TP Three.js 5

Licence Informatique 3ème année

Année 2023-2024
durée : 3h

L'objectif de ce cinquième TP est d'effectuer une introduction à l'utilisation des textures en Three.js. Dans un premier temps, récupérez l'archive jointe à cet énoncé et extrayez les dossiers et fichiers qui s'y trouvent dans un sous-dossier nommé TP5 qui se trouvera dans le dossier où vous réalisez vos TP d'informatique graphique. Après extraction vous disposez :

- d'un dossier `js` qui contient les scripts nécessaires à ce tp ;
- d'un dossier `Images` qui contient les exemples d'images à utiliser comme textures ;
- un fichier html pour chacun des exercices à réaliser.

Ce TP devra être rendu à la fin de la séance, sous la forme d'une archive nommée `TPIG5-XXX`, où `XXX` sera remplacé par votre nom de famille. Elle devra contenir le dossier TP5 et les dossiers et fichiers qui y seront contenus et sera à envoyer via un site tel que `wetransfer`.

Pour tous les TPs Three.js, vous pourrez consulter la documentation officielle en ligne, à l'adresse :
`https://threejs.org/docs/index.html`

Exercice 1 - Chargement d'une texture

On rappelle qu'une texture 2D est une image qui peut être appliquée sur la surface d'un objet, de manière à pouvoir y faire figurer de nombreux détails à un coût moindre que celui, par exemple, de modéliser en 3D chacun de ces détails.

Application d'une première texture

`Three.js` dispose d'une classe nommée `TextureLoader` qui dispose des fonctionnalités nécessaires au chargement d'une image et à la création d'une texture. Après création de l'objet corrrespondant, sa méthode `load` permet de charger l'image dont on fournit l'adresse ou l'emplacement, puis de retourner la texture correspondante. L'utilisation la plus simple de cette classe correspond alors au code suivant (en utilisant une image disponible sur un site externe) :

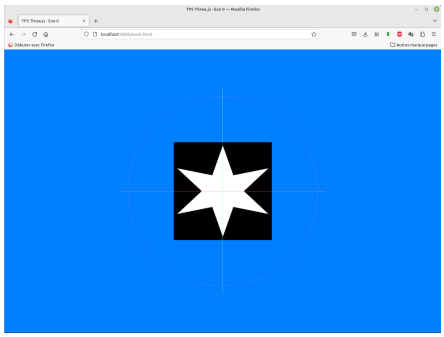
```
// création de "loader"
 const loader = new THREE.TextureLoader();

// chargement d'une image et création de la texture correspondante
 var texture = loader.load('https://cdn.pixabay.com/photo/2012/04/12/19/56/
               six-pointed-star-30398_960_720.png');
```

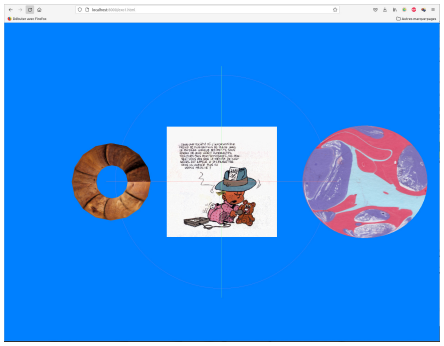
En complément de ces opérations de chargement et de création de texture, les géométries standard fournies dans l'API disposent du code nécessaire pour appliquer la texture sur leur surface. Pour illustrer cet aspect, créez un objet de type `PlaneGeometry` de taille 2.0 à l'origine de votre repère, puis associez lui un matériau de type basique, sans couleur, disponible sur les deux faces, et auquel vous ajouterez un attribut `map` qui contiendra la texture chargée et créée dans l'exemple de code fourni ci-dessus. Vous devez alors obtenir l'image apparaissant dans la figure 1a.

Utilisation d'une image locale

Les navigateurs n'autorisent pas un script à lire des données présentes sur la machine sur laquelle ils s'exécutent. Pour pouvoir néanmoins tester vos développements en local avec `Three.js`, différentes possibilités sont listées sur la page suivante :
`https://threejs.org/docs/manual/en/introduction/How-to-run-things-locally`



(a) Rendu après modélisation du carré et application de la première texture.



(b) Rendu après modélisation des trois objets et application des 3 textures locales.

FIGURE 1 – Rendus à obtenir pour l'exercice 1.

La façon la plus simple est d'utiliser la méthode utilisant un serveur web python (partie `Python server` dans la page précédente) qui est résumée ci-après [1].

