Travail pratique #3

Graphiques avec Three.js et WebGL

INF5071 - Infographie

Automne 2022

Table des matières

Exercice 1 : Visualiser des neurones en 3D (7 pts)

Pour cet exercice, vous devez utiliser l'API `Three.js`

Le *Allen Institute* est un groupe de recherche oeuvrant dans le domaine de la neuroscience et ayant pour visée de mieux comprendre l'organisation du cerveau et son fonctionnement. Un de leurs projets de recherche phare, le *Mouse Brain Connectivity Project*, a permis de cartographier les connexions neuronales dans un cerveau de souris. Pour ces expériences, un virus fluorescent a été injecté à divers endroits dans le cerveau. Ce virus se propage ensuite le long des neurones pour rendre visible toute la projection neuronale dans le cerveau de souris. Ensuite, à l'aide de microscopie optique et de méthodes avancées d'analyse d'images, il est possible de créer une version numérique 3D des neurones injectés. Le but de cet exercice est d'utiliser des données Open Source partagées par le **Allen Institute** pour visualiser des neurones en 3D dans un cerveau de souris.

Note : utilisez les gabarits `tp3_ex1.html` et `tp3_ex1.js` pour commencer votre travail.

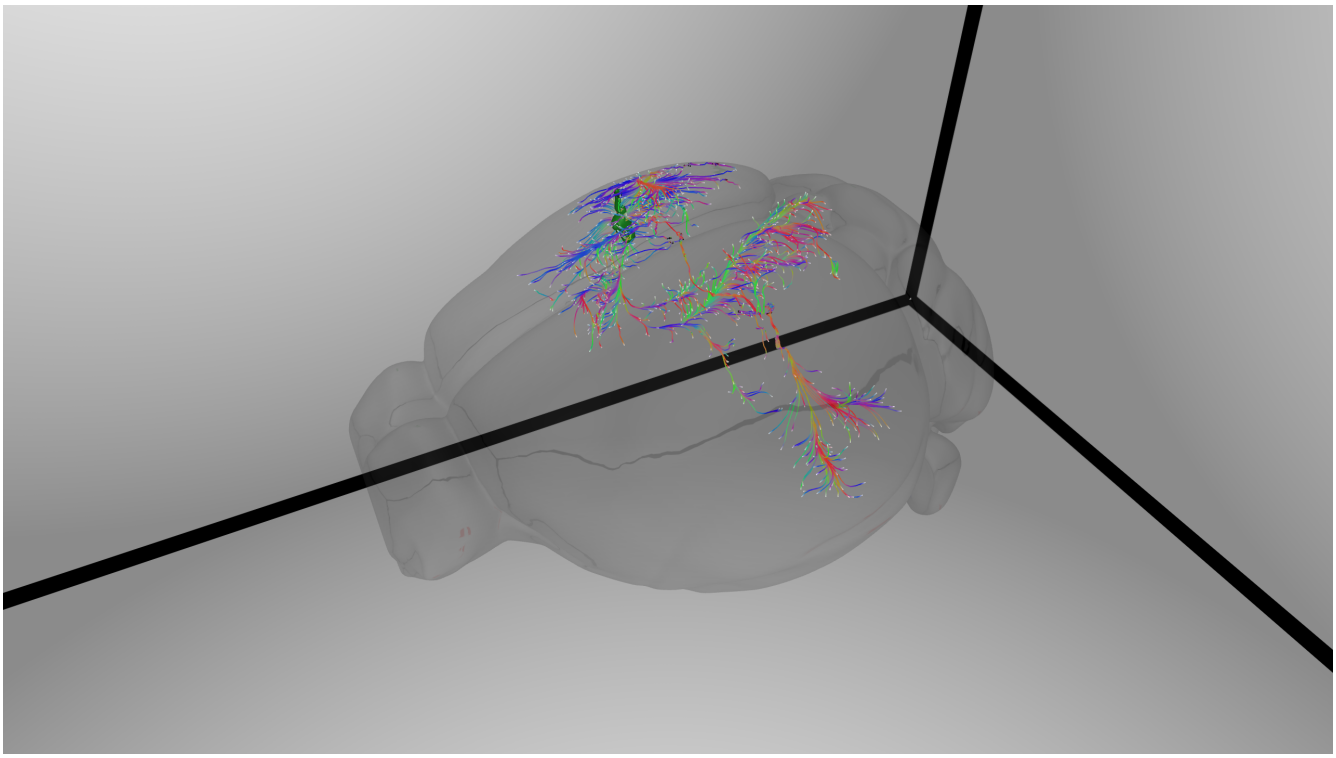

FIGURE 1 – Résultat attendu pour l'exercice 1. Une version vidéo est disponible : https://youtu.be/ivKdmvGeqXs

a) Préparation de la scène 3D

