Université Lille1 Sciences et Technologies
UFR IEEA

M3DS

2013–2014
Master ST-A MS2

Examen 2013-2014
1ere session
Durée : 2 heures
Tout document papier autorisé

Toute réponse doit être **justifiée**. Le barème est donné à titre indicatif.

Exercice 1 : Changements de repères *(4 points)*

Les 3 questions sont indépendantes.

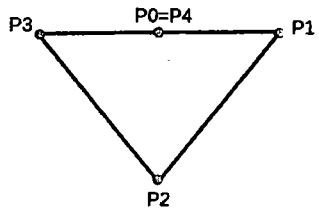
Q 1. Pour le tracé d'une scène, la position de la caméra est donnée par rapport au repère du monde et est traduite par une matrice homogène $M_{world \to cam}$. Un objet est placé dans le repère du monde ce qui se traduit par la matrice homogène $M_{world \to object}$. Comment obtenir les coordonnées dans **le repère de la caméra** d'un point P quelconque qui est connu dans le repère de l'objet ?

Q 2. On place l'objet directement par rapport à la caméra (i.e. on connait donc directement la matrice $M_{cam \to object}$). Le placement de la caméra, lui, se traduit toujours par $M_{world \to cam}$. Comment obtenir les coordonnées dans **le repère du monde** d'un point P quelconque qui est connu dans le repère de l'objet ?

Q 3. On dispose du tracé d'un cylindre `drawCylinder()` dont l'axe local est l'axe z et ses 2 bases sont en $z = 0$ et $z = 1$ (i.e. cylindre de hauteur 1 sur l'axe z). On ne se préoccupe pas du rayon.

On se donne 2 points A et B dans le repère du monde. Comment tracer le cylindre (peu importe le rayon) qui relie les points A et B ? Vous proposerez un pseudo-code en utilisant ce qui vous semble utile parmis : `translate(un vecteur)`, `rotate(un angle, un vecteur)`, `rotate(un quaternion)`, `scale(un vecteur)` (ces 4 instructions modifient le repère courant comme vu en Cours/Tps ; considérez qu'on se trouve déjà dans le repère du monde), `dot(u,v)`, `cross(u,v)`, `length(u)` ainsi que `quaternion(u,v)` qui donne directement le quaternion qui permet de transformer le vecteur **unitaire** u en le vecteur **unitaire** v (10 lignes **maxi** ; toute réponse "lourde" ne sera pas évaluée).

Exercice 2 : Courbe *(5 points)*



Q 1. Reprenez à main levée les 5 points de contrôle $(P_0, P_1, P_2, P_3, P_4)$ (le premier P_0 et le dernier P_4 sont confondus), et faites apparaitre la construction de De Casteljau pour le point $P(\frac{3}{4})$ de la courbe de Bézier correspondante à ces points de contrôles (construction à main levée, mais **très clairement**).

Q 2. Tracez, toujours à main levée, la forme de la courbe de Bézier.

Q 3. La courbe se raccorde avec elle même au point $P_0 = P_4$: en ce point, le raccordement est-il de continuité C_0 ? C_1 ?

Q 4. Reprenez à nouveau ces 5 points, mais cette fois vous tracerez grossièrement la forme, à main levée, d'une interpolation par des courbes de hermites (une courbe de hermite entre P_0 et P_1, puis entre P_1 et P_2, puis entre P_2 et P_3 et enfin entre P_3 et P_4). Comme tangentes pour ces hermites, vous prendrez en P_0 le vecteur $\overrightarrow{P_0P_1}$, en P_1 $\overrightarrow{P_1P_2}$ en P_2 $\overrightarrow{P_2P_3}$, en P_3 $\overrightarrow{P_3P_4}$ et en P_4 $\overrightarrow{P_4P_1}$ (autrement dit le vecteur reliant le point et son successeur).

Exercice 3 : Lancer de rayons et BSP *(6 points)*

On souhaite faire un lancer de rayon sur une scène constituée uniquement de triangles et représentée par un BSP.

