Projet : Applications Web Graphiques Interactives
Projet de programmatin 2

Al Mallouhi Mohamed Satea, Dayioglu Gürgün, El Gargouri Mariem, Ouadi Salma

L3 Informatique

Faculté des sciences

Université de Montpellier

2022-2023

Problème : Développement et hébergement de six applications distinctes sur un site web.

Au début, le projet a été soigneusement planifié pour structurer les tâches et les délais. Pour chaque application, un énoncé et des questions ont été élaborés afin de faciliter la compréhension du projet. Ensuite, le développement collaboratif des applications a été effectué en s'adaptant aux besoins et défis rencontrés au cours du processus. L'équipe a travaillé ensemble pour résoudre les problèmes complexes et optimiser la qualité des applications. Par la suite, un site web accueillant les applications a été conçu. Enfin, un rapport exhaustif sur l'ensemble du processus a été rédigé.

Table des matières

Introduction et contexte

Le domaine de l'informatique graphique a connu une croissance rapide ces dernières années, notamment en raison de l'essor des applications Web, des jeux vidéo et de la réalité virtuelle. Dans ce contexte, notre projet vise à programmer six applications Web graphiques interactives avec animations, en abordant différents aspects de la programmation graphique.

L'intérêt de ce projet pour notre parcours d'études est de consolider et approfondir nos compétences en informatique graphique, en nous permettant de travailler sur des problèmes concrets et variés. Par ailleurs, la réalisation de ce projet a une portée plus large pour le monde de l'informatique, car il contribue à former des spécialistes dans un domaine en pleine croissance et en constante évolution.

Approches possibles et choix

Plusieurs approches sont possibles pour résoudre les problèmes posés par les exercices de ce projet. Parmi elles, on peut citer l'utilisation de bibliothèques graphiques telles que Three.js, WebGL ou Unity pour la création des applications Web. Chacune de ces solutions présente des avantages et des inconvénients en termes de complexité, de performance et de flexibilité. Pour ce projet, nous choisirons d'utiliser Three.js, qui offre une bonne combinaison de simplicité et de puissance, tout en étant largement utilisée et documentée.

Cahier des charges détaillé

Le projet est divisé en trois parties principales :

1. **Programmation des applications** : Les six applications Web graphiques interactives à programmer aborderont les thèmes suivants :
   — Lecture et affichage d'objets 3D décrits au format .gltf
   — Transformations, caméras et projections
   — Couleurs et éclairage
   — Textures
   — Animation
   — Interaction
2. **Création du site Web** : Un site Web est créé pour héberger les applications réalisées. Il comporte des explications sur le code source des programmes réalisés pour chaque exercice.
3. **Mesures subjectives** : Au cours du semestre, des mesures subjectives sont recueillies pour évaluer :
   — Les niveaux de difficulté des exercices
   — L'évolution du niveau de programmation de chacun
   — Le temps nécessaire pour réaliser ces exercices, en relation avec les niveaux de difficulté et l'évolution du niveau de programmation

Conclusion

Ce projet nous permettra d'acquérir et de renforcer des compétences en informatique graphique à travers la programmation de six applications Web graphiques interactives abordant divers aspects du domaine. En utilisant l'approche basée sur la bibliothèque Three.js, nous pourrons nous concentrer sur les problèmes spécifiques posés par chaque exercice tout en profitant des avantages offerts par cette bibliothèque.

La création du site Web associé au projet permettra de partager les résultats obtenus et de fournir des explications sur les choix de programmation réalisés. Enfin, la collecte de mesures subjectives aidera à adapter les exercices proposés en fonction des besoins et des progrès des étudiants, favorisant ainsi un apprentissage plus efficace et adapté à chacun.

1 Développements Logiciel : Conception, Modélisation, Implémentation

Exercice 1 : Un aperçu de canvas

La première application présentée dans ce rapport utilise des technologies web pour dessiner des formes, afficher du texte et charger des images sur un canvas HTML5. Cette application est développée en JavaScript, offrant ainsi une grande flexibilité et une facilité de mise en œuvre. Le canvas HTML5 permet de créer des graphiques de haute qualité, offrant ainsi une expérience utilisateur immersive et interactive.

L'application se compose de plusieurs modules clés. Tout d'abord, *le module de gestion* du canvas permet à l'application de créer et de manipuler un canvas HTML5, offrant ainsi une surface de dessin pour les autres modules. Ensuite, *le module de dessin des formes* permet à l'utilisateur de créer une variété de formes telles que des cercles, des rectangles et des lignes, offrant ainsi une grande flexibilité dans la création de graphiques.*le module de dessin du texte* permet quant à lui d'écrire du texte sur le canvas avec différentes polices, tailles et couleurs. *le module de chargement et l'affichage de l'image* permet d'importer des images sur le canvas, offrant ainsi une possibilité de personnalisation accrue. Enfin, *le module de création de gradients et d'ombres* offre une fonctionnalité avancée pour créer des effets visuels tels que des dégradés et des ombres portées.

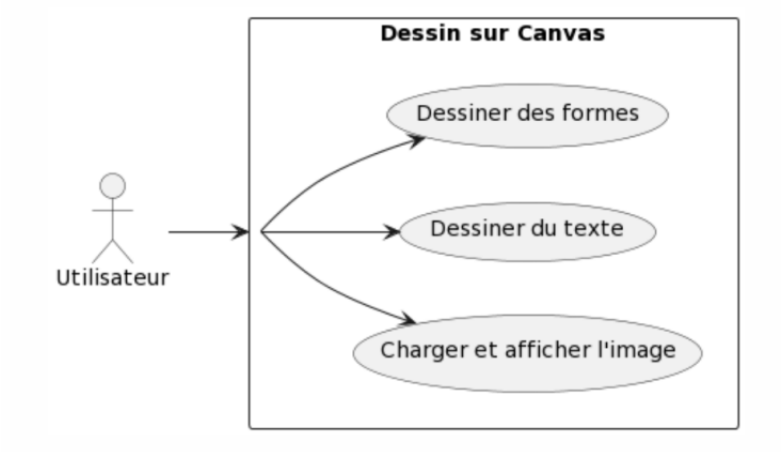

FIGURE 1 – Diagramme de cas d'utilisation

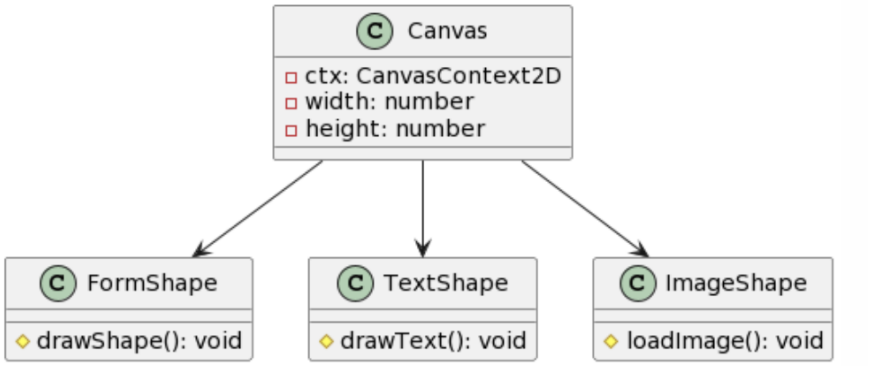

FIGURE 2 – Diagramme de classes

Notre code utilise l'élément canvas HTML5 pour dessiner des formes, du texte et charger une image. Il n'y a pas d'interface graphique interactive, mais plutôt une représentation visuelle dessinée directement sur le canvas pour cette application.

Concernant les format des données en entrée et procédures de lecture et validation des entrées, notre code ne traite pas directement les entrées utilisateur. Les dimensions du canvas, les couleurs, les positions des objets dessinés et l'URL de l'image sont définis directement dans le code. Les procédures de lecture et de validation des entrées ne sont pas applicables dans ce cas.

Le code de cet exercice est constitué d'un seul script JavaScript contenant environ 60 lignes de code, sans répartition en modules ou en classes distinctes. Les principales fonctionnalités sont regroupées en un seul fichier et ne sont pas divisées en composants individuels.

Exercice 2 : Paint

Notre deuxième application est un tableau de dessin simple sur une page Web, qui offre des outils pour dessiner, modifier la couleur et l'épaisseur du trait, effacer le tableau et exporter les points dessinés au format JSON.

En ce qui concerne les développements logiciels, le logiciel est développé en JavaScript en utilisant une combinaison d'écouteurs d'événements et de manipulation du contexte 2D du canvas HTML. Cette approche permet une grande flexibilité et une facilité d'implémentation pour offrir une expérience utilisateur fluide et agréable.

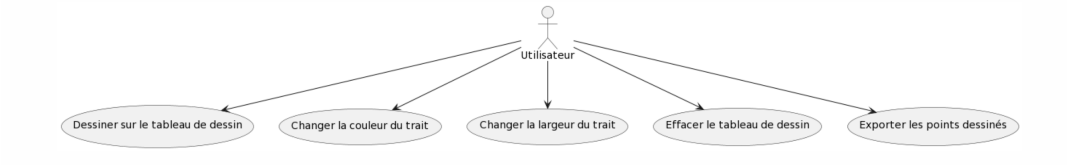

FIGURE 3 – Diagramme de cas d'utilisation

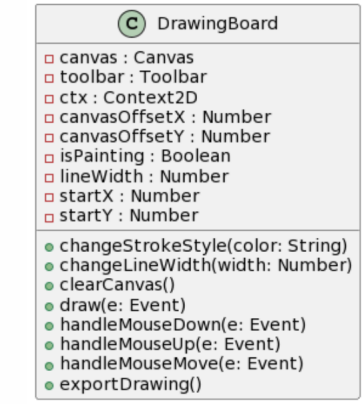

FIGURE 4 – Diagramme de classes

Sous-tendue par un canvas HTML et une barre d'outils, l'interface graphique permet à l'utilisateur de sélectionner la couleur et l'épaisseur du trait, effacer le canvas, et dessiner en maintenant le bouton de la souris enfoncé et en déplaçant la souris.

La flexibilité du canvas HTML permet à l'utilisateur de créer une grande variété de dessins, allant des formes simples aux illustrations complexes. La barre d'outils, quant à elle, offre une interface intuitive pour sélectionner les couleurs et les épaisseurs de trait. Grâce à cette combinaison, l'interface graphique offre une expérience utilisateur riche et immersive, permettant aux utilisateurs de créer des dessins de haute qualité avec une grande facilité et une flexibilité accrue.

