RAPPORT

UNIVERSITÉ DE BOURGOGNE

Fatima Zahra Talibi
Imane Choukri

SOMMAIRE

Nos Choix Techniques :

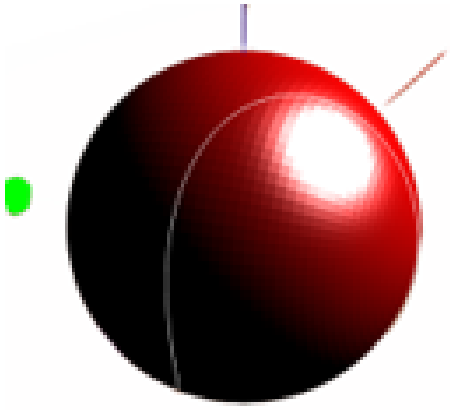
-Après avoir récupérer les fichiers nécessaires à notre projet comme demandé en cours (le fichier libs.tgz) puis le fichier init.js qu'on va implémenter au fur et à mesure de l'avancement de notre projet.

-On a commencé par définir l'entête de la fonction init.js : on commence par déclarer notre scène qui est essentielle pour la visibilité des autres objets. ensuite le moteur de rendu pour notre scène.

-Puis, on ajoute une caméra pour filmer la scène, on a choisi d'utiliser une caméra de type perspective qui sera animée via la Souris grâce à la fonction "animate".

LA BOULE :

Puis nous avons décidé de commencer la partie géométrique par la conception de la boule dont on définit la couleur de l'objet, la couleur spéculaire et sa brillance. puis on a associé la courbe et la balle de tennis dans un même groupe pour créer un objet unique (comme la boule de tennis vu en tp).

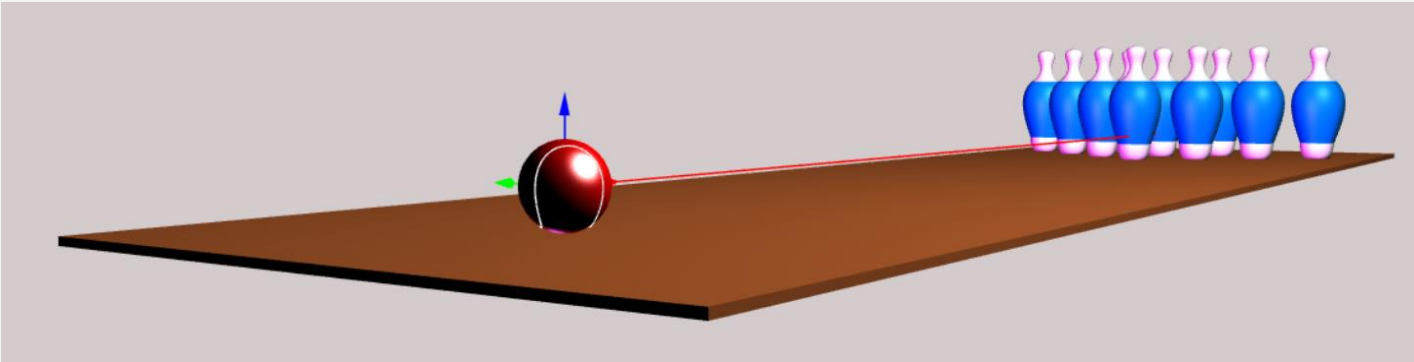


Les Pistes :

On entame ensuite la création de la piste qui est un cube allongé pour donner une sorte de parallélogramme et ainsi créer l'illusion de la piste de bowling. pour créer un cube, nous avons besoin d'une boxgeometry.

C'est un objet qui contient tous les points et le remplissage (faces) du cube et un mesh (maillage) qui est un objet qui prends une forme (géométrie), et qui y applique un matériau (matériel), que nous pouvons ensuite insérer dans notre scène, et déplacer librement.

