joue un rôle important dans l'ambiance du lieu. On s'attend en effet qu'il fasse jour aux entrées de la caverne et que l'on soit plongé dans le noir quand on se dirige vers le fond. Afin de simuler cela, on supprime la lumière ambiante de l'environnement Unity et on génère la lumière aux entrées avec des lumières Unity de type Spotlight. Dans l'idéal, il aurait fallu utiliser une ou deux grosses lumières pour mieux rendre une direction cohérente du soleil. Cependant comme le modèle de notre grotte est transparent de l'extérieur cette solution ferait passer la lumière du "soleil" à travers les murs ce qui n'est pas souhaitable. On a donc utilisé plusieurs lumières plus petites qui éclairent juste les endroits souhaités. Une autre solution aurait été de faire en sorte que les murs soient visibles des 2 cotés mais cela aurait demandé de doubler la taille du maillage ce qui n'est pas souhaitable dans notre cas pour des raisons de performances.

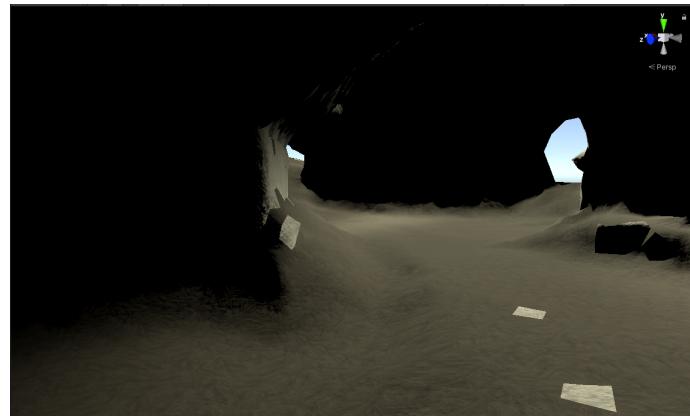


FIGURE 4 – Effets de lumière à l'entrée

Afin de tout de même pouvoir explorer les parties de la grotte plongées dans l'obscurité, plusieurs modes d'éclairage ont été prévus :

- * Un éclairage type lampe frontale pour avoir un rendu proche de ce que l'on a aujourd'hui en visitant la grotte. (fixation d'une Spotlight sur la caméra)
- * Un éclairage à la torche pour avoir un rendu plus proche de ce que pouvait voir les premiers visiteurs de la caverne. (ajout d'un modèle de torche et d'un PointLight au bout)
- * Un éclairage global, où la grotte est totalement mise en lumière. (ajout de plusieurs DirectionnalLight)

Ces différents modes d'éclairage permettent de jouer sur l'ambiance visuelle de l'environnement.

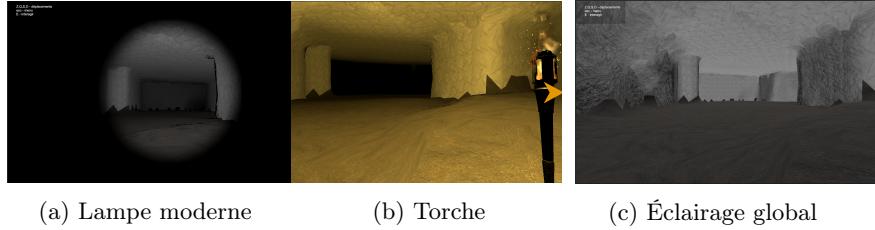


FIGURE 5 – Aperçu des différents mode d'éclairage

2.1.3 Autres éléments

Afin de compléter notre environnement, plusieurs ajouts ont été effectués :

- * La présence de gouttes d'eau qui tombent de temps en temps en faisant du bruit.
- * Le son des pas de l'utilisateur quand ce dernier se déplace.
- * La possibilité d'ajouter des points d'eau dans la grotte.

L'ajout de sons pose plusieurs problèmes. Tout d'abord l'écho de la grotte, en effet on s'attend à ce que les sons résonnent mais pas de la même manière à tous les endroits. Unity permet de générer cet écho avec une zone de réverbération (ReverbZone) qui peut être ajustée pour faire diminuer progressivement l'écho sur les bords de la zone. On place donc cette zone de réverbération centrée sur les salles les plus grandes et on fait décroître progressivement l'effet lorsqu'on s'éloigne de plus d'une certaine distance du centre de la zone (c'est à dire lorsque l'on se trouve dans les salles plus petites).

