

Construction de surface lisse : Dossier d'analyse

Les structures de données globales :

Le programme manipulera des points et des vecteurs qui sont définis par :

```
typedef {float x, y, z;} point ;
```

De plus on utilisera aussi deux variables globales :

```
point **reseau_e , **reseau_s ;
```

qui serviront aux mises à jour successives des points.

Déroulement de la fonction main :

Elle utilise un pointeur sur un fichier (fich), une chaîne de caractère (nom), un réel (epsilon) et deux entiers (N, M).

```
> nom <- entrer_utilisateur_du_nom_du_fichier_à_ouvrir ;
> fich <- ouverture_du_fichier_en_mode_lecture ;
> lecture(fich, adresse(N), adresse(M)) ;
> init_affichage(N, M) ;
> tant_que_nom_n'est_pas_«_oui_»_faire
>     epsilon <- entrer_utilisateur_de_la_sensibilité ;
>     tant_que_(moyenne_des_distances_en_deux_points_adjacents > epsilon) faire
>         catmull_clark(N, M, adresse(N), adresse(M)) ;
>         init_affichage(N, M) ;
>         nom <- entrer_utilisateur_de_«_oui_»_pour_terminer_ou_«_non_»_pour_continuer ;
> nom <- entrer_utilisateur_de_«_oui_»_ou_«_non_» ;
> si nom = «_oui_»
>     visualisation(N, M) ;
```

Les différents modules :

La liste est donnée par ordre alphabétique :

affichage : Il ne renvoie rien et prend en argument la taille (N, M), un point u et une chaîne de caractère nom. Il utilise un pointeur sur un fichier fich, deux matrices réelles Xp et Yp, deux points v et w et deux entiers i et j. Ce module crée un fichier au format gif en utilisant la bibliothèque graphique gd2.0.15 sur www.boutelle.com/gd. Le point mis en entrée est en fait le vecteur du regard.

```
> fich <- ouverture_du_fichier_nommé_«_nom_»_en_mode_«_wb_» ;
> u, w <- calcul_tel_que_(u, v, w) soit_une_base_orthonormée_directe ;
> pour i de 1 à N faire
>     pour j de 1 à M faire
>         Xp[i][j] <- produit_scalaire(v, reseau_e[i][j]) ;
>         Yp[i][j] <- produit_scalaire(w, reseau_e[i][j]) ;
```

```

> pour i de 1 à N-1 faire
>   pour j de 1 à M
>     dessin du trait entre (Xp[i][j],Yp[i][j]) et (Xp[i+1][j],Yp[i+1][j]) ;
> pour i de 1 à N faire
>   pour j de 1 à M-1 faire
>     dessin du trait entre (Xp[i][j],Yp[i][j]) et (Xp[i][j+1],Yp[i][j+1]) ;
> enregistrement du dessin dans fich ;