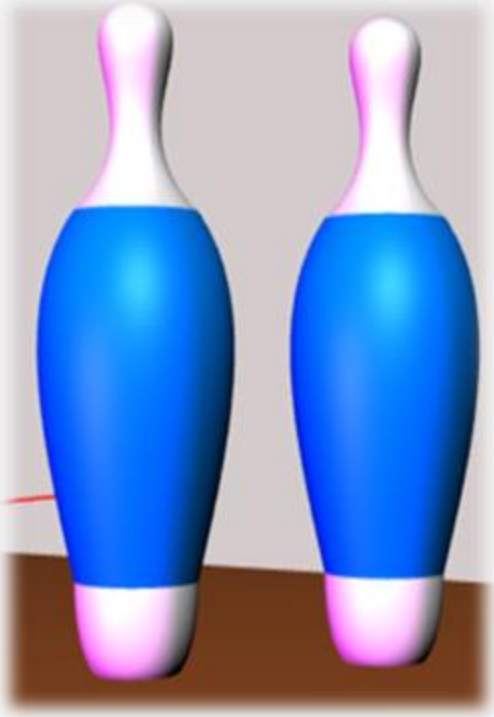
On place donc ce cube dans la scène dans une position en accord avec la boule qui est déjà présente (ainsi la boule sera au-dessus de la piste et on évitera la superposition de la piste et la caméra) après lui avoir accordé des propriétés de couleur, de brillance et de matière.



Les Quilles :

Création:

Pour créer les quilles nous avons tout d'abord créer 3 lathes qu'on va par la suite superposer afin de les assembler et ainsi créer la quille comme demandé dans le sujet.
On a donc créer une fonction nommé "quille" qui appelle à chaque niveau de la quille (à chaque lathe) la fonction lathe qui prend en paramètres le nombres de points sur la courbe de bézier, le nombre de segments de circonférence à générer,4 points de contrôle et une couleur.



Cette fonction fait appel à un constructeur nommé : lathegeometry , qui prend à son tour les paramètres suivants :

- **Points;**

Tableau de vector2s. la coordonnée x de chaque point doit être supérieure à zéro. La valeur par défaut est un tableau avec (0,-0.5), (0.5,0) et (0,0.5) qui crée une forme de losange simple.
le nombre de segments de circonférence à générer. la valeur par défaut est 12.

- **l'angle de départ en radians:** qu'on a fixé à 0.
- **La plage de radian (0 à 2pi) de la section du vase:** qu'on a fixé à 0.2pi.

La courbe de bézier :

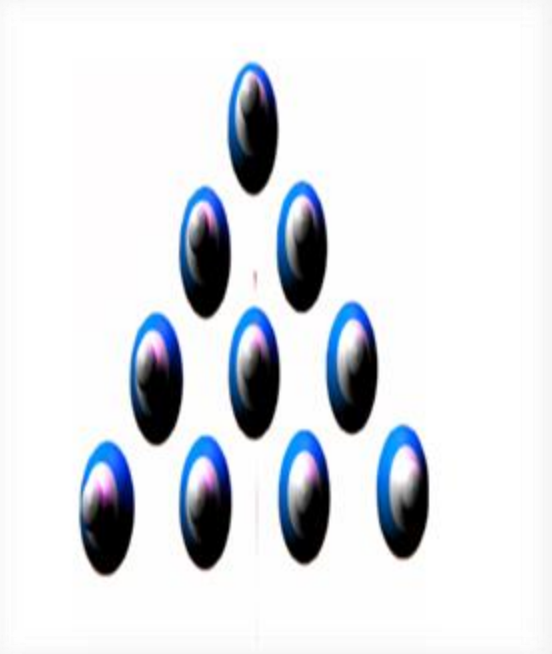
On a choisi d'utiliser une courbe de bézier cubique car elle permet d'assurer la continuité en tangence et en courbure de deux courbes raccordées. la courbe se trace en partant du point p0, en se dirigeant vers p1 et en arrivant au point p3 selon la direction p2-p3. la forme paramétrique de la courbe s'écrit comme une combinaison affine des points de contrôle. Qu'on définit grâce à la formule suivante :

$p = (1-t)3p1 + 3(1-t)2\,tp2 + 3(1-t)t2p3 + t3p4.$

Grace a cette fonction on a pu créer 3 lathes dans la fonction "quille" qui différent l'une de l'autre afin de créer le haut de la quille, le milieu et le bas de la quille. puis on les a regroupés grâce à "grouplathe" qu'on va retourner à la fin de la fonction.

Insertion:

Afin de placer les quilles sur la piste nous utilisons de boucles afin de les placer sous forme de triangle vu d'en haut dont le somment et dirigé vers la boule. On donne à chaque quille un emplacement (20+i) ce qui placera chaque quille assez loin de la boule puis on donne à chaque quille un y qui varient de -i jusqu'a i de la précédente boucle. On termine par ajouter la quille créée dans le tableau de quilles puis à la scène et donc dans une position afin qu'elle soit sur la piste.