Pour les gouttes d'eau, on les fait apparaître aléatoirement autour de l'utilisateur, et lorsqu'elles touchent le sol, on déclenche un son de goutte d'eau. Afin d'éviter trop de calculs, on réutilise toujours la même goutte qui se téléporte en hauteur près du joueur à des intervalles aléatoires. Les bruits de pas sont quant à eux générés quand l'utilisateur se déplace dans l'environnement. Notre ambiance sonore souffre cependant d'un défaut : comme on dispose d'un nombre d'assets sonores réduit (un bruit de pas et un bruit de goutte), ces bruits peuvent devenir vite répétitifs. Comme il n'est pas forcément possible de récupérer une large variété de sons correspondants à notre situation, on utilisera une solution un peu plus simple : avant de lancer les sons, on change aléatoirement leur volume et leur fréquence (ce sont des paramètres facilement modifiables dans Unity). On obtient ainsi un son légèrement différent pour chaque pas et chaque goutte sans avoir à générer ou trouver une banque de sons complète.

Pour ajouter d'éventuels points d'eau dans la grotte, on peut utiliser simplement les solutions des assets standards Unity⁴. L'asset fonctionnant le mieux

4. <https://assetstore.unity.com/packages/essentials/asset-packs/standard-assets-for-unity-2018-4-32351>

pour notre cas étant le "WaterProNighttime"⁵ qui a en particulier l'avantage de permettre de régler les effets de courants de vagues etc...



FIGURE 6 – Exemple de rendu d'eau obtenu

Cependant dans la partie de la carrière modélisée ici, il n'y a pas de plan d'eau, par conséquent le développement de cet aspect n'a pas été poussé plus loin (pas de son de pas spécifique à la marche dans l'eau par exemple).

2.2 Actions de l'utilisateur

Un second axe de développement concerne l'utilisateur et la manière dont il va évoluer dans l'environnement. Pour commencer, afin de pouvoir lui permettre de visiter notre monde virtuel, il faut lui permettre de regarder autour de lui et de se déplacer. Tout d'abord comme le but à terme est de faire une application sur casque de réalité virtuelle, on choisit une vue à la première personne. On peut donc reprendre les codes classiques du déplacement clavier-souris pour ce type de vue (à savoir la souris pour orienter le regard et les touches ZQSD pour se déplacer). Plus concrètement, on oriente la caméra en fonction des mouvements de la souris. On bloque cependant les mouvements de la caméra au delà d'un certain angle vers le haut et le bas pour éviter de se retrouver avec une caméra à l'envers. Pour les mouvements du personnage, on utilise simplement un CharacterController qui est un composant Unity simplifiant la gestion des déplacements des personnages. Au niveau du rendu le personnage se déplace de manière continue comme si il marchait dans la carrière. Enfin, pour empêcher le joueur de sortir du maillage (et de tomber à l'infini), on ajoute des murs invisibles au niveau des entrées.

On souhaite ensuite laisser la possibilité de choisir l'une des trois ambiances lumineuses précédemment présentée. On ajoute pour cela un menu accessible en

5. Standard Assets/Environment/Water/Water/Prefabs/WaterProNighttime.prefab

appuyant sur une touche. Ce menu permet grâce à des boutons de sélectionner le type d'éclairage souhaité, on a ajouté de plus un élément permettant de régler l'intensité de la lampe frontale (comme cela est possible avec une vraie lampe).

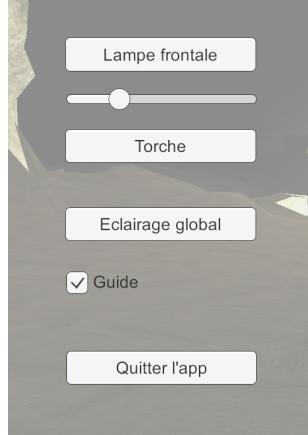


FIGURE 7 – Présentation du menu

2.3 Guider l'utilisateur dans l'environnement

On dispose maintenant d'un environnement dans lequel l'utilisateur va pouvoir évoluer et tester plusieurs choix d'éclairage. Pour compléter notre application de visite, il ne nous reste plus qu'à ajouter des informations sur l'environnement ainsi que d'indiquer à l'utilisateur quels sont les points d'intérêts à visiter.

2.3.1 Carte et points d'intérêt

La première étape pour l'ajout d'informations est d'ajouter un élément qui attire l'oeil et auquel l'utilisateur va intuitivement s'intéresser. On a choisi pour cela d'utiliser un cube vert qui émet une faible lumière (le rendant ainsi visible dans le noir). Quand on se rapproche du cube, on peut appuyer sur une touche pour afficher des informations sur la carrière. Plus précisément pour pouvoir interagir avec le cube, le joueur doit être assez proche et regarder globalement dans la direction du cube (on accorde un certain angle de tolérance).



(a) Cube indicateur



(b) Page d'information

FIGURE 8 – Visuels du système d'affichage d'informations

Le panneau qui s'affiche est prévu pour contenir un titre, un paragraphe, une photo et éventuellement une vidéo. Une fois les informations consultées, on change la couleur du cube (il devient jaune) pour marquer qu'il a été visité (les informations restent consultables cependant). Afin de pouvoir dupliquer facilement cet élément, on en fait un préfabriqué Unity.