Pour préparer la scène 3D, vous devez ajouter une caméra de type Perspective utilisant un FOV de 45, un ratio de `canvas.width/canvas.height`, et des limites `near=0.1` et `far=100`. Placez cette caméra à la position $(x, y, z) = (-1, 2, 2)$. Ajoutez ensuite une lumière directionnelle blanche (intensité = 0.8) pointant dans la même direction que la caméra. Pour que la lumière se déplace avec la caméra lors de l'interactivité avec la souris, vous devez ajouter cette lumière directionnelle comme *enfant* de la caméra, qui doit être elle-même un objet dans la scène. Ajoutez également une lumière ambiante blanche (intensité =0.2).

Utilisez une texture *cubemap* que vous devez générer vous-même avec Inkscape et importer dans votre application. Chaque face de votre `CubeTexture` doit être un carré avec un gradient radial pour la couleur de remplissage et un contour noir. Votre `CubeTexture` doit aussi contenir des lettres rouges / vertes / bleues au centre de chaque face pour indiquer l'orientation du cerveau (R / L en rouge pour les faces droite / gauche, A/P en bleu pour les faces antérieure et postérieure, et S / I en vert pour les faces supérieure et inférieure). Référez-vous à cette figure pour connaître la convention utilisée pour l'orientation des cerveaux de souris.

Vous devez compléter la fonction `createScene`. Recherchez aussi les `TODO` associés à l'exercice 1.a dans le fichier `tp3_ex1.js`.

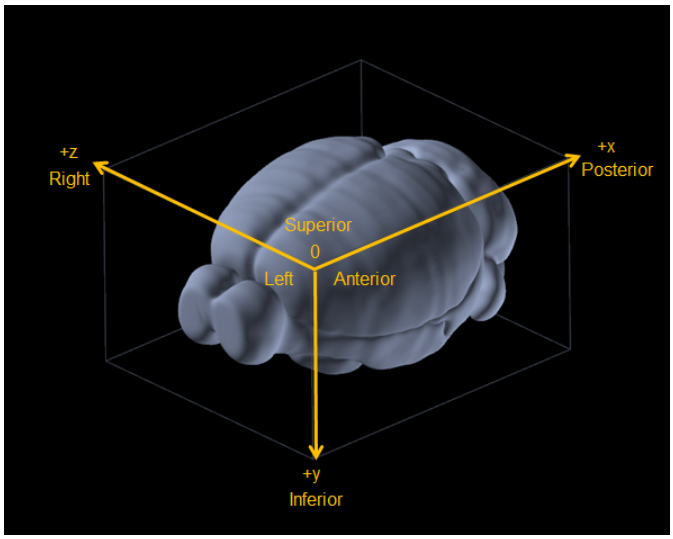

FIGURE 2 – Convention d'orientation pour les cerveaux de souris (Allen Institute)

