Sorbonne Universit
3M235 - Calcul matriciel numrique
Travaux pratiques - feuille 1

Les quaternions forment une structure algbrique semblable aux nombres complexes. Ils ont t introduits au 19me sicle par W.R. Hamilton et sont rests bien des gards une curiosit mathmatique jusqu' ce que leur intrt fut mis en lumire la fin du 20me sicle avec l'essor des programmes de visualisation 3D.

Un quaternion q arbitraire s'crit sous la forme $q = a + bi + cj + dk$, o a, b, c, d sont des rels quelconques. Du point de vue vectoriel, l'espace $\mathbb{H}$ de tous les quaternions est donc isomorphe $\mathbb{R}^4$. On munit $\mathbb{H}$ d'une structure d'algbre en imposant les lois de produits : $i^2 = j^2 = k^2 = -1$, ainsi que $ij = k, jk = i, ki = j$ et inversement $ji = -k, kj = -i, ik = -j$. Cette multiplication est associative mais n'est pas commutative.

Exercice 1. On reprsente un quaternion par un vecteur *numpy* de longueur 4. Ecrire une fonction *mult* prenant en entre deux quaternions q_1, q_2 et renvoyant en sortie le quaternion q_1q_2 (attention respecter l'ordre, la multiplication n'tant pas commutative).
Etant donn $r \in \mathbb{R}$ et $v \in R^3$, on note $q(r, v)$ le quaternion dont r est la *partie relle* et v la *partie imaginaire*. Inversement, tant donn une quaternion arbitraire q, on note $r(q) \in R$ et $v(q) \in \mathbb{R}^3$ ses parties relles et imaginaires. Par commodit d'criture, on crit aussi r la place de $q(r, 0)$ et v la place de $q(0, v)$ (autrement dit on peut penser n'importe quel rel comme un quaternion (dont la partie imaginaire est nulle) et galement n'importe quel vecteur de $\mathbb{R}^3$ comme un quaternion (dont la partie relle est nulle).

Exercice 2. Etablir une formule dcrivant le produit q_1q_2 o $q_1 = q(r_1, v_1)$ et $q_2 = q(r_2, v_2)$ sans qu'apparaissent explicitement les composantes de q_1 et q_2, mais uniquement des oprations vectorielles entre ces deux vecteurs de $\mathbb{R}^3$, combines avec les rels r_1 et r_2). Rcrire la fonction de l'exercice 1 en tenant compte de cette formule.

La norme d'un quaternion $q(r, v)$ est simplement la norme du vecteur de $\mathbb{R}^4$ correspondant, soit $\|q\|^2 = r^2 + \|v\|^2$. De manire quivalente, $\|q\|^2 = q^*q$, o $q^* := q(r, -v)$ dsigne le quaternion conjugu de q. Si q est un quaternion de norme unit, alors on peut toujours crire q sous la forme $q = q(cos(\theta), sin(\theta)u)$ o $\theta \in \mathbb{R}$ et u est un vecteur unit de $\mathbb{R}^3$. Cette dcomposition n'est pas unique, on peut bien sr modifier θ par un multiple de 2π, mais on peut aussi remplacer conjointement θ par $-\theta$ et u par $-u$.

Exercice 3. Vrifier sur papier que si $q = q(cos(\theta), sin(\theta)u)$ est un quaternion unitaire donn, alors l'application $\mathrm{rot}_q : \ v \in R^3 \mapsto q^*vq \in \mathbb{R}^3$ est bien dfinie (c'est--dire que le quaternion q^*vq est bien partie relle nulle lorsque v est partie relle nulle) et que $\mathrm{rot}_q(v)$ reprsente le vecteur obtenu en appliquant v une rotation anti-horlogique d'angle 2θ autour de l'axe dtermin par u. Ecrire une fonction python *conj* prenant pour entre un quaternion q et renvoyant son quaternion conjugu, et une fonction *rot* prenant en entre (dans l'ordre) un vecteur v et un quaternion q et renvoyant le vecteur q^*vq.

C'est cette dernire proprit qui est l'origine du succs des quaternions dans toutes les oprations de rendu 3D. Elle permet d'appliquer des rotations tous les points de la scne (correspondant un changement de point de vue de l'observateur) en moins d'oprations machine que ce que ncessiterait le produit matriciel avec la matrice dont il est question dans l'exercice suivant.

Exercice 4. En utilisant le resultat de l'exercice prcdent, construire la matrice 3×3 de l'application linaire de $\mathbb{R}^3$ dans $\mathbb{R}^3$ correspondant la rotation anti-horlogique d'angle θ autour d'un vecteur unit donn $u \in \mathbb{R}^3$. Vrifier, pour chacun des trois cas particuliers o u est respectivement e_x, e_y et e_z, que la matrice ainsi obtenue est bien celle laquelle on s'attend.

Exercice 5. Tlcharger le script python 3M235_TP1.py l'adresse
`http://www.ljll.math.upmc.fr/ smets/3M235_ TP1.py` et complter les parties laisses vides pour terminer de raliser notre mini moteur de rendu 3D. Le script commence par lire une scne (ici un cube dterminer par une liste de points (ses sommets) et une liste de faces (faisant rfrence aux indices des points). Du module *matplotlib* on n'utilise que les capacits de tra 2D. La scne est suppos dcrite initialement dans un repre de base orthonorm (x, y, z), et rendue au travers d'une camra dont le repre propre $(\hat{x}, \hat{y}, \hat{z})$ est tel que $(\hat{x}, \hat{y})$ est parallle au plan de la camra et $\hat{z}$ est perpendiculaire ce dernier (c'est donc l'axe de vise). La rotation permetant de passer du repre de la scne au repre de la camra est encode sous forme d'un quaternion (cfr exercice 3), dans le script python il est appel *point_de_vue*. La souris est utilise pour faire varier le repre de la camera, matplotlib permettant d'tre "prvenus" (fonctions dites de "call-back") des actions de la souris sur la figure. A chaque appel suivant une action de la souris, le repre camra est mis jour (fonction *maj_point_de_vue*), et la scne est ensuite projete dans le plan $(\hat{x}, \hat{y})$ de ce repre en guise de rendu (fonction *maj_rendu*).