RAPPORT DE PROJET INFORMATIQUE

Synthèse d'image : *Jeu de Bowling en THREE.JS*

Equipe

MEILLIER Léo

AFANVI KODJO Roland

Enseignant

GARNIER Lionel

Année 2022 – 2023

Préface :

Ce document a pour but de décrire le déroulement de notre projet de synthèse d'image en informatique Info3B.

Ce rapport contient l'ensemble des éléments du projet. D'un point de vue technique, nous verrons dans un premier temps la construction de la zone de jeu puis les spécifications plus détaillées qui en découlent. Nous décrirons ensuite le fonctionnement de notre projet dans son ensemble. Enfin, nous présenterons nos impressions sur le projet concernant les difficultés techniques rencontrées.

La deuxième partie de ce rapport est consacrée à des schémas expliquant les constructions des quilles du jeu ainsi que la manière dont nous avons géré les chocs entre les quilles et les boules de bowling. Leur type et leur mise en œuvre dans le programme seront également présentés en détail.

Nous espérons que vous prendrez autant de plaisir à lire ce rapport que nous en avons pris durant le déroulement de ce projet.

SOMMAIRE

Scénario :

Pour bien comprendre le projet, un scénario est décrit ci-dessous, dans lequel nous devons créer un jeu de bowling en trois dimensions.

En utilisant la librairie THREE.js (une bibliothèque JavaScript permettant de créer des scènes 3D dans un navigateur web), nous devons réaliser 2 mènes par équipe d'une partie de bowling. Une mène est composée au plus de 2 lancers pour faire tomber toutes les quilles.

Il y a deux équipes de couleurs différentes, ici Rouge et Bleue.

Si toutes les quilles sont renversées au premier lancer, l'équipe marque 30 points. En revanche, si toutes les quilles sont renversées au second lancer, l'équipe marque 15 points.

Sinon, les points marqués par une équipe correspondent au nombre de quilles tombées.

I. Rapport technique

1. Cahier des charges

Il nous a été fourni au début du projet un cahier des charges, des contraintes précises à respecter.

A. Contraintes

1.1) Construction des quilles

- A l'aide d'au moins trois surfaces de révolution avec un raccourci G^1 entre chaque surface, l'utilisation d'au moins deux "lathe" lisses est obligatoire et celles-ci doivent se joindre entre elles ;

- Pour chaque quille, la "lathe" intermédiaire a une couleur différente des deux autres surfaces de révolution ;

1.2) Propriétés des quilles et des boules

- Lors de la chute d'une quille par une boule de bowling, cette quille sera considérée comme un parallélépipède rectangle le plus petit possible ;

- Les boules sont aux couleurs de l'équipe et contiennent une courbe comme par exemple, celle d'une balle de tennis, de la couleur de l'autre équipe.

1.3) Contraintes sur les déplacements lors des animations

- Il faut au moins un déplacement rectiligne ;

- un déplacement plan non rectiligne via une trajectoire définie par au moins deux courbes de Bézier faisant une jointure G^1 ;

- un menu GUI permettant de modifier la trajectoire de la boule sur la piste ;

- après chaque lancer, le score est ajouté dans un tableau dans la page HTML. La couleur du texte est celui de l'équipe tandis que les bordures des cellules sont de la couleur de l'équipe adverse ;

- Il est interdit d'arrêter une boucle en utilisant le mot "break".

2. Spécifications fonctionnelles et détaillées

Dans un premier temps, avant de commencer toute modélisation, nous avons créé la scène ainsi qu'un repère orthonormé direct dans une scène. Ce dernier a été associé à un menu GUI permettant de modifier la position de la caméra (3 degrés de libertés), la direction de visée de la caméra (3 degrés de libertés) et d'actualiser la scène. A noté que la plupart des fonctions vues en TP telles que celles sur la lumière de la caméra, les courbes de Bézier ou encore les Géométries utiles ont été importées dans ce projet.

