

RAPPORT DE PROJET

PROJET L3B1 : RUBIK'S CUBE



MATHÉMATIQUES ET INFORMATIQUE

Sciences

Université Paris Cité

Rapport de projet

Projet L3B1 : Rubik's Cube

Les informations d'identification du document :

Référence du document :	Rapport_Projet_L3B1
Version du document :	1.1.0
Date du document :	20/04/2023
Auteurs :	MBAYE Fatima Lucie LATTAB Nassim PENN Clément BU Éric

Les éléments de vérification du document :

Validé par :	
Validé le :	
Soumis le :	23/04/2023
Type de diffusion :	Document électronique (.pdf)
Confidentialité :	Réservé aux étudiants UFR Maths-Info de l'université Paris cité

Les éléments d'authentification :

Maître d'ouvrage :	Chef de projet :
Date / Signature :	Date / Signature :

Sommaire

1. Introduction	4
1.1. Objectifs du document.....	4
1.2. Documents de référence.....	4
2. Objectifs du projet.....	5
3. Fonctionnalités du produit.....	5
3.1. Fonctionnalités obligatoires	5
3.2. Fonctionnalités optionnelles	6
3.3. Fonctionnalités facultatives	6
4. Organisation et gestion du projet.....	7
4.1. Conception et développement.....	7
4.2. Répartition des tâches	8
5. Outils utilisés.....	9
5.1. Langage de programmation, IDE et moteur de jeu	9
5.2. Gestionnaire de Version.....	11
6. Difficultés rencontrées et solutions.....	12
6.1. Sauvegarde du cube.....	12
6.2. Autres cubes.....	13
7. Conclusion	15
7.1. Apports pédagogiques.....	15
7.2. Poursuite et suggestions d'amélioration	15
8. Glossaire.....	16
9. Références	17
10. Index.....	18
11. Résumé	19

Table des figures

<i>Figure 1 : Écran de jeu avec le cube 3x3.....</i>	<i>5</i>
<i>Figure 2 : Exemple d'une figure de couleur pour le cube 3x3.</i>	<i>6</i>
<i>Figure 3 : Ajout de plusieurs cubes.....</i>	<i>7</i>
<i>Figure 4 : Cycle de vie en V.....</i>	<i>7</i>
<i>Figure 5 : Diagramme de GANTT du projet.</i>	<i>8</i>
<i>Figure 6 : Visual Code : la méthode permettant le mélange des cubes (en C#).....</i>	<i>9</i>
<i>Figure 7 : Unity 3D : Interface du menu.</i>	<i>10</i>
<i>Figure 8 : Dossier trunk du svn.</i>	<i>11</i>
<i>Figure 9 : Dossier branches du svn.....</i>	<i>11</i>
<i>Figure 10 : Dossier tags du svn.....</i>	<i>12</i>
<i>Figure 11 : Le cube miroir résolut.</i>	<i>13</i>
<i>Figure 12 : Le cube miroir mélangé.</i>	<i>13</i>
<i>Figure 13 : Screenshot du cube de taille 4x4.....</i>	<i>14</i>
<i>Figure 14 : Algorithme "layers by layers".</i>	<i>16</i>

1. Introduction

Dans le cadre du projet de troisième année de licence Informatique et Applications à l'Université Paris Cité, nous avons été amenés à concevoir et développer une application. Cette dernière a pour objectif de proposer un Rubik's Cube 3D manipulable ainsi que de fournir une aide à la résolution pour ce célèbre casse-tête.

Ce projet, encadré et dirigé par Monsieur LATTAUD, réunit une équipe de quatre développeurs dans le but de créer une application intuitive et facile à utiliser qui permettra aux utilisateurs de résoudre différentes tailles de Rubik's Cube.

Cette équipe est composée des étudiants suivants :

- BU Éric
- LATTAB Nassim
- MBAYE Fatima Lucie
- PENN Clément

Le présent rapport a pour objet de présenter le projet L3B1, qui vise à répondre aux besoins et aux objectifs définis au préalable. Nous commencerons par exposer les concepts fondamentaux du projet afin de mieux comprendre les enjeux et les objectifs visés.

Ensuite, nous décrirons les fonctionnalités du produit final, ainsi que les outils et méthodes utilisés pour sa mise en place. Nous présenterons également l'organisation du projet, en mettant en avant les différentes étapes clés de son déroulement.

Nous aborderons ensuite les difficultés rencontrées et les solutions que nous avons mises en place pour y faire face. Nous discuterons également des améliorations potentielles qui pourraient être apportées à notre application dans le futur.

Enfin, nous conclurons en mettant en évidence les acquis et les apprentissages que nous avons tirés de cette expérience.

1.1. Objectifs du document

Ce document a pour objectif de retracer l'ensemble des étapes clés de notre projet, en rappelant les points importants de tous les documents rendus précédemment. En complément, nous parlerons des nouveaux éléments ajoutés qui ne figurent pas dans les autres documents. De plus, une partie du document sera dédiée aux outils qui nous ont servi au développement de notre application. Le but étant d'enrichir cette rétrospective et de donner une vision globale de l'évolution du projet depuis ses débuts jusqu'à aujourd'hui.

1.2. Documents de référence

Vous trouverez ci-dessous une liste des documents rédigés qui ont servi à l'élaboration de notre projet :

- **Étude de marché** : Vise à analyser les produits existants dans le but de comprendre les besoins et les attentes des utilisateurs.
- **Cahier des charges** : Définit les besoins, les objectifs, les contraintes du projet, ainsi que les fonctionnalités attendues, les critères de qualité et les délais de réalisation.
- **Cahier de recette** : Décrit l'ensemble de tests à effectuer avant la livraison du produit afin de s'assurer de la conformité de ce dernier.
- **Diagramme de GANTT** : Permet de planifier et visualiser les différentes tâches du projet, leur durée, leur enchaînement et la répartition des tâches au sein de l'équipe de développement.
- **Plan de développement** : Détaille les différentes étapes du développement ainsi que les ressources nécessaires utilisées pour chacune d'elles.

- **Manuel d'utilisation** : Facilite la prise en main du produit livré en décrivant de manière détaillée la façon dont l'application doit-être utilisée.
- **Manuel d'installation** : Guide l'utilisateur dans les étapes à suivre pour l'installation et la configuration de notre produit.
- **Documentation interne** : Permet de garantir une cohérence et une transparence du code source de l'application, en fournissant une description détaillée de ce dernier dans le but de faciliter sa compréhension, sa maintenance et son évolution.
- **Plan de test** : Sert à vérifier que le produit à livrer fonctionne correctement afin de garantir sa conformité. On ne peut pas dire "produit livré" car le plan de test se fait avant livraison du produit.

