



Master bioinformatique

Génie Logiciel

**Semaine 1 - Travaux Pratiques****TP1**

## 1 Utilisation de Three.js

Three.js est une bibliothèque graphique basée sur WebGL. Elle facilite grandement le rendu 3D dans un navigateur car il s'agit d'une bibliothèque haut-niveau. Par exemple, les shaders sont gérés automatiquement dans la bibliothèque. Il ne reste donc qu'une interface en Javascript.

Pour l'utilisation de la bibliothèque, incluez dans le fichier HTML le fichier Javascript à l'adresse <https://raw.githubusercontent.com/mrdoob/three.js/master/build/three.min.js>

Une documentation très bien fournie est accessible à l'adresse : <https://threejs.org/docs/>.

### 1.1 Un cube avec Three.js

Vous aurez besoin d'un fichier html contenant uniquement les balises head et body et qui inclue les deux fichiers js. Dans votre fichier javascript, commencez par instancier quelques objets correspondant aux différentes grandes entités nécessaires au rendu 3D (caméra, scène, objet) :

1. Une `THREE.Scene`
2. Une `THREE.PerspectiveCamera` (cf. documentation pour les arguments)
3. Et enfin un `THREE.WebGLRenderer` qui correspond au contexte WebGL.

Après avoir créé ces trois objets, il reste à effectuer le rendu :

1. On ajoute le canvas dans la page HTML, c'est-à-dire l'élément HTML permettant de faire le rendu, avec `document.body.appendChild( renderer.domElement );`
2. Puis on affiche la scène : `renderer.render( scene, camera );`

A ce stade, le canvas devrait s'afficher en noir.

**Questions** A partir de la documentation :

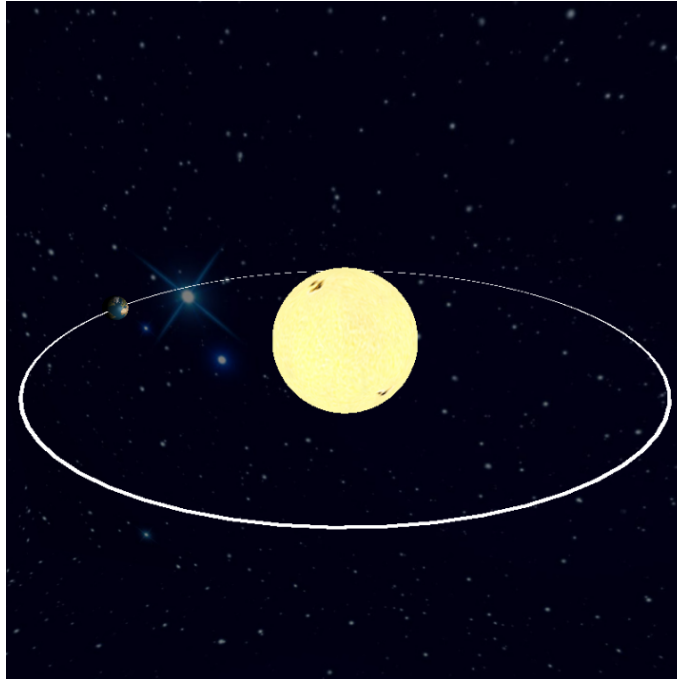
1. Modifier la taille du canvas
2. Modifier la couleur d'effacement du canvas (chercher dans la documentation). Indice : il est nécessaire d'utiliser deux fonctions : une pour spécifier la couleur d'effacement et l'autre pour effacer.
3. Faites tourner le cube à l'aide de l'attribut rotation de `THREE.Mesh`.
4. Créer une animation qui fasse tourner le cube selon les axes x et y. Indice : utiliser la fonction `requestAnimationFrame`, qui permet d'appeler une autre fonction en arrière-plan.

**Création du cube** Un objet en Three.js est défini par un maillage, ou mesh, (ensemble de triangles formant une surface) qui correspond à une géométrie (position des sommets) et un matériau (la couleur, la texture).

