IFSIC **Rennes le 25 janvier 2007**
Master Recherche Informatique

EPREUVE DE LR2V
DUREE : 2 heures
Documents autorisés

Le sujet est rédigé en français (version française) et en anglais (english version)

VERSION FRANCAISE

Problème 1 : Questions de cours

- **Question 1.1**
  Quelle est la différence entre une HVE (Hiérarchie de Volumes Englobants) et une SS (Subdivision Spatiale) telle qu'une grille 3D ou un Kd-tree ? Est il possible de les combiner pour plus d'efficacité ? Comment ?
- **Question 1.2**
  Quels sont les facteurs qui rendent le lancer de rayon très coûteux en terme de temps de calcul ?
- **Question 1.3**
  De combien de niveaux de récursion avez vous besoin, dans un algorithme de lancer de rayon, pour montrer la réflexion sur une sphère d'une texture plaquée sur un mur d'une pièce ? Combien de niveaux faudrait il si une sphère transparente se trouve entre le mur texturé et la sphère réfléchissante ? Justifier votre réponse à l'aide d'une figure montrant les rayons.
- **Question 1.4**
  Répondez par Vrai ou Faux et donnez des justifications à vos réponses.
  - (a) L'arbre BSP associé à une scène, est il calculé indépendemment du point de vue ?
  - (b) Les performances d'un arbre BSP ne dépendent pas de la manière dont les plans de découpage sont choisis.
  - (c) Le *Bump mapping* prend en compte l'occultation propre.

Problème 2 : Tampon de Lumière

la méthode de synthèse d'image utilisée est celle du lancer de rayon. On suppose que la scène est constituée de N_t triangles. Elle est rangée dans un tableau à une dimension $TRI[num], num \in [0, N_t - 1]$. La scène est éclairée par une source ponctuelle. Nous voulons accélérer le calcul de l'éclairage direct. Pour cela, la source ponctuelle est englobée par un cube de telle façon qu'elle se trouve au centre de ce cube. Chacune des 6 faces du cube englobant est divisée en $L \times L$ pixels. A chaque face est associé un tableau à deux dimensions, par exemple $Pixel_1[i, j]$ pour la face 1, $Pixel_2[i, j]$ pour la face 2 et $Pixel_k[i, j]$ pour la face k, où $i, j \in [1, L]$. Un élément d'un tableau $Pixel_k[i, j]$ contient le numéro du triangle (c'est-à-dire l'indice du triangle dans $TRI[num]$) vu de la source.

- **Question 2.1**
  Expliquer le principe de remplissage des six tableaux $Pixel_k[i,j], k = 1,6$.
- **Question 2.2**
  Soit P un point d'intersection entre un rayon (différent d'un rayon d'ombre) et la scène. Nous désirons décider rapidement si P est vu ou non par cette source, sans calculer l'intersection entre un rayon d'ombre et la scène. Proposer un algorithme.

Problème 3 : Tracé de Particules

Au lieu de lancer des rayons à partir de l'observateur, nous souhaiterions inverser le problème. C'est à dire, nous partons d'une source de lumière surfacique puis nous traçons des rayons transportant des particules de lumière à partir de cette source. Le principe est le suivant. Nous sélectionnons d'abord un point P_i sur la source de manière aléatoire. Ce point transportera un flux $\Delta\Phi$ égal au flus total Φ de la source divisé par le nombre de points échantillonnés sur la source. Ensuite, nous choisissons de manière aléatoire une direction D_i d'émission. Cette direction correspond en fait à un rayon dorigine P_i et de direction D_i. Ce rayon intercepte la scène en un point P_i^1 appartenant à une surface. A partir de ce point nous avons le choix (mis en oeuvre grâce à la fonction *Choix1*) de lancer un rayon ou pas. Si un rayon est lancé dans la direction D_i^1, on multipliera le flux du rayon incident (c'est à dire $\Delta\Phi$ pour le rayon émis de la source) par un facteur $\rho_d = R_d/\pi$, R_d étant la composante diffuse de la BRDF de l'objet contenant P_i^1. Le flux résultat sera associé au point P_i^1 . Nous obtenons donc une nouvelle particule associée au point P_i^1 et qui sera lancée dans la direction D_i^1 sous forme de rayon. Ce rayon intercepte la scène en un point P_i^2 appartenant à une surface, et le processus de récursion continue jusqu'à ce qu'un certain critère soit vérifié. Ensuite on choisit un autre point P_j sur la source puis on choisit de manière aléatoire une direction D_i^2. et on répète le processus. Un point d'intersection ainsi que certaines données associées à ce point (direction incidente et flux incident) sont tockés dans une structure de données appelée *enregistrement*. Ces enregistrements sont stockés dans une structure de données spatiale *SDD*.

Dans une deuxième phase, appelée phase de rendu, on lance un ou plusieurs rayons primaires à travers un pixel p_{ij}. Ce rayon intercepte la scène en un point P_{ij}. La luminance au point P_{ij} sera calculée en évaluant la somme pondérée des flux associés aux enregistrements voisins de P_{ij}.