2. Objectifs du projet

L'élément principal de notre application est un Rubik's Cube 3D manipulable librement. L'utilisateur a la possibilité de mélanger le cube par lui-même ou de se servir du mélange automatique proposé par l'application. Concernant la résolution, l'utilisateur peut décider de résoudre par lui-même le casse-tête ou de faire appel à l'aide à la résolution mise à disposition. Dans le but d'améliorer l'expérience de l'utilisateur, l'application offre d'autres fonctionnalités tels qu'un chronomètre ou encore la possibilité de réaliser des figures de couleurs.

3. Fonctionnalités du produit

Dans cette section, nous allons expliciter chaque fonctionnalité de notre produit en prenant soin de les classer selon leur importance : obligatoires, optionnelles ou facultatives.

3.1. Fonctionnalités obligatoires

L'objectif ici était de proposer un cube en 3D manipulable dans l'espace avec la possibilité de faire pivoter chacune de ses faces. Nous devons également proposer à l'utilisateur d'afficher s'il le souhaite une aide à la résolution du cube.

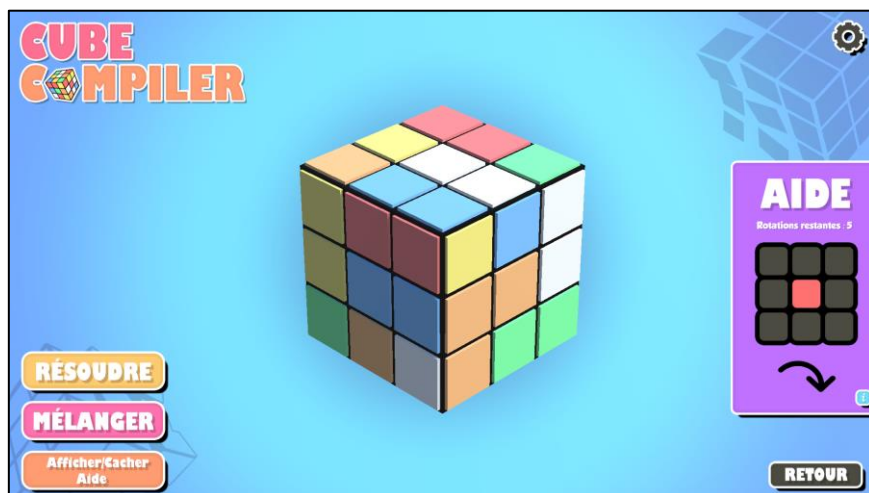


Figure 1 : Écran de jeu avec le cube 3x3.

3.2. *Fonctionnalités optionnelles*

Il nous a été suggéré initialement de proposer à l'utilisateur des options à savoir la mise en place d'un chronomètre et la possibilité de réaliser des figures de couleurs.

La page d'accueil propose donc un bouton "Options" donnant accès à l'interface des options.

Cette interface est divisée en différents sous-menus :

- Le sous-menu "Figures" offrant à l'utilisateur différentes figures de couleurs à réaliser pour le cube 3x3.
- Le sous-menu "Gameplay" offrant à l'utilisateur la possibilité d'utiliser un chronomètre en jeu qu'il peut démarrer ou arrêter lorsqu'il le souhaite. Bien-sûr, le chronomètre s'arrête automatiquement si l'utilisateur parvient à résoudre le cube, enregistrant un nouveau temps. Ce temps peut aussi apparaître en tant que meilleur temps si l'utilisateur bat son précédent record.



Figure 2 : Exemple d'une figure de couleur pour le cube 3x3.

3.3. *Fonctionnalités facultatives*

En plus des fonctionnalités optionnelles viennent s'ajouter d'autres fonctionnalités dites "facultatives". Ces fonctionnalités ont été ajoutées en complément des fonctionnalités optionnelles et n'étaient pas prévues à la base.

Dans le menu, vous pourrez y retrouver le bouton "Crédits" qui affiche brièvement les informations ainsi que le contexte dans lequel s'inscrit ce projet informatique. Vous retrouverez également un bouton "Statistiques" affichant dans une nouvelle interface le meilleur temps de résolution pour chacun des cubes.

Dans les options, le sous-menu "Cubes" offre à l'utilisateur la possibilité de sélectionner des Rubik's Cube de tailles différentes : initialement le 3x3 (sélectionné par défaut), mais aussi le 2x2 et le 4x4 que nous avons implémenté par la suite. En ce sens, le sous-menu "Figures" propose à l'utilisateur de choisir parmi 6 figures de couleurs différentes en fonction du cube sélectionné.

Par ailleurs, le sous-menu "Gameplay" offre en plus du chronomètre, la possibilité à l'utilisateur de modifier le nombre de rotations à effectuer lors du mélange automatique (entre 1 et 30 rotations, par défaut à 20).

Finalement, les dernières options ajoutées ont été la possibilité de résoudre le Rubik's Cube avec l'algorithme de résolution couche par couche, et la possibilité d'ajouter ou enlever des faces miroir au cubes courant.

Enfin, le dernier sous-menu “Audio” permet à l'utilisateur de modifier à sa guise le volume du thème et des effets sonores.

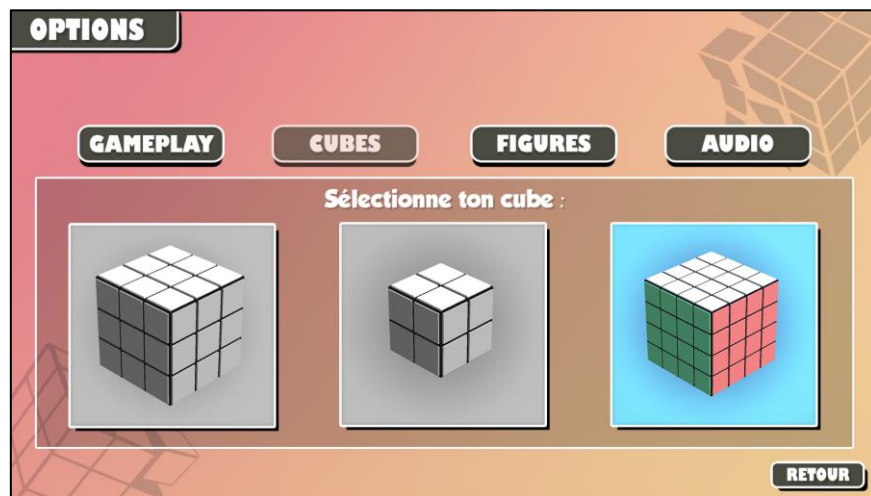


Figure 3 : Ajout de plusieurs cubes.