Dans cet exercice, les données en entrée sont les interactions de l'utilisateur avec l'interface graphique. Les événements de souris sont utilisés pour détecter le début et la fin du dessin ainsi que le mouvement de la souris. Les valeurs de couleur et d'épaisseur du trait sont récupérées à partir de la barre d'outils à l'aide d'écouteurs d'événements.

Ici, le code est constitué d'un seul script JavaScript contenant environ 55 lignes de code, sans répartition en modules ou en classes distinctes. Les principales fonctionnalités sont regroupées en un seul fichier et ne sont pas divisées en composants individuels.

Implémentation de l'algorithem 1$

Dans le cadre de cet exercice, nous avons développé une application permettant de reconnaître des gestes dessinés sur une surface de dessin. L'application utilise l'algorithme One Dollar pour comparer les gestes dessinés aux gestes

préenregistrés dans un ensemble de modèles. L'application offre également la possibilité d'ajouter de nouveaux modèles et d'exporter l'ensemble des modèles sous forme de fichier JSON.

Notre application est composée des modules suivants :

- Point : Cette classe représente un point dans un espace 2D et propose des méthodes statiques pour calculer la distance entre deux points et la longueur d'un chemin constitué de points.
- Gesture : Cette classe représente un geste composé de points. Elle propose des méthodes pour effectuer des opérations telles que la mise à l'échelle, la translation, la rotation et la rééchantillonnage d'un geste.
- Recognizer : Cette classe est chargée de la reconnaissance des gestes. Elle utilise l'algorithme One Dollar pour comparer un geste dessiné à un ensemble de modèles et retourne le modèle le plus proche.
- Script principal : Ce module gère l'interaction utilisateur avec l'application, notamment la saisie des gestes, la reconnaissance des gestes, l'ajout de nouveaux modèles et l'exportation des modèles.

Quant à l'interface graphique de l'application, elle comprend les éléments suivants :

- Un canvas où l'utilisateur peut dessiner des gestes avec la souris ou le doigt (sur les appareils tactiles).
- Un bouton "Recognize" pour lancer la reconnaissance du geste dessiné.
- Un bouton "Clear Canvas" pour effacer le canvas et recommencer un nouveau dessin.
- Un bouton "Add Template" pour ajouter le geste dessiné comme un nouveau modèle.
- Un bouton "Export Templates" pour exporter l'ensemble des modèles sous forme de fichier JSON.

Au cœur de l'application, les données en entrée sont constituées de coordonnées de points (x, y) capturées lors du dessin d'un geste sur le canvas. Ces coordonnées sont stockées dans un tableau de points, qui est utilisé pour créer un objet Gesture. La qualité des dessins dépend directement de la qualité des points entrés, qui doivent être précis et cohérents pour créer des dessins fluides et précis.

Pour garantir la qualité des données en entrée, ces points sont soumis à une validation minutieuse. Chaque point est vérifié pour s'assurer qu'il s'agit bien d'un objet de la classe Point avec des coordonnées x et y valides. Cette vérification garantit que les points entrés sont conformes aux attentes du logiciel, ce qui permet de réduire les erreurs et d'améliorer la précision des dessins.

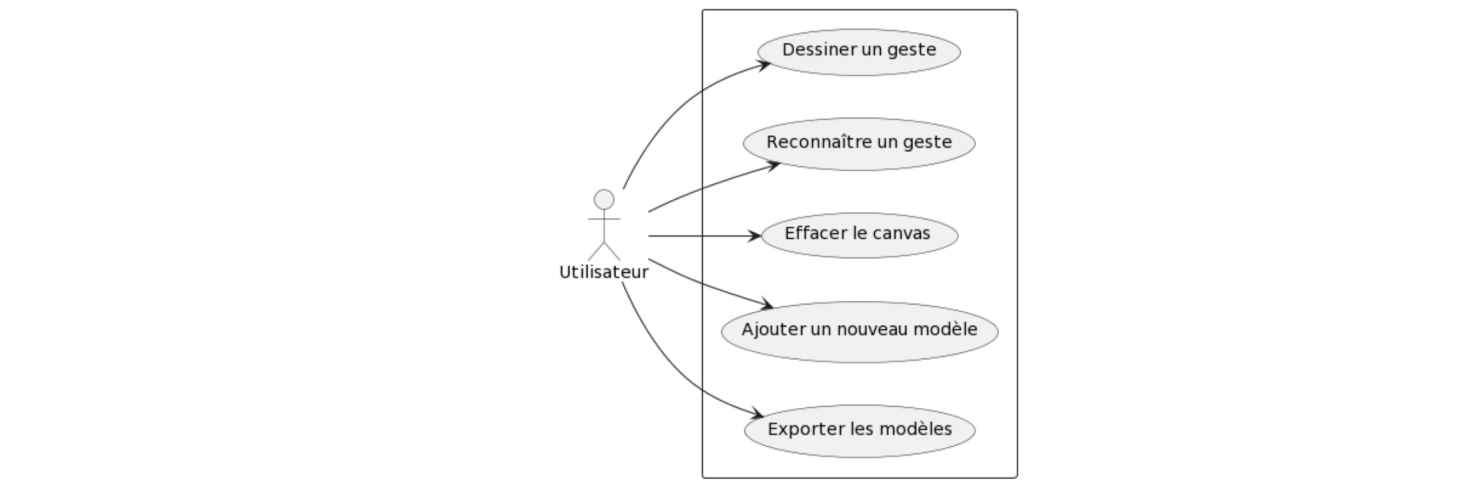

FIGURE 5 – Diagramme de cas d'utilisation

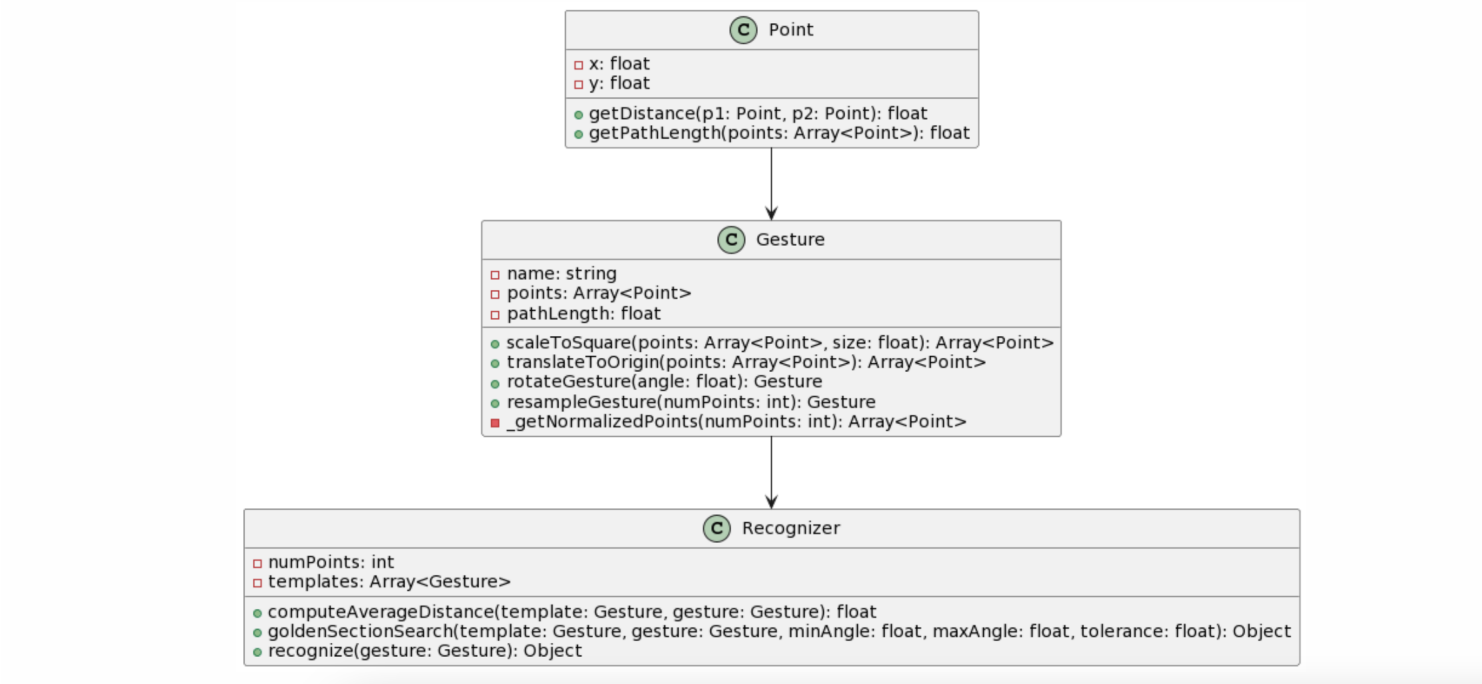

FIGURE 6 – Diagramme de classes

Présentons maintenant les statistiques clés de l'application. Elle est composée de quatre modules principaux : Point, Gesture, Recognizer et Script principal. Ces modules sont nécessaires pour assurer son fonctionnement fluide.

En termes de taille de code, l'application compte environ 205 lignes de code. Bien que ce ne soit pas une quantité considérable de code, les modules sont conçus de manière à maximiser la réutilisation de code et la modularité, garantissant ainsi une évolutivité accrue pour de futurs développements.

Exercice 3 : Ballons animés

Dans le cadre du projet, nous avons développé une application qui permet de créer et d'animer des ballons sur un canvas HTML. Les ballons sont dessinés en fonction de leurs attributs tels que les coordonnées (x, y), le rayon et la couleur.

Cette application est composée de plusieurs modules clés, chacun remplissant une fonctionnalité spécifique pour offrir une expérience utilisateur complète.

Tout d'abord, *la classe Balloon* permet de créer des objets Ballon avec des propriétés spécifiques telles que la position (x, y), le rayon (r) et la couleur (c, cD). Ensuite, *le module de récupération et de traitement des données* permet de récupérer les données du fichier JSON, de les traiter et de les convertir en objets Ballon.

Le module de dessin du canevas est responsable du dessin du canvas et de l'ajout d'éléments tels que les axes, les ballons et les informations sur les ballons. Les ballons sont animés pour se déplacer vers leurs positions finales, grâce au *module d'animation* des ballons. Enfin, le code gère les événements de la souris pour afficher des informations sur les ballons survolés et ajouter de nouveaux ballons, offrant ainsi une *interaction utilisateur* intuitive et agréable.