On se donne un rayon `ray`. Son origine est donnée par `ray.point()` et sa direction par `ray.direction()`. Si `bsp` est de classe `BSP` (représente un noeud du BSP), on dispose du pseudo-code :

- `bsp.empty()` : vrai si l'arbre est vide.
- `bsp.negative()` : le sous arbre négatif.
- `bsp.positive()` : le sous arbre positif.
- `bsp.triangle()` : le triangle du noeud `bsp`

Pour un triangle `t` donné, on dispose de `t.sign(un point)` (donne `POSITIVE` ou `NEGATIVE` selon le signe du point par rapport au triangle), et de `t.intersection(un rayon)` qui donne soit `EMPTY` s'il n'y a pas d'intersection avec le rayon, soit un réel qui correspond au λ de l'intersection du triangle avec le rayon (i.e. l'intersection I est donnée par $I = ray.point() + \lambda * ray.direction()$).

L'objectif est de trouver, pour l'ensemble des triangles, le λ le plus petit et qui est strictement supérieur à 1 (i.e. recherche de l'intersection avec la scène pour un rayon primaire ; on retournera `EMPTY` si aucun $\lambda > 1$ n'existe). On souhaite exploiter le BSP pour éviter des intersections inutiles avec les triangles.

Q 1. On se donne un noeud `bsp`, et le rayon `ray`. On note `inter` le résultat de `bsp.triangle().intersection(ray)` (`inter` est donc soit un nombre, soit `EMPTY`). Comment éviter des intersections inutiles dans les cas suivants (justifiez ; schéma 2D rapide bienvenu) :

1. lorsque `inter` est strictement plus petit que 1.
2. lorsque `inter` est strictement plus grand que 1.
3. lorsque `inter` est `EMPTY`, il sera préférable de commencer à parcourir l'un des 2 sous-arbres : lequel et pourquoi ?

Q 2. En déduire le pseudo-code de `intersect(BSP bsp, Ray ray)` qui donne l'intersection entre `bsp` et `ray` (pseudo-code récursif ; le résultat est soit `EMPTY`, soit un nombre strictement supérieur à 1).

Q 3. Pour un arbre BSP quelconque, dans quel cas tous les triangles sont parcourus ?

Q 4. On choisit d'englober tout le BSP dans une AABB (Axis-Aligned Bounded Box). Une AABB `b` sera représentée par la classe `Box` qui possède 2 points `b.min` et `b.max` qui sont les 2 sommets de la diagonale de la boite. Les sommets d'un triangle `t` peuvent être récupérés par `t.A`, `t.B` et `t.C` (par exemple `t.B.y` donne la coordonnée y du sommet B). Donnez le pseudo-code de la fonction `AABB(BSP bsp)` qui donne l'AABB du bsp (si un traitement est identique sur A, B, C d'un triangle, ou sur les coordonnées x, y, z, contentez vous d'inclure `Idem pour ...` dans votre pseudo-code).

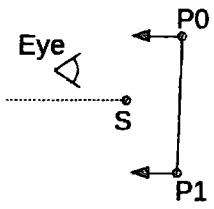
Q 5. On dispose de l'intersection d'un rayon avec une AABB, répondez alors à nouveau à la question : dans quel cas tous les triangles sont parcourus ?

Exercice 4 : Eclairement

(5 points)

Les réponses doivent être brèves, claires et justifiées (schémas obligatoires).

Q 1.



Le schéma 2D représente un polygone avec les sommets P_0 et P_1 (les flêches indiquent les normales en ces sommets), la caméra `Eye` ainsi qu'une source lumineuse S. Le trait en pointillés représente le mouvement possible de la source lumineuse.

1. On considère uniquement la réflexion diffuse avec un éclairage par sommet (interpolation de Gouraud) : lorsque la source S est proche du polygone, il apparait sombre, et lorsque la source est éloignée le polygone apparait clair (ce qui peut sembler, à priori, contre intuitif). Justifiez pourquoi.
2. On effectue à présent un éclairage par pixel : qu'observe t'on sur le polygone ?
3. On revient à un éclairage par sommet, mais on ajoute le spéculaire. Dans la configuration du schéma, on ne voit pas le spéculaire : pourquoi ?
4. On passe à l'éclairage par pixel : où se trouve approximativement la tâche spéculaire dans la configuration du schéma ?

Q 2. On implémente les ombres portées :

1. Avec la technique de la depth map, quel est le problème si la source lumineuse se trouve au "centre" de la scène (des objets se trouvent tout autour de la source lumineuse) ?
2. Le problème se présente-t-il pour les shadow volumes ?
3. Donnez un argument pour justifier que les ombres portées demandent moins d'efforts pour les implémenter en lancer de rayons par rapport à une visualisation en OpenGL.