Voici une capture d'écran présentant le menu GUI au début du projet :

En complément, un tableau de score entre l'équipe rouge et l'équipe bleue a été réalisé en dessous de la scène. On y retrouve quatre colonnes associées à quatre lancers et une dernière colonne consacrée au score total afin de comparer facilement les scores des deux équipes et de distinguer rapidement le vainqueur.

Une fois ces manipulations faites, nous pouvions enfin attaquer la modélisation de la piste, des quilles et des boules de bowling.

A. La piste

Nous avons commencé par réaliser la piste de bowling puisque c'est la base sur laquelle les quilles et les boules viendront se reposer. Il nous semblait alors judicieux de la créer en premier.

Pour se faire, quatre constantes associées respectivement à la largeur du plan, la hauteur du plan, le nombre de segments dans la largeur et le nombre de segments dans la hauteur ont été posées.

Ces dernières sont essentielles dans la fonction "plansol" qui permet de créer une palette de la piste de bowling. (En effet, nous avons cherché à reproduire un parquet en bois similaire à une vraie piste.)

Afin de réaliser chaque palette du sol, nous avons appelé la fonction pour créer une palette beige foncé puis une palette beige clair en alternance. Dans le but de modéliser une deuxième piste, on a simplement utilisé les positions opposées dans les conditions de la fonction.

Cette manière de réaliser la piste est triviale puisque l'on aurait simplement pu créer une base et appliquer une image dessus. Néanmoins, elle nous paraissait plus appropriée en cas de rajout, d'élargissement ou d'allongement de la piste.

B. Les quilles

Après avoir créé la piste, il était essentiel de modéliser les quilles car c'est autour de ces dernières que les principales actions du jeu (notamment les collisions avec les boules) vont se dérouler. Afin de modéliser une quille, nous avons décidé de séparer cette dernière en trois parties, sa tête, son tronc et sa queue. Pour cela, trois fonctions respectivement associées à chaque partie de la quille ont été créées (Nous ne parlerons pas des Courbes de Bézier par respect de l'énoncé du projet mais seulement des points de contrôle choisis). Une fonction "latheBezTab" a aussi été définie au préalable.

Tout d'abord, nous avons défini des points de contrôles fixes afin de pouvoir les réutiliser lors de la création de chaque partie de la quille.

```
let P0 = new THREE.Ve( any )0.5,0.02);
let P1 = new THREE.Vector3(1,1);
let P2 = new THREE.Vector3(0.71,1.57);
let P3 = new THREE.Vector3(0.5,1.984);

let M1 = new THREE.Vector3(0.29,2.29);
let M2 = new THREE.Vector3(0.33,2.65);
let M3 = new THREE.Vector3(0.374,2.88);

let N1 = new THREE.Vector3(0.43,3.13);
let N2 = new THREE.Vector3(0.5,3.344);
let N3 = new THREE.Vector3(0,3.5);
```

Par la suite, nous avons défini une fonction “creerQueue” dans laquelle nous prenons quatre points de contrôles définis auparavant. Une première lathe, créée grâce à la fonction “latheBezTab”, subit une rotation puis une translation par axes

Puis, nous avons répété ce même processus pour la fonction “creerTronc” en prenant d’autres points de contrôles sauf le premier qui est le même que le dernier point de la queue. Cela permet de relier facilement les deux lathes entre elles. Les actions de rotation et de translation sont à nouveaux répétées sur cette nouvelle lathe.

Enfin, nous avons créé une dernière fonction “creerTete” en prenant d’autres points de contrôles sauf le premier qui est le même que le dernier point du tronc, afin de relier à nouveau cette nouvelle lathe à celle précédemment créée. Des actions de rotation et de translation similaires aux deux autres lathes sont à nouveaux répétées.

Distingué ces trois parties pour une quille nous donne plus de facilité pour changer un endroit de la quille. Il nous paraissait bien plus judicieux que d’utiliser seulement deux lathes.

