TP2 - THREE.JS

Ouvrez votre TP précédent. Lors des changements, mettez en commentaire les instructions précédentes au lieu de les effacer.
L'objectif maintenant est d'ajouter des éléments au carrousel. La modélisation par primitives n'étant pas toujours évidente, nous allons ajouter un modèle à la scène en utilisant une librairie d'importation d'objets OBJ. Pour importer des objets 3D, il est nécessaire d'ajouter des « loaders » pour chaque format.

```
<script src="./libs/MTLLoader.js"></script>
<script src="./libs/OBJLoader.js"></script>
```

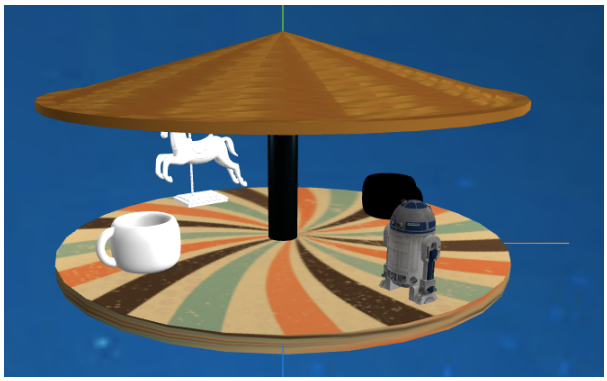
Le code suivant vous permet d'ajouter deux tasses au carrousel. Modifiez les transformations pour les adapter au carrousel que vous avez modélisé :

```
// Cup
 var mtlLoader = new THREE.MTLLoader();
 mtlLoader.setResourcePath('./models/Cups/');
 mtlLoader.setPath('./models/Cups/');
 mtlLoader.load('cup2.mtl', function (materials) {
    materials.preload();
    objLoader = new THREE.OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.setPath('./models/Cups/');
    objLoader.load('cup.obj', function (object) {
       object.scale.set(0.1,0.1,0.1);
       object.position.set(10,2,0);
       whiteCup = object;
       carousel.add(whiteCup);
    });
 });

mtlLoader.load('cup.mtl', function (materials) {
    materials.preload();
    objLoader = new THREE.OBJLoader();
    objLoader.setMaterials(materials);
    objLoader.setPath('./models/Cups/');
    objLoader.load('cup.obj', function (object) {
       blackCup = object;
       object.scale.set(0.1,0.1,0.1);
       object.position.set(-10,2,0);
       carousel.add(blackCup);
    },
     // called when loading is in progresses
    function ( xhr ) {
         console.log( ( xhr.loaded / xhr.total * 100 ) + '% loaded' );
     },
     // called when loading has errors
    function ( error ) {
          console.log( 'An error happened' );
     }
)});
```

Les deux dernières fonctions ("//called when loading…") ne sont pas obligatoires mais vous permettent de suivre dans la console du navigateur le chargement de votre objet.
Testez votre script. Ajoutez une capture d'écran à votre compte rendu.

Animez les tasses pour qu'elles tournent autour d'elles-mêmes, et ajoutez ensuite deux autres objets de votre choix (ceux proposés ou des objets de votre choix : https://free3d.com/3d-models/). Copiez le code de votre fonction d'animation dans le compte rendu, ainsi que les liens vers les modèles utilisés si nécessaire.



LES LUMIERES

Il est possible dans Three.js d'utiliser des lumières « correctes » du point de vue physique et ajouter donc du réalisme à la scène. En effet, la lumière ambiante permet de récréer l'éclairage indirect de l'environnement mais elle éclaire de la même manière dans toutes les directions. Il est donc indispensable d'ajouter plusieurs lumières directes mais cela implique notamment une baisse de performance de l'application.
Nous allons donc changer de stratégie ! Pour cela, après la création du renderer, ajoutez la ligne :

```
renderer.physicallyCorrectLights = true;
```

Maintenant rendez vous dans la fonction CreateLights. Three.js propose un autre type de lumière ambiante, HemisphereLight. Ce type de lumière agit comme dans la réalité, ou la lumière s'estompe du haut en bas, le sol étant toujours plus sombre que le plafond ou que le ciel. HemisphereLight définit donc une lumière plus lumineuse en hauteur et une plus pâle en bas. La lumière va s'estomper entre les deux couleurs. Remplacez votre lumière ambiante avec ce nouvel objet.

```
const ambientLight = new THREE.HemisphereLight(
0xddeeff, // sky color
0x202020, // ground color
3, // intensity
);
scene.add( ambientLight);
```

Même s'il est possible de conserver que cette lumière, le rendu sera encore « plat » car elle ne produit pas de reflets. J'ai donc conservé la lumière directionnelle pour que la colonne du manège affiche son aspect métallisé (pas visible dans l'image précédente) :

```
var columnMaterial = new THREE.MeshStandardMaterial({color: 0x3287a8, roughness: 0.3, metalness: 1});
```

Testez votre script. Ajoutez une capture d'écran à votre compte rendu.