4. Organisation et gestion du projet

Dans cette section, nous allons expliciter les étapes de conception et de développement de l'application ainsi que la manière dont nous nous sommes réparti les tâches à effectuer.

4.1. Conception et développement

Dans le cadre de notre projet, nous avons opté pour une méthode dite de cycle en V. Ce cycle vise à garantir la qualité et la conformité de l'application. Il s'agit d'une méthode en cascade, qui consiste en une série de phases successives, allant de la conception à la validation en passant par le développement et les tests.

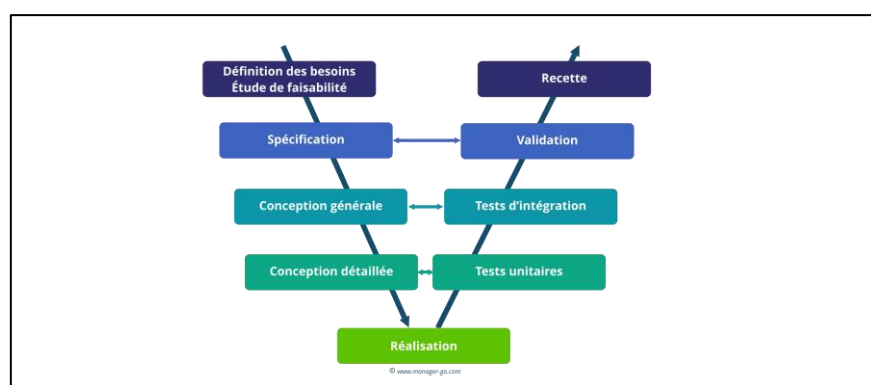


Figure 4 : Cycle de vie en V.

D'autre part, la phase de conception et développement est une étape cruciale dans tout projet informatique car elle permet de définir les objectifs et les exigences du projet, ainsi que les spécifications techniques nécessaires à la réalisation de l'application. Ainsi, les documents de conception permettent de formaliser les différentes étapes du projet, d'organiser les idées, les fonctionnalités, et les besoins, et de déterminer les contraintes ainsi que les ressources nécessaires.

Il a donc fallu rédiger en amont de la phase de développement un ensemble de documents de conception nécessaires afin de fournir une base solide et cohérente pour la réalisation de l'application. Ces documents ont permis de garantir la qualité, la fiabilité et la conformité de l'application, tout en visant à faciliter la maintenance et l'évolution ultérieure de l'application pour les années à venir. Ces documents sont explicités dans la partie "Documents de référence".

La phase de développement s'appuie donc en grande partie sur ces documents et implique aussi l'utilisation d'outils et de technologies spécifiques que nous expliciterons plus tard dans la partie "Outils utilisés". Afin d'atteindre nos objectifs, nous avons développé notre application par petites étapes successives, en répartissant les tâches dans un ordre précis. Grâce à cette séquence d'étapes bien définies, nous avons pu établir une ligne directrice solide pour la création de notre application. Cette répartition sera explicitée dans la partie suivante.

Enfin, les réunions hebdomadaires avec notre encadrant, considéré comme le client, nous ont permis de suivre l'avancement du projet en temps réel tout en prenant ses remarques en considération.

4.2. Répartition des tâches

La répartition des tâches est une étape importante du développement d'une application. Elle permet d'optimiser le temps et les compétences des membres de l'équipe ainsi que de donner une certaine chronologie au développement de notre application. Ainsi, certaines tâches ont une priorité plus grande que d'autres tâches. Par exemple, il est clair que les fonctionnalités facultatives sont à développer en dernier lieu. La planification des tâches est associée à un diagramme de GANTT explicité dans la partie "Documents de références".

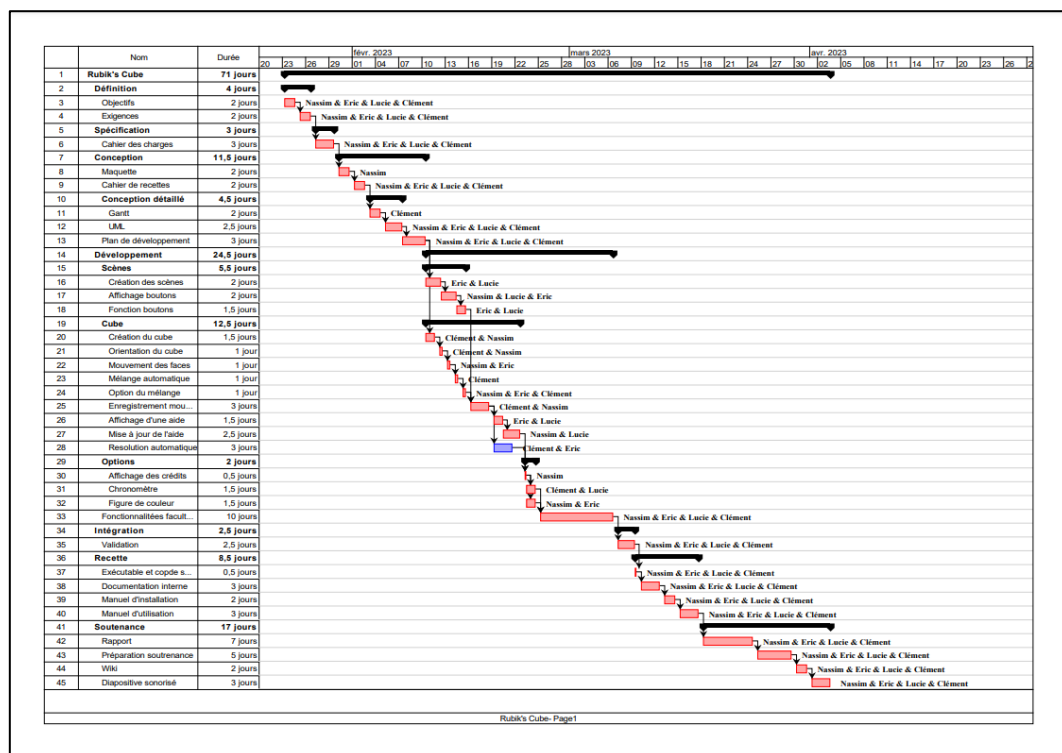


Figure 5 : Diagramme de GANTT du projet.