Néanmoins, ces fonctions à elles seules ne suffisaient pas à créer une quille au complet. C’est pourquoi nous avons défini une fonction “QuilleInit” qui créer des quilles (dans un tableau) dans la scène en fonction de leurs positions dans l’espace. Cela nous permet donc d’obtenir les 10 quilles alignées comme dans un vrai jeu de bowling.

Autrement dit, nous avons modélisé trois lathes lisses raccordées entre elles pour chaque quille.

C. Les boules de bowling

Pour former chaque boule, nous avons défini une constante R puis tracé une sphère de centre l'origine et de rayon R avec un matériau Phong. Ainsi, elles contiennent chacune une courbe similaire à une balle de tennis. En outre, nous avons simplement réutilisé les fonctions vues en TP en changeant leur opacité, leurs couleurs (celles des deux équipes). De plus, nous avons choisi, par simple problème d'esthétisme, de rendre les courbes de balles de tennis visibles et de les colorer des couleurs de l'équipe adverse.

Toutes les modélisations demandées ont été faites. Place maintenant à tout ce qui est en rapport avec le déroulement du jeu. Autrement dit, nous parlerons des trajectoires des balles, des actions de collisions ou encore de la réaction des quilles face à ce chocs avec une balle.

3. Fonctionnement

A. Trajectoires des boules

Parlons maintenant des choix faits concernant la trajectoire des balles.

Dans un premier temps, nous avons posé sept points de contrôle fixes puis défini deux courbes de Bézier de degré 3 en utilisant les points créés précédemment. Puis, nous avons défini deux courbes géométriques en faisant appelle à la fonction "Geometry".

Cependant, nous voulions que la trajectoire choisie par l'utilisateur soit visible sur la scène afin qu'il puisse visualiser rapidement et ajuster son lancer en cas de mécontentement. C'est pourquoi nous avons défini un matériau faisant appelle à la fonction "LineBasicMaterial".

Pour finir, nous avons terminé en définissant deux sommets représentant les segments des lignes dans laquelle on utilise comme paramètres les deux courbes géométriques ainsi que le matériau.

B. Action de collisions et réaction des quilles

Une fois les trajectoires des balles posées, il ne nous restait plus qu'à discuter de la manière dont nous allions gérer les collisions entre la balle et les quilles. D'autant plus que les quilles, lors de leur chute, doivent être considérées comme des parallélépipèdes rectangles le plus petit possible.

Dans un premier temps, nous avons défini une fonction “detecteCollisionCubes” qui détecte un choc entre deux objets. Plus précisément, elle compare les objets entre eux et renvoie vrai ou faux si elle trouve qu'ils se croisent. Au sein de cette dernière, les fonctions “computeBoundingBox” et “updateMatrixWorld” sont appelées deux fois lors de la création des box. Idéalement, ces fonctions seraient appelées une seule fois, plutôt qu'à chaque détection comme ici, pour améliorer les performances. Nous avons préféré rallonger de peu notre code plutôt que se perdre dans des détails sans importance, d’autant plus que nous n’avons que deux objets à traiter.

Enfin, nous avons défini une fonction “CreerParallelepipede” qui transforme les quilles en parallélépipède rectangle. Au sein de celle-ci, nous avons posé plusieurs constantes consacrées à la largeur, la hauteur, la profondeur, jusqu’à la profondeur des segments afin de créer des rectangles précis. Puis, nous avons défini une boite géométrique dans laquelle nous avons appelé la fonction “BoxGeometry”. Nous lui avons attribué une couleur grâce à la fonction “MeshBasicMatérial” qui remplit la géométrie à partir d’une couleur ou d’une texture sans tenir compte de l’ombrage. Pour finir, nous définissons un parallélépipède comme une association de la boîte géométrique et du matériau créé précédemment pour ensuite mettre à jour la position des quilles touchées par un parallélépipède rectangle.

C. Actions utilisateur

Lors du jeu, l’utilisateur peut effectuer de nombreuses actions. En outre, il pourra cliquer sur un des deux boutons, en fonction de la couleur de son équipe, dans le but de lancer sa boule. Pour cela, nous avons créé un bouton rouge qui, lorsqu’il est cliqué, modifie la position de la boule en fonction des choix faits dans le menu GUI. Un calcul du score est également effectué en détectant quelle quille est tombée ou plutôt combien de quilles sont tombées. Ces actions sont les mêmes pour le bouton bleu utilisé par l’adversaire.

