IUT Nancy Charlemagne
Université de Lorraine
2 ter boulevard Charlemagne
BP 55227
54052 Nancy Cedex

Département Informatique

Application VR : Déplacements en 4 dimensions.

Projet tutoré de 3ème année.

Rapport de fin de projet.

Jawad KOMODZINSKI
Hugo RYSAK
Stanislas TROHA

Tuteur : Laurent DUPONT
Année universitaire : 2024-2025

Table des matières

Introduction

Pour commencer, nous préférons préciser que notre camarade Jawad KOMODZINSKI a été présent et impliqué dans ce projet durant les 3 premières itérations, après lesquelles il a quitté l'IUT. Par conséquent son nom apparaîtra dans ce rapport.

Ce document a pour objectif de faire un bilan détaillé de l'avancement de notre projet à la fin de ces 7 itérations, marquant la conclusion du projet tutoré. Il présente à la fois les choix techniques et fonctionnels réalisés, ainsi que les difficultés rencontrées et les solutions apportées.

Présentation du projet

Dans le cadre de notre projet tutoré en troisième année de BUT Informatique, nous avons entrepris le développement d'une application interactive permettant de visualiser et manipuler des objets en quatre dimensions (4D). Ce projet s'inscrit dans une démarche exploratoire, visant à rendre plus accessible la perception de la quatrième dimension à travers des outils numériques et une approche ludique.

L'objectif principal de cette application est de permettre à un utilisateur d'explorer et d'interagir avec des figure 4D dans un environnement en trois dimensions. Comme il est impossible d'observer directement un objet en 4D, notre solution repose sur des techniques de projection et des mécanismes d'interaction permettant d'étudier ces objets sous différentes perspectives.

L'application est construite autour de plusieurs fonctionnalités clés :

- Visualisation d'hypercubes projetés en 3D, avec différentes méthodes de projection (stéréographique, perspective, orthogonale).
- Manipulation des objets à travers des transformations (translations et rotations sur un ou plusieurs plans).
- Interface utilisateur intuitive, permettant de modifier les paramètres d'affichage et d'interaction.
- Mécanique inspirée du jeu Fez, permettant d'effectuer des rotations dans des sous-espaces tridimensionnels.
- Exploration en Réalité Virtuelle, permettant une meilleure immersion de l'utilisateur et une compréhension accrue

Structuration du développement

À l'origine, le projet était prévu en deux grandes phases :

1. Première phase : Implémentation de la visualisation des hypercubes, application de transformations (translations, rotations), et développement d'une interface permettant de modifier la projection et les interactions.
2. Deuxième phase : Ajout de la navigation et du déplacement dans un espace 4D, en intégrant une mécanique de rotation similaire à Fez, ainsi que l'exploration d'une implémentation en réalité virtuelle (VR).

Toutefois, suite à notre étude préalable et aux premiers défis techniques rencontrés, nous avons ajusté cette organisation pour prioriser un produit fonctionnel à la fin du premier semestre. Et donc lors du second semestre nous avons priorisé l'implémentation de la VR. Nous avons donc concentré nos efforts surs :

- L'optimisation de la visualisation des hypercubes et leur comportement dans l'environnement 3D.
- L'amélioration de l'interface utilisateur pour garantir une interaction fluide et intuitive.
- L'intégration de la mécanique Fez en 3D, permettant d'expérimenter les rotations dimensionnelles sans encore introduire la VR.

Et enfin lors du second semestre

- Implémentation de la VR : sûrement une des parties les plus importantes du projet, aussi une des parties les plus compliquées
- Adaptation du projet en VR : beaucoup de partie ont changé de la VR au projet bureau
- Implémentation de nouvelle figure : en parallèles nous avions aussi fait évoluer le projet sur bureau pour implémenter de nouvelle figure, notamment grâce aux fichier PLY sur lesquels nous reviendront plus tard.

L'aspect réalité virtuelle et certaines fonctionnalités avancées ont donc été abordés dans la seconde partie du projet, comme prévu lors de l'étude préalable et la fin de l'itération 4.

Analyse

Découpage fonctionnel du projet

Tout d'abord pour notre projet il existe deux applications : l'application bureau et l'application VR

Voici les modules principaux qui composent notre application bureau

- **Scène Principale** : C'est le monde où l'utilisateur va pouvoir se déplacer et interagir avec des objets.
- **Menu Principal** : Menu où l'utilisateur va pouvoir configurer la scène principale en choisissant les objets qu'il veut voir, leurs nombres, leur type de projection.
- **UI** : Dans le menu principal, l'interface utilisateur va permettre à l'utilisateur de sélectionner les objets qu'il veut voir. Dans la scène principale, l'utilisateur va pouvoir interagir avec les objets en utilisant l'UI par exemple en bougeant des sliders ou en appuyant sur des boutons pour modifier des valeurs (coordonnées, angle de rotation, …).
- **Sous-Monde** : Les sous-mondes permettent de délimiter un espace 3D. Un objet appartenant à ce sous monde disparaitra lorsque qu'il quittera les limites de ce dernier. On le définit comme une « fenêtre 3D » sur le monde en 4D qui contient la projection en 3D d'un objet 4D. C'est pour cela que si on le déplace trop selon une certaine coordonnée, il disparaîtra de la fenêtre de projection.
- **Hypercube.gd** : Bien que le nom du fichier soit hypercube.gd, il n'est en fait pas utilisé que pour l'hypercube, le fonctionnement est simple, il utilise un parser pour lire les informations venant d'un fichier ply, de ce fichier il va pouvoir récupérer les sommets et les arrêtes qu'on lui donne, à partir de ça il va donc modéliser cette figure 4D et s'appuyer sur ces points pour effectuer transformations ou autre. Pour l'instant nous avons réussi à modéliser des hypercube, hypersphère, tritriduoprisme, et hexadecachore
- **CharacterView** : La caméra qui représente en quelques sortes l'utilisateur. C'est en la déplaçant qu'on permet à l'utilisateur de se déplacer dans le monde afin d'explorer les différents hypercubes.