Maintenant que l'on dispose d'un média pour ajouter des informations, il nous faut un moyen d'aider l'utilisateur à trouver ces informations et à s'orienter dans l'environnement. La solution la plus simple pour cela est d'utiliser une carte où l'on place la position des cubes d'informations ainsi que celle du joueur. Pour mettre en place cette carte, on utilise une caméra en projection orthographique qui filme la scène de haut et envoie sa vision dans une texture. On utilise ensuite la texture obtenue pour créer une image que l'on place sur le menu de pause. Afin d'indiquer la position des points d'intérêt et du joueur, on ajoute au dessus de chaque élément un grand modèle 3D. Ainsi l'objet est bien visible sur la carte.

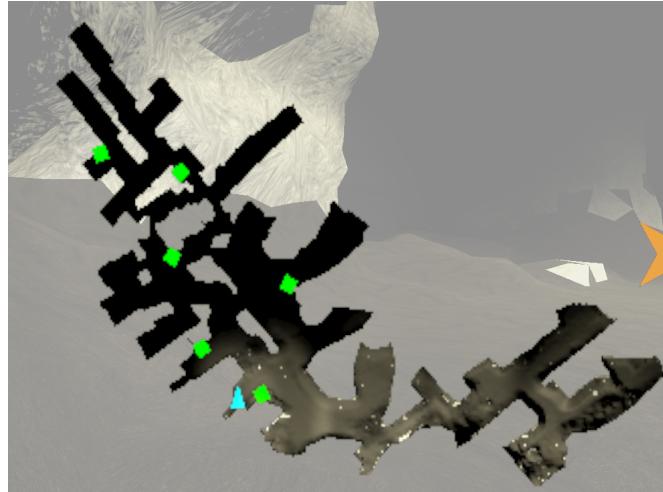


FIGURE 9 – Aspect de la carte obtenu

La carte est une première manière qui permet de mieux se retrouver dans la carrière. On a pu de plus développer une fonctionnalité de téléportation qui permet en cliquant sur un point de la carte de s'y retrouver téléporté. Pour implémenter cela, on transforme la carte en bouton et quand on clique sur le bouton, on effectue les actions suivantes :

- * On récupère la position de la souris sur l'écran.
- * On converti cette position pour obtenir la position correspondante dans l'environnement virtuel.
- * On vérifie qu'il y a bien du sol à l'endroit choisi (en utilisant un Raycast vers le bas)
- * Si la position est valide, on téléporte l'utilisateur.

2.3.2 Ajout d'un guide

Si la carte est une manière simple de guider l'utilisateur dans l'environnement, elle peut être assez lourde dans son utilisation car nécessite de mettre l'application en pause pour vérifier son chemin et n'indique pas forcément d'itinéraire précis. On a donc décidé de développer une sorte de guide qui évolue directement dans l'environnement et amène le visiteur vers les points d'intérêt. Il était de plus important de créer un guide qui laisse l'utilisateur assez libre d'explorer la grotte à son rythme. Pour l'aspect du guide, on a opté pour un petit feu-follet de couleur rose-rouge. L'idée étant qu'il soit assez visible et reconnaissable sans pour autant prendre trop de place dans le champ de vision.

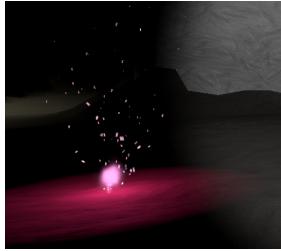


FIGURE 10 – Aspect du guide

Le comportement du guide est assez simple :

- * Si il est suffisamment proche du joueur, il se dirige vers le point d'intérêt non-visité le plus proche du joueur.
- * Si il est un peu trop loin du joueur, il se dirige vers le joueur jusqu'à en être suffisamment proche. L'objectif étant de l'attendre si ce dernier décide de s'arrêter à un endroit.
- * Si il est beaucoup trop loin du joueur, il se téléporte à coté de lui. Ce comportement étant là pour gérer le cas où le joueur se téléporte et éventuellement le cas où le guide se bloquerait.

Maintenant que nous avons défini le comportement du guide, il faut que notre guide puisse trouver son chemin dans la carrière. Pour cela, on utilisera une fonctionnalité de Unity appelée le NavMesh. L'idée est de créer un maillage de la surface où notre guide va se déplacer et d'utiliser ce maillage pour appliquer des algorithmes de pathfinding qui permettront de déterminer le chemin à suivre. On crée donc un NavMesh du sol de notre environnement (que l'on paramètre pour correspondre aux dimensions de l'utilisateur). On ajoute ensuite un composant NavmeshAgent à notre guide et il ne nous reste plus qu'à lui indiquer (via un script) le comportement à adopter.