Au sein de notre équipe, nous avons choisi de distribuer les tâches en tenant compte des préférences et des compétences de chacun de nos membres. Ainsi, chaque tâche est attribuée à un ou plusieurs membres de l'équipe suivant ses compétences et sa motivation. Cette approche nous permet d'optimiser l'efficacité de chacun.

Nonobstant, nous avons fait en sorte de garantir une répartition équitable des tâches au sein du groupe. En outre, cette méthode nous a permis de concentrer et d'optimiser les forces et les atouts de chaque membre de l'équipe. Cette approche favorise ainsi la cohésion et la collaboration de notre groupe.

Cependant, il était également important de s'assurer que chaque membre de l'équipe ait une charge de travail équilibrée. Par ailleurs, nous avons fait en sorte de permettre à chacun de contribuer à des tâches de nature différentes afin de rester polyvalent et ainsi acquérir de nouvelles compétences, tout en se familiarisant avec les différents aspects du développement du projet.

5. Outils utilisés

Dans le cadre de notre projet informatique, nous avons dû utiliser à la fois le langage de programmation C#, l'IDE (Integrated Development Environment) Visual Studio et le moteur de jeu Unity 3D. Ce fut un défi car nous devions acquérir une compréhension complète de ces nouveaux outils, leur utilisation étant une exigence essentielle du projet.

Pour le développement, une contrainte a également été d'utiliser le gestionnaire de version SVN.

5.1. Langage de programmation, IDE et moteur de jeu

Le langage de programmation C#, est un langage de programmation orienté objet développé par Microsoft. Il est largement utilisé pour le développement d'applications sur la plateforme Unity 3D. Le C# offre une syntaxe moderne et puissante, ainsi que de nombreuses fonctionnalités avancées pour faciliter le développement d'applications complexes.

Visual Studio, est un IDE développé par Microsoft qui est devenu très populaire dans la communauté des développeurs. Il offre un large éventail de fonctionnalités avancées pour faciliter le développement de logiciels, y compris la prise en charge de plusieurs langages de programmation dont C#, des fonctionnalités d'édition avancées ou encore différents outils de débogage.

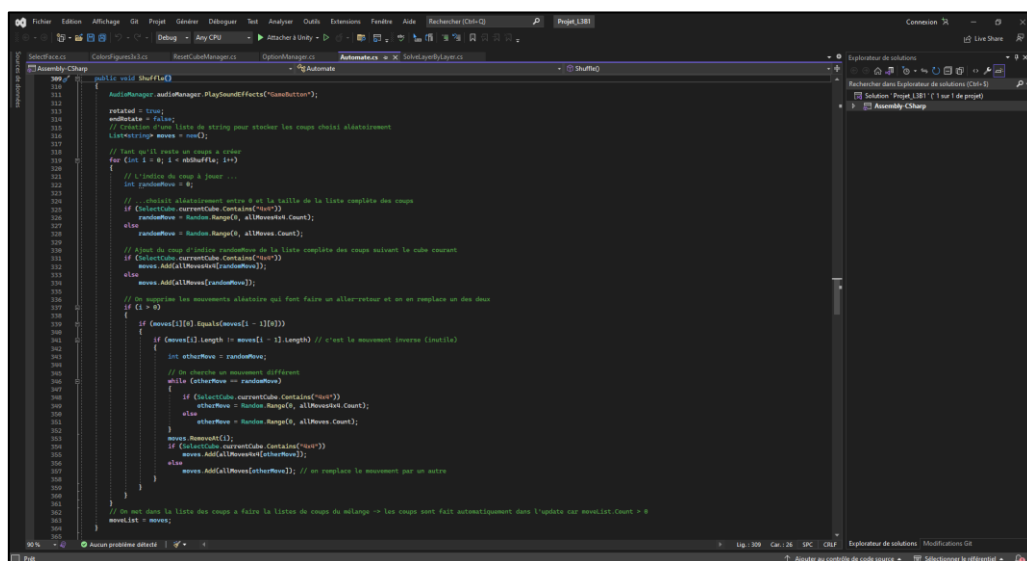


Figure 6 : Visual Code : la méthode permettant le mélange des cubes (en C#).

Unity 3D est un moteur de jeu très populaire qui permet de développer des applications interactives. Il offre de nombreuses fonctionnalités pour la création de jeux, tels que la gestion des graphismes, la physique, l'animation, les effets sonores, ainsi que des outils de développement et de débogage avancés. L'un des avantages clés d'Unity 3D est sa flexibilité et sa convivialité pour les développeurs. Il prend en charge plusieurs langages de programmation, dont C#, pour la création de fonctionnalités de jeu personnalisées.

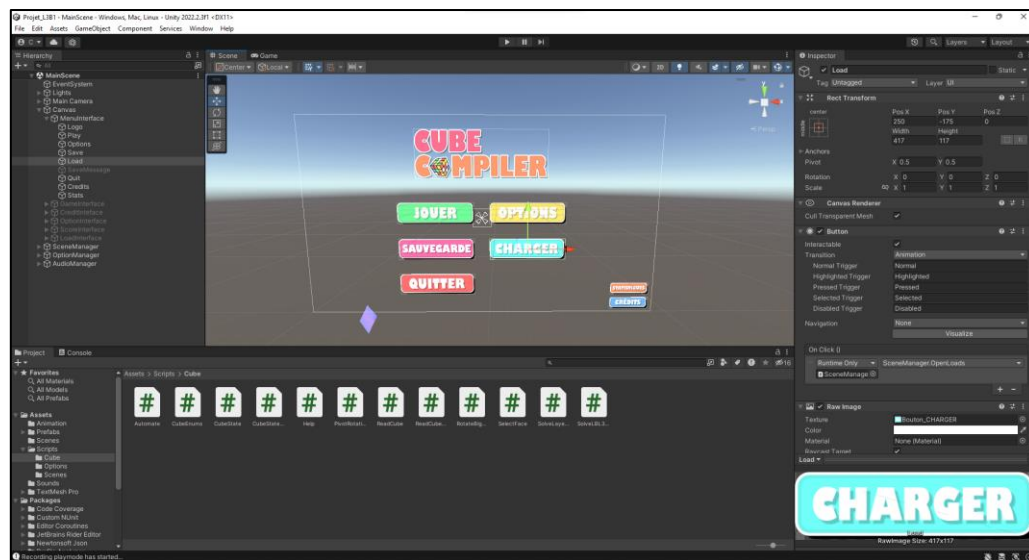


Figure 7 : Unity 3D : Interface du menu.

