Jeudi 02 Décembre 2021

Rapport Projet Info3B

Lien du GitHub : https://github.com/Sudo-Rahman/projetinfo3B

YILMAZ Rahman DUBOST Lucie

IE3-00-2b

Plan du rapport

1. Introduction

Nous avons eu à faire, lors notre troisième semestre de Licence en Informatique-Électronique, un projet portant sur une partie de curling, réalisé avec le langage JavaScript, plus précisément avec une bibliothèque spécifique, Three.js.

Three.js (ou ThreeJs) est une bibliothèque JavaScript utilisée pour créer des scènes 3D dans un navigateur web. Créée en 2010 par mrDoob, son code source est hébergé sur GitHub, dont vous trouverez le lien dans la Bibliographie.

Ce projet nous a été confié par notre professeur Lionel Garnier, achevant le module dont il a la charge : Info3B – Synthèse d'images.

Nous vous présentons donc aujourd'hui le résultat final d'un travail de plusieurs semaines, et espérons qu'il saura vous divertir.

Voici cependant quelques points à prendre en compte avant de vous lancer dans une partie.

2. Comment jouer

Inutile de vous le cacher, notre jeu n'est pas le plus minimaliste qui soit. Avec tous les paramètres qu'il offre, nous nous voyons dans l'obligation d'éclaircir vos lanternes.

Pour choisir la couleur de votre équipe, il vous faudra sélectionner votre équipe dans le menu « Partie », puis modifier les paramètres que vous souhaitez, comme la taille et la couleur de vos pierres et balais. La couleur représentant votre équipe sera la couleur du centre de vos palet, alors choisissez bien !

Avant de lancer une pierre, vous avez la possibilité de modifier sa trajectoire, qu'elle soit rectiligne ou qu'elle suive une courbe de Bézier. Adaptez la force de votre lancer et la force de frottement de vos balais pour tirer le plus proche de la maison.

Le tableau situé sous la piste contient la distance à la maison des palets lancés, selon l'équipe ayant tiré. Si votre équipe se retrouve mal en point vis-à-vis du score, n'hésitez pas à pousser les pierres de vos adversaires !

Remarques :

i) Il vous sera impossible de modifier les paramètres après le lancement de la partie.

ii) La couleur de la police du tableau est celle de l'équipe menant la partie.

Tableau des informations sur la partie

Lancers	1	2	3	4	5
Equipe 1	0.7 m du centre de la maison	0.93 m du centre de la maison	0.25 m du centre de la maison	0.72 m du centre de la maison	1.03 m du centre de la maison
Equipe 2	Hors piste	0.75 m du centre de la maison	0.32 m du centre de la maison	0.85 m du centre de la maison	0.98 m du centre de la maison

Figure 2.1 – Tableau affichant les distances à la maison des palets

3. Construction des objets

I - Construction de la piste

Pour implémenter la piste, nous avons utilisé la classe `CubeGeometry`, car elle nous a permis de facilement créer le rectangle servant de piste. Vous pourriez vous demander pourquoi nous avons décidé d'utiliser `CubeGeometry` au lieu de `BoxGeometry`, une méthode moins désuète. C'est tout simplement car nous avons utilisé cette dernière dans la création du balai, et il nous a paru plus agréable de changer de méthode, nous adaptant à la polyvalence de Three.js.
Son matériel est un `MeshPhongMaterial`, Elle est centrée en (0, 0, 0).

Au départ, nous avions l'ambition d'apposer sur cette piste une texture de glace, pour plus de réalisme. Malheureusement, bien que cela fonctionnait à merveille sur l'ordinateur de l'un de nous, le second était bloqué par le « CORS policy », donc au lieu de nous prendre la tête avec ce contretemps, nous avons ôté la texture.

Figure 3.1.1 – Piste finale (avec maison)

II - Construction de la maison

La maison est construite avec trois anneaux, tracés avec la classe `RingGeometry`, le premier, bleu, le second, blanc et le troisième, rouge. Le centre de la maison, lui, est construit avec un `CircleGeometry`, blanc.