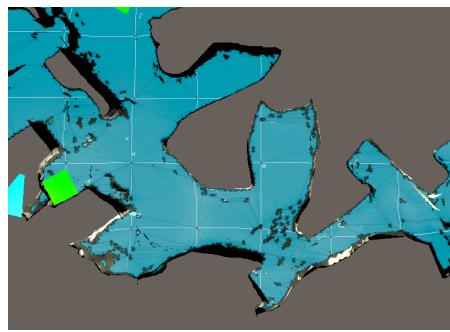


FIGURE 11 – Partie du NavMesh générée

On précise que l'on rend le guide un peu plus rapide que le visiteur pour éviter que ce dernier ait besoin de l'attendre quand il décide de le suivre. Pour

aider l'utilisateur à retrouver le guide si ce dernier n'est pas en face de lui, on affiche un indicateur en forme de flèche dans la direction du guide. Enfin, on ajoute la possibilité de désactiver le guide dans le menu au cas où l'utilisateur n'en ait pas l'utilité et le trouve dérangeant.

3 Version 2 : Adaptation de l'environnement pour casque de réalité virtuelle

Une fois la première version terminée, il était temps de passer au portage sur un casque de réalité virtuelle. L'idée étant de reprendre tout les aspects de la version précédente en y ajoutant les modifications nécessaires pour qu'ils restent fonctionnels dans la nouvelle version. Les axes de travail principaux pour cette version ont été :

- * La réflexion sur le choix du dispositif utilisé.
- * L'adaptation des actions et déplacements au nouveau dispositif.
- * L'adaptation des éléments d'interface au casque de VR.

3.1 Choix du casque de réalité virtuelle

Pour le développement de cette nouvelle version, nous avions à notre disposition deux dispositifs différents : le HTC Vive et l'Oculus Quest 2. La principale différence entre les deux est que le Vive nécessite d'être branché sur un ordinateur pour fonctionner tandis que le Quest fonctionne de manière autonome. Notre choix s'est porté sur l'Oculus Quest 2 pour les raisons suivantes :

- * Le casque ne nécessite pas un ordinateur puissant pour fonctionner.
- * Il ne nécessite pas l'installation d'une structure particulière pour fonctionner et peut donc être emmené à peu près partout (L'utilisation est déconseillée en extérieur cependant).
- * Dans le contexte sanitaire actuel, il permettait de travailler sur le projet sans se déplacer à l'INSA.

Il faut cependant bien être conscient que ce choix implique des contreparties :

- * La création et l'installation de l'application sur le support demandent la création d'un compte Facebook Developers et d'un compte Oculus. Ce qui dans notre cas complique la procédure d'installation sur le casque.
- * La puissance du casque est moins bonne que ce que l'on aurait avec le Vive.
- * On ne dispose pas de retour sur ce qu'affiche le casque pour les autres personnes dans la salle (la fonctionnalité n'est en tout cas pas présente de base).

Enfin d'un point de vue développement, plusieurs manipulations ont été nécessaires. En effet l'oculus Quest fonctionnant sous Android, il est nécessaire de vérifier que les modules Unity correspondant ont bien été installés. De plus en fonction de la version Unity utilisée il peut être nécessaire d'installer Android Studio (seulement pour des versions assez anciennes). Les informations relatives à ces changements peuvent être trouvées dans la documentation Oculus⁶. Ensuite, dans le projet Unity, il est nécessaire d'importer les assets de l'ensemble Oculus Integration⁷

3.2 Actions de base

Avant de commencer le développement de cette nouvelle version, il a été nécessaire d'ajouter au projet des bibliothèques pour nous simplifier la tâche. On utilise les bibliothèques Unity XR Interaction Toolkit⁸ et VRTK⁹. Les principaux rôles de ces packages sont de prévoir l'adaptation de l'application à d'autres casques ainsi que de faciliter la récupération des actions liées aux manettes ou la mise en place des déplacements de l'utilisateur. Enfin les changements décrits plus loin seront faits dans une nouvelle scène Unity afin de ne pas perdre la première version. Afin de relier les 2 versions, un préfabriqué Unity contenant la partie commune aux deux scènes (Caverne, points d'intérêt, certain effets de lumière...) a été créé, l'optique étant de pouvoir faire une modification dans l'environnement impactant les deux versions d'un coup.

3.2.1 Représentation de l'utilisateur et interactions

Le premier travail à effectuer est de récupérer les mouvements et actions que le casque enregistre. Pour cela, Oculus nous fournit un objet Unity nommé le OVRCameraRig. Cet objet est divisé en plusieurs parties qui vont reproduire les mouvements de l'utilisateur dans le monde virtuel. Si on peut utiliser ce composant tel quel, on préfère l'associer au composant TrackedAlias de VRTK. Le rôle du TrackedAlias sera de reproduire les mouvements du CameraRig qu'on lui associe. L'intérêt étant que si l'on souhaite adapter l'application à un autre modèle de casque, il suffit d'associer la capture du mouvement du nouveau casque au TrackedAlias et notre application peut directement marcher. Le TrackedAlias peut même prendre en paramètre une liste de CameraRig et choisir celui qui convient à chaque situation ce qui permet d'utiliser une même scène pour toutes les versions VR de l'application. Pour récupérer les actions relatives aux boutons des manettes associées au casque, on utilisera les composants OpenVR.RightController (ou LeftController) du paquet Unity XR. L'intérêt étant là aussi que ces composants sont compatibles avec la grande majorité des casques existants.