L'un des avantages les plus notables de l'utilisation du C#, Visual Code et Unity 3D pour notre projet est la capacité à les utiliser simultanément. Cela nous a permis de développer efficacement notre application sur Unity 3D pour la physique des objets tout en utilisant les fonctionnalités avancées du C# pour la programmation des fonctionnalités du jeu, et les outils puissants de Visual Code pour le développement, le débogage et la gestion des scripts sur Unity.

De plus, C# offre une syntaxe claire et expressive qui nous a permis de développer du code propre, efficace et maintenable pour notre application. La richesse des fonctionnalités avancées du C#, telles que l'héritage ou les événements, nous a permis de développer facilement des fonctionnalités complexes pour notre application.

Quant à Visual Studio, son interface utilisateur conviviale et ses nombreuses fonctionnalités d'édition avancées, comme la complétion automatique du code et la navigation intelligente, ont grandement facilité notre flux de travail lors du développement.

L'utilisation du C# et Visual Studio pour notre projet sur Unity 3D s'est donc avérée être un choix judicieux, malgré notre ignorance sur le sujet. Leur utilisation simultanée nous a permis de développer efficacement notre application tout en exploitant les fonctionnalités avancées du C# pour la programmation des fonctionnalités du jeu et les outils puissants de Visual Studio pour le développement et la gestion du code source. Leur syntaxe claire, leurs fonctionnalités avancées et leur convivialité ont grandement contribué à la réussite de notre projet informatique.

5.2. Gestionnaire de Version

Le gestionnaire de version SVN, également connu sous le nom de Subversion, est un outil essentiel dans le domaine du développement logiciel. Il permet de gérer les versions de fichiers et de dossiers, ainsi que de faciliter la collaboration entre les membres d'une équipe de développement.

Le fonctionnement de SVN repose sur un système centralisé, ici géré par l'Université, où un serveur détient le référentiel central de code source. Les développeurs peuvent alors effectuer des check-out pour obtenir une copie locale du code, apporter leurs modifications et effectuer des commit pour les intégrer au référentiel central. SVN offre également des fonctionnalités telles que la fusion de branches et la gestion des conflits, ce qui facilite la coordination des travaux de développement parallèles.

Le SVN permet une gestion efficace des versions, avec la possibilité de revenir à des versions antérieures, de gérer les branches pour le développement de fonctionnalités isolées et de fusionner les modifications.

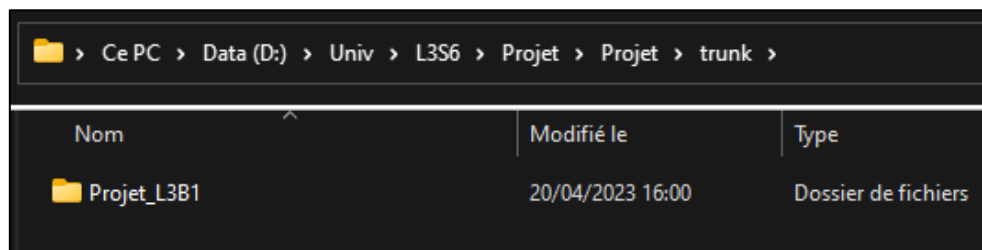
Cependant, SVN présente également quelques inconvénients. Par exemple, son système centralisé peut entraîner des conflits de fusion complexes, en particulier lorsque plusieurs développeurs travaillent sur la même partie du code.

Le gestionnaire de version SVN est donc un outil puissant pour la gestion des versions de code source et la collaboration en équipe. Ses fonctionnalités en font un choix populaire pour de nombreux "petits" projets de développement logiciel. Cependant, ses limitations en termes de conflits de fusion peuvent être des inconvénients potentiels pour de gros projets.

Finalement, SVN nous a permis de partager le code source de l'application et de déposer différentes versions du code. Le gestionnaire de version étant séparé en trois parties :

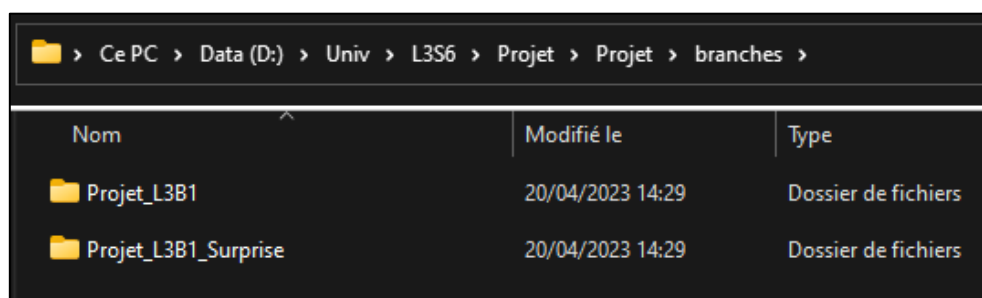
- Le dossier trunk : Contenant la dernière version stable du code
- Le dossier branches : Contenant les versions courantes sur lesquelles nous travaillons
- Le dossier tags : Contenant les anciennes versions stables du code

Cette méthode de pilotage de projet a grandement facilité le partage des données et la cohésion du groupe, favorisant ainsi un développement optimal de notre application.



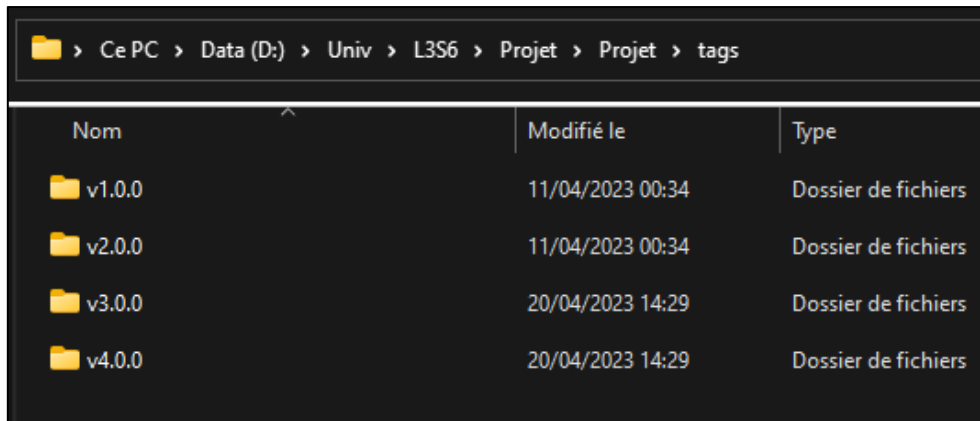
Nom	Modifié le	Type
Projet_L3B1	20/04/2023 16:00	Dossier de fichiers

Figure 8 : Dossier trunk du svn.



