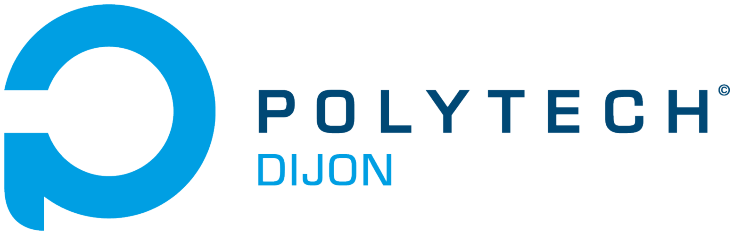
Polytech Dijon
Ingénierie Logicielle Connaissance 4A

Lionel Garnier

Projet de Synthèse d'Images

Auteurs :
NGUYEN Lionel
MARTINEZ Roméo



2023-2024

Sommaire

Table des Figures

Chapitre 1

Introduction

1.1 Contextualisation

La réalisation du projet de Synthèse d'Images repose sur l'utilisation de la librairie THREE.js, visant à créer une expérience immersive en 3D. Cette section présente de manière générale les objectifs et les enjeux de la réalisation, mettant en avant la pertinence de l'utilisation de la librairie THREE.js pour atteindre ces objectifs.

1.2 Définition du Problème

L'énoncé du sujet met en lumière la nécessité de créer une scène 3D interactive, avec un accent particulier sur la modélisation d'une table de ping-pong. Les contraintes techniques imposées soulignent l'utilisation de la librairie THREE.js, une bibliothèque JavaScript dédiée à la création d'environnements 3D dans le navigateur.

Parmi les aspects cruciaux du projet, on retrouve la mise en place d'une table de ping-pong réaliste avec des pieds modélisés en utilisant des courbes de Bézier. Ainsi qu'une simulation d'une partie de ping-pong utilisant elle aussi les courbes de Béziers.

Chapitre 2

Analyse des Contraintes Techniques

2.1 Composants du Jeu de Tennis de Table

Nous avons initié la conception de la scène principale, comprenant la table de ping-pong, les raquettes, et la balle. Pour réaliser cela, nous avons utilisé l'architecture suivante :

2.1.1 Architecture du Projet

Le projet est organisé en plusieurs fichiers JavaScript, chacun responsable d'une partie spécifique de la scène :

- `ball.js` : Gère la logique de la balle.
- `guiMenu.js` : Gère l'interface utilisateur du menu.
- `net.js` : Gère la création du filet de ping-pong.
- `ping-pong.js` : Fichier principal qui coordonne l'ensemble du jeu.
- `racket.js` : Gère la logique des raquettes.
- `table.js` : Gère la création de la table de ping-pong.

L'idée d'utiliser une architecture comme celle-ci a était pour simplifier et ordonner notre projet ainsi nous respectons les principes du clean code. Cette approche méthodique a non seulement facilité l'ajout de nouvelles fonctionnalités, mais a également rendu les adaptations nécessaires pour la partie courbe de Bézier plus accessibles. En suivant cette architecture bien définie, notre projet demeure cohérent, favorisant ainsi une maintenance sans heurts et une compréhension claire du code.

Fichier ping-pong.js

Ce fichier explique comment la courbe de Bézier fonctionne et comment la scène de la table de ping-pong est créée. De plus il gère la création des pieds de la table. En organisant de cette manière, on simplifie la gestion de la courbe de Bézier tout en la reliant de manière fluide à la création globale de la scène. Cela rend le code plus clair et plus facile à maintenir.

Il est le coeur du projet.

2.1.2 Phases de Jeu et Trajectoires

Dans le cadre de la simulation, nous avons associé la trajectoire complète de la balle, couvrant depuis le service jusqu'à la faute.

Chapitre 3

Conception et Réalisation

3.1 Choix Techniques

Nos choix pour la réalisation de ce projet ont consisté à créer un code orienté objet avec différentes classes qui permettront de créer différentes instances et ainsi réutiliser ces classes.

Trajectoires : Les trajectoires ont été conçues pour garantir des mouvements fluides et naturels dans le contexte d'un jeu de ping-pong. En adoptant des classes distinctes pour les trajectoires, nous avons simplifié la gestion des déplacements de la balle et des raquettes.