La caméra:

Dans une application 3D, implémenter le contrôle de la caméra est une étape indispensable pour proposer du contenu interactif et intuitif. Nous avons donc utilisé la classe de contrôle OrbitControls permet de faire orbiter la caméra autour d'une cible. Son constructeur prend en paramètres la caméra et le moteur de rendu de notre application.Nous avons aussi utilisé la fonction animate qui va permettre à l'utilisateur de tourner autour de la scène.

Animation :

Pour faire avancer la boule vers les quilles, nous avons utilisé une fonction nommée ' avance', cette dernière permet au centre de la boule de se déplacer tout au long de la piste et ainsi créer l'illusion du mouvement de la boule. Nous rajoutons à cette action, la rotation de la boule afin de simuler un mouvement réaliste de la boule de bowling.

Cette fonction ne sera utilisée que lors du lancement de la boule, comme demandé au sujet, la boule se déplacera selon un plan non rectiligne via une trajectoire définie par au moins deux courbes de Bézier faisant une jointure. Quand la boule est lancée, on lance une boucle qui va vérifier si chaque boule du tableau (contenant toutes les quilles sur la scène) est entrée en collision avec la boucle, si oui, la quille en question sera supprimée de la scène. Finalement, on utilisera la méthode setTimeout qui permet de définir un minuteur qui exécute une fonction ou un code donné après la fin du délai indiqué.

Comment on vérifie si la quille est touchée par la boule ?

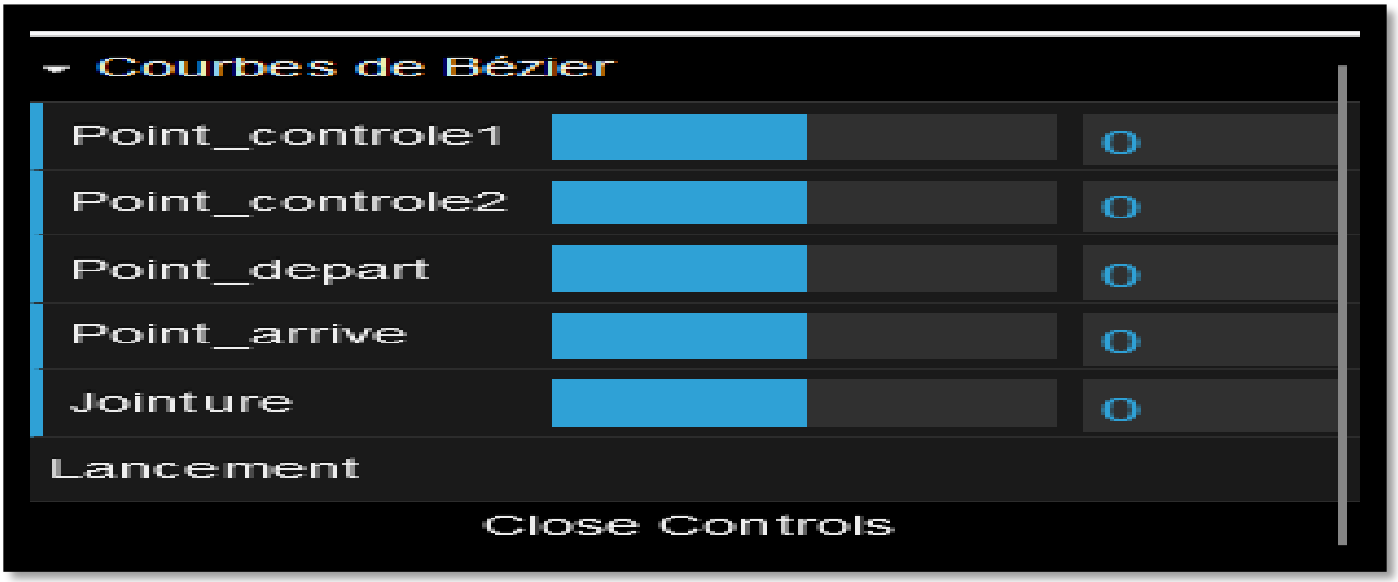
Pour cela, on a utilisé la fonction 'Collision' qui prend en paramètres deux objet de la scène.
Dans ce cas, la boule et la quille. La logique ici est d'envelopper les deux objets chacun de son coté d'une box, puis vérifier si ces deux box rentrent en collision grâce à la méthode "intersectsBox" puis retourner True si c'est le cas.

Comment avons-nous défini les courbes que la boule va suivre ?