Nom	Modifié le	Type
Projet_L3B1	20/04/2023 14:29	Dossier de fichiers
Projet_L3B1_Surprise	20/04/2023 14:29	Dossier de fichiers

Figure 9 : Dossier branches du svn.



Nom	Modifié le	Type
v1.0.0	11/04/2023 00:34	Dossier de fichiers
v2.0.0	11/04/2023 00:34	Dossier de fichiers
v3.0.0	20/04/2023 14:29	Dossier de fichiers
v4.0.0	20/04/2023 14:29	Dossier de fichiers

Figure 10 : Dossier tags du svn.

6. Difficultés rencontrées et solutions

Lors de la création de notre application, nous avons dû relever plusieurs défis techniques. Dans cette partie, nous aborderons les difficultés rencontrées ainsi que les solutions mises en place pour les surmonter.

6.1. Sauvegarde du cube

Au sujet des fonctionnalités facultatives, il était envisagé d’implémenter une sauvegarde permettant de garder en mémoire l’état courant du cube sélectionné afin de pouvoir le recharger ultérieurement lors du lancement de l’application. La première chose à faire était donc de pouvoir “écrire” la configuration que l’on souhaite sauvegarder dans un fichier. C’est là qu’apparaît la notion de sérialisation.

La sérialisation en programmation orientée objet est le processus de transformation d'un objet en une suite d'information qui peut être stockée ou transmise sur un réseau. Cette suite d'information peut être un flux d'octets, une chaîne de caractères ou tout autre format de données.

L'objectif de la sérialisation est de conserver l'état de l'objet afin de pouvoir le restaurer plus tard, soit dans la même application, soit dans une application différente. La sérialisation est utilisée dans de nombreuses situations, notamment pour le stockage de données, le partage de données entre des applications, la communication entre des composants distants d'un système, la mise en cache d'objets, etc.

En général, la sérialisation en orienté objet implique la conversion de l’état interne d’un objet en une séquence d’octets (binaire). Inversement, la désérialisation implique la reconstruction de l’objet original à partir de cette séquence d’octets. Cette opération peut être effectuée de manière automatique par des bibliothèques de sérialisation fournies par le langage de programmation.

En prenant cela en compte, nous avons pu sérialiser notre objet (ici, notre cube) et le sauvegarder quelque part dans un fichier d’extension “.dat”. En général, les fichiers ".dat" peuvent être utilisés pour stocker des données importantes, telles que des fichiers de configuration système, des sauvegardes de fichiers, ou des informations de base de données.

Une fois cela fait, il était facile de recréer notre objet à partir de ces données. Cependant, un objet sans script ne fait rien. En effet, un objet a besoin d'au moins un script afin d'effectuer une tâche ou un comportement. Or, la sérialisation de scripts étant complexe en raison de la nature dynamique de ces derniers et la diversité des attributs, il était difficile de pouvoir sauvegarder chacun d'eux et de les recharger par la suite tout en les rattachant au cube. En ce sens, la sérialisation a permis de sauvegarder l'état de notre cube afin de pouvoir le recréer ultérieurement mais sans aucun script associé, ce qui rendait le cube sauvegardé inutilisable. Nous n'avons à l'heure actuelle pas trouvé de solution à ce problème complexe que nous laissons ouvert aux autres futurs groupes de développeurs qui nous succéderont.

Finalement, une méthode plus simple a été considérée. L'idée est non pas de sérialiser le cube et ses scripts mais simplement la liste des mouvements effectués. Il suffit donc de charger la liste et d'effectuer chacun des mouvements à l'envers sur notre cube afin de retrouver son état sauvegardé.

6.2. *Autres cubes*

Après avoir implémenté toutes les fonctionnalités obligatoires et optionnelles, la première difficulté à laquelle nous avons été confrontés a été de trouver quoi faire après avoir implémenté avec succès le Rubik's Cube de taille 3x3. Très vite, nous avons pensé à créer de nouveaux cubes, notamment d'abord en introduisant de nouvelles tailles. C'est pour cela que nous avons commencé par ajouter le cube 2x2 à notre application.

L'implémentation du cube 2x2 a présentée quelques défis. En effet pour plus de facilité nous avons voulu utiliser les scripts que nous avons déjà créés pour le Rubik's Cube 3x3, mais nous avons dû les adapter pour tenir compte des différences de taille et de fonctionnement du cube 2x2. Heureusement, grâce à une bonne compréhension des concepts de base de la manipulation de Rubik's Cube et à notre expérience précédente, nous avons pu surmonter ces défis et ajouter avec succès le cube 2x2 à notre application.

Après avoir implémenté avec succès le cube 3x3 et le cube 2x2, nous avons réalisé que simplement créer des cubes de différentes tailles pourrait ne pas être suffisamment stimulant d'un point de vue de la créativité et de la complexité du projet. Nous avons alors cherché à ajouter une nouvelle dimension en introduisant l'idée d'un cube miroir.

Un cube miroir est une variation du Rubik's Cube traditionnel dans laquelle les stickers colorés sur les faces du cube sont remplacés par des surfaces réfléchissantes de différentes tailles.



Figure 11 : Le cube miroir résolu.



Figure 12 : Le cube miroir mélangé.

Cependant, lors de l'implémentation, nous avons rencontré des défis techniques liés aux raycasts (rayon utilisé pour détecter les faces du cube), ce qui rendait le cube miroir trop compliqué à mettre en œuvre sans tout revoir depuis le début et le temps de développement restant ne nous le permettait pas. Nous avons donc dû trouver une alternative pour maintenir un niveau approprié de complexité dans notre projet, ce qui nous a conduit à choisir de créer un cube de taille 4x4 à la place.

Un cube de taille 4x4 est tout aussi complexe car il apporte une nouvelle difficulté, il possède 6 faces de plus. Cela a posé problème car les scripts que nous avons développés pour les cubes de taille 2x2 et 3x3 ne pouvaient pas être directement adaptés pour le cube 4x4 en raison de ces nouvelles faces dites "faces internes" au cube. Pour surmonter ce défi, nous avons opté pour la duplication de certains scripts et leur adaptation spécifique pour n'être utilisés que par le cube 4x4.