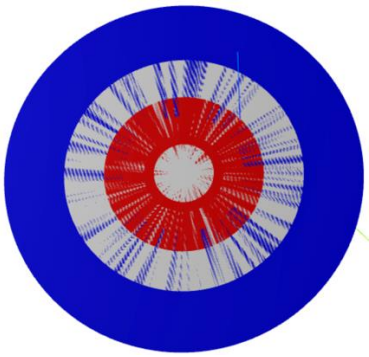

Figure 3.2.1 - Première tentative de la maison

Figure 3.2.2 - Maison finale

Comme vous pouvez le constater dans la Figure 3.2.1, nous avons tout d'abord eu quelques problèmes avec la texture de la maison. Nous avions en effet de prime abord choisi

de la coder en utilisant uniquement des `CircleGeometry`, c'est pourquoi les matériaux, se superposant, créaient des interactions entre eux, sans que nous ne sachions gérer ce genre de problème.
Nous avons alors modifié la méthode utilisée, nous penchant davantage sur des `RingGeometry` (voir Figure 3.2.2). Nous avons alors réuni tous ces anneaux et le cercle dans un unique groupe, ce qui nous a permis de manipuler l'ensemble bien plus facilement. Ainsi notre maison fut créée.

III - Construction des balais

Afin de créer les balais nécessaires au curling, nous avons décidé de coder chaque élément un par un. En effet, le manche, les poils, ainsi que le porte-poils de la brosse ont vu le jour séparément. C'est par la suite que nous les avons regroupés dans un seul et même objet : le balai. Il nous a fallu pour cela ajuster les emplacements de chaque élément vis-à-vis des autres.

Le manche, ainsi que les poils, sont créés par un `CylinderGeometry` ; la brosse par un `BoxGeometry`. Encore une fois, le choix des matériaux est un `MeshPhongMaterial`, car il offre plus de paramètres qu'un `MeshBasicMaterial`.

Nous avons par la suite tout rassemblé dans un unique groupe, afin de créer un unique objet, et ceci grâce à la méthode `groupe.add(élément)`.

Tout comme pour la piste, l'idée de départ était de donner au manche un aspect plus réaliste en passant par une texture, mais comme vous vous en doutez, un même problème mène au même résultat, c'est-à-dire un abandon de cette idée.

Remarque : l'utilisateur peut choisir le nombre de poils figurant sur son balai, de 10 à 1 000, bien que cela ne change strictement rien à la partie.

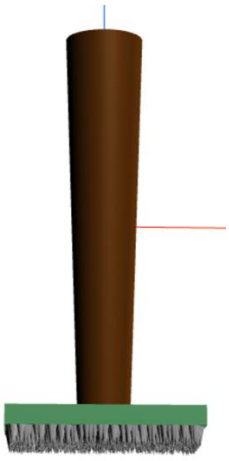

Figure 3.3.1 – Balai final

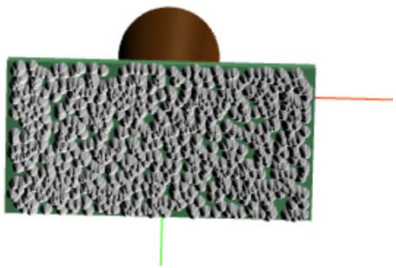

Figure 3.3.2 – Répartition des poils

IV - Construction des palets

Tout comme pour les balais et la maison, chaque élément fut codé à part, mais quelques divergences fondamentales existent entre ces groupes. En effet, la construction des palets fut soumise à de nombreuses consignes :

a) Utilisation de lathes et d'au moins trois surfaces de révolution lisses raccordées par un raccord G1 ;
b) La couleur de la pierre de chaque équipe est différente et la lathe intermédiaire arbore une couleur différente de celles qui l'entourent.

Nous avons décidé de laisser la liberté au(x) joueur(s) de choisir la couleur du palet (retrouver comment procéder dans la section « Comment jouer »). Ainsi, chaque équipe pourra jouir du palet de la couleur qu'il souhaite.

Passons à leur construction.