b) Modélisation de la surface du cerveau de souris

Représentez la surface du cerveau de la souris à l'aide d'un maillage gris utilisant un matériau de Phong semi-transparent (opacité = 25%) et un ombrage lisse. Le matériau utilisé pour la surface du cerveau doit également réfléchir l'environnement (réflexivité de 1.0) et posséder un ratio de réfraction de 0.25. Le fichier `allenMouseBrain.obj` contient le maillage triangulaire de la surface d'un cerveau de souris. La fonction `loadBrain` fournie avec ce TP doit être utilisée pour importer les données. **Vous devez compléter la fonction `add_brainMesh`. Recherchez aussi les `TODO` associés à l'exercice 1.b dans le fichier `tp3_ex1.js`.**

c) Modélisation du volume d'injection

Représentez le volume d'injection du virus fluorescent à l'aide d'un maillage vert utilisant un matériau de Phong, un ombrage lisse, et une brillance de 30. Le fichier `volumeInjection.obj` contient le maillage triangulaire de la surface englobant le volume d'injection. La fonction `loadInjection` fournie avec ce TP doit être utilisée pour importer les données. **Vous devez compléter la fonction `add_injectionVolumeMesh`. Recherchez les `TODO` associés à l'exercice 1.c dans le fichier `tp3_ex1.js`**.

d) Modélisation des streamlines

Le parcours des neurones a été extrait des données de microscopie à l'aide d'un algorithme nommé **tractographie**. Le résultat obtenu suite à la tractographie est un ensemble de *streamlines* 3D plus ou moins longues. Les streamlines sont décrites par une suite de points dans l'espace 3D.

La façon conventionnelle de représenter des streamlines en neurosciences est en leur assignant une couleur selon leur orientation dans le cerveau.

— Les segments des neurones orientés selon l'axe gauche-droite (direction `Z` pour les données

fournies) doivent être rouges;
— Les segments orientés selon l'axe inférieur/supérieur (direction `Y` pour les données fournies) doivent être verts;
— Les segments orientés selon l'axe antérieur/postérieur (direction `X` pour les données fournies) doivent être bleus

Pour cette application graphique, vous devez représenter chaque streamline à l'aide d'une primitive de type `THREE.Line`. Utilisez un matériau de type `THREE.LineBasicMaterial` avec une épaisseur de ligne = 2. Finalement, pour votre géométrie vous devez fournir une liste de sommets ainsi qu'une liste de couleurs pour chaque sommet. Définissez les composantes RGB de la couleur d'un sommet en considérant la différence absolue des positions x, y, z pour ses 2 voisins les plus proches et normalisez le vecteur $[I_R,I_G,I_B]$. **Vous devez compléter la fonction `add_singleStreamline`. Recherchez les `TODO` associés à l'exercice 1.d.**

$$I_{r,k} = |z_{k+1} - z_{k-1}| \tag{1}$$
$$I_{g,k} = |y_{k+1} - y_{k-1}| \tag{2}$$
$$I_{b,k} = |x_{k+1} - x_{k-1}| \tag{3}$$

où k est le k^e point d'un streamline.

Livrable : Vous devez remettre le fichier `tp3_ex1.html` complété, les fichiers JavaScript `three.js` et `TrackballControls.js`, ainsi que vos fichiers de texture (`posx.png`, `negx.png`, `posy.png`, `negy.png`, `posz.png`, `negz.png` et `cubemap.svg`). Indiquez en commentaire dans le fichier `tp3_ex1.html` quel fureteur web (Chrome, Firefox, Edge) vous avez utilisé pour développer votre application graphique.

Exercice 2 : Programme de peinture simple (8 pts)

Pour cet exercice, vous devez utiliser WebGL.

Pour cet exercice, il faut développer une application de peinture web simple. Cette application doit permettre de dessiner sur un canevas blanc en utilisant un pinceau numérique circulaire. Il faut pouvoir modifier la taille du pinceau et son opacité, et l'application doit comporter un bouton pour effacer le dessin.