Maintenant nous allons voir les modules principaux composant l'application VR qui diffèrent de l'application bureau car beaucoup de modules sont communs :

- **MondeVR** : Le Monde Vr est différent de l'application bureau, contrairement à cette dernière, l'utilisateur n'en choisit pas les figures qu'il visualisera, nous avons choisis de créer 3 îles, que l'utilisateur peut accéder ou des figures ont déjà était placé et que l'utilisateur peut aller observer et manipuler
- **UI** : l'UI est différente en VR que dans l'application bureau puisque nous avons dû adapter cette dernière pour l'application VR

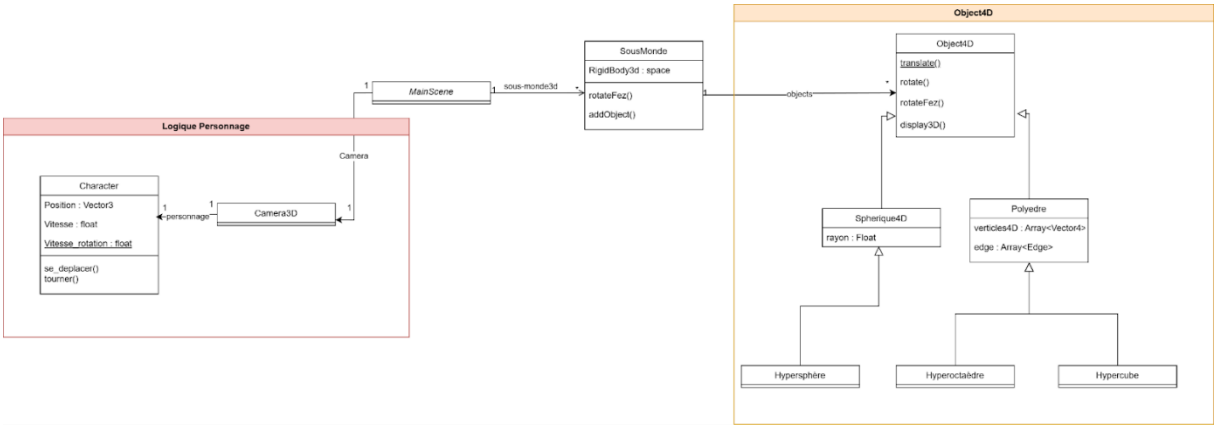
- **XROrigin** : Remplace le CharacterView de l'applciatiion bureau, XROrigin est adapté pour la VR, et permet de contrôler les inputs de l'utilisateur, de le déplacer, de bouger ses amin ect…

Technologies utilisées :

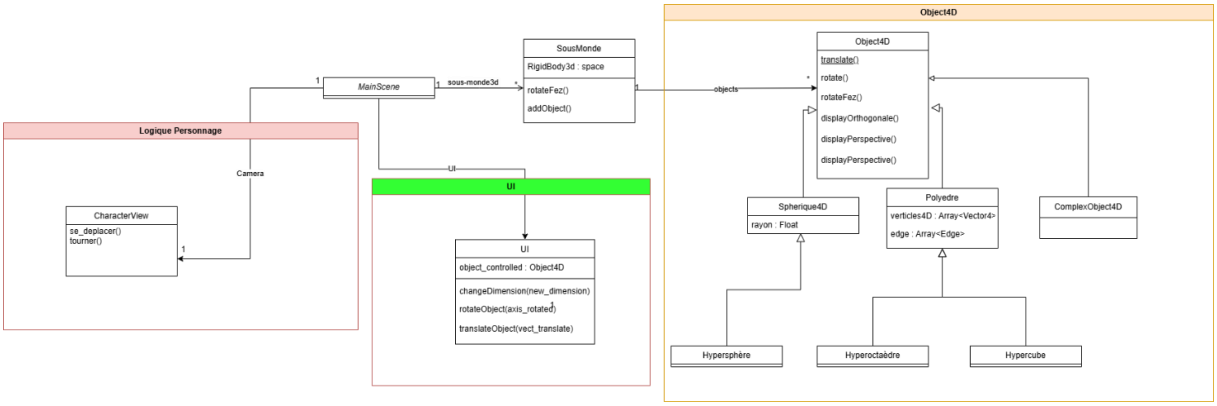
- **Godot** : Moteur de jeu
- **GDScript** : Langage de programmation
- **MetaQuest** 3 : Casque de réalité virtuelle

Diagrammes UML

Notre projet étant exploratoire, durant l'étude préalable nous avions seulement réalisé un diagramme UML pour représenter l'architecture de notre projet sur Godot :