1. ouvrir une console (interpréteur de commande) et se déplacer dans le dossier dans lequel se trouvent les fichiers html des exercices du TP 5 ;
2. lancer la commande : `python3 -m http.server` ; elle ouvre alors un server http local sur le port 8000 (conserver la console ouverte jusqu'à la fin du TP) ;
3. dans le navigateur, ouvrir le fichier html souhaité en donnant l'adresse `localhost:8000/`, suivie du nom du fichier html à ouvrir (par exemple `localhost:8000/exo1.html`).

Application

1. modifiez la texture appliquée sur votre carré de manière à utiliser le fichier `bb.png` qui se trouve dans le dossier `Images` récupéré dans l'archive associée au TP ;
2. créez une sphère de rayon 1.0 unité, placée en $x = 2.5$ et qui utilisera le fichier `10.jpg` du dossier `Images` comme texture ;
3. créez un tore de rayon extérieur 0.5 unité, de rayon intérieur 0.2 unité, placé en $x = -2.0$ et qui utilisera le fichier `woodV.jpg` du dossier `Images` comme texture.

On précise que, comme pour la classe `PlaneGeometry`, les classes représentant la géométrie d'une sphère et d'un tore, disponibles dans `Three.js`, intègrent la gestion des textures. Vous pouvez donc utiliser la même approche que pour le carré traité au départ du TP. A l'issue de ces 3 étapes, vous devez obtenir l'image qui apparaît dans la figure 1b.

Exercice 2 - Les coordonnées de texture

Lorsque vous définissez vos propres géométries et que vous souhaitez y appliquer des textures, il est nécessaire de préciser la manière dont l'image doit être appliquée sur cette géométrie. Cette technique, appelée **placage de texture** consiste à définir, pour chaque sommet 3D de la géométrie, quel point de l'image représentant la texture doit y être appliqué. Le moteur de rendu peut ensuite interpoler les points à choisir dans la texture pour tous les autres points qui apparaîtront dans l'image (ceux qui sont à l'intérieur du triangle qui représente une facette). La figure 2 illustre ce procédé sur un cas très simple.

A chaque sommet du carré, de coordonnée (x, y, z), il est nécessaire de faire correspondre un point de la texture, qui, elle, est définie dans un repère (u, v) et dont les coordonnées 2D des pixels sont toutes comprises dans $[0, 1] \times [0, 1]$. Au niveau du code, pour définir la facette (A, B, C) de la figure 2 qui illustre ce procédé sur un cas très simple, on doit donc déclarer et initialiser un tableau de sommets qui prendra la forme suivante, déjà vue dans les TP précédents :

```
const geometry = new THREE.BufferGeometry();
const vertices = new Float32Array( [
  xa, ya, za, xb, yb, zb, xc, yc, zc
] );
```

puis associer ce tableau à la géométrie de l'objet :

1. Cela nécessite évidemment que Python soit installé sur votre machine ....

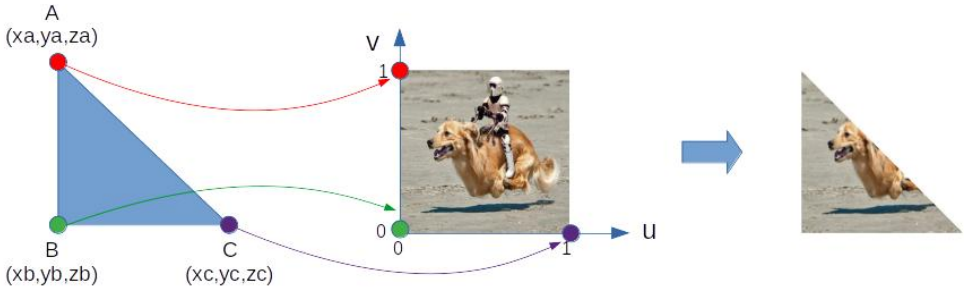


FIGURE 2 – Principe du placage de texture.

```
geometry.setAttribute( 'position', new THREE.Float32BufferAttribute( vertices, 3 ) );
```

Il faut utiliser un procédé similaire pour l'image des points A, B et C dans la texture :

```
const uvs = new Float32Array( [
  ua, va,  ub, vb,  uc, vc
) ];
```

avec dans l'exemple de la figure 2 , $(u_a, v_a) = (0, 1)$, $(u_b, v_b) = (0, 0)$ et $(u_c, v_c) = (1, 0)$. On associe ensuite ces coordonnées de texture à la géométrie de l'objet de manière similaire, en lui passant le tableau `uvs` en précisant qu'il s'agit de coordonnées de texture (paramètre `'uv'`).

```
geometry.setAttribute( 'uv', new THREE.Float32BufferAttribute( uvs, 2 ) );
```

Application Le script `exo2.html` permet d'afficher un carré modélisé à partir de la liste des deux triangles qui le composent, sans utiliser la classe `PlaneGeometry`. Elle utilise pour se faire la fonction `carre` définie dans le fichier `mesObjets.js` du sous-dossier `js`, qui prend comme paramètre la taille du côté du carré.