1. Créer la géométrie avec `THREE.BoxGeometry(largeur, hauteur, profondeur)`
2. Créer le matériau avec `THREE.MeshBasicMaterial(couleur)`
3. Enfin, réunissez géométrie et matériau dans le maillage avec `THREE.Mesh(geometrie, materiau)`
4. Ajouter le maillage à la scène avec `scene.add(maillage)`

## 2 Système solaire

Le but de cette partie est de réaliser un système solaire simple composé d'un soleil, au centre, autour duquel les planètes gravitent.



## 2.1 Représentation des corps célestes

Les planètes et soleil peuvent être représentés par des sphères texturées. Le but est pour l'instant de créer le soleil et la Terre. Vous pouvez récupérer les images de texture correspondantes à l'adresse suivante : <http://planetpixlemporium.com/planets.html>.

Les différentes étapes à la création de ces deux objets sont les suivantes (référez-vous à la documentation pour l'utilisation de chacune des fonctions) :

1. On crée la géométrie avec une `SphereGeometry`.
  2. On récupère la texture avec une instance de classe `TextureLoader`. Quelle méthode de cette classe permet de spécifier le chemin vers l'image de texture à charger ?
  3. Appliquer la texture sur le matériau en changeant l'attribut `map` du matériau.
  4. Fusionner géométrie et matériau pour créer un maillage.
- Faites-en sorte que le soleil soit 10 fois plus grand que la Terre, et que la Terre soit suffisamment éloignée du soleil (par exemple, fixer sa distance à 10 fois son rayon).
  - Incliner légèrement la Terre : son axe de rotation n'est pas exactement l'axe des ordonnées, mais est légèrement incliné vers l'axe des abscisses.

## 2.2 Mouvement

La Terre tourne à la fois sur elle-même (cette rotation définit les cycles jour-nuit) et autour du Soleil (cela définit le cycle annuel).

1. Créer une animation qui permette de faire tourner la Terre sur elle-même.
2. Créer l'animation qui permette de faire tourner la Terre autour du soleil. Du point de vue de la définition mathématique de la rotation, quelle est la différence avec la question précédente ? **Indice** : On peut associer un objet à un autre sur le plan d'une relation parent/enfant avec `mesh.add(otherMesh)`.
3. Afficher l'orbite de la Terre. La géométrie de l'orbite peut être modélisée à l'aide d'une `RingGeometry`.

## 2.3 Lumière

Pour l'instant, la Terre est éclairée uniformément. Il est donc impossible de voir quelle partie de la Terre est plongée dans la nuit. Nous allons ajouter la source de lumière dans le soleil.

1. Ajouter une source lumineuse à la scène à l'aide d'une instance de `PointLight`. Placer cette source lumineuse dans le soleil.

2. Le `MeshBasicMaterial` définit un matériau qui n'est pas impacté par les sources lumineuses. Utiliser `MeshPhongMaterial` qui permet d'utiliser l'ombrage de Phong.

## 2.4 Pour aller plus loin

1. Réaliser les déplacements dans la scène (translation, rotation, zoom), à l'aide de `OrbitControls.js`. Cette bibliothèque permet, en une ligne de code, de gérer les déplacements de la caméra à l'aide des déplacements classiques de la souris (clic, molette, etc). Des exemples d'utilisation de la bibliothèque sont disponibles sur la documentation de Three.js.
2. Ajouter la visualisation des galaxies lointaines en utilisant un cube de texture que vous transmettez à la scène (attribut `background` de `scene`). Les images de texture des galaxies peuvent être trouvées sur Internet.
3. Modéliser le déplacement d'une comète selon un chemin que vous définissez. Vous pourrez, pour cela, utiliser la classe `Path`.
4. Utiliser les `Sprite` pour ajouter des particules autour du soleil afin simuler les effets de jets de plasma.