```

catmull_clark : Il ne renvoie rien et prend en argument la taille (N,M) du réseau de points à mettre à jour et les adresses de ces deux nombres. Il utilise deux entiers i et j qui serviront à parcourir `reseau_e` et `reseau_s`. Le module met à jour `reseau_e` en utilisant `reseau_s` comme intermédiaire.

```

> on alloue l'espace necessaire pour reseau_s : une matrice de taille (2N,2M) ;
>
> reseau_s[1][1] <- reseau_e[1][1];
> pour j de 1 à M-1 faire
>   reseau_s[adr[1][2*j] <- 3/4*reseau_e[1][j]+1/4*reseau_e[1][j+1];
>   reseau_s[adr[1][2*j+1] <- 1/4*reseau_e[1][j]+3/4*reseau_e[1][j+1];
> reseau_s[1][2*M] <- reseau_e[1][M];
>
> reseau_s[2*N][1] <- reseau_e[N][1];
> pour j de 1 à M-1 faire
>   reseau_s[2*N][2*j] <- 3/4*reseau_e[N][j]+ 1/4*reseau_e[N][j+1];
>   reseau_s[(2*N)[2*j+1] <- 1/4*reseau_e[N][j]+3/4*reseau_e[N][j+1];
> reseau_s[2*N][2*M] <- reseau_e[N][M];
>
> pour i de 1 à N-1 faire
>   reseau_s[2*i][1] <- 3/4*reseau_e[i][1]+ 1/4*reseau_e[i+1][1];
>   reseau_s[2*i+1][1] <- 1/4*reseau_e[i][1]+3/4*reseau_e[i+1][1];
>
> pour i de 1 à N-1 faire
>   reseau_s[2*i][2*M] <- 3/4*reseau_e[i][M]+1/4*reseau_e[i+1][M];
>   reseau_s[2*i+1][2*M] <- 1/4*reseau_e[i][M]+3/4*reseau_e[i+1][M];
>
> pour i de 2 à N faire
>   pour j de 2 à M faire
>     reseau_s[2*(i-1)][2*(j-1)] <- 1/16(9*reseau_e[i-1][j-1]+ 3*reseau_e[i-1][j]+
3*reseau_e[i][j-1]+reseau_e[i][j]);
>     reseau_s[2*(i-1)][2*(j-1)+1] <- 1/16(3*reseau_e[i-1][j-1]+9*reseau_e[i-1][j] +
reseau_e[i][j-1]+ 3*reseau_e[i][j]);
>     reseau_s[2*(i-1)+1][2*(j-1)] <- 1/16(3*reseau_e[i-1][j-1]+reseau_e[i-1][j] +
9*reseau_e[i][j-1]+3*reseau_e[i][j]);
>     reseau_s[2*(i-1)+1][2*(j-1)+1] <- 1/16(reseau_e[i-1][j-1]+3*reseau_e[i-1][j] +
3*reseau_e[i][j-1]+9*reseau_e[i][j]);
>
> on libère l'espace occupé par reseau_e ;
> on alloue l'espace pour reseau_e : une matrice de taille (2N,2M) ;
> pour i de 1 à 2N faire
>   pour j de 1 à 2M faire
>     reseau_e[i][j] <- reseau_s[i][j] ;
> on libère l'espace occupé par reseau_s ;
> contenu(adresse(N)) <- 2N ;
> contenu(adresse(M)) <- 2M ;

```

init_affichage : Le module prend en argument la taille de `reseau_e`. Il utilise un point `u` et une chaîne de caractère `nom`. Il va permettre de faire un affichage à la demande.

```
> u <- rentrer_utilisateur_du_vecteur_de_regard ;
> nom <- rentrer_utilisateur_du_nom_du_fichier_pour_le_sauvegarde ;
> affichage(N,M,u,nom);
```

lecture : Le module ne renvoie rien et prend en argument un nom de fichier et deux pointeurs vers des entiers. Il utilise deux entiers `i` et `j` et deux tableaux de réels `U` et `V`. Il va lire les valeurs de `N` et `M` dans le fichier et ensuite il va stocker les valeurs de $P_{i,j}$ dans `reseau_e`.

```
> lecture_de_N ;
> lecture_de_M ;
> on_alloue_l'espace_necessaire_pour_reseau_e : une_matrice_de_taille (N,M) ;
> on_alloue_l'espace_necessaire_pour_U_et_V : des_tableaux_de_taille_N_et_M ;
> pour i de 1 à N faire
>   lecture_de_la_valeur_dans_le_fichier ;
>   stockage_dans_U[i] ;
> pour j de 1 à M faire
>   lecture_de_la_valeur_dans_le_fichier ;
>   stockage_dans_V[j] ;
> pour i de 1 à N faire
>   pour j de 1 à M faire
>     lecture_de_la_valeur_dans_le_fichier ;
>     reseau_e[i][j] <- (U[i], V[j], valeur) ;
```

visualisation : Le module ne renvoie rien et prend en argument la taille de `reseau_e`. Il utilise un `u`, un réel `theta`, deux chaînes de caractères `nom` et `nom_def` et un entier `i`. Il permet d'effectuer une visualisation complète de la surface en prenant douze cliqués chacun tourné d'un angle de $\pi/6$.

```
> theta <- pi/6 ;
> u.x <- sin(pi/12) ;
> u.y <- cos(pi/12) ;
> u.z <- entrer_utilisateur_de_la_coordonnee_z_du_vecteur_de_regard ;
> nom <- entrer_utilisateur_du_nom_commune_a_tous_les_fichiers_de_sorties ;
>
> pour i de 0 à 11 faire
>   nom_def <- concatenation(nom, i, « .png ») ;
>   affichage(N,M,u,nom_def) ;
>   u <- rotation_de_u_dans_angle_theta ;
```