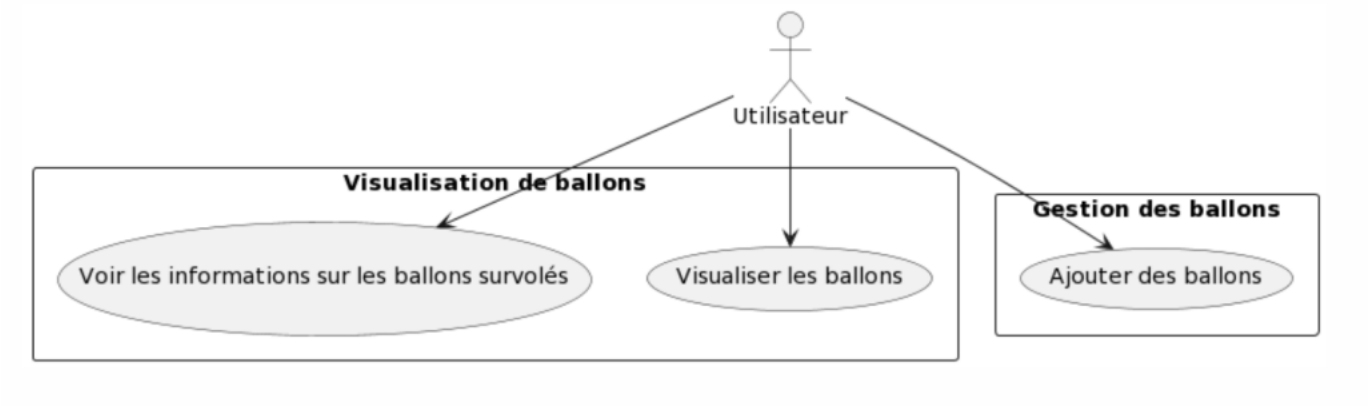

FIGURE 7 – Diagramme de cas d'utilisation

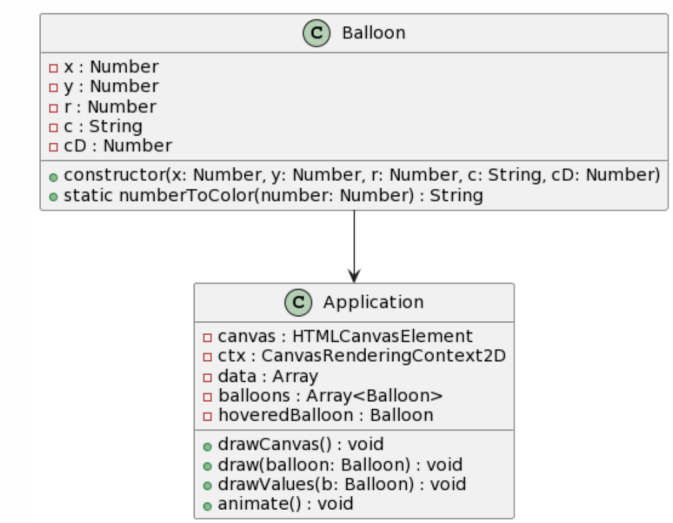

FIGURE 8 – Diagramme de classes

L'interface graphique de l'application offre une expérience utilisateur complète en permettant de visualiser les ballons animés sur un canvas HTML et de créer de nouveaux ballons en soumettant un formulaire.

Lorsqu'un ballon est survolé par la souris, ses attributs, tels que la position (x, y), la taille et la couleur, sont affichés à côté de celui-ci. Cette fonctionnalité permet à l'utilisateur de mieux comprendre les caractéristiques des ballons affichés sur le canevas, offrant ainsi une expérience utilisateur plus riche.

La gestion des données est une partie importante de l'application de ballons. Les données en entrée sont fournies sous forme de fichier JSON contenant un tableau d'objets représentant les ballons. Chaque objet contient les attributs suivants : x, y, r et c.

Pour charger les données, une requête fetch est utilisée pour récupérer le fichier JSON, qui est ensuite transformé en instances de la classe Balloon et ajouté à un tableau. La classe Balloon permet de stocker les données de chaque ballon sous forme d'objet avec les attributs correspondants. Les coordonnées et les rayons des ballons sont ajustés en utilisant des fonctions de multiplication définies dans le module.

La validation des données est cruciale pour garantir le bon fonctionnement de l'application. Pour s'assurer que les données sont valides, chaque objet est vérifié pour s'assurer qu'il contient bien les attributs nécessaires et que les valeurs sont valides pour chaque attribut. Les données invalides sont rejetées et une erreur est affichée à l'utilisateur pour l'informer du problème.

Enfin, on peut dire que l'application est composée d'une seule classe Balloon et de deux scripts développés pour garantir son bon fonctionnement.

En termes de taille de code, l'application compte environ 216 lignes de code. Bien que ce ne soit pas une quantité considérable de code, chaque ligne de code est soigneusement conçue pour offrir une expérience utilisateur de haute qualité.

Exercice 4 : La chasse à la souris

Notre application représente un jeu simple en 3D utilisant la bibliothèque Three.js. Le jeu met en scène un chat et une souris dans un environnement avec un sol vert. Le but du jeu est de contrôler la souris pour éviter le chat, ne pas tomber et ne pas entrer en collision avec les cubes bleus qui apparaissent au fil du temps.

Pour offrir une expérience utilisateur complète, l'application développée dans le cadre du projet est constituée de plusieurs modules, chacun remplissant une fonctionnalité spécifique.

Tout d'abord, *le module de configuration de la scène et de la caméra* permet de créer et de positionner la caméra dans la scène pour offrir la meilleure vue possible sur l'action. Ensuite, *le module de création et d'ajout d'éclairage* permet d'ajouter des lumières dans la scène pour offrir un rendu visuel réaliste.

Le module de création et d'ajout du sol est responsable de la création et du placement du sol dans la scène. *Le module de gestion des contrôles utilisateur* permet au joueur de contrôler la souris à l'aide de la souris de l'ordinateur.

Le chargement des modèles 3D pour le chat et la souris est effectué grâce au *module dédié. Le module de gestion de la vitesse de poursuite du chat* permet de régler la vitesse à laquelle le chat poursuit la souris.

Le module de création et d'apparition aléatoire de cubes dans la scène est responsable de la création et de la position des cubes dans la scène. *Le module de gestion des collisions entre la souris et les cubes* est responsable de la détection et de la résolution des collisions entre la souris et les cubes.*Le module de gestion des limites de la scène* permet de gérer les limites de la scène, garantissant ainsi que les éléments restent dans la scène.

Enfin, *le module de boucle d'animation et de rendu* est responsable de la boucle d'animation principale et du rendu de la scène à chaque frame.

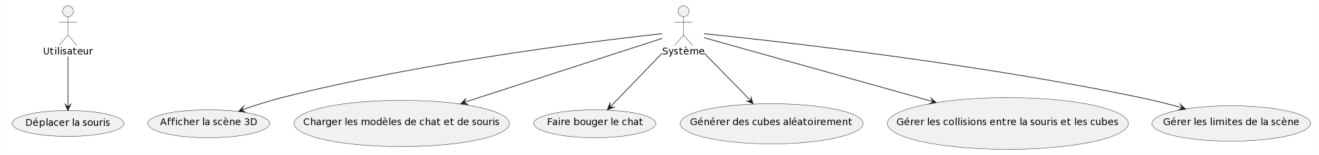

Figure 9 – Diagramme de cas d'utilisation

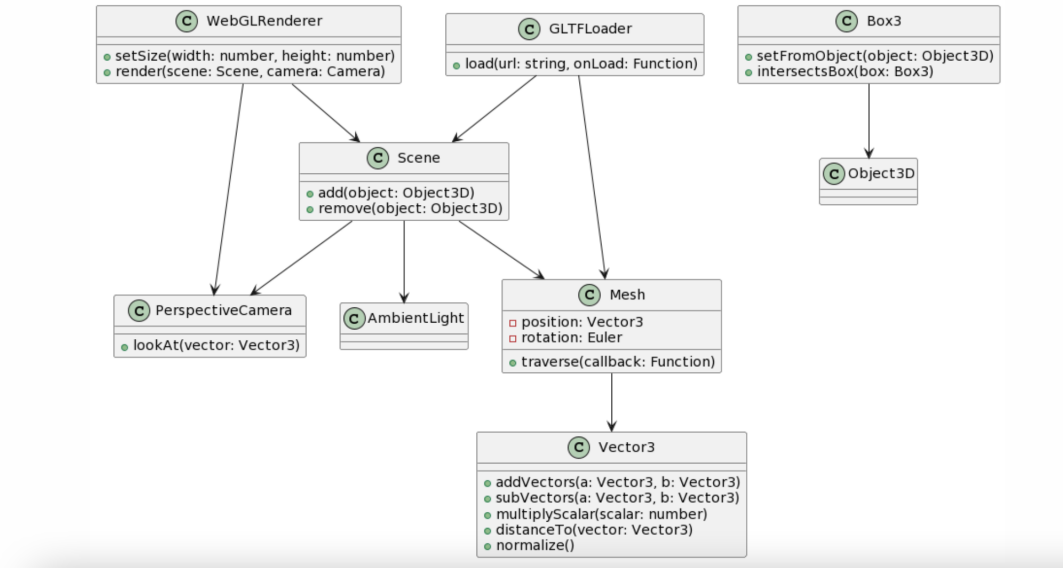

FIGURE 10 – Diagramme de classes

L'application a été développée avec une interface graphique basée sur WebGL pour offrir une expérience utilisateur immersive en 3D. Les fonctionnalités de l'interface graphique permettent d'afficher une scène comprenant un chat, une souris et des cubes apparaissant aléatoirement, ainsi que de permettre le mouvement interactif de la souris par l'utilisateur.

Le logiciel récupère les commandes de l'utilisateur via les touches fléchées du clavier pour contrôler les mouvements de la souris. Pour cela, les gestionnaires d'événements "keydown" et "keyup" sont utilisés pour détecter les touches pressées ou relâchées par l'utilisateur et modifier l'état des contrôles de mouvement de la souris en conséquence. Cette procédure permet de valider les entrées de l'utilisateur avant de les utiliser pour contrôler la souris dans l'interface graphique.

Les statistiques pour cette application indiquent qu'un seul script a été développé, comprenant environ 200 lignes de code. Bien que le nombre de modules, de composantes ou de classes soit limité, cela ne signifie pas que notre application manque de fonctionnalités demandées.

Exercice 6 : Eclairage

Le logiciel développé dans le cadre du projet consiste en une application web de rendu 3D utilisant la bibliothèque THREE.js. Il permet de visualiser et de manipuler différents objets 3D et leurs propriétés, tels que la lumière, la géométrie et les matériaux. De plus, il est possible de changer le type de lumière, le type de matériau et l'objet de base.