Nous avons choisi de simplifier le problème de la balle qui atterri à n'importe quel endroit d'un rectangle à une ligne afin de poser position de longueur des points de manière absolue et donc d'uniquement s'occuper de la coordonnée z. Nous verrons le placement de ces points dans la partie 5.2.

Pour créer la trajectoire nous utilisons 4 fonctions :

— `service()` : Gère le service de la balle.
— `droit()` : Gère les trajectoires d'un rectangle à celui d'en face .
— `diagonale()` : Gère les trajectoires d'un rectangle à celui en diagonale.
— `fail()` : Gère la balle qui rate. Elle ne s'active qu'à la fin des n-1 échanges.

Pour créer les courbes de Béziers, nous utilisons les fonctions QuadraticBezierCurve3 et CubicBezierCurve3 de la librairie THREE.js. Pour les combiner, nous utilisons la fonction CurvePath qui est aussi de la librairie THREE.js

Pieds de la Table : La conception des pieds de la table a été intégrée dans une classe distincte pour garantir une gestion efficace de cet aspect visuel. Au sein de cette classe, nous définissons trois jeux de points permettant de construire des courbes de Bézier. Ces courbes sont ensuite transformées en Lathe avec une révolution autour d'elles. Bien que la création de la courbe de Bézier soit codée directement plutôt que de passer par THREE, cette approche n'est pas compliquée, car elle peut être réalisée en se référant aux cours ou aux ressources disponibles en ligne.

Au cours de l'exécution de cette classe, les géométries des pieds sont générées à l'aide de la méthode `generateLegGeometries`, où chaque lathe correspond à une courbe de Bézier définie par les points spécifiés. Les matériaux des mesh correspondant à chaque lathe sont ensuite créés, avec des couleurs distinctes pour une meilleure

distinction visuelle.

La méthode `render` place correctement les mesh dans la scène en fonction de la position spécifiée. Notons que la méthode `createFeet`, bien que commentée, semble être une tentative d'ajouter une représentation plus détaillée des pieds, utilisant une courbe linéaire et un tube. Cependant, cette partie est actuellement désactivée dans le code.

La classe offre également des fonctionnalités de modification des points de contrôle de la courbe de Bézier avec la méthode `setControlPoints`. Enfin, la méthode `dispose` permet de supprimer les mesh existants de la scène, offrant ainsi la possibilité de mettre à jour ou de réinitialiser la géométrie des pieds.

3.2 Menu GUI et Interactivité

Dans cette partie, nous allons expliquer la création et le fonctionnement du menu GUI.

Le menu GUI est créé à l'aide de la bibliothèque GUI de THREE.js. Il offre plusieurs fonctionnalités pour ajuster les paramètres de la scène.

3.2.1 Choix de la Couleur de la Table

Nous avons inclus un menu permettant à l'utilisateur de choisir la couleur de la table de ping-pong. Deux options sont disponibles : vert et bleu. L'utilisateur peut sélectionner la couleur souhaitée à l'aide du contrôleur approprié. Tout changement de couleur déclenche une mise à jour visuelle instantanée.

3.2.2 Contrôle des Points de Bézier

Un autre aspect crucial du menu GUI est la possibilité de modifier les points de contrôle des courbes de Bézier utilisées pour générer les pieds de la table. Nous avons créé des contrôleurs pour chaque point de manière à permettre des ajustements précis. Toute modification déclenche une mise à jour instantanée de la représentation visuelle de la table.

3.2.3 Position de la Caméra

Enfin, le menu GUI offre la possibilité de régler la position de la caméra dans l'environnement 3D. Des contrôleurs distincts sont fournis pour ajuster les coordonnées X, Y et Z de la position de la caméra. Les changements sont immédiatement reflétés dans la scène.

Ce menu GUI enrichit l'expérience utilisateur en permettant des ajustements dynamiques, offrant ainsi un contrôle interactif sur plusieurs aspects du jeu.

3.3 Gestion du Score

Structure HTML :

Le tableau est structuré en utilisant la balise `<table>`, avec une section d'en-tête (`<thead>`) contenant les noms des équipes Jaune et Rouge, et une section de corps (`<tbody>`) avec une ligne représentant les scores actuels des deux équipes.

Style CSS :

Les cellules du tableau sont stylisées avec des couleurs distinctes pour chaque équipe. La couleur jaune est attribuée à l'équipe Jaune, tandis que la couleur rouge est attribuée à l'équipe Rouge. Des bordures sont également ajoutées pour accentuer visuellement chaque cellule, avec des couleurs de bordure correspondant à l'équipe adverse.