LES OMBRES

Les ombres permettent d'augmenter le réalisme d'une scène en donnant des indices sur le placement relatif des objets, leur géométrie, etc. Plusieurs étapes sont nécessaires pour activer les ombres. Il faut commencer par activer les ombres dans le renderer, après sa création :

```
renderer.shadowMap.enabled = true;
renderer.shadowMap.type = THREE.PCFSoftShadowMap;
```

Puis, dans votre fonction createLights, activez les ombres pour la lumière directionnelle et configurez les paramètres :

```
light.castShadow = true;

//Set up shadow properties for the light
light.shadow.mapSize.width = 512;  // default
light.shadow.mapSize.height = 512; // default
light.shadow.camera.near = 0.5;    // default
light.shadow.camera.far = 500;  // default
```

Maintenant, il faut activer le calcul des ombres pour chaque objet en décidant s'il peut en recevoir et/ou en générer, juste après la création de l'objet :

```
ground.castShadow = false;  // l'objet ne génère pas d'ombre
ground.receiveShadow = true;  // l'objet peut recevoir les ombres
```

Pour les modèles importés, il est nécessaire d'activer les paramètres pour tous ses maillages, avant d'ajouter l'objet à la scène :

```
object.traverse( function( node ) { if ( node instanceof THREE.Mesh ) { node.castShadow = true; } } );
```

Testez votre script. Ajoutez une capture d'écran à votre compte rendu.

LA SKYBOX

Pour améliorer l'environnement global, qui semble vide, il est possible de créer une skybox en utilisant un cube, vu de l'intérieur. Pour cela, nous allons utiliser six images, combinées dans un MeshFaceMaterial. Modifiez votre fonction createLights en désactivant l'ancienne skybox et en ajoutant la nouvelle version (faites correspondre le code à la skybox choisie sur celles disponibles dans le TP ou ici http://www.humus.name/index.php?page=Textures) :

```
var cube = null;
 textureURLs = [  // URLs of the six faces of the cube map
        "./textures/cube/posx.jpg",
        "./textures/cube/negx.jpg",
        "./textures/cube/posy.jpg",
        "./textures/cube/negy.jpg",
        "./textures/cube/posz.jpg",
        "./textures/cube/negz.jpg"
 ];

    // skybox
    var textures = loadTextures(textureURLs);
    var materials = [];
    for (var i = 0; i < 6; i++) {
       materials.push( new THREE.MeshBasicMaterial( {
          color: "white",
         // IMPORTANT: To see the inside of the cube, back faces must be rendered!
          side: THREE.BackSide,
          map: textures[i]
       } ) );
    }
cube = new THREE.Mesh(new THREE.CubeGeometry(100,100,100), materials );
 scene.add(cube);
```

Ajoutez à votre code la function loadTextures, qui permet de changer les fichiers de textures et qui est employée par le code précédent :

```
function loadTextures(textureURLs) {
  var loaded = 0;
  function loadedOne() {
     loaded++;
     if (loaded == textureURLs.length) {
        for (var i = 0; i < textureURLs; i++)
           textures[i].needsUpdate = true;
     }
  }
  var textures = [];
  for (var i = 0; i < textureURLs.length; i++) {
     var tex = new THREE. TextureLoader().load( textureURLs[i], undefined, loadedOne );
     textures.push(tex);
  }
  return textures;
}
```

Testez votre script. Ajoutez une capture d'écran à votre compte rendu en indiquant la skybox choisie.

LE SON

Dans un environnement interactif, le son joue un rôle important, permettant de créer une ambiance et de donner un retour à l'utilisateur. Le code suivant, fonctions initSound et toggle, vous permettent d'ajouter de la musique à votre scène. N'oubliez pas d'appeler la fonction initSound dans le main, après la création de la caméra. La fonction toggle permet de mettre en pause le son lorsque l'utilisateur appuie sur une touche du clavier. Vous pouvez modifier la musique utilisée en utilisant des ressources telles que freesound et playonloop.

```
function initSound() {
   // instantiate a listener
   var audioListener = new THREE.AudioListener();

  // add the listener to the camera
  camera.add( audioListener );

  // instantiate audio object
  ambientSound = new THREE.Audio( audioListener );

 // add the audio object to the scene
 scene.add( ambientSound );

// instantiate a loader
var loader = new THREE.AudioLoader();

// load a resource
loader.load('./sounds/ambient.mp3',
   // Function when resource is loaded
   function ( audioBuffer ) {
      ambientSound.setBuffer( audioBuffer );
      ambientSound.setLoop( true );
      ambientSound.setVolume( 0.5 );
      ambientSound.play();
  });

  document.body.addEventListener('keydown', toggle);
}
```

```
function toggle () {
  if ( ambientSound.isPlaying === true ) {
        ambientSound.pause();
  }
 else  {
        ambientSound.play();
  }
}
```

Déposez votre document et votre code sur Moodle ou envoyez le tout par mail à nancy.rodriguez@lirmm.fr