Par conséquent, nous avions globalement implémenté toutes les actions de bases du jeu.

Par envie de donner davantage de clarté au jeu et à l'expérience utilisateur, nous avons fait défiler un message dans la partie supérieure du tableau affichant l'équipe qui doit porter son prochain coup.

4. Difficultés techniques et solutions apportées

Les différentes trajectoires de la boule

C'était véritablement la plus grosse difficulté technique. Nous avions réussi à effectuer un déplacement rectiligne mais nous n'arrivions pas à faire un déplacement plan non rectiligne défini par au moins deux courbes de Bézier faisant une jointure G^1. Ce qui nous empêchait non seulement de compléter le menu GUI, les options de choix de trajectoire de la boule par l'utilisateur. Nous nous sommes finalement renseignés davantage à l'aide du cours et avons analysé précisément ce que devait faire la boule. Quant au menu GUI, il était maintenant simple d'ajouter un choix de type de ligne et un intervalle modifiant la trajectoire de la boule.

Communications

Cette difficulté se réfère plus largement à celle de devoir apprendre à programmer différemment. En effet, nous étions habitués à utiliser le langage JavaScript pour développer des sites internet mais pas pour modéliser des objets, créer une scène et un jeu en trois dimensions. Il a fallu apprendre une nouvelle facette de ce langage avec les ressources disponibles du cours ou sur internet, le tout dans un intervalle de temps limité.

5.Conclusion et perspectives

Finalement, nous avons une version de base du jeu. La majorité des fonctionnalités de base ont été implémentées et fonctionnent correctement mais il reste quelques améliorations à faire pour aboutir véritablement à une version utilisable du jeu qui pourrait être utilisable par tous (de manière optimisé).

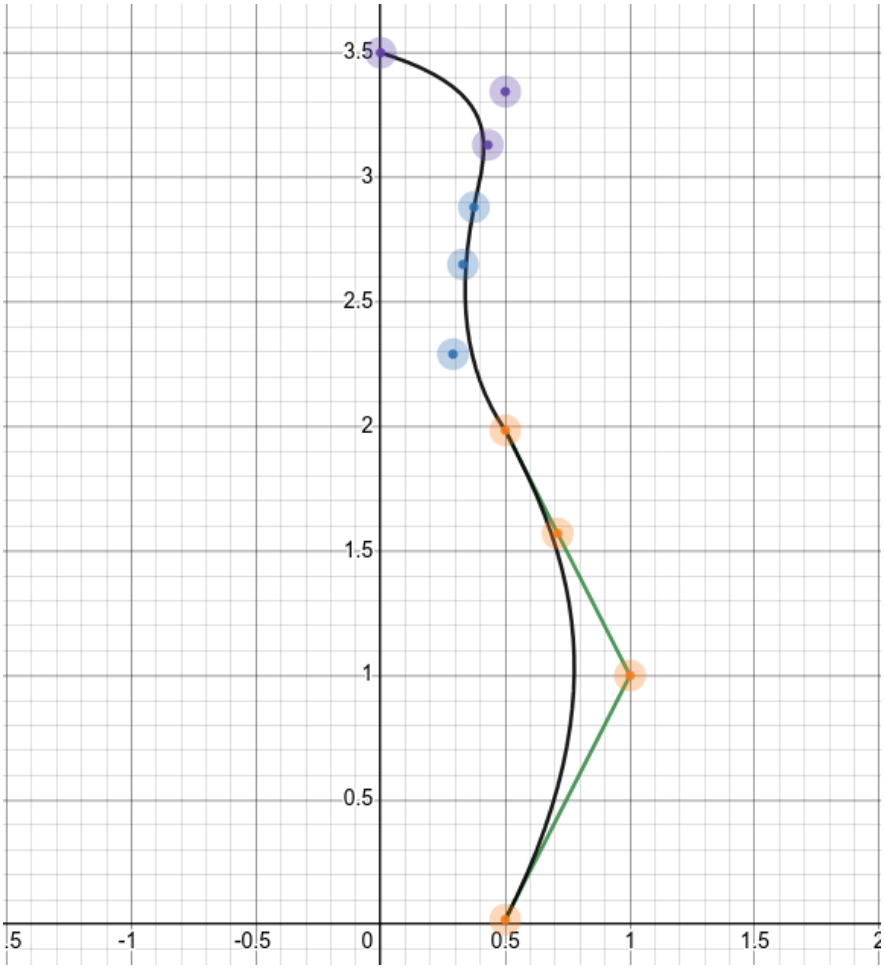
Quelques améliorations pourraient être ajoutées :

