Synthèse d'images avancée - Corrigé

9 Mars 2006

1 Synthèse de textures

1- [3 points] *Résumer en 5 lignes le principe des* polycube-maps*. Quelle est la particularité de la paramétrisation engendrée ?*

On attends un résumé concis qui contient le principe général :

Exemple : La méthode des *PolyCubeMaps* consiste à créer une paramétrisation d'une surface en s'inspirant des traditionelles *cubemaps*. La surface est d'abord plongée dans une surface intermédiaire formée de cubes parallèles aux axes. Une projection $\mathcal{P}$ permet d'associer a tout point a l'interieur d'un cube une coordonnée 3D à sa surface. Une seconde projection $\mathcal{M}$ transforme cette coordonnée en une coordonnée 2D dans une texture traditionelle.

A chaque vertex est associée une coordonnée de texture 3D, ce qui fait la particularité de cette paramétrisation.

2- [1 point] *La fonction qui a tout point x de la surface fait correspondre une coordonnée de texture (u, v) dans T^2 est-elle continue ?*

La projection $\mathcal{M} \times \mathcal{P}$ n'est pas continue puisque $\mathcal{P}$ est continue mais $\mathcal{M}$ ne l'est pas à cause du rangement des facettes des cubes dans l'espace de texture 2D.

3- [3 points] *Comment pourrait-on utiliser cette méthode de parametrisation pour améliorer la synthèse de texture sur une surface proposée par Wei et Levoy en 2001 ? (Citer deux améliorations importantes).*

On peut utiliser cette methode pour améliorer les points suivants de la méthode de Wei, du fait qu'elle génère une paramétrisation sur toute la surface :
- l'orientation intrinsèque des faces des cubes dispense de générer sur la surface un champ de vecteurs pour le parcours de type scan-line.
- le positionnement des points de couleurs peut etre directement parametré par leur position dans la face du cube correspondant : il n'est plus necessaire d'interpoler les couleurs sur un maillage.
- le rendu est grandement facilite, demême que l'export sous forme d'une texture et de coordonnées de texture.

2 Ombres

4- [3 points] *On veut calculer la proportion de la source visible depuis chaque point x du récepteur :*

$$s(x) = \frac{1}{\Omega_S} \int_{\Omega_S} v(x, \omega) d\omega$$

On utilise pour cela une méthode d'échantillonnage de la source pour discrétiser l'équation, en introduisant n points y_i sur la source et en calculant :

$$s_n(x) = \frac{1}{n} \sum_{i=1}^{n} v(x, y_i)$$

Selon quelle densité de probabilité faut-il disposer les échantillons sur la surface de la source pour que $s_n(x)$ converge vers $s(x)$ lorsque n tend vers l'infini ?

On a

$$\begin{aligned} s(x) &= \frac{1}{\Omega_S}\int_{\Omega_S} v(x,\omega)d\omega \\ &= \frac{1}{\Omega_S}\int_S v(x,y)\frac{\cos\theta'}{r^2}dy \end{aligned} \tag{1}$$

Soit p une densité de probabilité selon laquelle on distribue les y_i sur la surface de la source. On a

$$s(x) = \lim_{n\longrightarrow\infty}\frac{1}{n\Omega_S}\sum_{i=1}^{n} v(x,y_i)\frac{\cos\theta'_i}{r_i^2}\frac{1}{p(y_i)}\underbrace{\int_S p(y)dy}_{1}$$

Un seul choix possible de $p(y_i)$ rend cette expression égale à $s_n(x)$ pour tout n. Il s'agit de :

$$p(y_i) = \frac{\cos\theta'_i}{r_i^2\Omega_S}$$

ce qui correspond à distribuer les points selon la densité de probabilité :

$$p_x(y) = \frac{\cos\theta'}{r^2\Omega_S}$$

On vérifiera que

$$\int_S p_x(y)dy = 1$$

5- [2 points] *Démontrer que la méthode de Herf calcule $s_n(x)$ pour tout x à condition que le choix des y_i soit le même pour tous les points x du récepteur.*

La méthode de Herf calcule la moyenne des projections des obstacles depuis n points disposés sur la source. Pour un point x donné sur le récepteur, la méthode de Herf calcule donc $s_n(x)$.