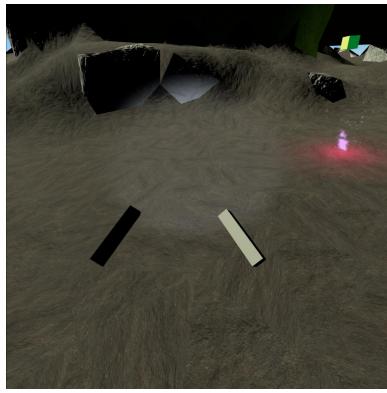
6. <https://developer.oculus.com/documentation/unity/book-unity-gsg/>

7. <https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022>

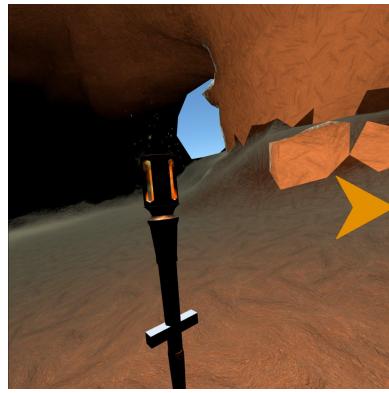
8. <https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@0.9/manual/index.html>

9. <https://github.com/ExtendRealityLtd/VRTK.Prefabs>

Maintenant que l'on dispose d'un moyen pour connaître les mouvements de l'utilisateur, On souhaite pouvoir lui donner un retour visuel sur ses mouvements. Pour cela on place au niveau des mains deux petits parallélépipèdes pour les représenter. Cette représentation rudimentaire des mains va nous permettre également d'interagir avec d'autres objets. Dans notre cas on fait en sorte que l'interaction avec les points d'intérêt ne se fasse plus en appuyant sur une touche mais si l'utilisateur touche le cube avec sa main virtuelle. De même dans un soucis d'immersion la torche qui avait une position fixe sur l'écran auparavant se retrouve fixée sur l'une des mains de l'utilisateur. L'utilisateur va pouvoir ainsi déplacer plus précisément la torche dans l'espace.



(a) Représentation des mains



(b) La torche fixée sur une main

FIGURE 12 – Représentation virtuelle des mains de l'utilisateur

3.2.2 Ajustement des déplacements

Une différence importante dans cette nouvelle version est la gestion des déplacements. En effet si on garde un déplacement continu similaire à celui de la première version, le visiteur risque d'être sujet à la cinétose (malaise lié à un décalage entre le mouvement perçu par les yeux et celui perçu par l'oreille interne). Afin d'empêcher cela, deux solutions sont proposées :

1. La première est la plus classique et passe par un système de téléportation à proximité. Quand l'utilisateur appuie sur un bouton, une courbe parabolique partant de sa main apparaît. Le point d'impact de cette courbe avec le sol est l'endroit où l'on souhaite se déplacer et peut être modifié en déplaçant sa main. Quand l'utilisateur relâche le bouton, il est téléporté vers l'endroit sélectionné. Cette méthode peut être facilement développée à partir des éléments de VRTK.
2. Une autre solution un peu moins efficace mais qui permet des ajustements de position plus précis est de garder un déplacement continu mais en réduisant le champ de vision de l'utilisateur (en plaçant des bandes noires

de chaque cotés de ses yeux). Cette solution peut-être facilement construite à partir de la mécanique de déplacement de la version 1.

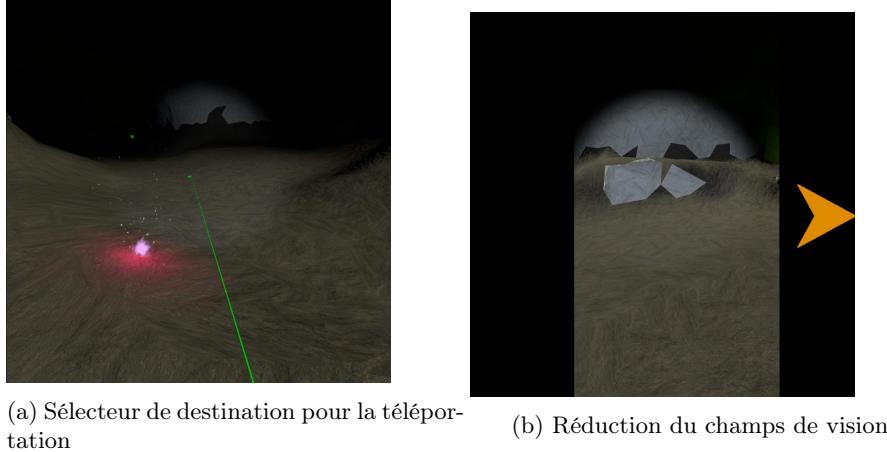


FIGURE 13 – Dispositifs d’adaptation des déplacements

Ces méthodes de déplacements sont toutes les deux intégrées à notre nouvelle version. Il est important de mentionner certains problèmes rencontrés lors de cette intégration :