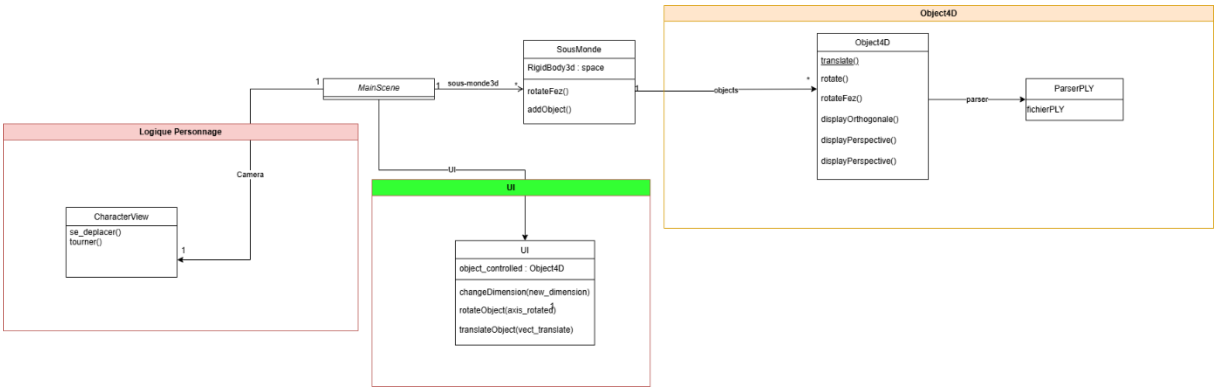
L'étude préalable ayant été réalisée lorsque nous n'avions encore aucune base en Godot, nous avions réalisé ce second diagramme lors de l'itération 4



On remarque une certaine cohérence avec ce que nous avions fait au début, mais un manque d'information, comme l'interface utilisateur, ou des fonctionnalités représentées différemment dans notre code actuel par rapport à ce que nous nous étions imaginés. Comme on peut le voir, nous avions choisi de réduire la logique personnage à seulement une caméra, puisque nous nous sommes rendu compte que pour la VR, il valait mieux contrôler un personnage à la

1ère personne qu'à la 3ème personne pour renforcer le sentiment d'immersion ainsi que la cohérence des commandes. A noter que nous avons rajouté la classe ComplexObject4D, elle représente en fait tous les objets 4D que nous souhaitons créer mais que ne sont pas polyèdres ou sphériques, ils hériteront alors directement de la classe Object4D. Et enfin nous avions rajouté la logique pour l'interface utilisateur.

Ce diagramme était déjà mieux mais pas totalement cohérent avec le projet final, voici donc le nouveau diagramme :



Ici ce que nous avons changé est la logique pour les objets 4D, Godot n'étant pas pratique pour la conception orienté objet, nous avons donc un script qui utilise un parser pour parser les informations du fichier PLY passé en paramètre et donc construire la figure en fonction des coordonnées.
Finalement les erreurs de conception que nous avions fait était de traiter Godot comme ce que nous avions vu avant et d'y appliquer la même logique de conception orienté objet, finalement le parser était la meilleur solution.

Réalisation

Architecture logicielle

Notre projet a été développé en utilisant Godot. Godot est un moteur de jeu qui a la particularité d'être complétement open source, permettant de créer des jeux 2D et 3D. Il fut un moteur de choix pour notre projet car il est facile à comprendre et à utiliser, ce qui fut important puisque nous ne savions pas utiliser de moteur de jeu. Il a donc fallu apprendre à l'utiliser ainsi que son langage GDScript qui est aussi très facile à apprendre puisqu'il possède fortes ressemblances à Python. Godot fonctionne avec des scènes et des nœuds, ce qui facilite la conception modulaire, et a l'avantage de toujours être facile à appréhender.

Développement des fonctionnalités clés

Affichage d'objets 4D

Notre application nous permet actuellement de visualiser plusieurs types d'objets 4D, les comme les hypercubes, hypersphère etc... Ces objets 4D peuvent être projeté en 3D via différentes techniques :

- **Projection orthogonale** : Si un point a comme coordonnée x y z w, on va le projeter sur l'espace 3D en fonction de ses coordonnées x y et z en oubliant le W. Cette projection garde les proportions mais n'a donc aucun effet de profondeur.
- **Projection perspective** : Consiste à projeter l'hypercube comme si on le regardait à travers une caméra placée dans un espace 4D. Il conserve l'effet de profondeur de l'hypercube mais ne garde pas les proportions. On peut le comparer à dessiner un cube sur une feuille, on projette un objet 3D sur un espace 2D.
- **Projection stéréographique** : C'est une autre manière de projeter un objet 4D en 3D. On projette les points de l'hypercube depuis un point spécifique sur un espace 3D. Cette projection permet de conserver les angles mais déforme fortement les distances à mesure que l'on s'éloigne du point de projection.

Vues

Il est possible de visualiser l'hypercube sous 3 vues qui sont les suivantes :

- Vue de face : C'est la vue où l'hypercube est aligné parallèlement aux axes 3D, En d'autres termes c'est la vue ou l'on ne peut voir qu'une des faces de l'hypercube parce que celle-ci éclipse toutes les autres. L'hypercube va donc apparaitre comme un cube à l'intérieur d'un autre cube relié par des arrêtes, cette vue permet une bonne compréhension de l 'hypercube
- Vue Point de fuite : Projection faite perpendiculairement à une face. Sous cette vue, le tesseract est formé de deux troncs de pyramide. Il est recommandé d'utiliser la projection perspective pour cette vue puisqu'on a besoin de la profondeur, cela marche cependant aussi avec une projection orthogonale.
- Vue par la pointe : On va orienter l'hypercube de manière que l'un de ses sommets soit orienté vers l'utilisateur. Toutes les autres parties vont être projetées autour de ce

point. Avec cette vue, le tesseract va être formé de quatre volumes hexaédriques entourant le sommet.