Cela nous a permis de résoudre les problèmes de manipulation des faces internes du cube 4x4 et d'ajouter avec succès cette fonctionnalité à notre application.

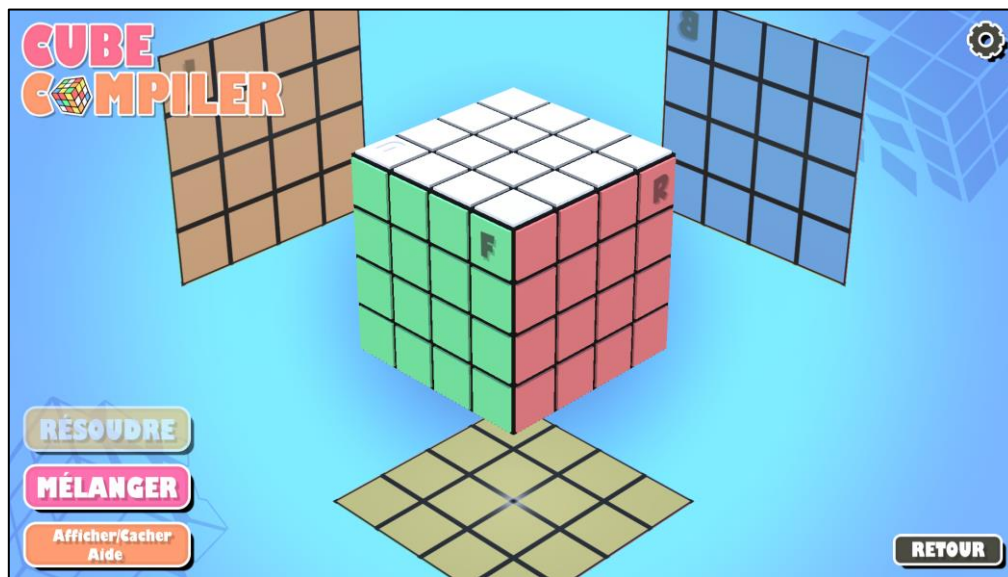


Figure 13 : Screenshot du cube de taille 4x4.

7. Conclusion

7.1. *Apports pédagogiques*

Dans le cadre de notre projet étudiant, nous avons pu tirer plusieurs apports pédagogiques enrichissants.

Tout d'abord, ce projet nous a permis de nous projeter dans le monde professionnel en nous confrontant à un projet de groupe concret avec un objectif à atteindre et des contraintes à respecter. Cela nous a également sensibilisés à l'importance du travail d'équipe et de la supervision par un client, qui ont été des éléments clés dans la réussite de notre projet.

Une autre dimension importante de notre projet a été la découverte de nouvelles technologies, notamment le langage de programmation C# et l'environnement de développement Unity 3D. Ces nouvelles compétences techniques acquises tout au long du projet sont un atout indéniable pour notre future carrière professionnelle, car elles nous ont permis de développer nos compétences en programmation, en gestion de projet et en résolution de problèmes.

7.2. *Poursuite et suggestions d'amélioration*

Dans le cas où le projet serait reconduit dans le futur, plusieurs éléments pourraient être ajoutés pour améliorer l'expérience des utilisateurs.

Tout d'abord, l'ajout de nouvelles tailles de Rubik's cubes, comme le 5x5, permettrait aux joueurs de s'entraîner sur des cubes plus complexes et ainsi de progresser dans leur capacité à résoudre les casse-têtes. L'intégration de faces "miroirs" permettrait aux joueurs de visualiser les faces qui ne sont pas directement visibles et ainsi permettre une représentation totale du cube. De plus, il serait également intéressant d'explorer la possibilité d'intégrer de nouveaux types de cubes plus complexes, comme le cube miroir, ou la pyramide, qui présentent un défi supplémentaire en raison de leur configuration unique. Enfin, l'implémentation d'autres méthodes de résolution peut être intéressante tel que la méthode CFOP qui est la méthode sur laquelle se base la résolution couche par couche que nous avons déjà implémentée. Cette méthode est très populaire en speedcubing, le sport qui consiste à résoudre le cube le plus rapidement possible, ce qui pourrait permettre d'attirer des utilisateurs plus aguerris.

Il s'agit là d'une liste non-exhaustive d'éléments à ajouter qui pourra s'élargir avec de nouvelles idées toujours plus créatives afin d'offrir une expérience de jeu encore plus intéressante et captivante.

8. Glossaire

Algorithme de résolution par retour en arrière : Le premier algorithme de résolution utilisé pour notre projet a été l'algorithme dit par retour en arrière. Cet algorithme prend en compte chaque mouvement effecteur sur les faces du cube et si ce mouvement ne participe à la résolution alors l'inverse de ce mouvement est ajouté sur le dessus la liste des mouvements à faire pour résoudre le cube.

Cet algorithme assez, cependant quelques subtilités sont à savoir, l'algorithme vérifie à chaque mise à jour de liste que des coups ne soient pas inutiles. Par exemple faire tourner 3 fois la même face dans la même direction correspond en réalité à faire tourner cette même face dans le sens inverse.

Algorithme de résolution couche par couche : Le second algorithme de résolution utilisé a été l'algorithme couche par couche. Cet algorithme intelligent peut résoudre n'importe quelle configuration de cube 3x3 en 7 étapes.

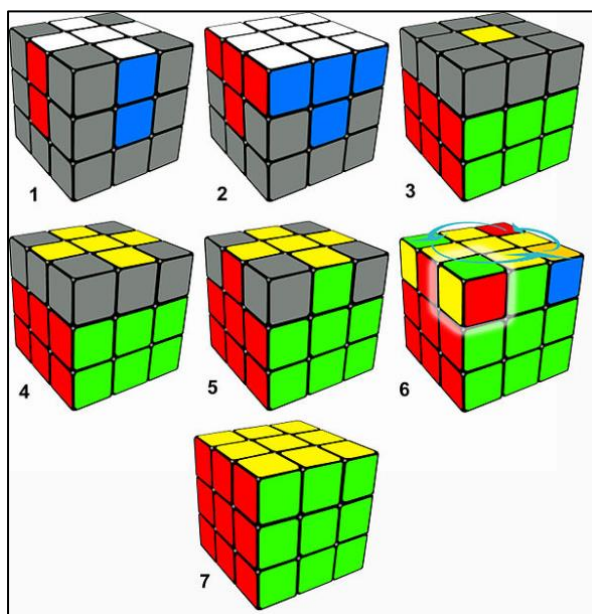


Figure 14 : Algorithme "layers by layers".