- Implémenter un menu qui permettrait de changer les points de vue de la caméra lors du lancer d'une balle.
- Visualiser un tableau des scores précis à la fin du jeu.
- Pouvoir accéder à une rediffusion animée après avoir effectué un strike.
- Modéliser un décor entier et y ajouter des effets de lumière plus soignés, etc...

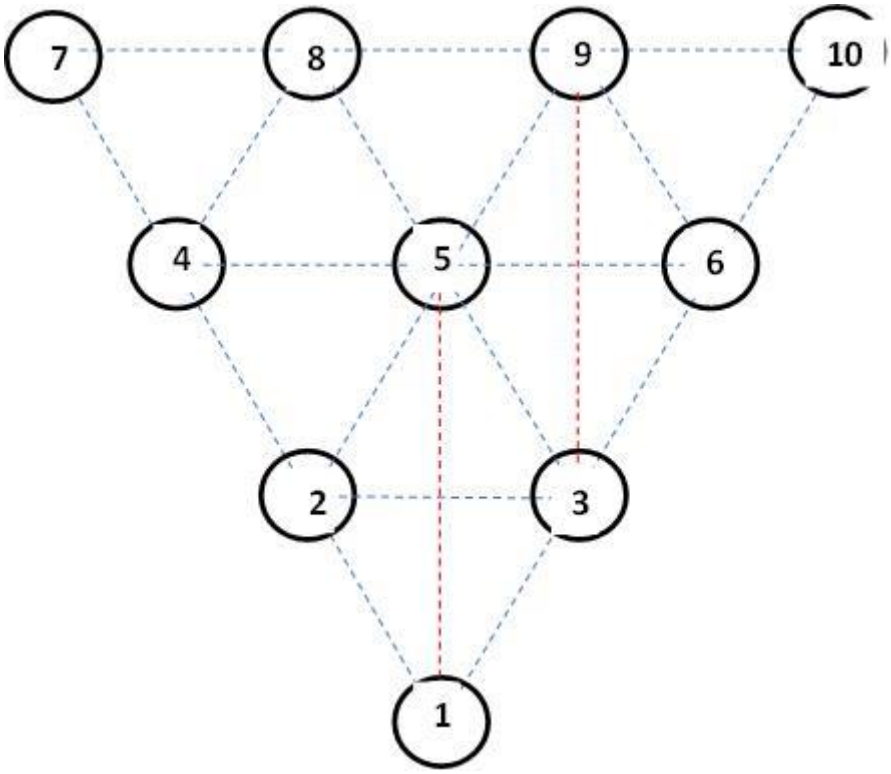
II. Visualisation et relation de collision

1. Schémas expliquant la construction des quilles

Voici un schéma représentant la construction d'une quille, faisant apparaître chaque point de contrôle. Ces derniers sont fixes et réutilisés pour chaque quille.