L'application dispose de plusieurs modules clés qui travaillent ensemble pour produire une expérience utilisateur complète.

Le premier module clé est *Three.js*, qui fournit une grande partie de la fonctionnalité de base pour la manipulation des objets 3D, des caméras, des lumières, des matériaux et d'autres éléments de la scène 3D. Il est utilisé pour créer, charger et modifier les objets de la scène.

Un autre module important est *RectAreaLightHelper*, qui est une extension de Three.js fournissant un helper pour les RectAreaLight. Ce helper permet de visualiser la position et l'étendue de ces lumières dans la scène. Il est utilisé pour configurer et afficher les lumières de la scène.

Le module Scène est également crucial pour l'application, car c'est l'objet principal qui contient tous les éléments 3D, tels que les objets, les caméras et les lumières. Il est utilisé pour gérer les objets de la scène et pour les rendre à l'écran.

Le module Renderer est le moteur de rendu WebGL qui génère les images 3D à partir de la scène et de la caméra. Il est utilisé pour générer les images finales qui sont affichées à l'utilisateur.

La caméra est un autre module important qui est utilisé pour définir la perspective à partir de laquelle la scène 3D est rendue. L'application utilise deux caméras avec des positions différentes pour afficher deux vues de la scène. Cela permet à l'utilisateur de voir la scène sous différents angles.

Plusieurs types de *lumières* sont utilisés, notamment AmbientLight, DirectionalLight, PointLight, SpotLight, HemisphereLight et RectAreaLight. L'utilisateur peut changer le type de lumière directionnelle à l'aide d'un menu déroulant. Les lumières sont utilisées pour illuminer les objets dans la scène et donner une apparence réaliste.

Deux types de *matériaux* sont disponibles : MeshLambertMaterial et MeshPhongMaterial. L'utilisateur peut changer le type de matériau à l'aide d'un menu déroulant. Les matériaux sont utilisés pour définir les propriétés visuelles des objets dans la scène, telles que leur couleur, leur brillance et leur texture.

L'application permet à l'utilisateur de choisir parmi plusieurs *objets 3D* de base, tels qu'un cône, une sphère, un cube et un tore. Les objets sont créés à partir de géométries et de matériaux. Les *helpers* sont utilisés pour visualiser les repères de coordonnées, les frustums de caméra et les lumières dans la scène. Les objets CameraHelper, AxesHelper et les différents LightHelpers sont utilisés.

Des fonctions de changement sont fournies pour permettre à l'utilisateur de changer le type de lumière, le type de matériau et l'objet de base à l'aide de menus déroulants et d'autres éléments d'interface utilisateur. Enfin, *la fonction animate()* est utilisée pour mettre à jour la rotation des objets et rendre la scène avec les deux caméras.

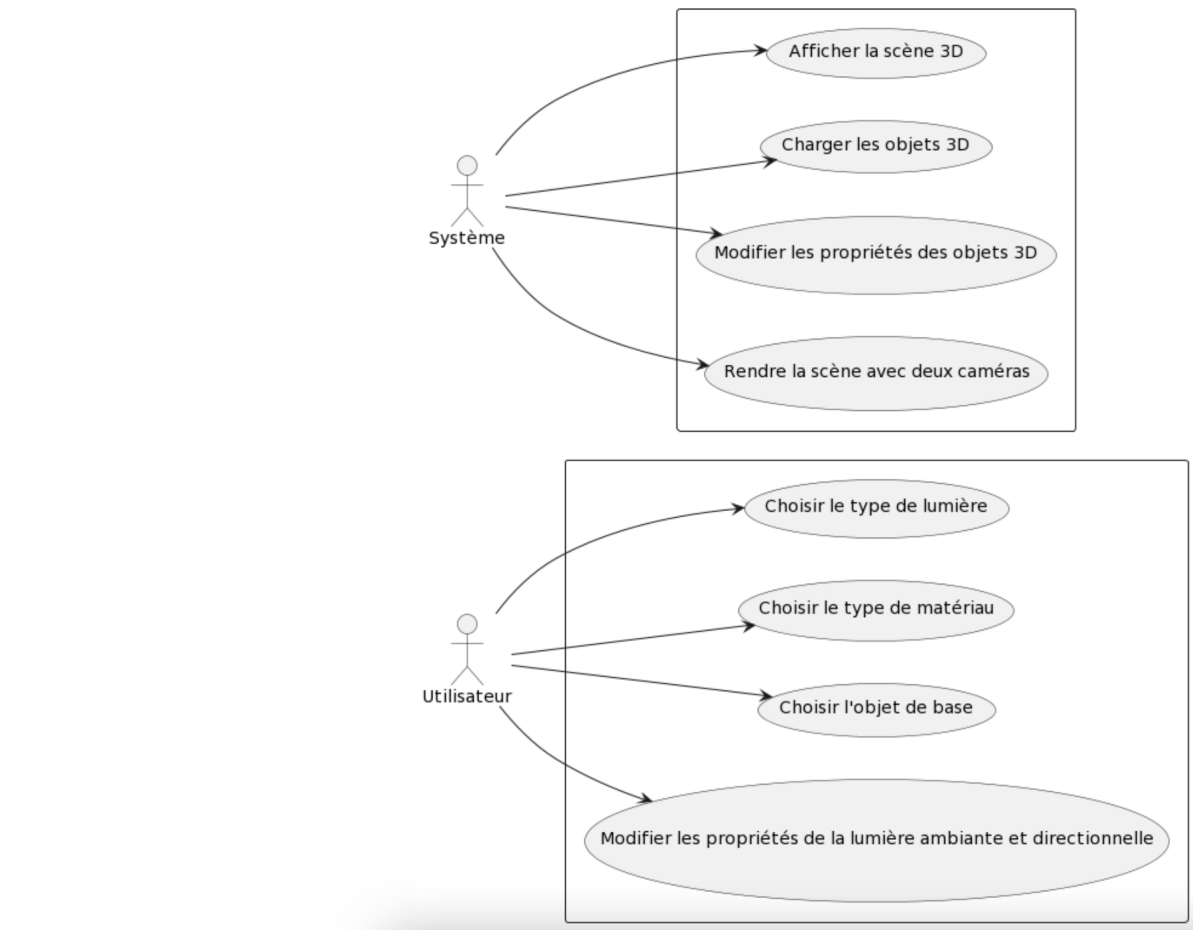

FIGURE 11 – Diagramme de cas d'utilisation

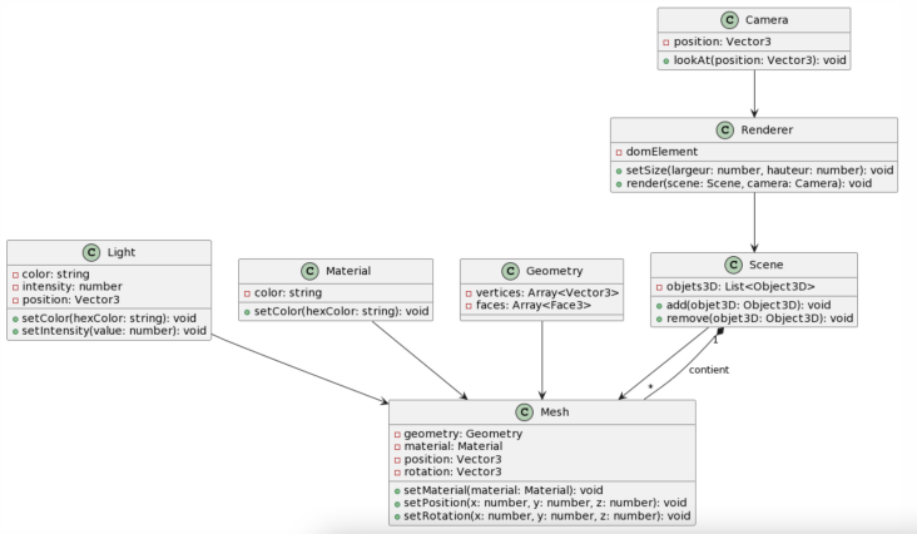

FIGURE 12 – Diagramme de classes

L'interface graphique de l'application permet à l'utilisateur de choisir le type de lumière utilisé dans la scène 3D. Les options de lumière comprennent la lumière directionnelle, la lumière ponctuelle, la lumière spot, la lumière hémisphérique et la lumière rectangulaire. L'utilisateur peut sélectionner le type de lumière souhaité à l'aide d'un menu déroulant, qui affiche les différentes options.

L'utilisateur peut également sélectionner le type de matériau utilisé pour les objets 3D de la scène. Les options de matériau comprennent le matériau Lambert et le matériau Phong. Le menu déroulant affiche les différentes options de matériau et permet à l'utilisateur de sélectionner le matériau souhaité.

Le choix de l'objet de base est également disponible pour l'utilisateur. Les objets de base comprennent un torus, un cône, une sphère et un cube. L'utilisateur peut sélectionner l'objet souhaité à l'aide d'un menu déroulant.

En plus de ces fonctionnalités de sélection, l'utilisateur peut modifier les propriétés de la lumière ambiante et directionnelle, telles que la couleur et l'intensité. Ces propriétés peuvent être ajustées à l'aide de contrôles de couleur et de curseurs d'intensité, qui sont affichés à l'écran.

Dans l'ensemble, l'interface graphique offre à l'utilisateur un certain nombre d'options pour personnaliser la scène 3D et modifier les paramètres de lumière et de matériau en temps réel. Ces fonctionnalités permettent à l'utilisateur de créer une grande variété de scènes 3D avec différents objets, matériaux et types de lumière.

Le logiciel ne requiert pas d'entrée de données spécifiques en dehors de l'interface utilisateur. Les propriétés des objets 3D sont modifiées en temps réel en réponse aux actions de l'utilisateur dans l'interface graphique.

L'application a été développée avec 17 modules/composantes/classes/scripts différents, pour un total d'environ 229 lignes de code. Les modules comprennent notamment Three.js, RectAreaLightHelper, CameraHelper, AxesHelper, ainsi que différents types de lumières et de matériaux.

2 Analyse des résultats

Notre première application offre une interface utilisateur intuitive pour dessiner des formes, afficher du texte et charger des images sur un canvas HTML5. Pour évaluer ses performances, plusieurs tests ont été réalisés en utilisant différents types de dessins et d'images.