Styles de l'hypercube

Nous avons aussi donné à l'utilisateur le pouvoir de choisir sous quel « style » il préférait visualiser l'hypercube. Nous en avons conçu trois :

- **Wireframe** : Ce style va représenter l'hypercube uniquement avec ses arêtes sous forme de lignes. Chaque sommet est connecté par des segments, sans face pleine. Cela donne une apparence ou toutes les connexions entre les points sont visibles.
- **Stylish** : Ce mode d'affichage est une version améliorée du style wireframe en ajoutant des sphères aux sommets et des cylindres pour les arêtes, il a l'avantage d'être plus visible et surtout plus agréable à regarder mais un peu plus gourmand en ressources. Lors des transformations (rotations, translations), c'est le style donnant une impression « mécanique » des transformations.
- **Solid** : Ici, l'hypercube est représenté en volume avec des face pleines, chaque face est colorée d'une différente couleur ce qui permet de mieux différencier les différentes facettes de l'hypercube. Contrairement aux modes stylish et wireframe, on ne peut voir l'intérieur de l'hypercube car aucune transparence n'est appliquée aux faces. Attention ce style ne marche pour l'instant que pour l'hypercube, car nous n'avons pas eu le temps de calculer les coordonnées des faces pour chaque figure

Transformations dans un univers 4D :

Concernant les transformations en 4D, bien que les objets soient projetés en 3D, les transformations qu'on peut leur appliquer concernent bel et bien un espace en quatre dimensions. Nous avons implémenté les transformations suivantes :

- **Translation** : L'utilisateur a la possibilité de déplacer l'hypercube selon 4 axes (x, y, z, w) ce qui lui permet d'explorer sa structure et d'observer comment l'objet évolue dans l'espace 4D.
- **Rotation** : L'utilisateur peut appliquer des rotations simples (autour d'un seul axe) ou double (autour de deux axes simultanément) à l'hypercube, lui permettant d'observer les transformations complexes de l'hypercube et d'appréhender sa structure sous différents angles de rotation.
- **Fez** : Inspiré du jeu Fez, ce mode permet à l'utilisateur de basculer entre différentes dimensions. L'utilisateur peut changer sélectionner une nouvelle dimension disponible pour basculer dans celle-ci et donc observer l'hypercube sous une autre dimensions 3D

Objets réalisés :

Voici les différents objets que nous avons pu implémenter grâce au parser et au script ply

- **Hypercube (ou Tesseract)** : Polytope en 4D, analogue du cube. Il a 16 sommets, 32 arêtes, 24 faces carrées et 8 cellules cubiques.

- **Hypersphère (ou 3-sphère)** : Sphère en 4D. Tous les points sont à une distance fixe d'un centre dans l'espace 4D. Sa "surface" est un espace tridimensionnel fermé.
- **Tritriduoprisme (duoprisme 3-3)** : Polytope en 4D formé par le produit cartésien de deux triangles. Il possède 9 cellules prismatiques (6 prismes triangulaires + 3 prismes hexagonaux). Il est symétrique et torique.
- **Hexadécachore (16-cell)** : Polytope régulier en 4D composé de 16 tétraèdres. C'est le dual du tesseract. Il a 8 sommets, 24 arêtes et 32 faces triangulaires.
- **Hypercube empilé (16 hypercubes) :** Assemblage de 16 tesseracts pour former un hypercube plus grand, toujours en 4D.

Interface Utilisateur

- **Menu Principal** : Permet de choisir les objets à visualiser ainsi que leur méthode de projection et leurs styles d'affichage au démarrage de l'application.
- **UI dans le monde** : Permet d'interagir avec les objets 4D pour effectuer des transformations.
- **UI VR** : permet d'interagir avec les objets 4D mais adapté pour la VR, l'utilisateur la trouvera devant lui constamment en train de le suivre et l'UI s'activera quand l'utilisateur s'approchera d'un objet 4D

Tests et validation

Nous avons effectué plusieurs tests pour valider la projection correcte des objets 4D :

- **Comparaison entre les différentes méthodes de projection** : Nous avons vérifié la fidélité de chaque projection (orthogonale, perspective, stéréographique) en comparant les résultats obtenus à des représentations théoriques.
- **Vérification de la cohérence des transformations appliquées** : En effectuant des tests, nous avons vérifiés que les translations et rotation appliquées aux sommets de l'hypercube étaient cohérentes en termes de coordonnées et projections
- **Comparaisons avec un outil en ligne** : Lors de la veille technologique nécessaire pour notre étude préalable, nous avions découvert un outil en ligne permettant d'effectuer des transformations sur un tesseract. Nous avons donc comparé nos résultats graphiquement à ceux obtenus avec cet outil en ligne.
- **Aide au CharlyLab** : nous avons pu nous appuyer sur le personnel du Charly lab. notamment pour la VR pour nous aider et nous guider, leur aide à était précieuse pour l'implémentation et la réalisation de la VR.