Fonction JavaScript (`updateScoreTable`) :

Une fonction JavaScript, nommée `updateScoreTable`, est utilisée pour mettre à jour les scores de manière dynamique. Cette fonction prend les scores actuels des équipes en paramètres et modifie le contenu textuel des cellules correspondantes dans le tableau.

Chapitre 4

Difficultés Rencontrées

4.1 Analyse des Challenges Techniques

Le plus gros obstacle du côté de la simulation a été de trouver la bonne méthode pour placer les points de contacts avec la table plus les points de contrôle afin qu'ils soient alignés.

De plus, les courbes sont G0 seulement et la vitesse de la balle est proportionnelle au nombres d'échanges car nous n'arrivions pas à implémenter les tangentes aux courbes.

Un autre défi rencontré était la mise en place du menu GUI, en particulier l'implémentation de la modification des points de contrôle des Lathe. Cela impliquait la gestion simultanée de la création d'une nouvelle instance du pied de table une fois que celle-ci était modifiée, ainsi que la gestion de la suppression de l'ancienne instance pour éviter les conflits dans le rendu de la scène. Pour simplifier cette gestion nous avons donc décidé d'implémenter la création des pieds dans le fichier ping-pong.js

Cette problématique requérait une gestion précise des transitions entre les différentes étapes de modification. Il était crucial de s'assurer que les modifications apportées aux points de contrôle se traduisent par une mise à jour fluide dans la représentation visuelle de la table.

Chapitre 5

Schémas Explicatifs

5.1 Construction des Pieds de la Table

Pour la réalisation des pieds de la table, nous avons eu recours à l'utilisation de GeoGebra afin de mieux appréhender le fonctionnement des jointures de type G1 entre les pieds et la table :

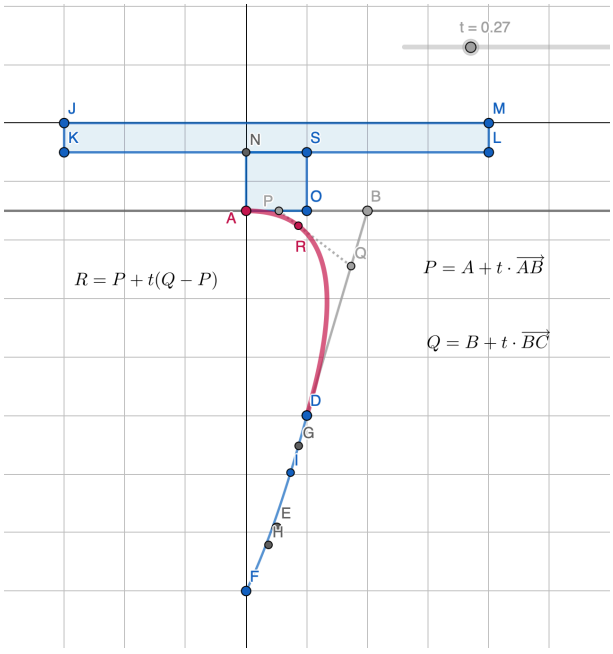


Figure 5.1 – Jointure G1

La jointure G1, mesure la position et la direction de la courbe aux extrémités. En d'autres termes, les deux courbes se touchent et se dirigent dans la même direction au point où elles se touchent.

La direction est déterminée par le premier et le deuxième point de chaque courbe. Si ces deux points tombent sur une ligne, les deux courbes sont tangentes au niveau de leur extrémité commune.

Il est à noter que sur la figure, les points de contrôle A, B, D et D, E, F sont disposés de manière à assurer la continuité G1 des pieds de la table. De plus, les tangentes aux points A sont confondues et collinaires, assurant ainsi une jointure G1 entre les points. Il est également notable que les points de contrôle partagent la même ordonnée que les points correspondants, renforçant la cohérence visuelle de la jointure.

5.2 Trajectoires de la Balle

Concernant le choix des points de contrôle j'ai utilisé le site `https://www.desmos.com/calculator/d1ofwre0fr?lang=fr` afin de choisir mes points de contrôle.