Le bas et le haut de la pierre sont construits avec deux `CircleGeometry`, placés en (0, 0, 0) quant au groupe. L'anse du palet est créée par deux `CylinderGeometry`, positionnés l'un par rapport à l'autre dans l'esthétique de créer une belle anse. Les surfaces de révolution sont créées à partir d'une courbe de Bézier, réalisant une jointure G1 lors de leur raccord, c'est-à-dire que la lathe du dessus est rattachée à celle du milieu avec cette jointure, et que celle du milieu est reliée de la même manière à la lathe du bas, lathes font par la suite une rotation complète. Les matériaux utilisés sont, naturellement, des `LatheGeometry` (prenant notamment en paramètres les points des courbes de Bézier et la fameuse rotation).

Expliquons ces jointures.

Comme nous le savons, pour qu'une jointure G1 soit réalisée, il faut que :
- le dernier point de la première courbe soit égal au premier point de la seconde ;
- ces même deux points sont alignés avec l'avant-dernier point de la première courbe.

Ainsi, pour créer ces palets, nous avons utilisé trois lathes différentes (dont la couleur de celle du milieu diffère). Elles sont raccordées entre elles comme suit :

- Premier raccord : les vecteurs G1 et A1 sont les mêmes, et ils sont alignés avec A2. G1 est le dernier vecteur de la première courbe, et A1 le premier vecteur de la seconde courbe. Nous avons décidé de la position de tous les points sur une feuille de papier.
- Second raccord : Il en est de même pour le raccord entre les deux autres lathes (celle du milieu et celle du bas), c'est-à-dire que le dernier vecteur de la courbe du milieu, G2, sert de point de jointure (et de point de départ de la courbe). Les points nécessaires furent eux aussi alignés sur une feuille.
  La dernière lathe étant exactement la même que la première, nous avons « retournée » cette dernière (qui est donc la première lathe).

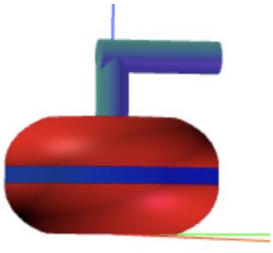

Figure 3.4.1 – Palet final

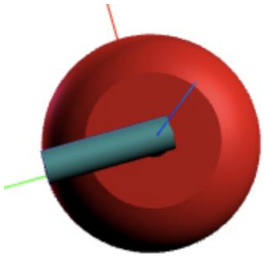

Figure 3.4.2 – Palet vu de haut

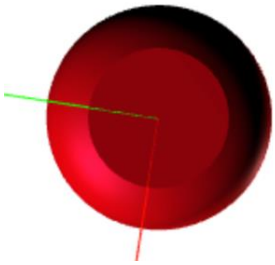

Figure 3.4.3 – Palet vu du bas

4. Construction et mise en place de la scène

La création de notre scène ne fut guère différente de celles utilisées tout au long du semestre, et ce rapport n'ayant lieu d'être une explication du code, nous serons brefs.

Tous les objets précédemment créés furent ajoutés dans une même scène, nommée « scene » (avec la méthode `scene.add(éléments)`). Ainsi, elle s'est vu être peuplé par la piste, les balais, la maison, et les palets (quand l'utilisateur les ajoute dedans). Il nous a simplement fallu choisir les tailles adaptées, telles que 0.1 pour les palets (bien que vous puissiez les modifier dans les menus GUI (voir la section « Menus GUI »)).

Nous avons ajouté les ombres des balais et des palets dans le fichier « objet.js » (oui, nous adorons le réalisme), à la toute fin du code (un commentaire vous indiquera l'emplacement de ces lignes). De plus, vous pouvez observer les différentes lumières constituant la scène dans le fichier « demarrage.js ».

Le reste se situe dans le fichier « CameraLumiere.js », mais vous devez certainement déjà bien le connaître celui-ci.

5. Animations

Les animations ont constitué un point particulièrement retors dans la création de ce projet.

I – Les déplacements

a) Les déplacements rectilignes

Les déplacements en rectilignes ne nous ont pas posé énormément de problèmes. Bien que ce rapport ne soit pas lieu d'être une explication du code, nous allons tout de même rédiger quelques lignes à ce sujet.

En récupérant la position de la pierre au départ, et en fonction de son point d'arrivée (doublée des différentes forces au choix de l'utilisateur), la distance de déplacement est divisée en une centaine de points. Le rendu d'animation est appelé jusqu'au dernier de ces points.
Notez que, toujours dans cet élan de réalisme qui nous tient à cœur, la vitesse de déplacement des palets est progressivement réduite.

b) Les déplacements selon des courbes de Béziers et leurs points de contrôle

Fini de rigoler, passons à Bézier.