Retours utilisateur

Pour tester l'interface utilisateur ainsi que les contrôle notamment pour la VR et vérifier qu'elle est facile et compréhensible pour tout néophyte, nous avons fait remplir une fiche à plusieurs personnes pour obtenir des retours sur l'interface utilisateur et les contrôles et les améliorer en fonction de ces derniers. Ces fiches nous ont permis de nous rendre compte par

exemple que les déplacements en VR était difficile à prendre en mains pour les personnes n'étant pas familière avec la VR

Difficultés rencontrées et solutions apportées

Compréhension de la 4D

L'un des plus grands défis de ce projet a été notre manque de connaissances sur la quatrième dimension. Contrairement aux dimensions 2D ou 3D, la quatrième dimension est complétement abstraite pour nous. En tant qu'humain, nous ne pouvons pas directement nous la représenter. Nous pouvons seulement nous représenter des projections de cette dernière. Il nous a donc fallu trouver les bonnes ressources pour appréhender ce concept, le comprendre et pouvoir avancer dans le projet.

Compréhension mathématique :

Manipuler des objets 4D, notamment pour les translations et rotations, requiert une compréhension avancée des mathématiques et de la géométrie, en particulier sur l'utilisation des matrices de transformation.

- Les matrices 4x4 sont essentielles pour représenter et appliquer des transformations linéaires dans l'espace 4D.
- La composition de rotations multiples implique des calculs complexes que nous avons dû tester et valider progressivement.
- Le fait de représenter la 4D en 3D était un grand défi pour nous. Au début, saisir les concepts des différentes projections (orthogonale, par perspective, stéréographique) était une étape essentielle.
- La représentation de ces données dans Godot, en GDScript, la façon dont les aborder était également une difficulté. Par exemple, comment représenter l'hypercube en termes de coordonnées ? Devons-nous stocker les faces ou les reconstruire à partir des sommets à chaque fois ?

Optimisation

Créer des objets 4D peut être couteux en performances, nous avons dû réserver une itération, la dernière de ce premier semestre, pour optimiser l'application du mieux possible, notamment pour les affichages d'objet en mode stylish qui consommait beaucoup de ressources et réduisait les performances de notre application. Dans cette même itération, nous avons également refactorisé du code, et fixé une multitude de bugs et warning relatifs aux manipulations des données dans notre code.

Implémentation VR

L'implémentation de la VR a sûrement été la plus grande difficulté à laquelle nous avons dû faire face. En effet, la plus grosse erreur que nous avons commise a été de croire que

l'implémentation serait relativement simple, qu'il suffirait de brancher le casque, de changer deux-trois choses, et que cela fonctionnerait. Mais cela n'a pas été le cas, et l'implémentation complète de la VR nous a occupés pendant la majeure partie des itérations 5, 6 et 7. Voici les différentes difficultés que nous avons rencontrées :

- **Lancer le projet en VR** :
  Cette première étape de l'implémentation est celle qui nous a pris le plus de temps. Il a fallu installer toutes les dépendances, suivre minutieusement les tutoriels, mais nous avons été confrontés à la relative nouveauté de Godot. En effet, les méthodes pour implémenter la VR avaient changé, et beaucoup de tutoriels n'étaient plus à jour. Nous avons heureusement pu nous appuyer sur le personnel du Charly Lab, qui nous a grandement aidés.

- **Implémentation des contrôles** :
  Nous avons également dû adapter les contrôles à la VR, ce qui n'a encore une fois pas été facile. Toute la logique d'interaction avec le monde a dû être repensée pour correspondre aux nouvelles contraintes.

- **Adaptation de l'interface** :
  Lorsque nous avons intégré notre première interface en VR, nous nous sommes rapidement rendu compte qu'elle n'était pas du tout adaptée. Nous avons donc dû modifier entièrement sa disposition et créer une nouvelle interface pensée spécifiquement pour la VR. Nous avons aussi rencontré des problèmes d'interaction, et avons dû faire en sorte que l'expérience utilisateur soit la plus fluide et intuitive possible.

En conclusion, l'implémentation de la VR a été très longue, et nous n'aurions sûrement pas réussi sans le soutien du personnel du Charly Lab. Mais maintenant que nous avons réussi à l'intégrer, le lancement en VR est devenu facile, Godot simplifiant cette tâche une fois la base bien mise en place.

Planning de déroulement du premier semestre

Lors de ce premier semestre, nous avons dû reconsidérer les planifications réalisées lors de l'étude préalable tout en gardant dans les grandes lignes notre plan global. Le planning prévu lors de la fin de l'itération 4 correspondait bien à ce qui a était réalisé pour les itérations 5 6 et 7.

1ère itération

Cette première itération, comme nous l'avions planifiée, était dédiée à notre apprentissage de Godot et son langage GDScript. Chaque membre du groupe a créé un jeu 2D pour se familiariser avec GDScript, les différentes structures de données, instructions possibles avec ce langage. Les jeux 2D ont également permis de découvrir le fonctionnement des nœuds de base, comment les lier entre eux, comment utiliser les signaux et évènements entre différentes entités. Pour poursuivre, chacun a ensuite créé un jeu en 3D pour intégrer une nouvelle notion de physique au développement du jeu, étant donné que nous étions assez familiers avec le moteur de jeu en lui-même. La gestion d'une nouvelle coordonnée a beaucoup impacté la physique, les graphismes et représentations d'objets dans le code, ce qui a été très utile pour la suite lorsqu'il était temps de commencer le développement du vrai projet.

2ème itération

Au moment de l'étude préalable et notre veille technologique et mathématique, nous avions pu découvrir les concepts clés de la 4D : Les transformations, projections, objets, etc.