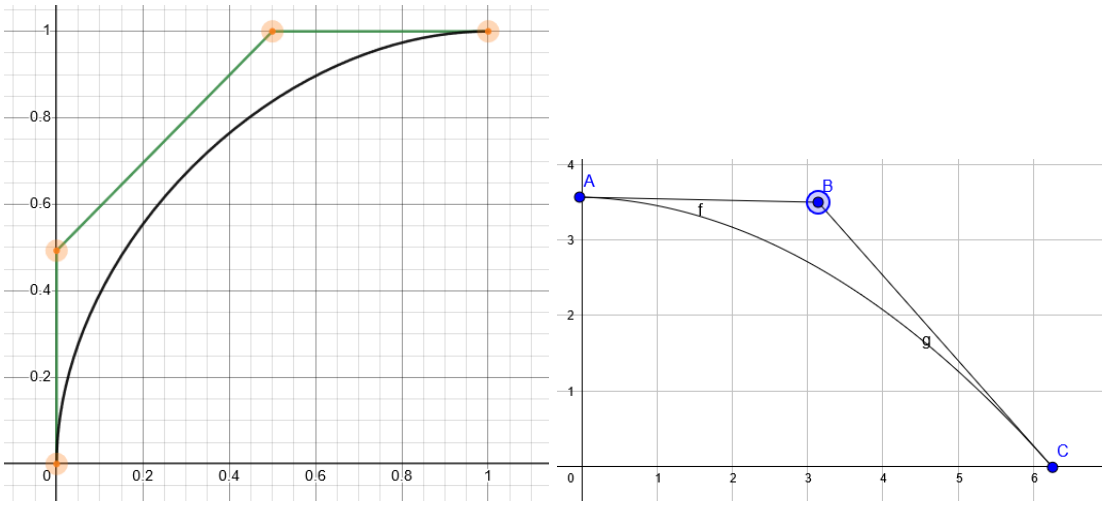


Figure 5.2 – Bézier cubique + quadratique

Ensuite je redimensionne par proportionnalité aux dimensions de la table ce qui me demande les coordonnées x et y correspondant à l'axe de longueur et de largeur de la table en prenant en compte que le point d'origine est le centre.

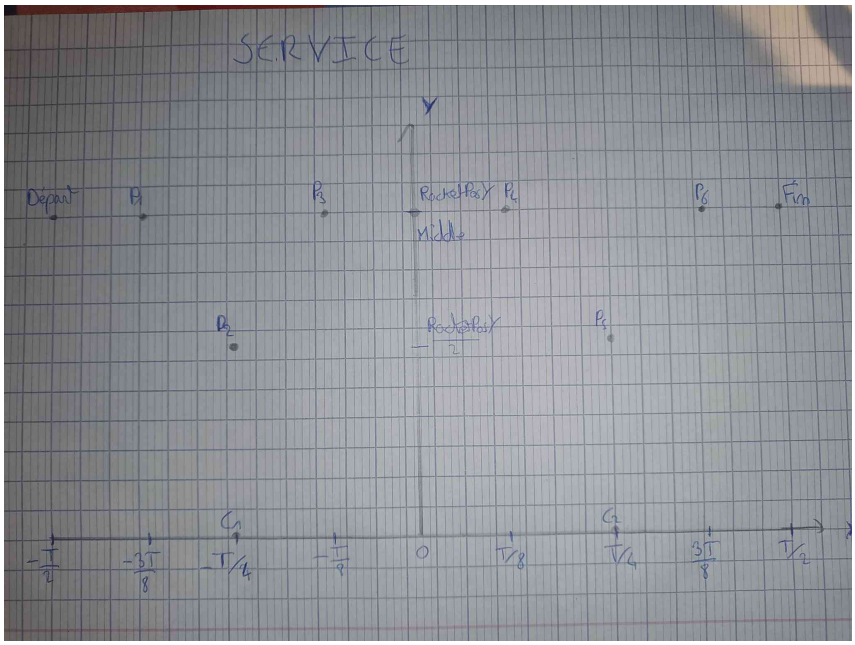


Figure 5.3 – Position des points de service

Ici, Pn correspond au n-ième point de contrôle et Cn au n-ième point de contact.

Passons à la coordonnée z qui correspond à l'axe de la largeur de la table. Afin de créer une trajectoire qui semble rectiligne vue du dessus et en connaissant 2 coordonnées z correspondant au point du début et au point de contact de l'autre côté de la table, nous avons choisi d'utiliser le théorème de Thalès.