1. recopiez le code de cette fonction dans une nouvelle fonction nommée `carreTexture`, qui prendra un second paramètre qui représentera la texture à appliquer sur le carré ;
2. déclarez et initialisez le tableau de coordonnées de texture de telle sorte qu'il corresponde bien aux différentes coordonnées des sommets fournis dans `vertices` (il est conseillé de faire un schéma) ;
3. associez ces coordonnées à la géométrie et activez le mode texture dans le matériau de l'objet ;
4. chargez et créez la texture dans le script principal, puis appelez cette nouvelle fonction à la place de la fonction `carre`.

Vous devez obtenir l'image de la figure 3.

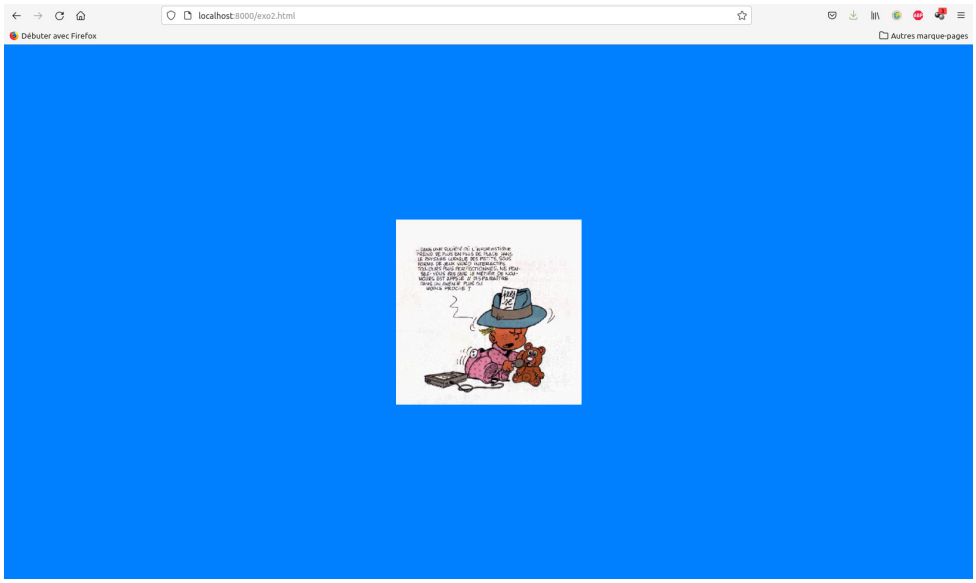
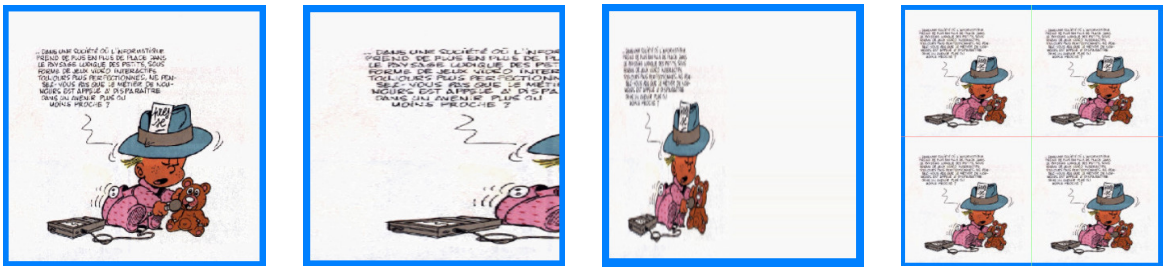


FIGURE 3 – Image après placage d'une texture sur le carré.

Exercice 3

Dans l'exercice précédent, l'image est appliquée sur la totalité du carré (figure 4a). Expérimentez les deux modifications suivantes dans un fichier nommé `exo3.html`, après y avoir recopié le contenu du fichier `exo2.html` :

1. changez la valeur de la coordonnée de texture u dans votre tableau `uvs`, de telle sorte que les valeurs qui étaient à 1.0 passent à 0.5. Pourquoi obtient-on ce résultat ? (voir figure 4b) ;
2. faites de même, mais en changeant la valeur 0.5 par 2.0. Même question ? (voir figure 4c).



(a) Rendu de la texture sur l'ensemble du carré. (b) Rendu de la texture avec la valeur $u = 0.5$. (c) Rendu de la texture avec la valeur $u = 2.0$. (d) Rendu de la texture dupliquée 2 fois.

FIGURE 4 – Différents aspects de l'application de la texture pour l'exercice 3.