- * La téléportation si elle est utilisée sous sa forme de base permet de se déplacer sur tout corps solide. Autrement dit il est nécessaire de rajouter des restrictions sinon l’utilisateur peut se téléporter sur certains murs. Pour régler cela on utilise la découpe faite pour ajouter les textures : on ajoute le tag "Floor" sur l’objet représentant le sol de la grotte et on spécifie que la téléportation ne marche que sur les objets avec le tag "Floor".
- * L’objet représentant le corps du CharacterController peut avoir tendance à entrer en collision avec les autres éléments du TrackedAlias ce qui mène à ce que l’utilisateur ait l’impression de flotter au dessus du sol. On peut régler cela en utilisant les couches (ou layers) de Unity. On place le CharacterController et les autres éléments du TrackedAlias sur deux couches différentes qui ne peuvent pas entrer en collision (les collisions entre les couches peuvent être réglées dans les paramètres du projet Unity).
- * Lors du déplacement par téléportation, il est important de vérifier que l’on déplace bien l’objet contenant le CharacterController pour que sa position soit toujours cohérente avec celle de l’utilisateur.

3.3 Adaptation des interfaces

Un travail important de cette version a été consacré aux interfaces utilisateur. En effet, la gestion de ces composantes est différente dans cette version car on

ne dispose plus de souris pour sélectionner les différents éléments. Une première modification nécessaire est que toutes les interfaces doivent être des composantes du monde virtuel et non plus de simples interfaces. Cependant cela ne pose pas la même difficulté pour toute les interfaces.

L'interface la plus simple à adapter est celle des points d'intérêt en effet, il suffit de passer les informations sur un affichage dit "WorldSpace" (qui fait partie de l'environnement). Ainsi lorsque l'utilisateur touche un cube, un panneau apparaît devant le cube. La seul interaction avec ce panneau se fait via un bouton et on peut donc facilement adapter l'interaction avec ce bouton en l'activant quand on détecte que la main virtuelle appuie dessus. Il est important de remarquer que le préfabriqué de ces points d'intérêt contient 2 affichages (un pour la version clavier-souris et un pour la version casque) et qu'il est nécessaire de modifier les 2 affichages lorsque l'on veut faire changer le texte ou les images.

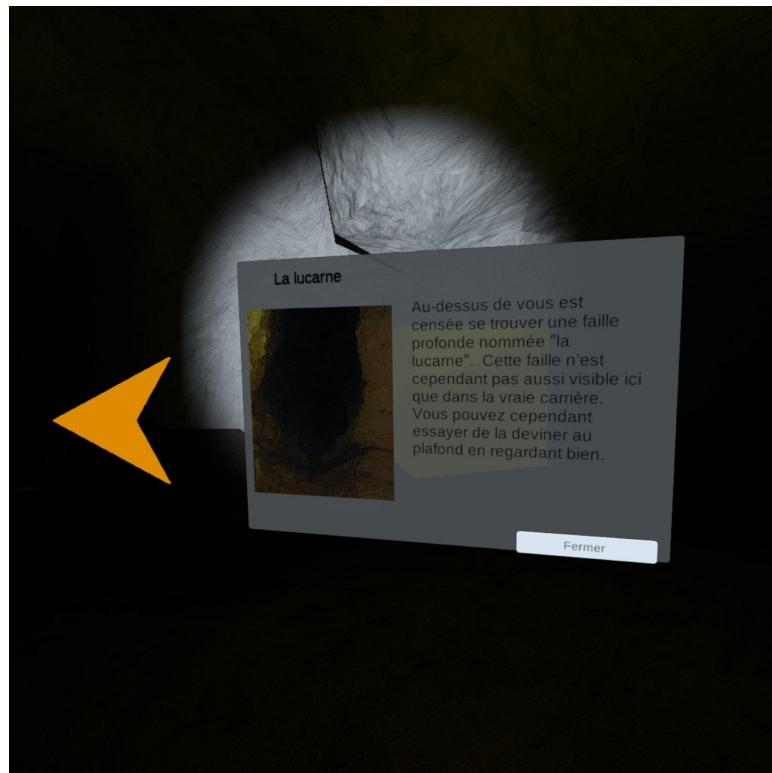


FIGURE 14 – Aspect du panneau d'information

Le menu de pose demande, lui, plus de travail. En effet en plus des boutons, on dispose d'éléments nécessitant plus de précision comme le sélecteur d'intensité de la lampe ou la téléportation via la carte. De plus on veut pouvoir avoir accès à tous moments à ce menu sans mettre de pause dans l'application. La solution choisie a été de fixer le menu à l'une des mains du visiteur. De cette manière, il