Quant aux autres trajectoires, nous avons choisi de trouver l'équation de la droite formée par 2 points.

Chapitre 6

Conclusion

6.1 Récapitulation des Points Clés

Ce rapport a exploré en détail la conception et la réalisation d'un projet de Synthèse d'Images, mettant en avant l'utilisation de la librairie THREE.js. Les objectifs du projet ont été présentés, mettant en lumière la modélisation réaliste d'une table de ping-pong, la simulation d'un jeu de ping-pong, et l'interactivité à travers un menu GUI.

6.2 Difficultés et Solutions

La réalisation de ce projet n'a pas été exempte de défis techniques. La gestion des trajectoires de la balle, notamment la synchronisation avec les courbes de Bézier, a demandé une approche minutieuse. La limitation des courbes G(0) et la vitesse proportionnelle aux échanges ont été des compromis nécessaires.

L'implémentation du menu GUI, en particulier la modification des points de contrôle des Lathe, a également posé des défis. La gestion des transitions entre les étapes de modification et la synchronisation avec la création des pieds de la table ont demandé une attention particulière.

Chapitre 7

Annexes

.1 Annexe : Concepts JavaScript Utilisés dans le Projet

Dans cette annexe, nous allons détailler les concepts JavaScript utilisés dans le code du projet. Ces concepts incluent des notions de programmation orientée objet (POO), la gestion des événements, la manipulation du DOM, les fonctions fléchée et d'autres principes.

.1.1 Variables

Dans le code, plusieurs variables sont déclarées pour stocker des données telles que la position de la caméra (`cameraYController`, `camera`, `cameraPosition`), la largeur de la table (`tableWidth`), l'instance de la table (`tableInstance`), etc.

.1.2 Fonctions et Méthodes

Des fonctions sont utilisées pour définir des comportements spécifiques. Par exemple, la fonction `onChange` est associée à des contrôleurs pour détecter les changements et déclencher des actions. D'autres fonctions, comme `set`, `lookAt`, `getWidth`, `getLength`, `getHeight`, `animate`, `requestAnimationFrame`, `update`, `render`, etc., sont également définies pour différentes tâches.

.1.3 Objets

L'objet `camera.position` est utilisé pour représenter la position de la caméra dans l'environnement 3D. Il est mis à jour en fonction des changements spécifiés par l'utilisateur dans le menu GUI.

.1.4 Fonctions Anonymes et Gestionnaires d'Événements

Des fonctions anonymes sont utilisées comme callbacks dans des événements tels que `onChange` et `addEventListener`. Ces événements sont des gestionnaires d'événements qui réagissent aux interactions de l'utilisateur, tels que le changement de couleur de la table ou le déplacement de la caméra.

.1.5 Animation

L'animation est gérée à l'aide de la fonction `requestAnimationFrame`, qui permet une animation fluide en synchronisation avec le taux de rafraîchissement du navigateur.

.1.6 Contrôle de la Caméra

Le contrôle de la caméra est assuré par l'objet `controls`, qui est mis à jour à l'intérieur de la fonction `animate`.

.1.7 Rendu de Scène

Le rendu de la scène est effectué avec la méthode `renderer.render`, qui prend en compte les différents éléments et les affiche dans la fenêtre du navigateur.

.1.8 Fonctions d'Instance

Des fonctions d'instance sont utilisées pour obtenir des propriétés spécifiques de l'instance de la table, telles que sa largeur, sa longueur et sa hauteur (`tableInstance.getWidth`, `tableInstance.getLength`, `tableInstance.getHeight`).

.1.9 Manipulation de la Fenêtre et du DOM

La taille de la fenêtre est prise en compte avec `window.addEventListener` pour détecter les changements de taille et ajuster la scène en conséquence. La fonction `resize` est appelée lors de l'événement de redimensionnement.

.1.10 Programmation Orientée Objet (POO)

L'utilisation de classes telles que `Gui`, `Table`, et `TableLeg` reflète la programmation orientée objet, où des objets sont créés à partir de classes avec des propriétés et des méthodes spécifiques.

.1.11 Méthodes de Chaîne

Des méthodes de chaîne sont utilisées pour définir la position de la caméra avec `camera.position.set`.

Ces concepts démontrent une utilisation variée des fonctionnalités de JavaScript dans le projet, contribuant à la création d'une expérience interactive dans un environnement 3D.