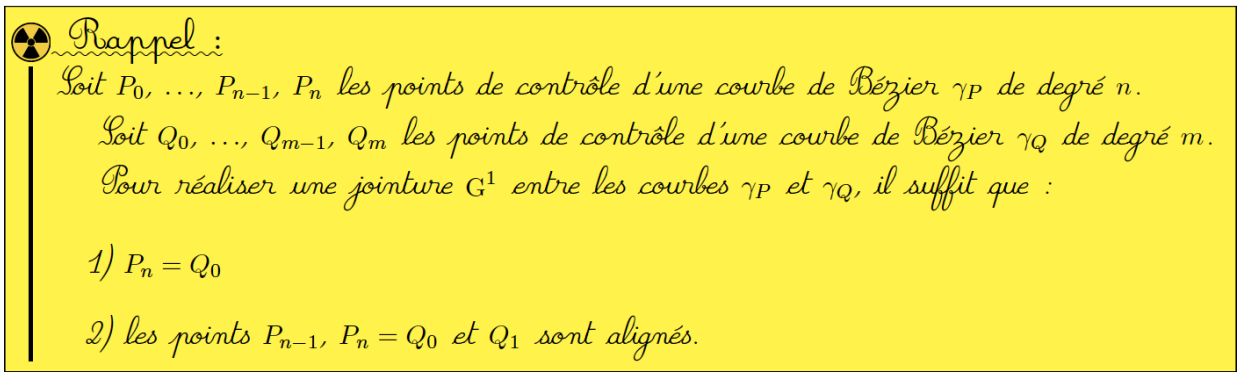

Figure 5.1.2 – Rappel concernant les jointures G1

En adéquation avec les consignes ci-dessus, voici ce que nous avons fait :

- Le dernier vecteur première courbe (sobrement nommée curve1), a comme points `(milieux, milieuy)`. Comme Pn = Q0, le premier vecteur de la courbe suivante (curve2) possède les mêmes points que le vecteur dont nous avons parlé juste au-dessus, à savoir `(milieux, milieuy)`.
- Bien qu'il soit difficile de l'expliquer, nous avons alignés trois points sur une feuille, puis nous les avons retranscrits dans le code (exactement comme pour la création de la pierre (voir section « Construction des palets »), donc nous sommes navrés de ne pas pourvoir plus vous en dire.

Ayant déjà réalisé des prouesses avec la première courbe de Bézier, nous en avons réalisé une seconde sans jointure. Après tout, vous avez la preuve que nous maîtrisons le concept des jointures G1, et cela offre plus de possibilités de tirs, ce qui donne un jeu plus amusant.

II - Les chocs

Nous avons eu un type de chocs à gérer : les chocs entre les différentes pierres lors des lancers.
Pour se faire, nous avons créé une fonction détectant les chocs, nommée « `chocDetected` », et une autres les animant, « `chocanime` ».

`chocDetected` : Un choc est détecté dès que le rayon de la première pierre, ajouté au rayon d'une seconde pierre (avec laquelle la première va rentrer en collision) est plus petit ou égal à la distance entre ces deux pierres.
`chocanime` : Les chocs sont animés dans cette fonction. Je vous laisse y jeter un œil.
Remarque :

Nous avons fait le choix de ne pas implémenter les collisions entre les balais et les pierres, car cela est interdit dans les règles du jeu, donc bien qu'il puisse vous sembler étrange de voir un balai traverser une pierre, sachez que c'est un choix que nous assumons.

6. Menus GUI

Nous avons pris de nombreuses libertés quant à l'implémentation du menu de notre projet, afin de permettre une totale personnalisation des parties à chaque utilisateur.

Le menu gérant la caméra est assez simple, mais il n'en reste pas moins complet. Il permet de modifier le point de vue de la caméra à tout moment, excepté lors d'un lancer (voir Figure 6.1).

