WebGL

OBJECTIF

Dernier de la série, ce TP sera consacré à la 3D en JavaScript avec l'utilisation de Three.js, une bibliothèque simplifiant grandement l'intégration et la manipulation d'une scène 3D au sein d'une page Web.

QUESTION 1

Nous allons créer une classe qui nous permettra de créer et afficher notre scène WebGL.

- Créez un fichier 'scene.js' dans lequel vous déclarerez une classe 'Scene'.

QUESTION 2

Préparons notre classe pour accueillir tout ce qui sera nécessaire à notre scène 3D.

Ajoutez les propriétés suivantes qui seront initialisées à null :

- rendu3D, générera le rendu de la scène
- scene3D, contiendra les éléments de la scène
- camera3D, déterminera le point de vue de la scène
- cube3D, un cube dans la scène

QUESTION 3

Nous allons à présent entrer dans le vif du sujet en initialisant l'environnement 3D. Pour cela, nous allons faire appel à une bibliothèque très utile : THREE.js. Celle-ci est déjà intégrée au projet, vous n'avez plus qu'à l'utiliser.

- Ajoutez une fonction 'initialiser' à la classe 'Scene' qui initialisera le rendu3D et créera une surface de dessin dans la page.

  Pour initialiser le générateur de rendu :

```
this.rendu3D = new THREE.WebGlRenderer();
```

QUESTION 4

- Pour afficher l'environnement 3D précédemment créé dans la page HTML, complétez la fonction précédente avec ceci :

```
$('#content').append(this.rendu3D.domElement);
```

- Dans la fonction 'main' du fichier 'main.js', créez une instance de la classe 'Scene' et appelez sa fonction 'initialiser'.

- Testez.

La taille par défaut du canvas créé n'est pas forcément adaptée à votre écran.

- Dans la fonction 'initialiser', appelez la fonction setSize du générateur de rendu en lui donnant en paramètre la largeur et la hauteur souhaitées.
- Testez.

QUESTION 5

- Ajoutez la ligne suivante à la fonction 'initialiser' de la classe 'Scene'. Cela aura pour effet de créer la structure qui contiendra les objets de notre scène.

```
this.scene3D = new THREE.Scene();
```

QUESTION 6

A présent nous allons créer une caméra qui représentera notre point de vue dans la scène.

- Modifiez la fonction 'initialiser' de la classe 'Scene' comme suit :

```
this.camera3D = new THREE.PerspectiveCamera(45, largeur / hauteur, 1, 1000);
```

  - 45 représente l'angle de vue de la camera
  - largeur / hauteur est le ratio entre la largeur et la hauteur de la surface d'affichage. Vous indiquerez ici les valeurs utilisées pour la dimension du canvas créé.
  - Le paramètre suivant indique la distance d'affichage la plus proche
  - Et le dernier paramètre indique la distance d'affichage la plus éloignée

- Puis positionnez la caméra aux coordonnées (0, 0, 500) :

```
this.camera3D.position.set(x, y, z);
```

- Testez. C'est tout noir ? C'est normal. On y est presque.

QUESTION 7

Nous avons à présent un environnement 3D, une scène qui contiendra nos objets, une caméra. Il ne manque plus que le contenu de la scène. Commençons par le plus simple : le cube !

Avec la bibliothèque THREE.js, la création d'un objet se fait en trois étapes :

- Tout d'abord on crée la géométrie, la forme de l'objet.

```
var geometrie = new THREE.BoxGeometry(largeur, hauteur, profondeur);
```

- Ensuite la matière dans laquelle l'objet est conçu (la couleur, la texture, les effets de lumières, ...)

```
var matiere = new THREE.MeshBasicMaterial({ color : 0x00ff00 }); //Vert
```

- Enfin, on combine la géométrie et la matière pour obtenir notre objet

```
var cube = new THREE.Mesh(geometrie, matiere);
```

Il ne reste alors plus qu'à ajouter l'objet créé à la scène :

```
this.scene3D.add(cube);
```

- Ajoutez à la classe 'Scene' une fonction 'creerCube' qui crée un cube comme vu précédemment et initialise la propriété 'cube3D' de la classe avec l'objet créé.
- Ajoutez à la classe 'Scene' une fonction 'construireScene' qui appelle la fonction précédente.
- Appelez cette fonction depuis la fonction 'main' du fichier 'main.js'.
- Testez. Toujours rien ? Patience...

QUESTION 8

Il faut maintenant réaliser le rendu de notre scène 3D. Pour cela :

- Ajoutez une fonction 'animer' à la classe 'Scene'.

  Pour réaliser le rendu de la scène, appelez la fonction suivante :

```
this.rendu3D.render(this.scene3D, this.camera3D);
```

- Appelez la fonction 'animer' dans la fonction 'main' du fichier 'main.js'.
- Testez. Oh ! Un carré vert !

QUESTION 9

Bien. Passons aux choses sérieuses à présent. Un peu de lumière, ce serait pas mal.