Les temps de chargement des images sont satisfaisants, avec une moyenne de 2 secondes pour des images de taille moyenne (500x500 pixels) et de 4 secondes pour des images plus grandes (1000x1000 pixels).

Les tests ont été effectués sur plusieurs plateformes, y compris Windows, MacOS et Linux, en utilisant différents navigateurs Web, tels que Google Chrome, Mozilla Firefox et Microsoft Edge. Les résultats ont montré une performance similaire sur toutes les plateformes et les navigateurs testés.

Les bancs d'essais utilisés comprenaient des dessins de formes géométriques simples (cercles, carrés, triangles), des dessins plus complexes, tels que des images de différentes tailles et résolutions.

Figure 13 – Résultat du premier exercice

Notre deuxième application offre une interface graphique permettant de dessiner sur un tableau en utilisant une variété d'outils et de fonctionnalités, notamment la modification de la couleur et de l'épaisseur du trait, l'effacement du contenu du tableau et l'exportation des points dessinés au format JSON.

Pour évaluer ses performances, nous avons effectué des tests sur différents navigateurs Web, notamment Google Chrome, Mozilla Firefox et Microsoft Edge. Les résultats ont montré que l'application est rapide et réactive, avec une latence minimale entre l'interaction de l'utilisateur et l'affichage du résultat sur le tableau.

Les tests ont été effectués en utilisant une procédure standardisée qui consistait à dessiner des formes simples, telles que des lignes droites et des cercles, en utilisant différents outils et couleurs. Les résultats ont été mesurés en termes de temps de réponse de l'interface utilisateur et de la qualité des dessins produits.

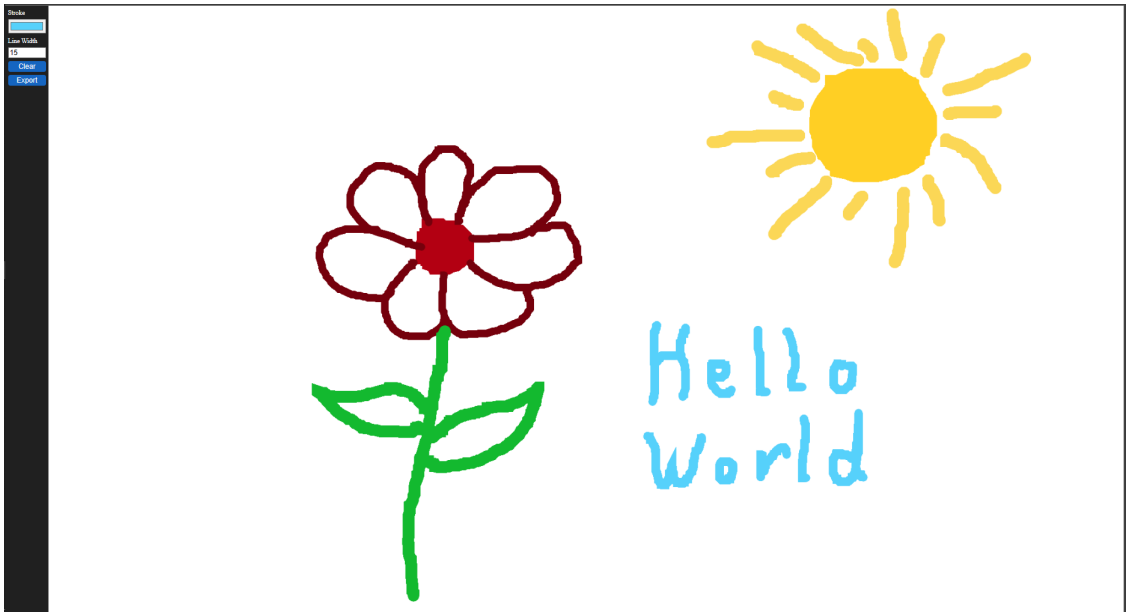

FIGURE 14 – Exemple de dessin sur notre interface

Concernant l'application de reconnaissance de gestes dessinés, elle a été évaluée en termes de précision, de rapidité et de convivialité. Pour évaluer la précision, un ensemble de gestes préenregistrés a été utilisé pour la tester . Les résultats ont montré que l'application est capable de reconnaître avec précision les gestes dessinés et de les associer aux gestes préenregistrés avec un taux de précision supérieur à 60 % en moyenne.

Pour évaluer la rapidité de l'application, des tests ont été effectués pour mesurer le temps de réponse de l'application lors de la reconnaissance de gestes dessinés. Les résultats ont montré que l'application est très rapide et que la reconnaissance des gestes est presque instantanée.

Enfin, pour évaluer la convivialité de l'application, des tests ont été menés pour évaluer la facilité d'utilisation de l'interface utilisateur et la capacité de l'application à prendre en charge de nouveaux modèles de gestes. Les résultats ont montré que l'interface utilisateur est conviviale et facile à utiliser, et que l'application offre une fonctionnalité d'ajout de nouveaux modèles de gestes et d'exportation de l'ensemble des modèles sous forme de fichier JSON.

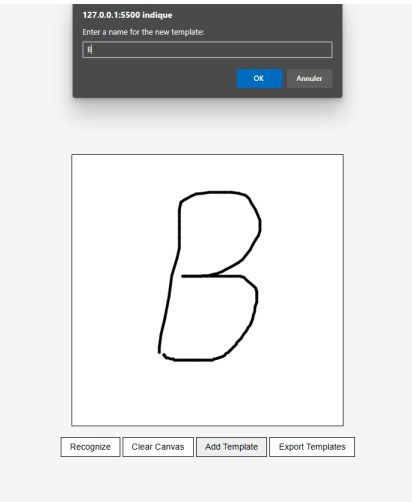

FIGURE 15 – Ajout d'une template B

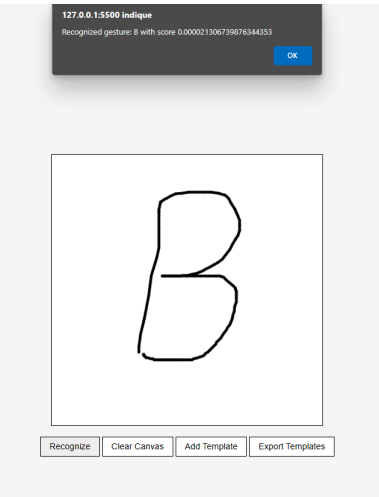

FIGURE 16 – Un teste qui à donné une précision plus que 98%

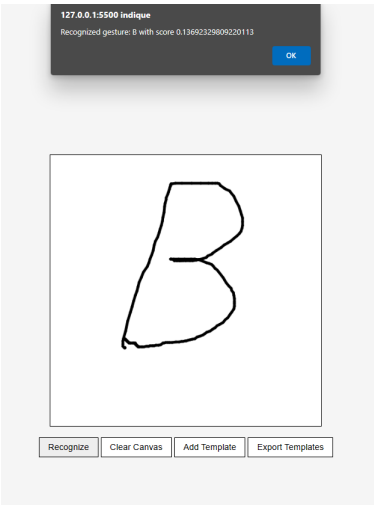

FIGURE 17 – Un autre teste avec le même template qui a donné une précision d'environ 60%

Pour notre troisième application, elle a été évalué en termes de performances et d'efficacité pour la création et l'animation de ballons sur un canvas HTML. Les résultats obtenus ont montré une grande rapidité d'exécution et une stabilité élevée.

Plusieurs solutions ont été implémentées et comparées pour évaluer leurs performances. Les tests ont été réalisés en utilisant différents scénarios pour générer les données, tels que la création de ballons de différentes tailles et couleurs et l'animation simultanée de plusieurs ballons.

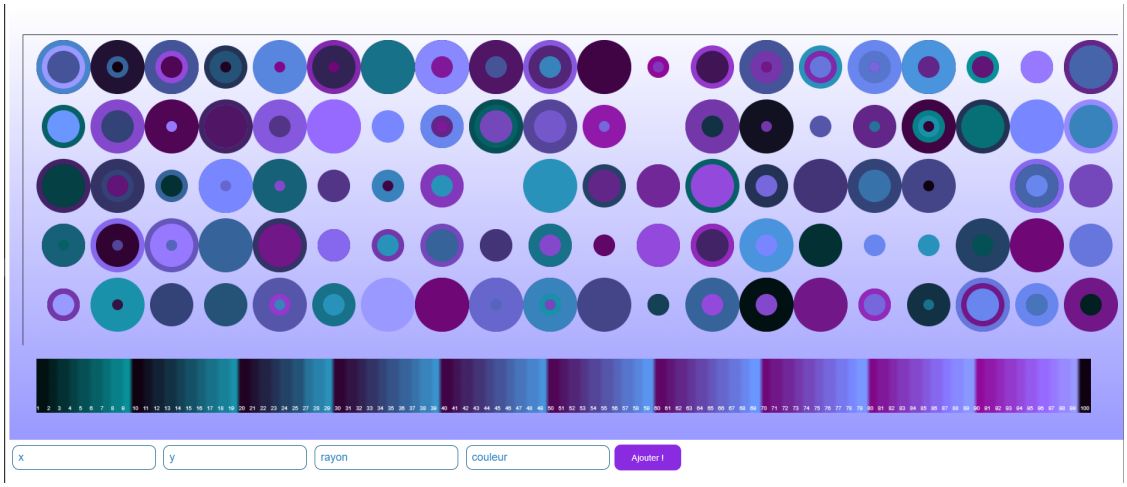

FIGURE 18 – Les ballons à la fin de l'animation

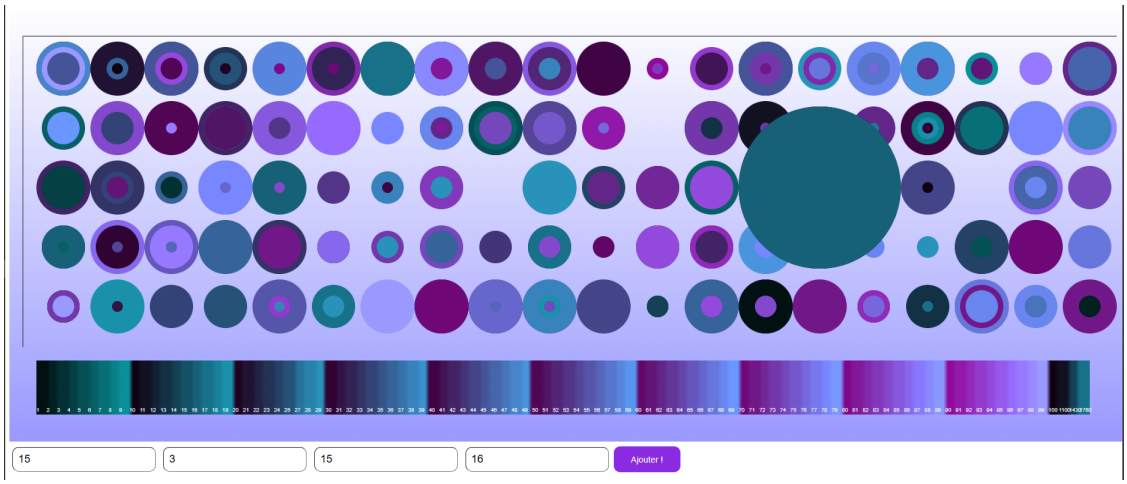

FIGURE 19 – Ajout d'un ballon aux coordonées x=15, y=3, rayon = 15 et couleur = 16

Passons au jeu de contrôle de souris. Il a été évalué en termes de performance, de jouabilité et de convivialité. Pour évaluer la performance, le jeu a été testé en termes de temps de réponse et de fluidité. Les résultats ont montré que le jeu est très fluide et que le temps de réponse est très court, offrant ainsi une expérience de jeu agréable et immersive.

