Nancy-Université
Université Henri Poincaré

LCIN4U73 - Bases géométriques de l'imagerie

TP 2 - SURFACES PARAMÉTRIQUES

Exercice 1 — Friandise

La surface paramétrique d'équation

$$\begin{cases} x &= ka(1+u)\cos v \\ y &= ka(1-u)\sin v \\ z &= au \end{cases} \quad (1)$$

où u et v sont les abscisses curvilignes variant de $-\pi$ à π, possède deux paramètres, k et a, qui contrôlent le comportement de la courbe. Le but de cet exercice est d'écrire une fonction `tracerAvecParametres(a,k)`, qui trace la surface pour des valeurs de a et k données. Vous devrez utiliser la fonction `eval3dp` (consulter l'aide de Scilab). Étudiez le comportement de la courbe pour différentes combinaisons de valeurs de a et de k.

Exercice 2 — Casteljau

Nous allons utiliser l'algorithme de Casteljau pour tracer des courbes de Bézier dont les points de contrôle sont définis à la souris.

1. Écrire la fonction `[x]=clickPolyLine()` qui :
   (a) efface le contenu de la fenêtre courante et dessine une figure vide dans un carré de taille 1 (`xbasc() ; plot2d(0,0,rect=[0,0,1,1]) ;`),
   (b) attend les clicks gauche de la souris (help `xclick`) et relie les points cliqués par des segments
   (c) au click droit, retourne la matrice `x` qui contient les coordonnées des points cliqués (n lignes, 2 colonnes).
2. L'algorithme de Casteljau vu en TD applique le principe de subdivision en identifiant les milieux des segments formés par $n = 4$ points de contrôle d'une courbe de Bézier, puis les milieux des milieux etc. jusqu'à obtenir un seul point dont on a montré qu'il est un point de la courbe d'abscisse curviligne $t = 1/2$.
   Ce résultat se généralise en fait à un nombre quelconque $n \geq 2$ de points de contrôle et $t \in [0, 1]$, c'est-à-dire qu'en calculant les barycentres de paramètres $\{t, 1-t\}$ des points de contrôle consécutifs de la courbe, puis les barycentres de même paramètres de ces barycentres et ainsi de suite itérativement, on définit de cette manière une suite de listes de points que nous indexons P_i^j, où P_i^j est le barycentre de $\{P_i^{j-1}, P_{i+1}^{j-1}\}$. La dernière liste ne contient qu'un seul point, qui est le point de la courbe d'abscisse curviligne t.
   En pseudo-code, cela donne :

```
Pour j de 1 à n-1 faire
|
| Pour i de 0 à n-1-j faire
| |
| | P(i) = t*P(i+1)+(1-t)*P(i)
| |
|
Retourner P(0)
```

   En utilisant cet algorithme (sans oublier que les indices commencent à 1 en scilab...), écrire la fonction `[b]=deCasteljau(x,t)` qui, à partir d'une matrice `x` de points de contrôles (n lignes, 2 colonnes) retourne le point d'abscisse curviligne `t` de la courbe de Bézier définie par ces points de contrôle.
3. Écrire la fonction `BezierInteractif()` qui
   (a) appelle `x=clickPolyLine()` pour permettre à l'utilisateur de définir des points de contrôle,
   (b) appelle `b(i, :)=deCasteljau(x,t)` pour 100 valeurs de `t` comprises entre 0 et 1,
   (c) trace la courbe de Bézier qui relie les points `b(i, :)`.
4. Réécrire la fonction `[b]=deCasteljau(x,t)` de manière à n'avoir qu'une seule boucle.