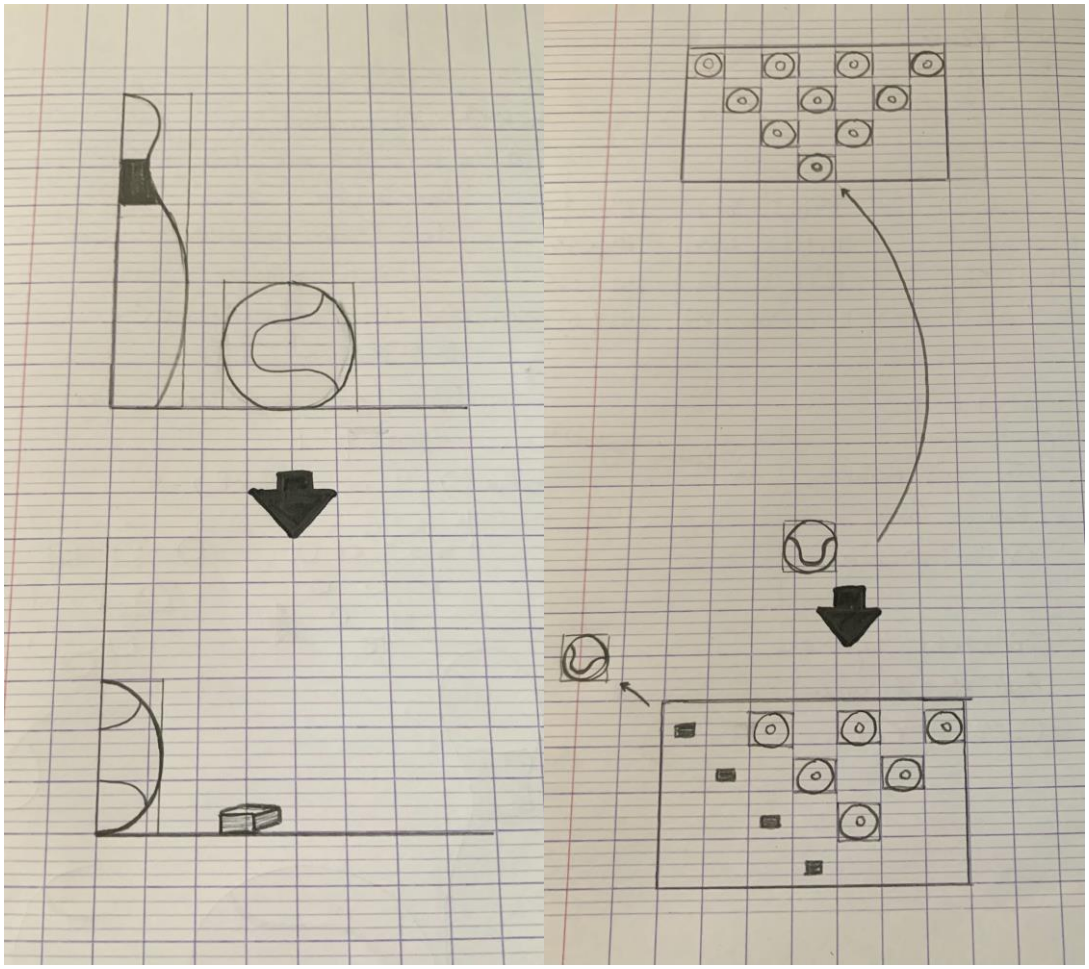


Concernant la position des dix quilles, son processus a été expliqué précédemment. Précisons seulement que 2 unités séparent chaque quille d'une même ligne et 1 unité entre chaque rangée.



2. Schéma expliquant les chocs

Voici deux schémas représentant les chocs entre la balle et les quilles :



III. Conclusion

C'est la première fois que nous programmons sur ce type de sujet. De l'avis général, nous avons pu consolider nos connaissances générales et appris à modéliser une scène plus ou moins complexe. Ainsi, nous sommes globalement satisfaits de ce que nous avons pu réaliser.

Au niveau de la gestion du projet en équipe, nous avons réussi à nous répartir les tâches afin de réaliser nos objectifs dans les temps. De plus, l'ambiance et la cohésion du groupe étaient très formidables.

BIBLIOGRAPHIE :

How to detect collisions three.js :
https://www.youtube.com/watch?v=9H3HPq-BTMo