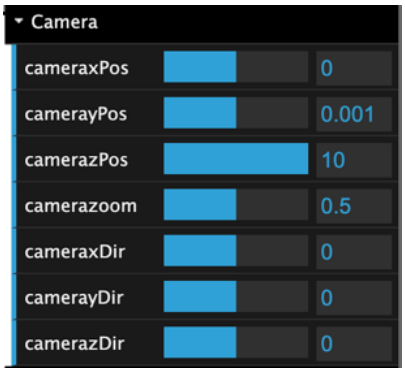

Figure 6.1 – Menu « Caméra »

Abordons à présent les menus maniant les différents objets.

Le menu régissant la piste vous donne l'occasion de jouer sur une piste de curling assez différente des autres : vous pouvez en modifier la longueur et la largeur à souhait, et, bien entendu, la couleur de la piste si celle par défaut ne vous plaît guère. À vous de voir si vous préférez jouer sur une très longue piste ou non (voir Figure 6.2).

Le menu nommé « Pierre » vous permet de gérer la taille de vos palets, la couleur de l'anse, de la lathe centrale, et des surfaces environnantes (voir Figure 6.3).

Le menu gui nommé « balais » permet de modifier la taille des composantes des balais, que ce soit la largeur du manche ; sa longueur ; sa hauteur ; le nombre de poils le constituant, la couleur de tous ses composants, et j'en passe (voir Figure 6.4).

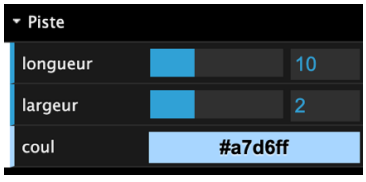

Figure 6.2 – Menu « Piste »

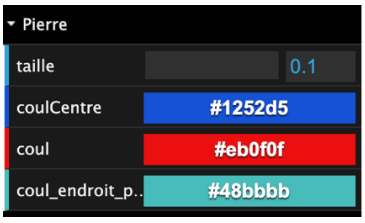

Figure 6.3 – Menu « Pierre »

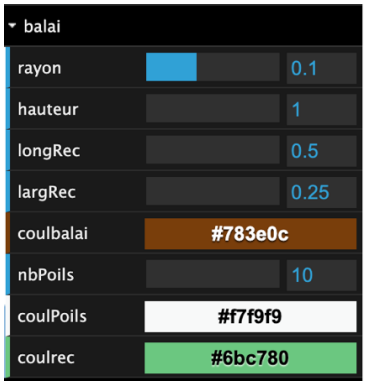

Figure 6.4 – Menu « balai »

Voyons les deux dernières parties de ce long menu, celles contrôlant le jeu.

Dans le menu « Partie », il vous est proposé trancher quelle sera votre équipe, entre la 1 et la 2. Nous avons déjà évoqué la particularité de cette option dans la partie « Comment jouer ». Viennent ensuite les options permettant de commencer ou recommencer la partie (voir Figure 6.5).

Le dernier menu est le plus important de tous, car c'est celui qui vous amènera peut-être la victoire : j'ai nommé le menu « lancer ». Avec lui, gérer les différentes forces de vos lancers, et choisissez leur trajectoire entre une ligne droite et une courbe de Bézier. Vous avez également la possibilité de changer les points de contrôle des deux courbes. Cliquez sur l'option « lancer » pour que les palets se mettent à glisser (voir Figure 6.6).

Figure 6.5 – Menu « Partie »

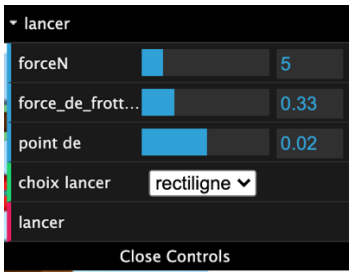

Figure 6.6 – Menu « lancer »

7. Arbre C.S.G.

L'arbre C.S.G. avec explications :

Le C.S.G. (pour Constructive Solid Geometry) est une technique de modélisation utilisant des opérations booléennes sur les solides. Il nous permet alors d'unir, de soustraire, d'intersectionner deux solides entre eux, donnant par la suite de nouvelles formes.

