TP 2 WebGL : Courbes de Bézier avec three.js

Groupe : "Les Chèvres"

- **Membres**:
  - Eudeline Nathan
  - Evain Sacha
  - Cacheux Nolan
  - Chiadmi Yassine

Introduction

Le TP2 s'est focalisé sur l'exploration et la mise en application des courbes de Bézier en utilisant la bibliothèque treeJS. Notre objectif était d'acquérir une meilleure compréhension du fonctionnement des courbes de Bézier tout en manipulant ThreeJS.

Répartition des Tâches

Lors de la première séance, nous avons travaillé sur le sujet individuellement, afin de se familiariser avec les exercices sans se reposer sur l'un des membres.

Pour ce rendu final, Sacha et Nathan ont travaillé sur l'ajout et la modification de points, en implémentant les deux méthodes proposées. Yassine a réalisé l'interface utilisateur et a collaboré avec Nolan sur le calcul puis la visualisation des courbes de bézier.

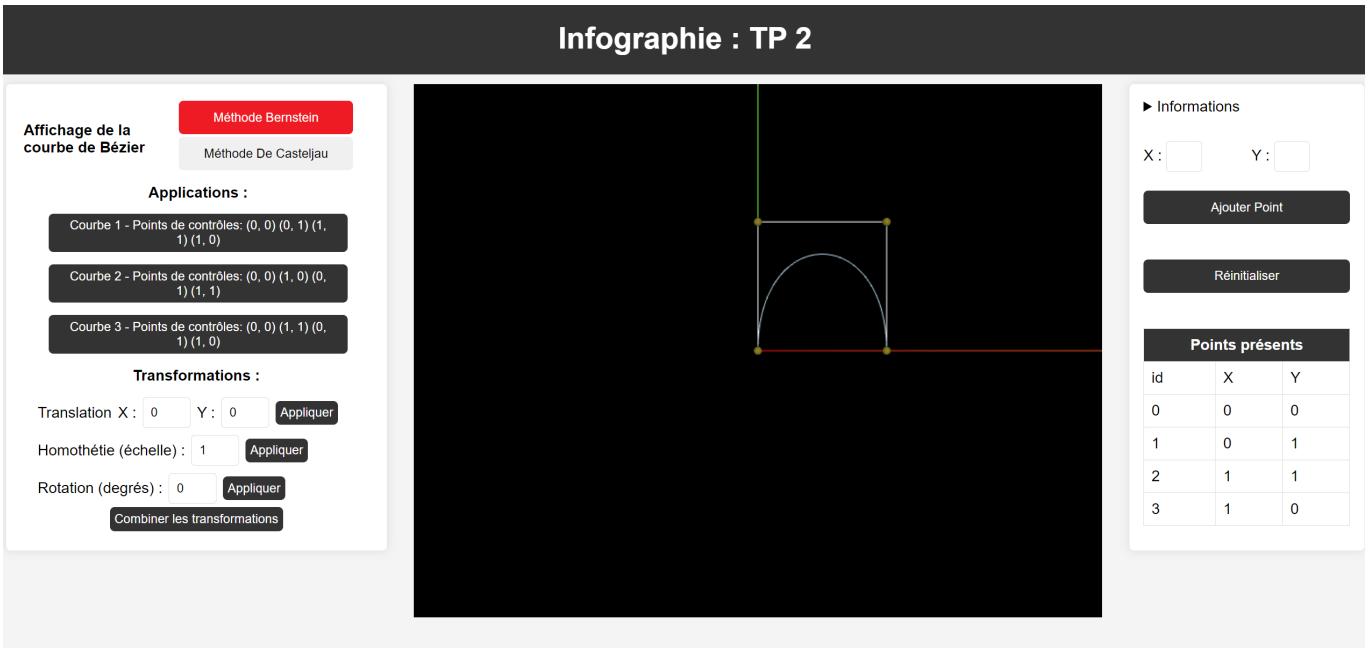
Travail Réalisé

Nous avons réalisé une page qui implémente le projet threejs dans un canvas affichant les courbes, un menu permettant d'ajouter des points ou de réinitialiser le graph, une table résumant les points présents et d'un second menu où l'on peut choisir quelle méthode sera utilisée pour déterminer les courbes de bézier.

Il est possible d'ajouter un point via le menu en donnant ses coordonnées ou par un clic-gauche de la souris.

Par un clic molette (avec une souris), on peut déplacer un point, ce qui met à jour les éléments de la page.

Pour tous les exercices, nous avons chargé la librairie `three.js`, puis nous avons créé une scène et une caméra décalée selon l'axe Z afin de voir les objets placés à l'origine.



Ajout et modification de points

Pour intéragir avec les points, nous avons utilisé un `listener` qui détecte les clics de la souris. Si le clic gauche est utilisé, la fonction `addPoint` est utilisée, et si c'est le clic molette, alors on va tenter de déplacer le point qui intersecte potentiellement avec notre souris.

La complexité de l'ajout d'un point était de convertir les coordonnées de notre souris en coordonnées dans le plan. Grâce à un `raycaster`, nous avons pu calculer les coordonnées de notre la dans le plan (O,X,Y).

Pour la modification de point, on a instancié une variable "draggable" ("trainable" en français, c'est à dire le point à déplacer.) qui contiendra les informations du point en mouvement jusqu'à sa position cible. En utilisant encore une fois le raycaster, on détermine si un objet *Mesh* intersecte la souris, puis on met ses informations dans la variable "draggable". Enfin, lorsqu'on re-clique sur la molette, la scène est modifée, les courbes sont recalculées et la variable est vidée.

Interface

Par l'utilisation de JQuery, une fonction récupère les points contenus dans notre scène et rempli une table contenant l'id et la position de chaque point.

On peut également réinitialiser le graphique, c'est à dire supprimer les points présents.

Calcul des courbes de bézier

Le but essentiel de l'application est de calculer des courbes de Bézier. Comme proposé dans l'énoncé, nous avons implémenté les deux méthodes :

- La méthode de Bernstein, en calculant les fonctions de base de Bernstein.
- L'algorithme de De Casteljau, pas à pas.

Ces méthodes nous ont permis de tracer des courbes de Bézier précises en fonction des points de contrôle ajoutés par l'utilisateur.

Conclusion

Ce TP était un bon moyen de mettre en pratique la théorie sur les courbes de bézier vu en cours, mais aussi un bon entrainement sur la gestion de la caméra, des plans et intersections entre éléments...