Note : Utilisez les gabarits `tp3_ex2.html` et `tp3_ex2.js` pour commencer votre travail.

Nous utiliserons une primitive de type `gl.POINTS` pour représenter le pinceau. La forme du pinceau doit être un cercle de couleur uni, pouvant être semi-transparent. La taille du cercle doit pouvoir varier entre `1` et `64`.

a) Création du shader de sommet

Pour compléter le shader de sommet, vous devez :

— Déclarer les variables de types `attribute` / `uniform`

FIGURE 3 – Traits fragmentés (gauche, ex 2.d) et traits continus (droite, ex 2.e) obtenus avec l'application de peinture web. Un vidéo du résultat attendu est disponible : https://youtu.be/f5qlmd0xGFs

— Convertir la position du curseur en coordonnées utilisées par WebGL
— Assigner les valeurs des variables `gl_Position`, et `gl_PointSize`.

b) Création du shader de fragment

Pour compléter le shader de fragment, vous devez :

— Déclarer les variables `uniform`
— Calculer la distance entre un pixel et le centre du point
— Représenter le point par un cercle de couleur unie et ayant une composante alpha

c) Initialisation

Complétez la fonction `initGL` et `init` pour que les propriétés de chaque variable du shader soient assignées et que les bonnes propriétés du contexte `WebGL` pour un programme de peinture 2D soient utilisées. Vous devez aussi compléter les fonctions `callback` utilisées pour l'interactivité entre l'artiste et WebGL.

d) Dessin (pour des mouvements lents)

Complétez la fonction `draw()`. Cette fonction sera appelée seulement lorsque la souris est en mouvement et que le bouton de gauche est enfoncé.

Pour cette question, tracez uniquement le point pour la position actuelle de la souris. Cette position est reçue en argument de la fonction et est donnée dans le système de coordonnées naturel du canevas.

e) Dessin (pour des mouvements rapides)

Finalement, pour éviter que des mouvements rapides de la souris causent des traits fragmentés et disjoints, utilisez la dernière coordonnée de la souris dessinée (contenue dans la variable

`lastCoordinates`) pour déterminer une série de points à dessiner lors d'un appel à `draw`. Pour ce faire, représentez la droite à dessiner sous forme paramétrique

$$x = (1 - t)x_0 + tx_f \tag{4}$$
$$y = (1 - t)y_0 + ty_f \tag{5}$$

où $t \in [0, 1]$ est un paramètre indiquant la position le long d'une droite allant de la position (x_0, y_0) (la dernière position de la souris) vers la position (x_f, y_f) la position actuelle de la souris.

Instructions générales

- Le travail pratique doit être réalisé en équipe de 2 maximum.
- Vous devez soumettre un seul fichier `zip` qui contient tous les fichiers nécessaires pour tester votre travail, et le nom de ce fichier `zip` doit inclure le nom de famille de chaque membre de l'équipe.
- Veuillez utiliser les gabarits fournis avec le TP pour commencer les exercices.
- Vous devez soumettre votre travail via Moodle obligatoirement. Les soumissions par courriel ne seront pas corrigées.
- Le plagiat ne sera pas toléré, écrivez votre propre code. Les normes de plagiat de l'université seront appliquées en cas de plagiat (voir la Politique 18).
- Il est important qu'il n'y ait pas d'erreurs d'exécution pour la correction. Les images doivent s'afficher.
- Indiquez dans votre rapport quel fureteur web vous avez utilisé pour réaliser le TP. Veuillez utiliser Firefox, Chrome ou Edge.
- Une partie des points pour chaque question est attribuée à la lisibilité du code. Mettre des commentaires pour expliquer votre démarche.
- La qualité artistique n'est pas un objectif du cours. Ne passez donc pas trop de temps sur l'esthétique de vos graphiques, puisque cela ne sera pas évalué.
- **Note :** ce travail compte pour 15% de la note finale.

Références

- WebStorm