- Subtract : Cette méthode permet de soustraire à un objet A un objet B. Pour appliquer cette méthode, il nous faut faire objetA.subtract(objetB).
- Union : Cette méthode fait l'inverse de la fonction subtract, puisqu'elle unie, fusionne deux objets afin de créer un objet C. La méthode relative s'écrit objetA.union(objetB).
- Intersect : Cette méthode-ci permet de conserver uniquement ce qui résulte de l'intersection de deux géométries. Sa méthode relative s'écrit objetA.intersect(objetB).

Cette technique a l'avantage de créer des formes aux bords parfaits (à la différence des techniques à base de polygones par exemple). De plus, elle permet d'accélérer certains calculs, comme les collisions, ou la projection d'ombres.

Il ne nous est plus demandé dans le sujet de l'utiliser, mais il nous faut en parler.
Un arbre C.S.G. (CsgTree) est une sorte de diagramme permettant une visualisation simple des opérations utilisées pour créer un objet final. En recherchant sur Inter des informations concernant cet arbre, nous avons trouvé ce qui suit (voir Figure 7.1).

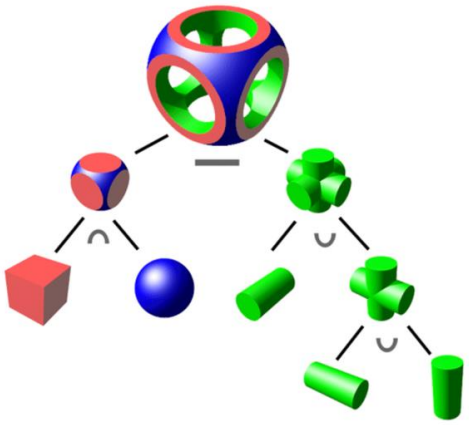

Figure 7.1 – Exemple d'un arbre CSG courageusement pris sur Internet

Explication : le cube rouge et la sphère bleue sont inter-sectionnés (objet A) ; les cylindres verts sont unis les uns aux autres, selon différents angles (objet B). Le résultat final est la soustraction de l'objet A et de l'objet B.

Imaginons-en un créant un palet :

Manche …… (création des deux cylindres constituant l'anse)
CSGCylinder1
CSGCylinder2
Manche = CSGCylinder1.union(CSGCylinder2)

Pierre …… (création d'une sphère et d'un cube)
CSGSphere
CSGBox
Pierre = CSGSphere.intersection(CSGBox)

Palet = Manche.union(Pierre) …… (union des deux éléments, ou plutôt on les assemble)

Nous nous doutons que ce procédé est bien plus complexe que ça, mais n'ayant jamais eu de travaux dirigés sur ce sujet, nous ne pouvons que vous fournir ce que vous venez de voir.

8. Conclusion

Ce projet nous a demandé beaucoup de réflexion, une grande autonomie, ainsi qu'un bon travail d'équipe. Nous sommes fiers d'être parvenu au bout de ce que nous voulions vous présenter, bien qu'il ne soit pas parfait. Nous avons certes quelques regrets (comme la texture de la piste ou des balais), mais ils sont moindres lorsque nous visualisons le résultat final de nos nombreuses semaines de travail. Les tracas furent nombreux, mais surmontables, ce qui nous a permis de grandement comprendre les notions apportées par le module d'Info3B.

Nous espérons qu'il saura vous divertir, et surtout vous rendre fiers de vos élèves.
À présent, place au jeu !

9. Bibliographie

Bibliothèque Three.js

RingGeometry :

https://threejs.org/docs/index.html?q=ring#api/en/geometries/RingGeometry

CircleGeometry :

https://threejs.org/docs/index.html?q=circ#api/en/geometries/CircleGeometry

BoxGeometry :

https://threejs.org/docs/#api/en/geometries/BoxGeometry

CubicBezierCurve :

https://threejs.org/docs/?q=bezier#api/en/extras/curves/CubicBezierCurve

QuadraticBezierCurve :

https://threejs.org/docs/?q=bezier#api/en/extras/curves/QuadraticBezierCurve

CylinderGeometry :

https://threejs.org/docs/?q=cyli#api/en/geometries/CylinderGeometry

Autres liens

https://threejsfundamentals.org/threejs/lessons/threejs-primitives.html

https://www.researchgate.net/figure/Exemple-darbre-CSG-Lobjet-solide-final-est-en-fait-le-resultat-dune-soustraction_fig20_321664988