Lorsque les coordonnées de texture sont plus grandes que 1.0, il est possible de spécifier que l'on souhaite dupliquer la texture [2], de manière à éviter le résultat obtenu dans la figure 4b, où le moteur de rendu complète par du blanc l'absence de pixels dans la texture au delà de la coordonnées $u = 1.0$. Pour ce faire, il suffit d'initialiser l'attribut `wrapS` de la texture à la valeur `THREE.RepeatWrapping`.

Application

1. Modifiez votre script principal pour que cette initialisation soit faite après création de la texture, soit ajoutez la ligne suivante :

   ```
   uneTexture.wrapS = THREE.RepeatWrapping;
   ```

   où `uneTexture` représente la texture à modifier. Vous ferez de même pour l'attribut `wrapT` qui spécifie que l'on souhaite aussi dupliquer la texture verticalement si une coordonnée de texture v est supérieure à 1.0.
2. Modifiez votre fonction `carreTexture` de telle manière que la texture soit présente 2 fois horizontalement et 2 fois verticalement (cf figure 4d).
3. Ajoutez au fichier `mesObjets.js` une nouvelle fonction nommée `carreTexture2` qui prendra trois paramètres : les deux premiers seront les mêmes que pour la fonction `carreTexture` et le troisième représentera le nombre de répétitions de la texture horizontalement et verticalement. Après codage de cette fonction, modifiez le script principal de manière à ce que le nombre de répétitions soit initialisé à 1.0, qu'à chaque appui sur la touche `+` ce nombre soit multiplié par 2 et à chaque appui sur la touche − il soit divisé par 2. L'affichage devra être modifié en conséquence.

Exercice 4

Ajoutez la fonction suivante à votre fichier `mesObjets.js` :

```
cubeTexture(c, texture1, texture2, texture3)
```

qui permet de construire un cube de côté `c`, centré à l'origine et où la première texture sera plaquée de manière continue sur les faces latérales du cube, tandis que la seconde texture apparaîtra sur la face supérieure et la troisième texture sur la face inférieure (voir figure 5). On précise que ceci implique de créer 3 `mesh` différents, chacun associé à un matériau différent. Vous modifierez en conséquence le script présent dans le fichier `exo4.html` et vous pourrez utiliser l'image `landscape.jpg` qui se trouve dans votre dossier `Images` comme texture pour les faces latérales du cube.

2. D'autres modes que la duplication existent, mais ne seront pas étudiés dans le cadre de ce TP.

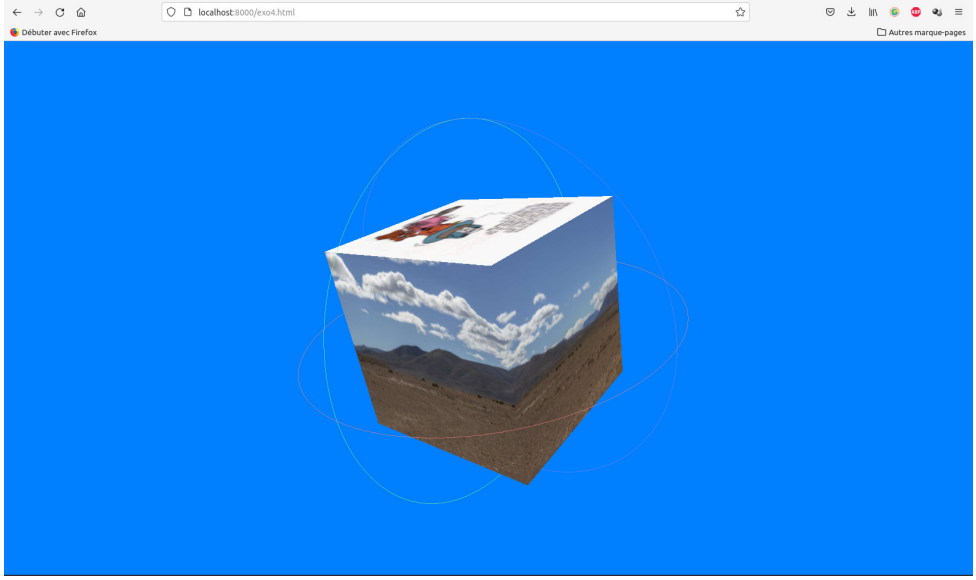


FIGURE 5 – Illustration du placage continu d'une image sur les 4 faces latérales d'un cube. Seules deux faces latérales sont visibles sur cette image..

Exercice 5

Complétez le script présent dans le fichier `exo5.html` de telle sorte qu'il anime un système solaire simplifié, comprenant le Soleil, la Terre et la Lune. On supposera ces astres parfaitement sphériques et vous pourrez utiliser les textures présentes dans le dossier `Images` pour recouvrir les objets géométriques les représentant.