6- [3 points] *En déduire que la méthode de Herf ne converge que vers une approximation de la proportion de la source visible lorsque le nombre de centres de projections choisis sur la source tend vers l'infini.*

Pour que $s_n(x)$ converge vers $s(x)$ il faut disposer les echantillons y_i selon une densité de probabilité qui, d'aprés ce qui précède, dépend de x ! ! On ne peux donc pas, en choisissant la même distribution de points pour tous les points du récepteur assurer la convergence de $s_n(x)$ vers $s(x)$ pour tout x. C'est ce que fait pourtant la méthode de Herf. Elle calcule donc une approximation de la proportion de la source visible depuis chaque point du récepteur.

3 Rendu expressif

7- [2 points] *Proposer une technique permettant d'afficher la scène en évitant les problèmes de cohérence temporelle.*

Il faut utiliser des textures avec prorité de manière à pouvoir produire une densité de traits égale au ton requis pour l'image tout en prenant en compte ses variations impliquées par la perspective.

8- [2 points] *On désire agrémenter le dessin en affichant les silhouettes dans un style variable et parametrable par l'utilisateur. Comment doit on les calculer ?*

Les silhouettes doivent être calculées dans l'espace objet. En effet, dans l'espace image, on ne récupère pas les silhouettes sous forme vectorielle, et donc on ne peut pas leur appliquer aisément un style particulier.

4 Éclairage global

9 *- **[2 points]** Expliquer pourquoi il est préférable que le support des fonctions Λ_i soit limité aux polygones de la scène de départ lorsqu'on ne peut pas considérer la normale comme continue d'un polygone à l'autre. Dans une grande pièce rectangulaire, par exemple, représenter Λ_i dans les cas où x_i est à un angle de mur, ou au centre d'un mur.*

La solution approximative de l'équation dans l'espace engendré par les Λ_i s'écrit :

$$B'(x) = \sum_i B_i \Lambda_i(x)$$

Sur tout domaine où fonctions de base sont continues, B' est donc continue. Or la radiosité n'est pas continue la où la normale n'est pas continue, d'où l'intérêt de limiter le support de ces fonctions aux polygones de la scène.

10 *- **[3 points]** On note $<,>$ le produit scalaire défini par*

$$< f, g >= \int_S f(x)g(x)dx$$

Calculer $< \Lambda_i, \Lambda_j >$ en fonction de l'aire $A_{i,j}$ de la zone formée des triangles partageant éventuellement les sommets x_i et x_j. Cette base est-elle orthogonale ?

Le produit scalaire entre deux fonctions de base Λ_i et Λ_j est égal à la somme des intégrales sur chaque triangle ayant x_i et x_j comme sommet. Notons L la base d'un triangle et H sa hauteur. Si $i == j$, on choisit comme base le coté du triangle sur lequel $\Lambda_i = 0$. Ces intégrales valent alors :

$$\begin{aligned} I_1 &= \int_0^H \frac{u^2}{H^2} \int_0^{L\frac{u}{H}} dxdu \\ &= \int_0^H \frac{Lu^3}{H^3} du \\ &= \frac{1}{2}A \end{aligned} \tag{2}$$

...où A est l'aire du triangle. On a donc

$$< \Lambda_i, \Lambda_i >= \frac{1}{2}A_{i,i}$$

Si $j \neq i$, on choisit pour L le coté du triangle contenant les points x_i et x_j. L'intégrale sur le triangle vaut :

$$\begin{aligned} I_2 &= \int_0^H \int_0^{L\frac{u}{H}} x(Lu/H - x)\frac{H^2}{L^2u^2} dxdu \\ &= \int_0^H \frac{Lu}{H}(\frac{1}{2} - \frac{1}{3})du \\ &= \frac{1}{6}A \end{aligned}$$

On a donc

$$< \Lambda_i, \Lambda_j >= \frac{1}{6}A_{i,j}$$

Cette base n'est donc pas orthogonale.

11 *- **[4 points]** discretiser l'équation de radiosité par la méthode de Galerkin en utilisant la base $\{\Lambda_i\}$, de telle facon que les inconnues du systeme obtenu soient les valeurs B_i. On donnera l'expression des facteurs de forme ainsi que celle des coefficients de la matrice.*

L'equation de radiosité s'écrit

$$B(x) = E(x) + \rho(x) \int_S v(x,y) B(y) \frac{\cos\theta \cos\theta'}{\pi r^2} dy$$