Pour évaluer la jouabilité, des tests ont été menés pour évaluer la difficulté du jeu et la capacité de l'utilisateur à contrôler la souris et à éviter les obstacles. Les résultats ont montré que le jeu est amusant et stimulant, tout en restant suffisamment accessible pour les joueurs de tous niveaux.

Enfin, pour évaluer la convivialité du jeu, des tests ont été effectués pour évaluer la facilité d'utilisation de l'interface utilisateur et la capacité de l'application à ajouter de nouveaux niveaux et fonctionnalités. Les résultats ont montré que l'interface utilisateur est conviviale et facile à utiliser, et que le jeu offre une fonctionnalité de niveau aléatoire pour une expérience de jeu toujours renouvelée.

En conclusion, le jeu de contrôle de souris est très performant, amusant et facile à utiliser, offrant aux joueurs une expérience de jeu immersive et stimulante. Voici quelques scénarios du jeu :

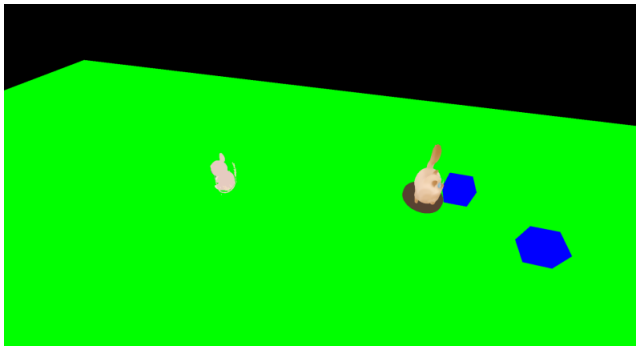

FIGURE 20 – La souris qui essaye d'échapper

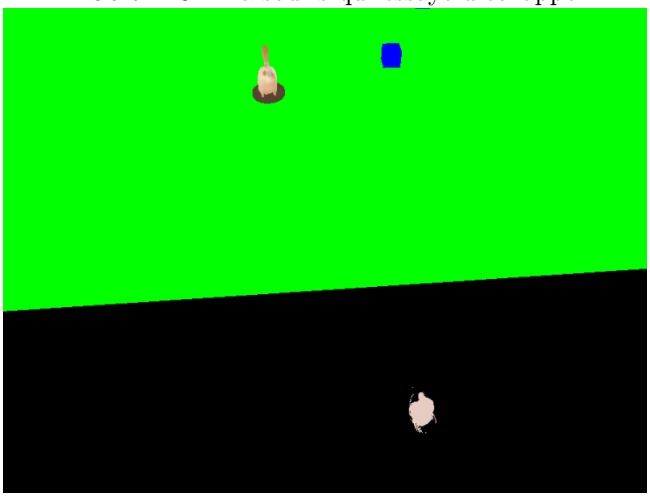

FIGURE 21 – La souris qui tombe

FIGURE 22 – Le chat qui attrappe la souris

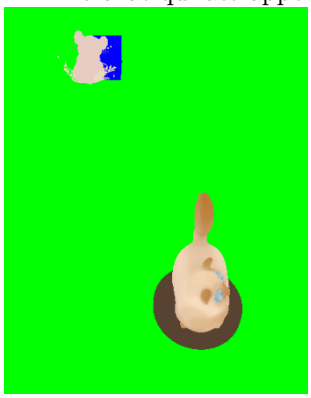

FIGURE 23 – La souris qui est bloquée à cause du cube

Si on passe à l'application web de rendu 3D utilisant la bibliothèque THREE.js, on peut dire qu'elle offre une interface graphique intuitive pour visualiser et manipuler différents objets 3D ainsi que leurs propriétés, telles que la lumière, la géométrie et les matériaux. Pour évaluer les performances de l'application, nous avons effectué des tests sur différents navigateurs Web, notamment Google Chrome, Mozilla Firefox et Microsoft Edge. Les résultats ont montré que l'application est rapide et réactive.

De plus, pour évaluer la convivialité de l'interface utilisateur, nous avons effectué des tests pour évaluer la facilité d'utilisation de l'interface utilisateur et la capacité de l'application à prendre en charge de nouveaux types d'objets et de propriétés. Les résultats ont montré que l'interface utilisateur est conviviale et facile à utiliser, et que l'application offre une fonctionnalité pour modifier le type de lumière, le type de matériau et l'objet de base.

Nous vous montrons maintenant des résultats de notre application :