peut facilement consulter la carte tout en se déplaçant dans la carrière et le menu reste discret mais accessible à tout moment (on précise qu'une commande est prévue pour activer/désactiver le menu). Reste alors le problème des interactions avec ce menu, pour cela, on utilise le préfabriqué UIHelpers de l'ensemble Oculus Intégration (qui reste normalement compatible avec d'autres casques puisque le paramétrage est fait sur le TrackedAlias et non sur le OVRCameraRIG). Cet objet permet de créer un rayon entre la main du joueur et le menu et de se servir de ce rayon comme d'un pointeur qui remplace la souris. Cela permet donc d'interagir avec le menu plus précisément. On précise qu'ici la gestion des interactions avec la carte nécessite plus d'adaptations que les autres éléments. En effet l'endroit sélectionné sur la carte ne dépend plus du pointeur de la souris. Il est donc nécessaire de se baser sur le point d'impact du rayon sur le menu (qui est donné heureusement par un objet du UIHelpers) et de projeter sa position sur le plan de la carte pour pouvoir récupérer la bonne position.



FIGURE 15 – Aspect du menu adapté au dispositif de réalité virtuelle

Bilan et pistes d'améliorations

A la fin de ce projet, on dispose de 2 versions de notre visite virtuelle, une pour ordinateur et l'autre pour casque de réalité virtuelle. L'environnement virtuel se veut un minimum cohérent avec la carrière réelle et dispose d'éléments permettant au visiteur d'en apprendre plus sur la grotte. De plus le développement a été fait de manière à essayer de simplifier le portage sur de nouveaux dispositifs . On peut donc considérer notre objectif de réaliser une visite virtuelle de la carrière de Caumont comme réussi. Cependant l'application proposée comporte beaucoup de points pouvant être améliorés. On propose donc ici quelques axes de travail possibles pour une éventuelle suite du projet :

- * Mettre en place un système permettant d'avoir un retour sur ce que la personne avec le casque voit.
- * Trouver des solutions pour faciliter l'installation du projet sur un casque (en le mettant sur le store Oculus par exemple)
- * Améliorer les textures de l'environnement pour qu'elle se rapproche plus de la réalité.
- * Rendre l'apparence des points d'intérêt plus cohérent avec l'environnement. (sous la forme d'objets en rapport avec l'information donnée)
- * Ajouter une dimension plus ludique à l'application (intégration de petites épreuves au niveau des points d'intérêts)

On peut donc clairement dire que le travail rendu ouvre la porte à des possibilités d'améliorations. Ainsi même si l'application se suffit à elle-même, il existe de nombreuses voies d'amélioration qu'il serait dommage de ne pas explorer.

Annexes

Manuel d'utilisation

3.3.1 Version clavier-souris

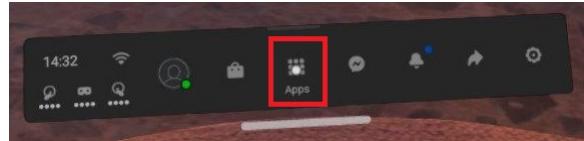
Cette version ne nécessite pas d'installation particulière : pour la lancer allez dans le dossier du projet et lancer l'exécutable s'appelant "CaumontVR" (ou "CaumontVR.exe") se trouvant dans le dossier `release/windows`. On donne la liste des commandes utiles pour cette version :

- * Le déplacement se fait avec les touches Z,Q,S,D du clavier (Z = avancer, S = reculer, Q = aller à gauche, D = aller à droite)
- * L'utilisateur peut orienter son point de vue en bougeant la souris
- * `échap` permet d'ouvrir et de fermer le menu de pause
- * E permet d'interagir avec les points d'intérêt

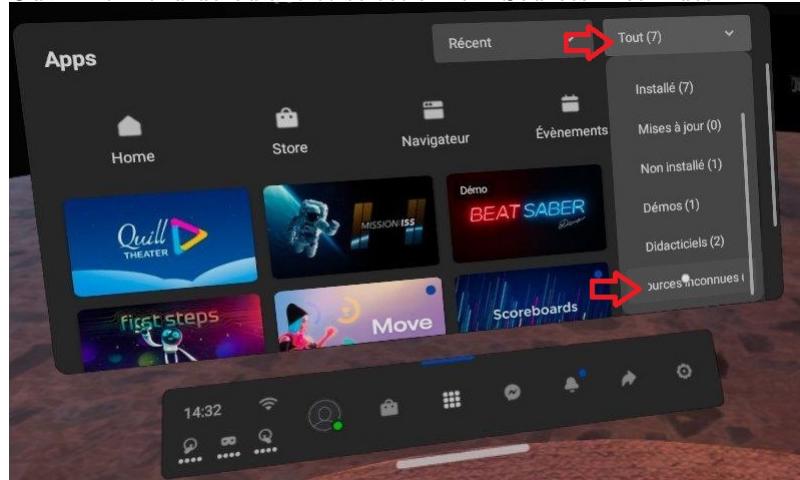
3.3.2 Version casque de VR (oculus quest)