On effectue une discretisation par collocation aux points x_i. Pour cela on cherche une solution de l'equation sous forme d'une combinaison lineaire des fonctions de base $B' = \sum B_i \Lambda_i$ plus un résidu ΔB. Demême, on approche la réflectance sur le support de chaque Λ_i par un terme $\hat{\rho}_i$ indépendant de x sur chaque domaine d'intégration :

$$B'(x) = E(x) + \hat{\rho}_i \int_S G(x,y) B'(y) dy + \underbrace{\rho(x) \int_S G(x,y) \Delta B dy - \Delta B(x) - \Delta\rho_i \int_S ...}_{\Delta}$$

... et on annule le résidu Δ en exprimant qu'il est orthogonal aux fonctions de base. On obtient :

$$\begin{aligned} \forall i \quad \sum_{j \neq i} \frac{1}{6} A_{i,j} B_j + \frac{1}{2} A_{i,i} &= \underbrace{< E, \Lambda_i >}_{E_i} + \hat{\rho}_i \int_{\Lambda_i(x) \neq 0} \Lambda_i(x) \int_S v(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} B'(y) dy dx \\ &= E_i + \hat{\rho}_i \sum_j F_{ij} B_j \end{aligned}$$

Les facteurs de forme s'écrivent donc :

$$F_{ij} = \iint_{\Lambda_i(x)\Lambda_j(y) \neq 0} v(x,y) \frac{\cos\theta \cos\theta'}{\pi r^2} B'(y) \Lambda_i(x) \Lambda_j(y) dx dy$$

N.B : Du fait que les supports des fonctions sont disjoints lorsque les normales sont différentes, ces intégrales représentent bien des transfers d'énergie, et non des coefficients de couplage comme dans le membre de gauche.

5 Visibilité et niveaux de détails

12- [3 points] *Qu'est que la méthode de* frustum culling *? Donner les éléments clés.*

On teste si les objects sont dans le champ de la caméra avant de les envoyer à la carte graphique. Les points clefs sont :
- on teste des volumes englobants (bbox, bsphere) pour que le test soit simple et pour qu'il coûte bien moins cher qu'afficher l'objet ;
- on utilise une hiérarchie de volume englobants pour attteindre un coût sub-linéaire.

13- [3 points] *Qu'est que la méthode d'*horizon culling *? Comment l'utiliserait t'on dans notre cas ?*

C'est une méthode adaptée au rendu de scène type 2.5D. On rend les objets de l'avant vers l'arrière. On maintient une ligne d'horizon. Quand un objet est entièrement dessous, il n'est pas rendu. sinon il est rendu et on met à jour l'horizon.

14- [4 points] *Comment utiliserait-t-on les* occlusion queries *pour tester rapidement les objets en mouvement qui sont visibles depuis le point de vue courrant ? Comment peut-on faire pour que les résultats de ces calculs de visibilté soit réutilisables sur plusieurs frames ?*

On rend d'abord tous les bâtiments et arbres. Puis pour chaque objet, on teste la visibilité d'un volume englobant en le faisant rasteriser par la carte qui nous retourne combien de fragments passent le z-test. Si ce nombre est en dessous d'un seuil fixé (0 pour être conservatif, $< \varepsilon$ pour être aggressif) on n'affiche pas l'objet.
En testant des volumes englobants étendus dans la direction de mouvement des objets, le résultat pourra être réutilisé à la frame suivante.
On utilise comme occluders potentiels ce qui était visible au frame précédent.

15- [3 points] *Vaut-il mieux stocker la liste des triangles visibles ou la liste des bâtiments/arbres visibles ?*

En stockant par triangle, on aura une granularité plus fine de la visibilité que par bâtiments. Ca évitera de mettre dans le PVS les 10 000 triangles d'un arbre qui n'est visible que par le bout d'une feuille. Par contre, ca a deux inconvénients majeurs :
– ca fait des PVS beaucoup plus gros, mais on pourrait éventuellement lancer un algorithme de compression de PVS pour regrouper les triangles qui sont visibles au même moment.
– ca casse l'optimisation des maillages par triangle strip, par matériaux, par display list et donc ca n'est pas bon.

On stockera les ID des buildings visibles (ca permet en outre une meilleure compression car ca augmente la cohérence). Si on a une hiérarchie sur les buildings, on stockera les idées des noeuds dans cette hiérarchie.

16- [2 points] *Comment vaut-il mieux placer les points ?*

Il est suffisant de les placer sur le pourtour de la cellule (ce qui est visible d'un point intérieur est forcément visible d'un point de contour).
On placera les points adaptivement : en place 2 points au hasard sur un côté et on calcule ce qu'ils voient. Si ce qu'ils voient est très différent, on rajoute un point au milieu. Ca n'est pas exacte (on peut rater une petite rue en enfilade) mais c'est une heuristique raisonnable.