Lors de cette deuxième itération, n'ayant pas encore assimilé précisément toutes ces notions, nous avons décidé de commencer la programmation simple d'un hypercube dans un espace 3D. Pour cela, il nous a fallu valider la compréhension des différentes projections de 4D en 3D. Le fait de pouvoir entamer la programmation nous a permis de nous rendre compte de la façon dont nous allions représenter un hypercube en termes de nœuds Godot, mais aussi en termes de script. Les Node4D n'existant pas, nous avons décidé de représenter un hypercube avec un Node3D, auquel on attacherait un script comportant les données réelles en 4D de l'hypercube.

3ème itération

Possédant une très bonne cohésion d'équipe et gestion de projet ainsi qu'une base de code suite à l'itération précédente, cette 3ème itération nous a permis de prendre une petite avance par rapport au planning que nous avions créé. Durant cette dernière, nous avons pu implémenter les translations, rotations simples et doubles, ainsi que l'interface utilisateur permettant non seulement d'effectuer ces transformations, mais aussi de générer les hypercubes de notre choix.

Cette itération a été décisive dans l'avancement de notre projet. C'est à partir de celle-ci que nous avions l'impression d'avoir une réelle application permettant de visualiser des tesseracts selon notre choix.

Pour un résumé condensé, voici toutes les fonctionnalités qui ont été réalisées :

- Transformations (rotations simples et doubles, translations)
- Interface utilisateur (menu principal, menu de transformation pour hypercube)
- Lien backend – UI (gestionnaire d'évènements, signaux…)
- Visualisation d'un hypercube sous différents styles (traits, cylindres & sphères, faces solides de couleur remplies).
- Sous-zones, pour donner cette impression de fenêtre sur le monde 4D depuis la 3D, cohérentes avec les transformations effectuées à l'hypercube.

4ème itération

Suite au départ de notre camarade, nous ne souhaitions pas réaliser beaucoup de nouvelle fonctionnalité lors de cette itération puisque nous avions compris qu'il fallait un produit fini et délivrable pour ce premier semestre. Ainsi, nous avons décidé de prioriser deux choses principales.

1. **L'interface Fez**, permettant d'effectuer des rotations sur des espaces pour visualiser différemment un objet 4D.
2. **L'optimisation**, **refactorisation** du code ainsi que le **fix** de certains **bugs**, **incohérences** dans le code qui auraient été traînés dans les itérations suivantes.

A noter que suite à une demande « ergonomique » et « pratique » de notre tuteur, nous avons également implémenté les points de vue pour cette itération. Pour rappel, ils consistent à pouvoir cliquer sur un bouton, dans l'interface utilisateur de transformation d'un hypercube, afin que le tesseract soit orienté selon le point de vue sélectionné (vue de face, vue de fuite, vue par la pointe).

Finalement, nous avons obtenu un produit fini, sans soucis de performance comparables à ceux rencontrés à la 3ème itération, et surtout sans bug. L'interface Fez fonctionnant parfaitement avec de petites animations. Nous pouvons remarquer que nous nous sommes plutôt tenus à notre étude préalable puisque toutes les fonctionnalités prévues pour ce premier semestre ont été réalisées, mis à part le fait de pouvoir visualiser d'autres objets que des tesseracts.

Avec une application optimisée et un code propre, nous sommes prêts à débuter cette deuxième phase de projet et remplir nos objectifs.

5ème itération

Lors de la fin de la 4ème itération nous avions obtenu une première démo de projet fonctionnel, la 5-ème itération devons donc rajouter une nouvelle fonctionnalité, quitte à ce que celle-ci ne soit pas fonctionnel à 100% lors de la fin de l'itération. Nous avons donc priorisé deux choses principales.

1. **L'empilement d'hypercube**, une figure composée de 16 hypercube qui ensemble forme un hypercube plus grand, comme lorsque nous empilons 4 cubes pour former un plus grand cube
2. **L'implémentation de la VR**, commencer à implémenter la VR pour que l'utilisateur puisse explorer la 4D en VR, nous ne savions pas exactement jusqu'ou nous pourrions implémenter la VR lors de cette itération

A noter que à la fin de l'itération prenant place la journée de prote ouverte de l'IUT Nancy charlemagne, journée à laquelle nous participions, l'objectif était donc d'avoir un produit fonctionnel avec la VR pour permettre aux personnes venues découvrir l'iut de tester la VR, étant toujours une expérience plus ludique et intéressante que la version bureau.

Finalement à la fin de cette itération nous avons réussi à implémenter une partie de la VR, c'est à dire que l'utilisateur pouvait se déplacer, visualiser des figures dans un monde que nous avions créer mais pas encore interagir avec puisque nous n'avions pas réussi à encore adapter l'interface utilisateur pour la VR

Nous avions aussi réussi à implémenter les hypercubes empilés, à noter que nous avons utilisé notre script hypercub.gd pour les créer, cette logique sera réutilisée pour la création du parser lors de l'itération 6. Si nous nous référons au planning créer lors de l'itération 4, nous l'avions plutôt bien suivi.