Cette version nécessite d'être installée sur un casque Oculus Quest pour fonctionner. Pour l'installer, reportez-vous au manuel d'installation. Une fois installée, pour trouver l'application dans le casque il faut :

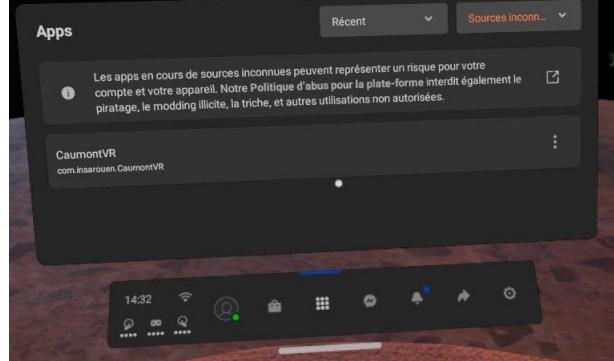
1. Sélectionner le menu "Apps" de l'Oculus :



2. Ouvrir le menu déroulant et sélectionner "Sources Inconnues" :



3. L'application devrait se trouver à présent dans le menu :



On donne ici encore les commandes et moyens d'interactions utiles :

- * Le déplacement continu se fait avec le stick de la main droite
- * Le menu s'ouvre et se ferme avec le bouton X
- * Le bouton A permet de sélectionner l'entrée pointée sur le menu
- * Pour interagir avec un cube passez votre main dedans.
- * La gâchette de la main droite permet de se téléporter.

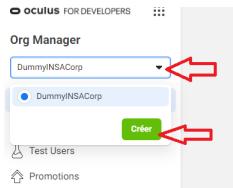
Manuel d'installation

Nous allons ici présenter comment installer l'application sur un casque Oculus Quest2. Les instructions sont basées sur le tutoriel suivant :

<https://stylistme.com/vr-ar-jeu-tuto/comment-installer-des-apk-sur-oculus-quest>

L'installation de notre application sur L'Occulus Quest 2 nécessite l'activation du mode développeur sur le casque. Nous allons détailler comment faire dans notre cas :

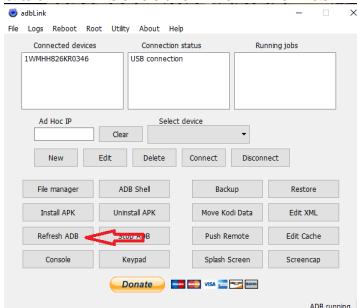
1. Pour fonctionner le casque demande un compte Facebook et un smartphone avec l'application Oculus. Le casque vous expliquera comment réaliser ces étapes.
2. Il va falloir créer un compte oculus développeur. Pour cela rendez vous sur le site <https://developer.oculus.com/> et connectez vous avec le compte facebook associé au casque (la création vous demandera votre numéro de téléphone pour se finaliser)
3. une fois votre compte créé, retournez à l'accueil du site oculus developer et on vous demandera de renseigner le nom de votre organisation (ce nom n'a aucune importance vous pouvez mettre ce que vous voulez). Si ce n'est pas le cas allez à <https://developer.oculus.com/manage> et créez une organisation :



- Une fois l'organisation créée, lancez l'application Oculus de votre smartphone, allez dans les paramètres, sélectionnez votre casque, puis sélectionnez "autres paramètres", puis "mode développeur" et activez l'option "mode développeur"

Votre casque est maintenant en mode développeur. Vous pouvez donc maintenant y installer l'application. Pour cela nous allons utiliser adbLink :

- Si ce n'est pas fait téléchargez et installez adbLink sur votre ordinateur depuis le site suivant : <http://www.jocala.com/>
- Branchez votre oculus quest à votre ordinateur via le cable fournit avec l'oculus. Mettez le casque et il devrait vous demander si vous acceptez les transferts de fichiers → acceptez.
- Sur votre ordinateur, lancez adbLink et vérifiez que vous voyez bien au moins un appareil détecté.
- Si l'appareil connecté est noté "unauthorized", mettez le casque (sans le débrancher) et acceptez les transferts puis, sur votre ordinateur, dans adbLink sélectionnez "refresh adb" :



- Suite à cela sélectionnez "Install APK", et quand adbLink vous demande de sélectionner un fichier, sélectionnez le fichier "Caumont_VR.apk". Ce fichier se trouve dans le projet dans le dossier `release/oculus` :