Nous avons tout simplement créé deux courbes de Bézier qu'on va associer grace à une Jointure G.
Pour cela, nous avons utilisé la fonction TraceBezier "Quadratique" qui prend 4 points de contrôles en paramètres, un nombre de points, une couleur et une largeur définissant l'épaisseur de la ligne de la courbe.
On a donc défini quatre points de contrôles, un nombre de points par courbe (50) et une épaisseur de 2.
Finalement, on ajoute les deux courbes créer dans la scène.
Ces deux derniers auront l'air d'une trajectoire rectiligne car les deux points de contrôles, le point d'arrivée, le point de départ et la jointure seront alignés par défaut sur la piste.

LE MENU GUI :

Comme vu au cours , nous avons créé un menu gui qui va nous permettre de contrôler les deux courbes qui déterminent la trajectoire de la boule , ainsi que le point qui les lie (la jointure), le point d'arrivée et le point de départ de la boule. On a implémenter la variable menuGUI gérant le menu dans la fonction init() en positionnant la caméra et en donnant des valeur par défaut au éléments du menu qui sont dans le même Folder nommé '' Courbes de Bézier''. Puis on simule un lancement par défaut avec une valeur de 0. Ensuite, on actualise en réaffichant le rendu dans la scène. Finalement, on ajoute le lancement au menu qui fait appel à la fonction ''lance()''.



Comment peut-on changer la valeur d'un élément dans le menu GUI ?

Pour cela, on a fait appel à la fonction ''CourbeGui'' qui prend en paramètres les deux courbes de Bézier et leurs points de contrôle , la boule , le variable menuGUI et la variable pour le menu nommée ''gui''. Cette fonction créer la variable ''guicourbe'' où on ajoute au fur et à mesure le Point_controle1 , le Point_controle2 ,le Point_arrive,le Point_depart et la Jointure. Les valeurs de ces éléments varient entre -3 et 3 .

On a aussi modifier la partie du point de départ pour que la boule soit supprimée de la scène avant la modification et ainsi permettre à l'utilisateur de bien placer son point (on ajoutera la boule ensuite) .

❖ LES DIFFICULTES RENCONTREES :

Grace aux cours ,aux tps et leur corrigé , nous avons pu avancé dans notre projet . Nous avons aussi fait beaucoup de recherches sur internet afin de trouver les méthodes les plus adéquates et logiques qui pourront nous aider dans notre projet .

Cependant ,on a eu des difficultés qu'on a pu surpasser comme: l'emplacement des fichiers , les classes utiles , la conception des quilles et des courbes de Bézier , mais aussi l'implémentation du menu GUI .

Malheureusement, on a fait face à des problèmes qu'on n'a pas su résoudre pour des raisons d'incompréhension ou juste par manque de temps :

- On voulait ajouter une fonction qui contrôle les collisions entre les quilles , cette dernière remplacera chaque quille entrée en collision avec une autre par un petit cube .
- Le score : nous n'avons pas eu le temps de faire le tableau qui affiche le score mais l'idée était de compter le nombre de cube sur la piste à chaque lancer .Vérifier au premier lancer si ce nombre est égale à 10 , si c'est le cas : on ajoute au score 30 points , sinon on ajoute au score le nombre nb de quilles transformées en cubes après le lancer . Au deuxième lancer , on aura deux cas :

-Si au premier lancer, le score est resté à zéro , on vérifie si après le 2e lancer toutes les quilles sont tombées. Si c'est le cas on ajoute 15 points au score.

-Si au premier lancer quelques quilles ont été déjà tombées. Dans ce cas on ne comptera que les nouveaux cubes qui apparaissent au 2e lancer , afin d'éviter de rajouter au score les points des quilles qui sont tombées au premier lancer .

- On aurait aussi voulu replacer chaque quille tombée par un petit cube cependant on a choisi de supprimer la quille définitivement de la piste . Pour cela , il fallait récupérer la position de la quille en question dans une variable et puis créer un cube en lui donnant comme position la variable créée.

Finalement, malgré les difficultés rencontrées et le manque de temps dont on a fait face, ce projet nous aidé à être plus à l'aise dans la manipulation de la librairie three.js et du javascript en général. Nous avons essayé d'expliquer chaque partie de notre code dans ce rapport (sans rentrer dans les petits détails vus en cours) . Nous avons aussi commenté quelques parties de notre code afin qu'il soit clair et bien organisé. Pour conclure , ce projet nous a donné envie d'approfondir nos connaissances et de découvrir de nouvelles librairies Javascript afin de maitriser le module INFO3B .

MERCI .