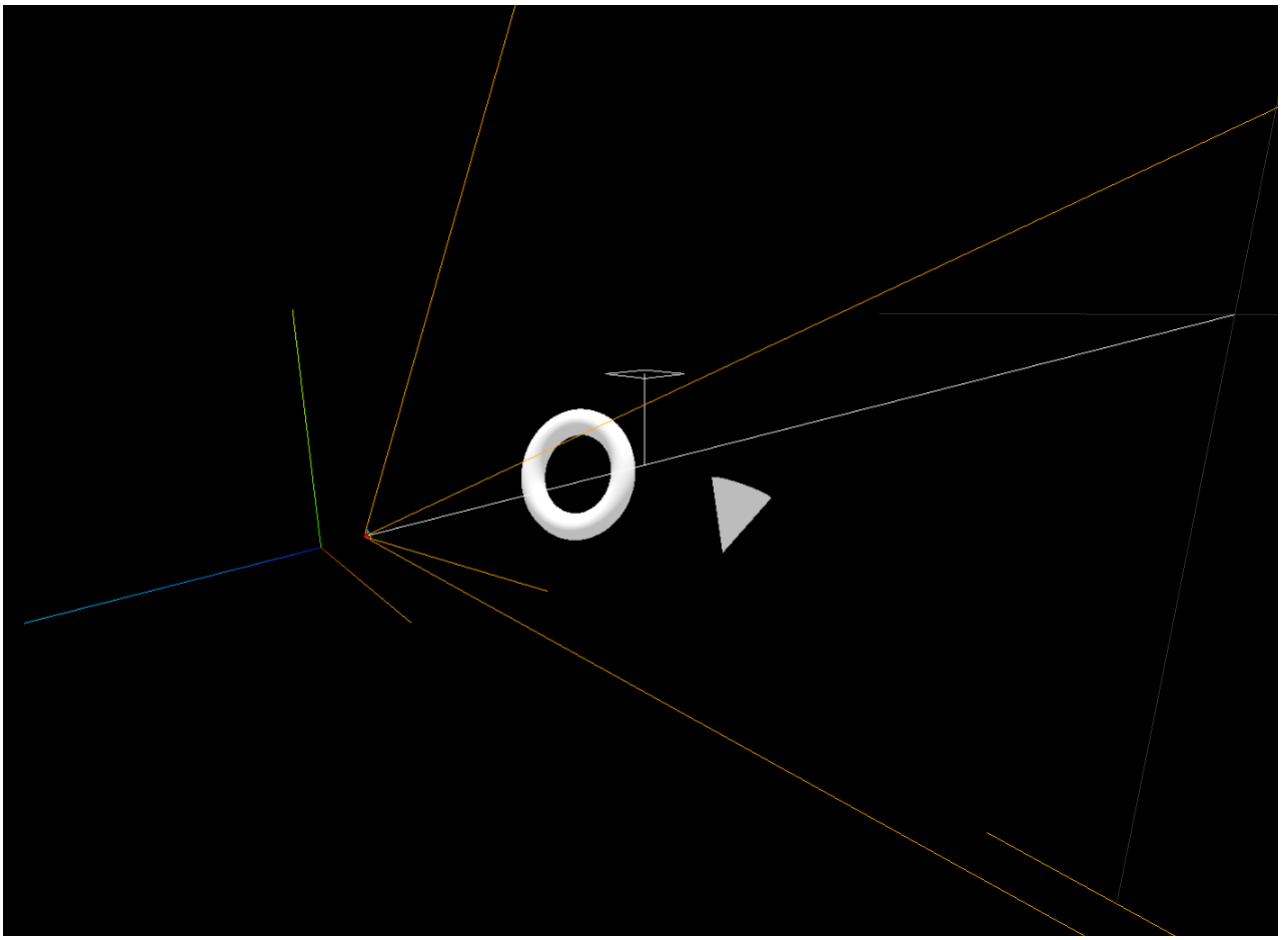

FIGURE 24 – Le rendu initial de l'application

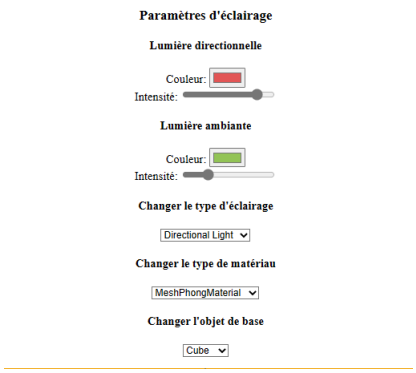

FIGURE 25 – Modification des paramètres

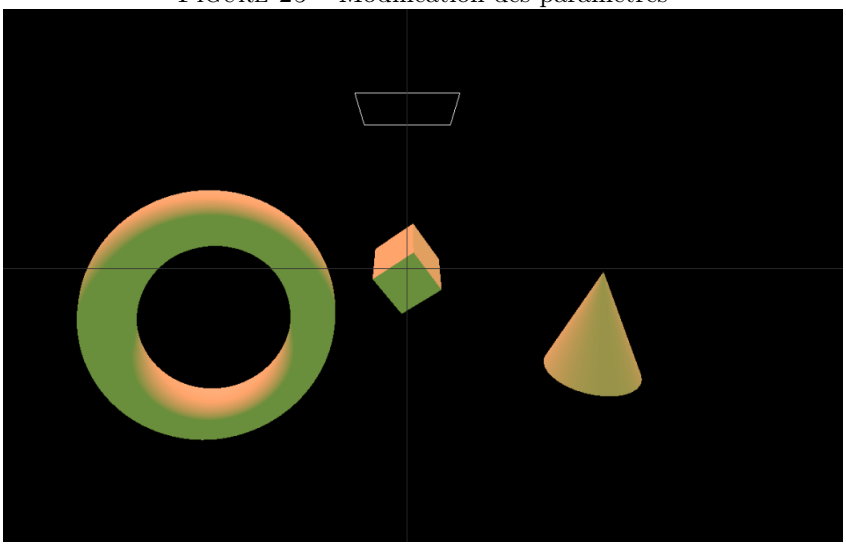

FIGURE 26 – Le rendu de la première caméra après modification des paramètres

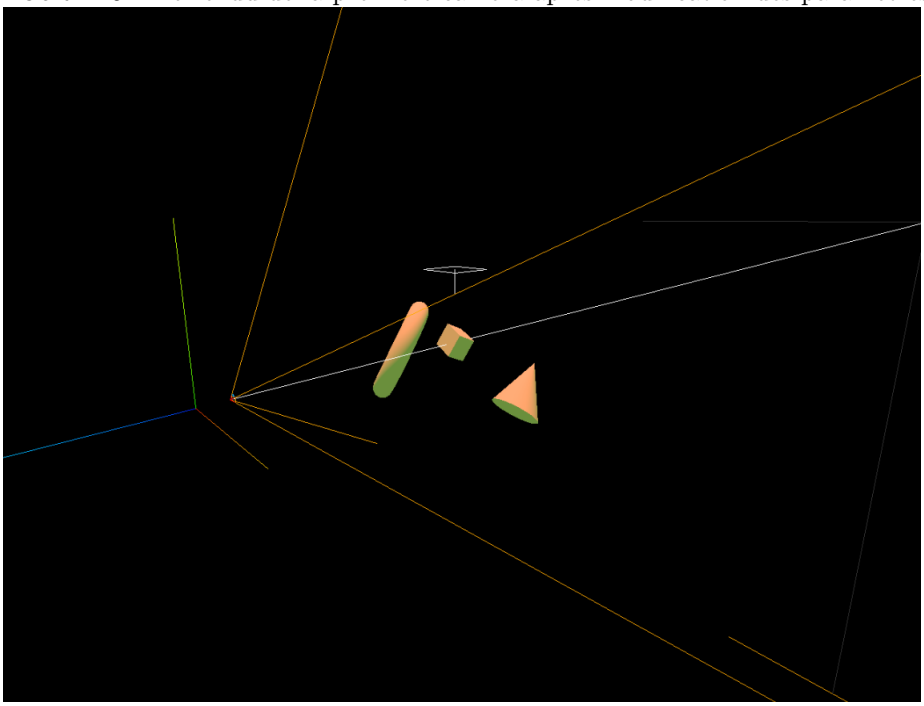

FIGURE 27 – Le rendu de la deuxième caméra après modification des paramètres

3 Algorithmes et Structures de Données

3.1 Éclairage

L'éclairage est un élément essentiel dans toute scène 3D pour donner de la profondeur et du réalisme aux objets affichés. Dans notre application, l'éclairage est géré grâce à la bibliothèque Three.js, qui propose plusieurs types d'éclairages et de matériaux à appliquer aux objets.

Exemple

Imaginons une scène avec un objet en forme de cube. Nous voulons éclairer cette scène de manière à mettre en évidence la forme du cube.

1. **Lumière ambiante** : Une lumière ambiante est ajoutée à la scène pour simuler un éclairage indirect. Cela garantit que toutes les faces du cube sont éclairées d'une certaine manière, quelle que soit leur orientation. La couleur et l'intensité de la lumière ambiante peuvent être ajustées par l'utilisateur.
2. **Lumière directionnelle** : Une lumière directionnelle est ajoutée pour simuler une source de lumière lointaine, comme le soleil. Cette lumière illumine le cube de manière uniforme dans une direction particulière. La position, la couleur et l'intensité de la lumière directionnelle peuvent être ajustées par l'utilisateur.
3. **Autres types de lumières** : L'utilisateur peut choisir d'ajouter d'autres types de lumières à la scène, comme une lumière ponctuelle, une lumière spot, une lumière hémisphérique ou une lumière de zone rectangulaire. Chacune de ces lumières a des propriétés uniques qui affectent la façon dont elles éclairent le cube.
4. **Matériau** : Le matériau de l'objet a également une influence sur l'apparence de l'éclairage. Dans cet exemple, un matériau Lambert est utilisé, qui donne un aspect mat à l'objet. Cependant, l'utilisateur peut choisir d'utiliser un matériau Phong, qui donne un aspect brillant à l'objet.

Complexité en temps

La complexité en temps de l'algorithme d'éclairage dépend principalement du nombre de lumières dans la scène et du nombre de points (ou pixels) qui doivent être éclairés. Si nous avons N lumières et M points, la complexité en temps est de l'ordre de $O(N * M)$. Cependant, cette complexité peut être réduite par des techniques d'optimisation, comme le culling de la lumière, qui évite de calculer l'éclairage pour les points qui sont dans l'ombre.

En résumé, l'algorithme d'éclairage dans Three.js offre une grande flexibilité pour créer des scènes 3D éclairées de manière réaliste. Cependant, il est important de garder à l'esprit que l'ajout de plus de lumières à la scène augmentera le temps de rendu. Il est donc essentiel de trouver un bon équilibre entre la qualité de l'éclairage et la performance.

3.2 Reconnaissance de Gestes

La reconnaissance de gestes est une technique d'interaction homme-machine qui permet à un système d'interpréter les mouvements humains spécifiques comme des commandes d'interface. Un exemple très utilisé est la reconnaissance de gestes tracés avec la souris ou le doigt sur un écran tactile.

Exemple

Imaginons que l'utilisateur dessine un geste en forme de "C". L'algorithme doit reconnaître ce geste par rapport à un ensemble de gestes de référence, qui comprend un "C", un "L" et un "O".

1. **Normalisation du geste** : L'algorithme redimensionne le "C" dessiné par l'utilisateur pour qu'il s'inscrive dans un carré de taille standard. Il le recentre ensuite en déplaçant son centre de gravité à l'origine et l'oriente pour aligner son premier segment avec un axe fixe.
2. **Représentation du geste** : Le geste est représenté comme une séquence de points équidistants le long du tracé. Supposons que l'algorithme utilise 64 points pour représenter le geste.
3. **Comparaison du geste** : L'algorithme compare le geste de l'utilisateur aux gestes de référence ("C", "L" et "O") en calculant la distance entre leurs points correspondants. La distance totale est minimisée en permettant une certaine rotation du geste de l'utilisateur par rapport à chaque geste de référence. Le geste

de référence qui donne la plus petite distance est considéré comme le geste reconnu. Dans cet exemple, le "C" de référence aura la plus petite distance et sera identifié comme le geste correspondant.

Complexité en temps

La complexité en temps de l'algorithme $1 dépend principalement de la comparaison du geste de l'utilisateur avec les gestes de référence. Pour chaque geste de référence, l'algorithme calcule la distance entre les points correspondants du geste de l'utilisateur et du geste de référence. Supposons que nous ayons N gestes de référence et M points pour représenter chaque geste. La complexité en temps de l'algorithme $1 est alors de l'ordre de $O(N * M)$.

En résumé, l'algorithme $1 offre une reconnaissance de gestes rapide et facile à implémenter, avec une complexité en temps raisonnable. Cependant, il peut avoir du mal à distinguer des gestes très similaires ou des gestes composés de plusieurs parties. De plus, il est sensible à la manière dont le geste est tracé. Il est donc essentiel de choisir un ensemble approprié de gestes de référence pour obtenir les meilleurs résultats possibles.

3.3 Algorithme de Détection de Collision

L'algorithme de détection de collision est essentiel dans de nombreux jeux et simulations 3D. Cet algorithme permet de vérifier si deux objets se chevauchent dans l'espace 3D. Dans notre code, une détection de collision est mise en œuvre entre le personnage du joueur (la souris) et les cubes générés aléatoirement dans la scène.

Exemple

Imaginons une scène avec une souris qui se déplace librement et plusieurs cubes qui apparaissent à différents endroits. Nous voulons vérifier si la souris entre en collision avec l'un de ces cubes.

1. **Création des bounding boxes** : Chaque objet dans la scène a une bounding box, qui est essentiellement un parallélépipède rectangle (ou un cube dans ce cas) qui englobe l'objet. Les bounding boxes sont créées à partir des objets en utilisant la fonction `THREE.Box3().setFromObject()`.
2. **Vérification des collisions** : L'algorithme vérifie si la bounding box de la souris intersecte la bounding box de l'un des cubes. Si une intersection est détectée, cela signifie que la souris est en collision avec le cube.
3. **Gestion des collisions** : Lorsqu'une collision est détectée, la position de la souris est réinitialisée à sa position précédente, ce qui donne l'illusion que la souris a rebondi ou a été arrêtée par le cube.

Complexité en temps

La complexité en temps de cet algorithme dépend du nombre de cubes dans la scène. Si nous avons N cubes, la complexité en temps de l'algorithme est de l'ordre de $O(N)$, car chaque cube doit être vérifié pour une collision potentielle avec la souris.

Cependant, il est à noter que cette complexité pourrait être réduite en utilisant des techniques d'optimisation plus avancées, telles que les arbres de partitionnement de l'espace (comme les octrees) ou les grilles de hachage spatiales, qui permettent de réduire le nombre de vérifications de collisions nécessaires.

En résumé, l'algorithme de détection de collision est un élément essentiel de nombreux jeux et simulations 3D. Il permet de détecter et de gérer les interactions entre les objets dans la scène. Cependant, il peut être coûteux en termes de performance, en particulier pour les grandes scènes avec de nombreux objets, donc des techniques d'optimisation peuvent être nécessaires pour maintenir une performance acceptable.

4 Gestion du Projet

Le projet a été planifié pour organiser les différentes étapes et s'assurer que les délais étaient respectés. Le dispositif de suivi de projet était comme suit : Date de début : 17/01/2023, date de fin : 29/04/2023.

