



R my Malgouyres,
Algorithmes pour la synth se d'images
et l'animation 3D,
DUNOD, 2002

TP n  8 : Surfaces de B zier

Objectifs :

Le but de ce TP est de construire une hi rarchie de classes pour repr senter les surfaces. Pour le moment, la hi rarchie ne comprendra que la classe surface de base, et les surfaces de B zier d'ordre $n \times m$ avec calcul des points par l'algorithme de Casteljau.

1 La classe Surface

La classe `surface` est la classe de base dont d riveront toutes les classes de surfaces. Elle contient le tableau (  deux dimensions) des points de contr le comme donn e prot g e.

```
class Surface {
protected:
    Point3D **ctrlpoints;
    int nblignes;
    int nbcols;
public:
    Surface();
    virtual ~Surface();

    virtual void getpoint(double s, double t,
                          double & xpos, double & ypos, double & zpos)=0;

    void affiche(CDC &cdc);
};
```

La m thode virtuelle `getpoint`, dont l'impl mentation d pendra du type de surface consid r , retourne le point $\sigma(s, t)$ de la surface, avec $s, t \in [0, 1]$.

La fonction `affiche` utilise la fonction `getpoint` pour r aliser un affichage filiforme de la surface.

Pour la fonction `affiche`, on pourra consid rer des valeurs de param tre s_i et t_j , pour $i, j = 0, \dots, \text{Echantillonage}$, r guli rement espac s sur l'intervalle $[0, 1]$. On dessinera les segments de droites $[\sigma(s_i, t_j), \sigma(s_{i+1}, t_j)]$ et $[\sigma(s_i, t_j), \sigma(s_i, t_{j+1})]$. Par ailleurs, on fera une projection en perspective en supposant que l'observateur est plac  en $O = (0, 0, 0)$, regarde vers l'axe des z , et que l'on projette les points sur le plan d' quation $z = d$ avec $d = 500$.

2 La classe SurfaceBezierCastel

La classe `SurfaceBezierCastel` représente les surfaces de Bézier avec algorithme de Casteljau.

```
class SurfaceBezierCastel : public Surface{  
  
public:  
  
    SurfaceBezierCastel(void);  
    ~SurfaceBezierCastel();  
  
    SurfaceBezierCastel(ifstream &fich);  
    void getpoint(double s, double t,  
                  double & xpos, double & ypos, double & zpos);  
};
```

Dans la fonction `getpoint`, on implémentera l'algorithme de Casteljau pour calculer $\sigma(s, t)$ à partir des points de contrôle.

Le constructeur à partir d'un fichier permet de lire un fichier texte contenant dans l'ordre :

- Le nombre de lignes, puis le nombre de colonnes de la matrice des points de contrôle ;
- Pour $i = 0, \dots, nblignes$ et pour $j = 0, \dots, nbcols$ les coordonnées x , puis y , puis z du point de contrôle $P_{i,j}$.

Des exemples de fichiers seront fournis.