Étape 1 : Faire la croix blanche.

Étape 2 : Finir la première couche.

Étape 3 : Faire la deuxième couche.

Étape 4 : Positionnement de la croix jaune.

Étape 5 : Permutation des arêtes jaunes.

Étape 6 : Positionnement des coins jaunes.

Étape 7 : Orientations des coins jaunes.

Une fois que toutes ces étapes sont finies le cube est résolu.

Cet algorithme fut le premier algorithme introduit au grand public. C'est l'algorithme de résolution rapide et intelligent le plus simple à comprendre et à appliquer. Cependant il est vrai qu'il existe plusieurs autres algorithmes plus rapides notamment utilisés pour faire des records de vitesse.

Figure de couleurs : Les figures de couleurs sont des résolutions différentes. Le but est ici de faire apparaître des dessins en jouant avec les faces de couleurs des cubes. Il en existe de toutes sortes, des plus ou moins compliquées à réaliser ou des plus difficiles que d'autres.

Les cubes : Le cube de taille 2x2 également appelé « Pocket Cube » est introduit pour la première fois en 1982. Il est développé par Erno Rubik et est conçu pour être une version plus simple que le cube 3x3.

Le Rubik's Cube de taille 4x4, également connu sous le nom de « Rubik's Revenge », a été inventé par Erno Rubik. Il a été créé en 1982 et est une version plus complexe du Rubik's Cube original de taille 3x3.

Le cube miroir, également connu sous le nom de « Rubik's Mirror Cube », a été inventé par Hidetoshi Takeji, un sculpteur et designer japonais, en 1999.

9. *Références*

Voici quelques références bibliographiques sur le Rubik's Cube et ses secrets :

- « The Simple Solution to Rubik's Cube » de James G. Nourse
- « Speedsolving the Cube: Fridrich, Roux, Zborowski, and More » de Dan Harris
- « The Rubik's Cube: A Toy, a Puzzle, a Mathematical Model » de Michael Reid
- « The Rubik's Cube: From Simple Twist to Complex Math » de Tony Fisher
- « The Rubik's Cube: A Cultural Symbol » de Keith Devlin

Ces ouvrages couvrent divers aspects du Rubik's Cube, allant de la résolution rapide à l'analyse mathématique en passant par l'histoire culturelle de ce célèbre casse-tête.

Quelques références utilisées lors du développement de notre application :

- <https://www.megalomobile.com/lets-make-and-solve-a-rubiks-cube-in-unity/> : Projet similaire au notre qui nous a aidé sur les premières fonctionnalités.
- <https://kewbz.fr/blogs/tips-tricks/cool-3x3-rubiks-cube-patterns?shpxid=bfe679a6-b19e-443e-9efa-b57223a2c74d> : Figures de couleurs cube 3x3.
- <https://thedukeofcubes.com/3x3-cube-patterns/> : Figures de couleurs cube 3x3.
- <https://thedukeofcubes.com/2x2-cube-patterns/> : Figures de couleurs cube 2x2.
- <https://thedukeofcubes.com/4x4-cube-patterns/> : Figures de couleurs cube 4x4.
- https://cubingcheatsheet.com/algs4x_patterns.html : Figures de couleurs cube 4x4.
- https://en.wikipedia.org/wiki/Layer_by_Layer : Algorithme de résolution intelligente.

10. Index

A

Aide (à la résolution)4, 5, 17, 19
Algorithme6, 16, 17

C

C#9, 10, 15
Cycle de vie7, 19

D

Documentation5
Développement4, 7, 8, 9, 10, 11, 14, 15, 17, 19

F

Fonctionnalité4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 17, 19

G

Gantt4, 8, 19

I

IDE9
Informatique4, 6, 7, 9, 10

R

Rubik's Cube4, 5, 6, 13, 15, 16, 17, 19

S

Scripts10, 13, 14

T

Test4, 5, 7

U

Unity 3D9, 10, 15, 17

11. *Résumé*

Dans le cadre de notre 3ème année de Licence, un projet de fin d'études nous a été proposé. Ce projet est à réaliser en groupe de 4 et encadré par un professionnel. Notre groupe est composé de BU Éric, LATTAB Nassim, MBAYE Fatima Lucie et PENN Clément et encadré par M. LATTAUD.

L'objectif est de proposer un Rubik's Cube 3D manipulable ainsi que de fournir une aide à la résolution pour ce célèbre casse-tête. Ce projet qui a pour but de nous faire entrer dans le monde professionnel nous à poser quelques problèmes. Notamment, la difficulté ici a été de s'adapter à des technologies qui ne nous étaient pas forcément familières.

Pour ce projet, nous avons travaillé sur un cycle de vie en V et avons utilisé un Gantt pour une planification et une répartition efficace du travail. Lors du développement qui a duré 8 semaines, nous avons pendant les 2 premières semaines implémentées les fonctionnalités obligatoires. Les 3 suivantes, les fonctionnalités optionnelles. Puis enfin les 3 dernières semaines, les fonctionnalités facultatives.

Ce projet nous a permis de valider nos acquis ainsi que nous améliorer sur de nombreux points. Il nous a permis de découvrir le monde professionnel au travers d'un projet tout autant professionnel. Enfin, ce projet a pour nous été très intéressant à travailler et nous pensons qu'il peut être amélioré sur bien des aspects si le projet venait à être reconduit.

As part of our 3rd year of the Bachelor's degree, we have been offered a final year project. This project is to be carried out in groups of 4 and supervised by a professional. Our group is composed of BU Éric, LATTAB Nassim, MBAYE Fatima Lucie and PENN Clément and supervised by Mr. LATTAUD.

The objective is to propose a 3D Rubik's Cube that can be manipulated and to provide a solution for this famous puzzle. This project, which aims to make us enter the professional world, posed some problems. In particular, the difficulty here was to adapt to technologies that were not necessarily familiar to us.

For this project, we worked on a V-shaped life cycle and used a Gantt for efficient planning and distribution of work. During the 8-week development period, we implemented the required features during the first 2 weeks. The next 3 weeks, the optional features. The last 3 weeks we implemented the facultative features.

This project allowed us to validate our knowledge and to improve on many points. It allowed us to discover the professional world through an equally professional project. Finally, this project was for us very interesting to work on and we think that it can be improved on many aspects if the project was to be renewed.