17- [3 points] *Quel(s) problème(s) pose cette approche ? Comment cela se manifestera-t-il ?*

Cette approche est potentiellement lente. Il ya 600 rendus par viewcell. Si il y a 2 km de route, chacune faisant 10 m de large, on aura 2000*10 viewcells. Si chaque rendu prend 30ms, le précalcul total coutera 100 heures.
Ensuite cette méthode n'est pas du tout conservative. On peut rater des rues en enfilade et des changements brusques de visibilité, notamment au niveau des carrefours. Dans l'application, cela se traduira par des apparitions soudaines de bâtiments quand on change de cellule.

18- [2 points] *Donner 2 autres critères pertinents qui pourrait être utilisés ? On pensera notamment au cas de bâtiments visibles à travers des arbres.*

Si on a utilisé de l'horizon culling ou des occlusions queries, qui peuvent renvoyer une information sur le pourcentage de visibilité, on peut utiliser cette information pour régler le niveau de détail.
Au choix on peut aussi utiliser :
– une estimation de la taille de la projection à l'écran ;
– la vitesse de déplacement de l'utilisateur ;
– l'importance de certains bâtiments ;
– l'orientation du bâtiment (un bâtiment vu de face peut utiliser un billboard plus qu'un bâtiment vu de côté)
– ...

19- [2 points] *Quels artefacts visuels sont d'après vous éventuellement observables avec notre choix de niveaux de détails ?*

C'est principalement le manque de parallax. Notamment, pour les bâtiments vus de côté, tout le relief des fenêtre est perdu. Même si l'erreur géométrique est faible, l'erreur visuelle est importante (on verra par exemple la couleur des carreaux des fenêtres)
Il y aura aussi les problèmes de popping que l'on peut atténuer en utilisant du blending.

6 Shading et image based rendering

20- [2 points] *Décrire des pseudo vertex et fragment shader qui permettent de faire ce bump mapping.*

On applique un modèle de Phong mais en obtenant la normale en un point par un texture lookup plutôt que par interpolation à partir des normales aux sommets.

21- [2 points] *Vaut-il mieux normaliser le vecteur dans le vertex shader ou dans le fragment shader ?*

Normaliser dans le fragment shader est plus coûteux : le fragment shader est exécuté beaucoup plus souvent que le vertex shader car il y a en général plus de fragment que de sommets(sauf si tout les triangles se projettent sur moins de 3 pixels). Mais on est obligé car l'interpolation (linéaire) de vecteurs normalisés ne donne pas un un vecteur normalisé.

22- ***[3 points]*** *Dans le fragment shader, quand on a cette direction, peut-on calculer l'intersection exacte avec la scène ? Même question avec une intersection approximative (indication : penser au rendu à base d'images).*

Calculer l'intersection exacte avec la scène nécessite un ray-tracing. Ce n'est pas implémentable dans un fragment shader car on ne peut pas parcourir la liste des triangles de la scène, et parce que le nombre d'instructions est limité. Par contre, on peut utiliser une représentation simplifiée de la scène qui permette d'approximer l'intersection. Le cas le plus simple est l'environment map. On peut augmenter cette envmap d'une carte de hauteur. On pourrait aussi utiliser un lightfield.

23- ***[2 points]*** *Qu'est ce que le* early Z rejection *et comment peut-on en tirer parti pour éviter des calculs inutiles ?*

C'est le fait que la carte graphique n'évalue un fragment shader que si le fragment passe le z-test. Pour en tirer parti, notre shader ne doit pas modifié le z. Or le bump et les réflections (type envmap) n'ont pas besoin d'altérer le z.

24- ***2 points*** *Qu'est ce que le* deferred shading *? Comment pourrait-on le mettre en oeuvre ?*

Le but est de ne pas évaluer un fragment shader pour des fragments qui seront *ensuite* recouvert par un fragment plus proche (c'est complémentaire du early z rejection).
Pour cela, on rend d'abord une image des paramètres utilisés par le shader. Dans notre cas, une image des normales et des attributs de matériaux. Puis on fait applique un filtre à ces images (par un deuxième rendu d'un quad couvrant l'écran et accédant ces valeurs par texture lookup) pour évaluer le shader complet.