- **Question 3.1**
  Comment échantillonner la source de lumière et les directions D_i^k, k étant le niveau de récursion ?
- **Question 3.2**
  Comment mettre en oeuvre la fonction *Choix1*, autrement dit, quelle méthode peut on utiliser pour terminer la récursion de manière efficace (c'est à dire comment décider de ne pas faire réfléchir une particule) ?
- **Question 3.3**
  Proposer une structure de données *enregistrement*.
- **Question 3.4**
  La structure *SDD* doit nous permettre de retrouver les enregistrements voisins d'un point donné. Proposer une structure de données et expliquer comment l'utiliser pour déterminer ces enregistrements voisins.
- **Question 3.5 : subsidiaire**
  Proposer un algorithme complet de tracé de particules. Nous ne nous intéressons qu'à la première Phase. Il n'est pas demandé dans ce problème de donner des algorithmes détaillés, mais uniquement des algorithmes de haut niveau expliquant le fonctionnement du système.

ENGLISH VERSION

Problem 1 : Questions on the course

- **Question 1.1**
  What is the difference between a BVH (Bounding Volume Hierarchy) and a SDS (Spatial Data Structure) such as a 3D grid or a Kd-tree? Is it posssible to combine them for more efficiency? How?
- **Question 1.2**
  What factors make ray-tracing computationally expensive?
- **Question 1.3**
  How many levels of recursion will you need in a ray tracer to show the reflection on a sphere of a texture that is mapped onto one of the walls of the room? How many levels if a transparent sphere lies between the textured wall and the reflective sphere? Justify your answers with a picture showing the rays.
- **Question 1.4**
  Answer by True or False and give justifications.
  - (a) The BSP tree for a scene is computed in a viewpoint independent manner.
  - (b) The performance of a BSP tree representation is not influenced by the selection of splitting planes.
  - (c) Bump mapping accounts for self occlusion.

Problem 2 : Light Buffer

Ray tracing is used as a rendering technique. We suppose that the scene is made up of N_t triangles stored in a linear array $TRI[num], num \in [0, N_t - 1]$. The scene is lit by a single point light source. The objective of this problem is to propose a methof for speeding up the computation of direct lighting. To this end, a cube is centered around the point light source. Each of the 6 faces of the cube is discretized into $L \times L$ pixels. With each face is associated a 2D array, say $Pixel_1[i, j]$ for face 1, $Pixel_2[i, j]$ for face 2 and $Pixel_k[i, j]$ for face k, where $i, j \in [1, L]$. An element of the array $Pixel_k[i, j]$ contains the number of the triangle (say the index of the triangle in $TRI[num]$) as seen from the light source.

- **Question 2.1**
  Explain the process of filling the 6 arrays $Pixel_k[i, j], k = 1, 6$.
- **Question 2.2**
  Let P be the intersection point between a ray (different from a shadow ray) and the scene. Our aim is to rapidly determine if P is either visible or not visible from the light source, without having to compute the intersection between a shadow ray and the scene. Propose an algorithm to do so.

Problem 3 : Particule Tracing

Rather than tracing rays from the viewer, we would like to reverse the problem. In other words, we start from the area light source then we trace rays transporting light particles from this light source. The principle is the following. First, we randomly select a point P_i on the light source. This point will transport a flux $\Delta\Phi$ equal to the total flux Φ of the light source divided by the number

of points sampled on the light source. Next, we randomly choose an emission direction D_i. This direction actually corresponds to a ray of origin P_i and direction D_i. This ray intersects the scene at a point P_i^1 lying on a surface. From this point we have the choice (implemented using the function *Choice1*) of whether tracing a ray or not. If a ray is traced in the direction D_i^1, we multiply the flux of the incident ray (say $\Delta\Phi$ for the ray traced from the light source) by a factor $\rho_d = R_d/\pi$, R_d being the diffuse component of the BRDF of the object containing P_i^1. The resulting flux will be associated with the point P_i^1. We get then a new particule associated with point P_i^1 which will be traced in direction D_i^1, under the form of ray. This ray intersects the scene at a point P_i^2 lying on a surface, and the recursion processus is repeated untill a certain criterion is met. Next, we choose another point P_j on the light source then we randomly choose a direction D_i^2. This processus is repeated. An intersection point as well as certain data associated with this point (incident direction and incident flux) are saved in a data structure called *record*. These records are saved in a spatial data structure *SDD*.

In a second phase, called rendering, we trace one or more primary rays through a pixel p_{ij}. This ray intersects the scene at a point P_{ij}. The radiance at point P_{ij} will be calculated by evaluating the weighted sum of the fluxes associated with the records close to P_{ij}.

- **Question 3.1**
  How to sample the light source and the directions D_i^k, k being the recursion level?
- **Question 3.2**
  How to implement the function *Choice1*, in other words what method can be used to stop the recursion in an efficient way (say how to decide to whether reflect or not a particle)?
- **Question 3.3**
  Propose a data structure *record*.
- **Question 3.4**
  The data structure *SDD* has to allow us to recover the records close to a given point. Propose a data structure and explain how it could be used to determine these close records.
- **Question 3.5 : subsidiary**
  Propose a complete particle tracing algorithm. We are just interested in the first phase. It is not requested to provide detailed algorithms, but just high level ones explaining the working of the system.