- Ajoutez une fonction 'creerLumiereAmbiance' dans la classe 'Scene'.
- Dans cette fonction, créez une lumière et ajoutez-la à la scène :

```
var lumiere = new THREE.AmbientLight(0xAAAAAA);
```

  Le paramètre de la fonction 'AmbientLight' permet de définir la couleur et la vivacité de la lumière.

- Appelez la fonction 'creerLumiereAmbiance' depuis la fonction 'construireScene'.
- Testez.

Rien ? C'est normal, la matière utilisée pour le cube ne gère pas la lumière.

- Modifiez la matière du cube en utilisant celle-ci :

```
var matiere = new THREE.Mesh Phong Material({ color : 0x00ff00 }); //Vert
```

- Testez.

QUESTION 10

Bon, depuis le début du TP on parle de 3D et pour le moment, on ne voit qu'un carré... Faisons tourner légèrement notre cube.

- Dans la fonction 'creerCube', après la création du cube, ajoutez la ligne suivante :

```
cube.rotation.y = 0.5;
```

- Testez.

QUESTION 11

Ok, je sais ce que vous allez dire : 'Paie ta 3D, c'est tout gris !'. Dans ce cas, ajoutons quelques ombrages.

- Ajoutez une fonction 'creerLumiereDirectionnelle' à la classe 'Scene'. Cette fonction prendra un paramètre qui représentera l'objet qui sera éclairé par la lumière.

```
creerLumiereDirectionnelle(cible)
{
	//Création de la lumière directionnelle
	var lumiere = new THREE.DirectionalLight(0xffffff, 1);

	//Positionnement de la lumière
	lumiere.position.set(200, 200, 200);

	//Orientation de la lumière vers un objet de la scène
	lumiere.target = cible;
}
```

- Dans la fonction 'construireScene', appelez la fonction précédente en lui fournissant le cube comme paramètre.
- Testez.

QUESTION 12

- Ajoutez une fonction 'creerSol' qui créera un sol sous le cube.
- Ajoutez à l'objet sol créé la propriété suivante (le sol recevra des ombres portées) :

```
sol.receiveShadow = true;
```

- Modifiez la fonction 'creerCube' pour que le cube créé reçoive la propriété suivante (le cube générera une ombre portée) :

```
cube.castShadow = true;
```

- Modifiez la fonction 'créerLumiereDirectionnelle' pour que la lumière créée génère des ombres portées :

```
lumiere.castShadow = true;
```

- Enfin, modifiez la fonction 'initialiser' pour que l'environnement 3D prenne en compte les ombres portées :

```
this.rendu3D.shadowMapEnabled = true;
```

- Testez.

QUESTION 13

Et si nous faisions bouger tout cela à présent ? Pour ce faire, nous allons modifier la fonction 'animer' pour que celle-ci tourne en boucle, mais sans figer la page web. JavaScript fournit une fonction qui permet d'appeler notre méthode 'animer' quand WebGL est prêt à réaliser un nouveau rendu.

- Ajoutez les lignes suivantes à la fonction 'animer' de la classe 'Scene' :

```
var that = this;
requestAnimationFrame(function() { that.animer(); });
```

- Testez.

Si, si, ça tourne en boucle, je vous assure. Rendons la chose plus visible :

- Toujours dans la fonction 'animer', ajoutez cette ligne :

```
this.cube3D.rotation.y += 0.01;
```

  Ceci aura pour effet de faire tourner le cube sur lui-même de 0.01 radian par affichage.

- Testez.

QUESTION 14

Notre cube tourne sur lui-même. Mais comment faire pour qu'il tourne autour d'un axe extérieur ? Il va falloir créer un pivot.

- Ajoutez une propriété 'pivot3D' à la classe 'Scene', dans le constructeur.
- Créez une fonction 'creerPivot' qui créera un objet 3D invisible :

```
this.pivot3D = new THREE.Object3D();
```

- Ajoutez le cube à ce pivot :

```
this.pivot3D.add(this.cube);
```

- Ajoutez le pivot à la scène :

```
this.scene3d.add(this.pivot3D);
```

- Dans la fonction 'creerCube', déplacez le cube sur l'axe des x. Par exemple :

```
this.cube3D.position.x = 50;
```

- Testez.

Mise à part la position du cube, pas beaucoup de changement, n'est-ce pas ?

- Dans la fonction 'animer' de la classe 'Scene', faites tourner le pivot précédemment créé, de la même panière que pour le cube.
- Testez avec des rotations différentes.

QUESTION 15

- Transformez le cube en sphère en utilisant la géométrie 'SphereGeometry'.
- Transformez la teinte unie de la sphère en une texture :

```
//Chargement de la texture
texture = THREE.ImageUtils.loadTexture('img/texture.jpg');

var matiere = new THREE.MeshPhongMaterial({ map : texture });
```

- Eclatez-vous ! Il y a plein de choses à essayer !