6ème itération

Pour cette itération, nous avions recueilli les retours d'utilisateur lors de la journée de porte ouverte, l'objectif de cette itération était donc d'adapter notre projet à ces retours

Voici la liste des fonctionnalités réalisés

- **Adaptation de l'UI pour la VR**, puisque nous n'avions pas pu le réaliser lors de l'itération 5, il convenait de réaliser cette fonctionnalité lors de cette itération pour permettre aux utilisateurs d'interagir avec les figures en VR
- **Création d'un parser de fichier PLY**, cela permettrait d'implémenter de nouvelle figure 4D beaucoup plus facilement, puisqu'ils nous auraient suffi d'un fichier ply avec les nouvelles coordonnées de la figure 4D pour la créer

A la fin de cette itération nous avions donc réussi à adapter l'UI pour la VR, nous avions aussi fait en sorte que l'UI soit « attachées » à la main de l'utilisateur pour faciliter l'interaction avec celle-ci

Nous avons aussi réussi à créer le Parser et adapter notre scène hypercube pour qu'elle marche avec toutes les figure tant qu'un fichier PLY était spécifié. A la fin de l'itération 6 nous avions donc le fichier

ply de l'hypercube, de l'hypersphère et de l'héxadecachore

7ème itération

Cette itération marquant la fin du projet tutoré, l'objectif était ismilaire à l'itération 4, avoir un produit finit sans bug, voici donc les fonctionnalités que nous avions réalises

- **Implémentation des nouvelles figures** sur la VR, ayant un projet bureau et VR séparé nous devons reporter els changement fait sur la version bureau à la version VR, nous voulions aussi agrandir le monde VR pour montrer ces nouvelles figures
- **Création des duoprisme 3-3**, nous voulions rajouter cette figure notre tuteur nous l'ayant demandé
- **Optimisation et correction des derniers bug**, pour avoir une version finie et correct

A la fin de cette itération nous avions donc obtenu un nouveau monde VR, avec les nouvelles figure, ainsi que le duoprisme 3-3.

L'optimisation et la correction des derniers bugs nous a permis d'obtenir une version du produit que nous jugeons terminé, bien que l'on puisse encore l'améliorer et la compléter, notre travail sur ce projet est quant à lui terminé.

Répartition du travail entre étudiants

Nous avons, comme expliqué lors de l'étude préalable décidé de séparer les responsabilités en désignant un chef de projet qui s'engagerait à prendre en charge les tâches d'organisation et planification, bien évidemment en consultant les autres membres du groupe.

Afin de garantir une coordination efficace et de maximiser la synergie au sein de notre petite équipe, nous avons décidé d'adopter une approche collaborative pour l'ensemble du projet. Dès le départ, nous avons convenu que chaque membre devait contribuer à l'ensemble des composantes du projet — qu'il s'agisse de la programmation, de l'interface utilisateur ou de l'optimisation — afin que chacun développe une compréhension globale du système et puisse intervenir en cas de difficulté dans n'importe quelle partie du code. Les scripts s'entremêlant et les dépendances étant mêlées, il est essentiel que chaque membre du groupe saisisse tous les concepts de sorte qu'il puisse intégrer sa solution/fonctionnalité sans perturber le reste du code.

Nom	**Rôle**	**Contributions clés sur le projet.**
KOMODZINSKI	Développeur	Déplacement de la caméra. Projection perspective. Rotations. Modes d'affichages.
RYSAK	Développeur	Projection orthogonale. Interface Utilisateur. Modes d'affichages. Interface Fez. Fix de bugs. Empilement d'hypercube. Implémentation de l'Interface Utilisateur en VR. Création du monde VR. Optimisation itération 7. Adaptation VR.
TROHA	Chef de projet / Développeur	Organisation et planification. Gestion des réunions et suivi du planning. Projection stéréographique. Translations. Lien backend UI. Modes d'affichages. Optimisation du code et fix de bugs. Implémentation VR. Création du parser PLY. Création des fichiers d'objet PLY. Création tritriduoprisme. Optimisation itération 7.

Conclusion

Ce projet tutoré nous a permis de relever un défi à la fois technique, scientifique et créatif. En partant d'une simple idée – rendre la quatrième dimension perceptible via une application immersive – nous avons progressivement construit une solution fonctionnelle, modulable, et surtout accessible. Le chemin parcouru a été semé de difficultés, notamment dans la compréhension des concepts mathématiques liés à la 4D ou encore dans l'implémentation de la réalité virtuelle, mais chaque obstacle a été l'occasion d'apprendre, de nous adapter et de progresser en équipe.

Grâce à notre approche méthodique et à notre esprit collaboratif, nous avons su tirer profit de nos compétences respectives, tout en élargissant notre champ de connaissances, tant sur le plan du développement que de la gestion de projet. Le soutien du Charly Lab (Antoine et Cyprien) a également joué un rôle essentiel dans la réussite de l'intégration de la VR.

Ce projet nous a aussi offert l'opportunité d'expérimenter des approches de modélisation avancées, notamment à travers l'utilisation de fichiers PLY et la conception d'un parser générique. Cette démarche nous a permis de structurer notre application autour d'une logique plus souple et évolutive, ouvrant la voie à l'ajout futur de nouvelles figures 4D sans refondre l'architecture du projet. Nous avons ainsi pu concilier exploration mathématique et contraintes techniques dans un cadre de développement concret.

Aujourd'hui, nous sommes fiers de livrer une application qui permet non seulement de visualiser et manipuler des objets en 4D, mais également de le faire dans un environnement immersif grâce à la réalité virtuelle. Même si le projet reste perfectible et pourrait accueillir encore de nombreuses améliorations, notamment en matière de figures supportées ou d'expérience utilisateur, nous considérons avoir atteint les objectifs fixés au début du projet.

Ce travail restera pour nous une expérience marquante, tant par la richesse des apprentissages que par la satisfaction d'avoir mené à bien un projet ambitieux et original, de la conception à la livraison finale.