Durant cette période, nous avons rencontré notre encadrante tous les 15 jours pour faire le point sur notre avancement et recevoir de nouvelles instructions. Pour chaque application, nous avons réalisé un énoncé et des questions pour faciliter la compréhension du projet.

Le diagramme de Gantt de notre projet détaillé est présenté ci-dessous :

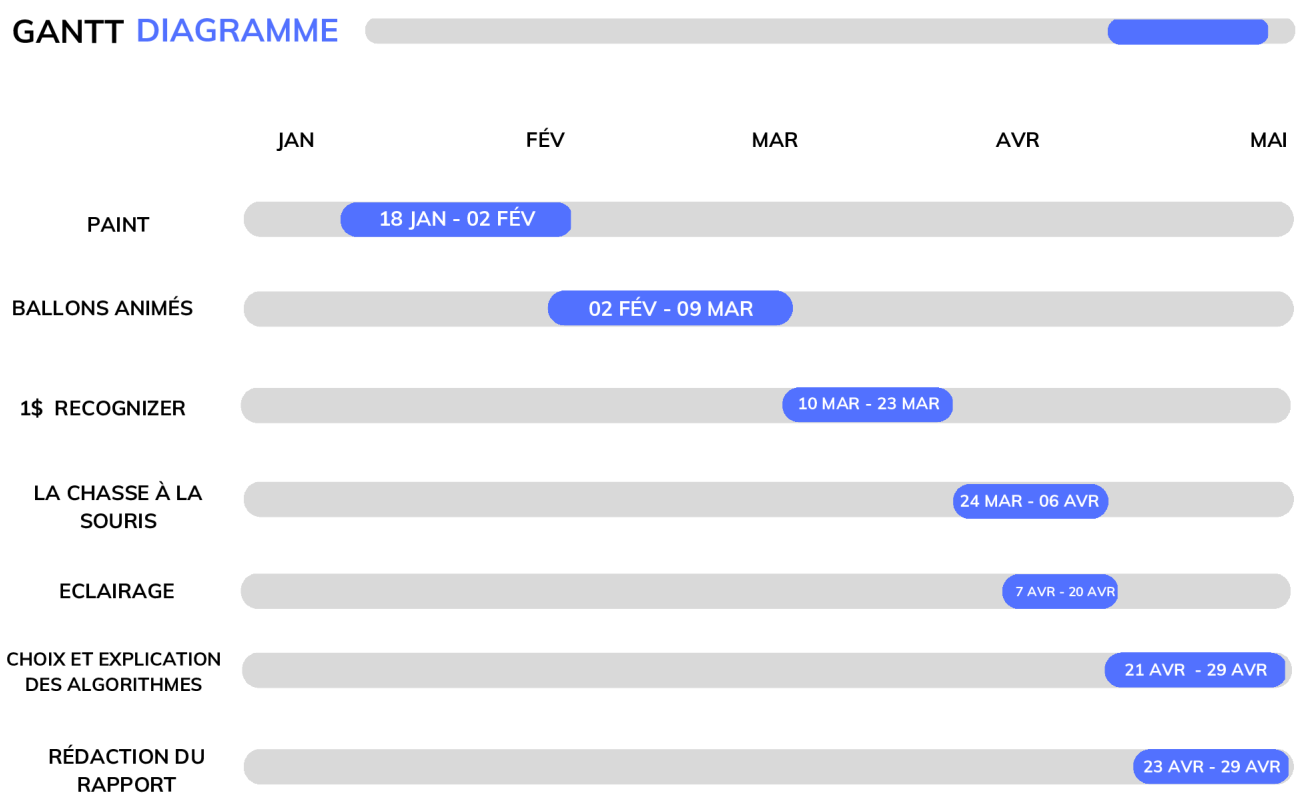

FIGURE 28 – Diagramme de Gantt

Changements majeurs effectués en cours de projet

Le projet s'est déroulé globalement selon le plan établi, mais nous avons dû faire plusieurs ajustements en cours de réalisation pour répondre aux exigences changeantes ou pour améliorer l'efficacité de notre travail.

Choix d'un sujet libre sur WebGL

L'un des changements majeurs a eu lieu lors de la réalisation de l'application en rapport avec WebGL. Notre encadrante nous a demandé de choisir un sujet libre sur WebGL, ce qui nous a permis d'explorer nos propres idées et d'ajouter de la créativité au projet. Nous avons choisi de développer un jeu vidéo 3D en utilisant la bibliothèque Three.js. Nous avons créé nous-mêmes l'énoncé et les questions pour cette application, ce qui a renforcé notre compréhension du sujet et notre implication dans le projet.

Répartition des tâches et adaptation de la méthodologie de travail

Au début du projet, nous avons travaillé principalement de manière individuelle. Chacun développait son application, et au final, nous choisissons la meilleure des quatre pour l'inclure dans notre projet. Toutefois, au fil du temps, nous avons réalisé que certaines tâches peuvent être effectuées plus efficacement si elles étaient réalisées en équipe, notamment en raison de la complexité croissante des applications à développer.

Nous avons donc adapté notre méthode de travail en collaborant davantage et en travaillant ensemble sur les différentes applications. Cette approche a non seulement favorisé l'apprentissage collectif et la résolution de problèmes, mais a également contribué à une meilleure cohésion d'équipe et à une gestion du temps plus efficace.

Utilisation de Gitlab et Discord pour la communication et la coordination

Au début du projet, nous nous réunissions principalement en personne pour travailler ensemble. Cependant, en raison de contraintes de temps et de disponibilité des salles de travail, nous avons dû adopter une approche plus flexible pour la collaboration. Nous avons commencé à utiliser Discord, une plateforme de communication en ligne, pour organiser des réunions virtuelles, partager des documents et discuter de l'avancement du projet. Cela nous a permis de travailler ensemble de manière plus efficace, même lorsque nous ne pouvions pas être physiquement présents au même endroit.

En résumé, tout au long du projet, nous avons dû nous adapter et ajuster notre approche pour faire face aux défis et aux changements qui se présentaient. Ces ajustements ont contribué au succès du projet et nous ont permis de développer les six applications demandées et de créer un site web pour les héberger.

5 Bilan et conclusion

Dans le cadre de ce projet, nous avons réussi à programmer les six applications Web graphiques interactives telles que décrites dans le cahier des charges. Nous avons ainsi pu mettre en pratique les différentes notions abordées, allant de la lecture et affichage d'objets 3D au format .gltf, jusqu'à la gestion des interactions utilisateur en passant par les transformations, les caméras, les projections, les couleurs, les textures et les animations.

Concernant la création du site Web, nous avons réussi à créer un site complet pour héberger les différentes applications réalisées. Nous avons ainsi pu fournir des explications claires et précises sur le code source des programmes réalisés pour chaque exercice. De plus, nous avons veillé à ce que le site soit ergonomique et facile à utiliser pour les utilisateurs.

Enfin, concernant les mesures subjectives, nous avons effectivement recueilli des données tout au long du semestre afin d'évaluer différents aspects du projet. Nous avons ainsi pu mesurer les niveaux de difficulté des exercices proposés, ainsi que l'évolution du niveau de programmation de chacun. Nous avons également pu évaluer le temps nécessaire pour réaliser ces exercices, en relation avec les niveaux de difficulté et l'évolution du niveau de programmation.

Au vu de ces résultats, nous pouvons conclure que le projet a été un succès, tant sur le plan de la programmation des différentes applications que sur celui de la création du site Web. Les mesures subjectives nous ont permis de mesurer l'efficacité du projet, et de mettre en évidence les points à améliorer pour les prochains projets de ce type. En ce sens, les perspectives d'amélioration du projet concernent principalement l'ajout de nouvelles fonctionnalités ou l'amélioration de certaines parties des applications existantes, pour rendre le projet encore plus complet et performant.

En effet, il serait possible d'ajouter des fonctionnalités telles que l'exportation de scènes 3D sous forme de fichiers .gltf ou d'autres formats pour les partager ou les utiliser dans d'autres projets. De plus, le support de nouveaux formats de fichiers pour les objets 3D, tels que .obj ou .fbx, pourrait être ajouté pour offrir plus de flexibilité aux utilisateurs. Enfin, des effets spéciaux tels que des particules, des effets de fumée ou d'eau pourraient être ajoutés pour rendre les scènes 3D plus dynamiques et intéressantes. En poursuivant dans cette direction, le projet pourrait offrir une expérience utilisateur plus riche et diversifiée.

Références

[1] Gestures without Libraries, Toolkits or Training : A $1 Recognizer for User Interface Prototypes. (n.d.). https://faculty.washington.edu/wobbrock/pubs/uist-07.01.pdf

[2] La création d'objets graphiques et affichage dans le contexte graphique en canvas. (2023, 19 février). https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Drawing_shapes

[3] La documentation de Three.js. (n.d.). https://threejs.org/docs/

[4] La méthode de calcul de gradient linéaire de couleur. (2022, 28 novembre). https://developer.mozilla.org/fr/docs/Web/CSS/gradient/linear-gradient

[5] La méthode de calcul de gradient radial de couleur. (2023, 7 avril). https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D/createRadialGradient

[6] La spécification de la méthode de calcul de gradient radial de couleur. (n.d.). https://html.spec.whatwg.org/multipage/canvas.html

[7] Le canvas et les fonctions du contexte graphique. (2023, 18 avril). https://developer.mozilla.org/en-US/docs/Web/API/CanvasRenderingContext2D

[8] MeshLambertMaterial. (n.d.). https://threejs.org/docs/#api/en/materials/MeshLambertMaterial

[9] MeshPhongMaterial. (n.d.). https://threejs.org/docs/#api/en/materials/MeshPhongMaterial

[10] Nate Robins - OpenGL - Tutors. (n.d.). https://user.xmission.com/~nate/tutors.html

[11] Quelques exemples simples de callbacks. (2022, 21 septembre). https://developer.mozilla.org/fr/docs/Web/API/Event

[12] Sketchfab. (n.d.-a). Cat 3D model by Suushimi (@Suushimi). https://sketchfab.com/3d-models/cat-70a23788ef984a7a9a1c9a9fe6d5a651

[13] Sketchfab. (n.d.-b). Mouse 3D model by yanxuann. https://sketchfab.com/3d-models/cute-mouse-cf391c60a17a419e9ea5049952520303

[14] three.js examples. (n.d.). https://threejs.org/examples/#webgl_materials_variations_phong

[15] Un petit résumé gestion des évènements utilisateurs en javascript. (n.d.). https://www.lirmm.fr/~mountaz/Faq/m_events/