Annexes

Perspectives de réutilisation du projet

Notre application, bien qu'aboutie et pleinement fonctionnelle, a été pensée dès le départ comme une base évolutive et modulaire. À ce titre, elle peut tout à fait être reprise et poursuivie par un autre groupe d'étudiants dans le cadre d'un nouveau projet tutoré.

Le code source est intégralement disponible sur GitHub à l'adresse suivante :
https://github.com/Stantrh/Projet_Tutore_4D_VR_KOMODZINSKI_RYSAK_TROHA
Il est commenté de manière à faciliter la compréhension de la logique générale du projet, bien qu'aucune documentation technique détaillée n'ait été formellement rédigée.

Nous recommandons aux futurs contributeurs de se familiariser dans un premier temps avec le moteur Godot et le langage GDScript, puis d'explorer les scripts principaux (notamment Hypercube.gd, parser.gd, et les scènes VR/Bureau) pour bien comprendre l'architecture globale du projet. Le dépôt Git contient l'ensemble des scènes, scripts et ressources nécessaires à l'exécution de l'application.

Plusieurs pistes d'amélioration peuvent être envisagées pour poursuivre ce projet :

- Fusion de plusieurs objets 4D pour construire des structures complexes (par exemple, créer des architectures 4D types « maison »).
- Refactorisation et modularisation du code pour faciliter l'ajout de nouveaux types de figures ou de styles d'affichage.
- Amélioration des performances de rendu, en particulier lors des transformations et dans les styles visuels plus gourmands comme le mode « Stylish » ou « Full ».
- Création d'un véritable éditeur d'univers 4D permettant à l'utilisateur de placer et assembler des figures à la volée.
- Optimisation du rafraîchissement dans les fonctions _process pour gagner en fluidité.
- Documentation technique et tutoriels pour améliorer l'accessibilité du projet aux prochains groupes.

Nous pensons que ce projet constitue une base solide pour développer une application éducative, pédagogique ou même scientifique autour de la quatrième dimension. Conçu à la fois pour des néophytes curieux de découvrir des objets 4D de manière ludique, mais aussi pour des chercheurs ou enseignants souhaitant analyser des comportements géométriques après différentes transformations ou projections, ce projet offre un fort potentiel d'extension et d'adaptation selon les usages.

Guide d'installation et utilisation

Installation

Prérequis

- Godot Engine version **4.3** (le projet a été réalisé sous Godot 4.3)
- Un casque VR **Meta Quest 3** (pour la version VR, facultatif si vous souhaitez tester uniquement la version bureau)
- Git (pour cloner le dépôt)

Étapes

1. Cloner le dépôt :
   ```
   git clone https://github.com/Stantrh/Projet_Tutore_4D_VR_KOMODZINSKI_RYSAK_TROHA.git
   ```
2. Ouvrir le projet dans Godot :
   a. Lancer Godot
   b. Cliquer sur « import »
   c. Naviguer jusqu'au dossier sélectionné
   d. Sélectionner le fichier « project.godot »
   e. Cliquer sur « Open », puis « Edit »
3. Lancer le projet :
   a. Pour la version Bureau : Lancer la scène « main_menu_world »
   b. Pour la version VR : lancer la scène « main_space_vr.tscn »
   c. Appuyer sur **F6** (ou cliquer sur « Play current scene »)

Mode d'emploi

Version Bureau

1. **Menu principal**
   a. Choisir les figures (Hypercube, Hypersphère, etc.)
   b. Choisir le style d'affichage : Wireframe, Stylish ou Full
   c. Choisir la projection : Orthogonale, Perspective, Stéréographique
   d. Cliquer sur "Lancer" pour démarrer la scène
2. **Scène principale**
   a. Cliquer sur la sous-zone (verte) d'un objet pour afficher son menu d'interactions
   b. Utiliser les sliders à gauche pour déplacer la figure selon les axes X, Y, Z, W
   c. Cliquer sur les boutons à gauche pour appliquer des rotations (deux plans à la fois maximum)
   d. Utiliser le menu Fez pour explorer la figure dans un autre sous-espace 3D
   e. Choisir un point de vue prédéfini (vue de face, par la pointe, etc.)

Version VR

1. **Préparer le casque**
    a. S'assurer que le casque est bien connecté au PC via AirLink ou câble USB
    b. Vérifier que SteamVR est désactivé (le projet utilise le plugin natif OpenXR de Godot)
2. **Lancer la scène** « main_space_vr.tscn »
    a. Le monde VR comporte 3 îles, chacune contenant des figures différentes à explorer
3. **Contrôles & interactions**
    a. Déplacements libres via la main droite en faisant un signe de pistolet (lever le pouce en l'air et l'index pointant devant soi, les deux doigts étant perpendiculaires). Faire un geste de pression sur la détente du pistolet imaginaire pour se déplacer. A noter qu'il faut que la main soit visible par le casque, et que les flèches partant de la main droite soient bien vertes.
    b. Interactions avec un objet en se rapprochant de ce dernier. Une fois proche de l'objet, lever la main gauche à hauteur du casque pour faire apparaître le menu d'interactions. Utiliser les flèches le plus sur les côtés pour changer de vue (Rotations, translations, Fez, etc.). A noter qu'il faut interagir avec cette UI avec la main droite puisque la main gauche est chargée de la faire apparaître.

Démonstration & Tutoriel d'utilisation

Une vidéo de démonstration de l'utilisation bureau ainsi que VR est disponible ici :

https://youtu.be/h_rhzZH0R5c