

Modélisation géométrique 2

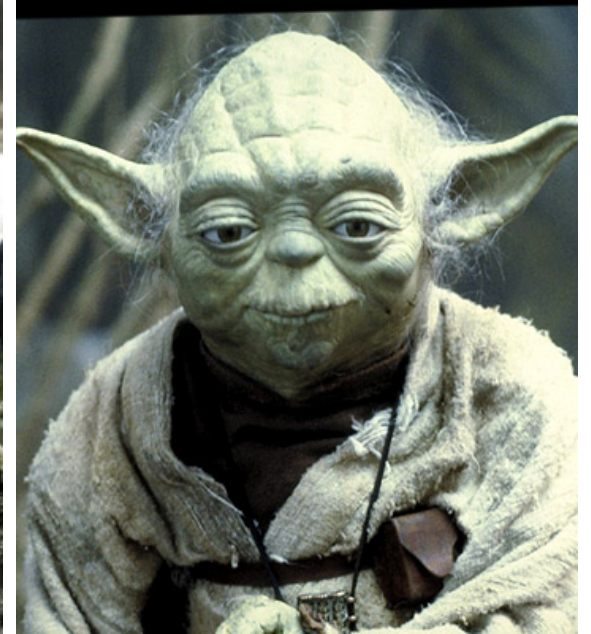


Sylvain Brandel

M1if27 – 5 mars 2021

- Qu'est ce qu'on a envie d'afficher
 - Courbes et surfaces paramétriques et courbes et surfaces de subdivision
 - Surfaces implicites / CSG

Pourquoi ?



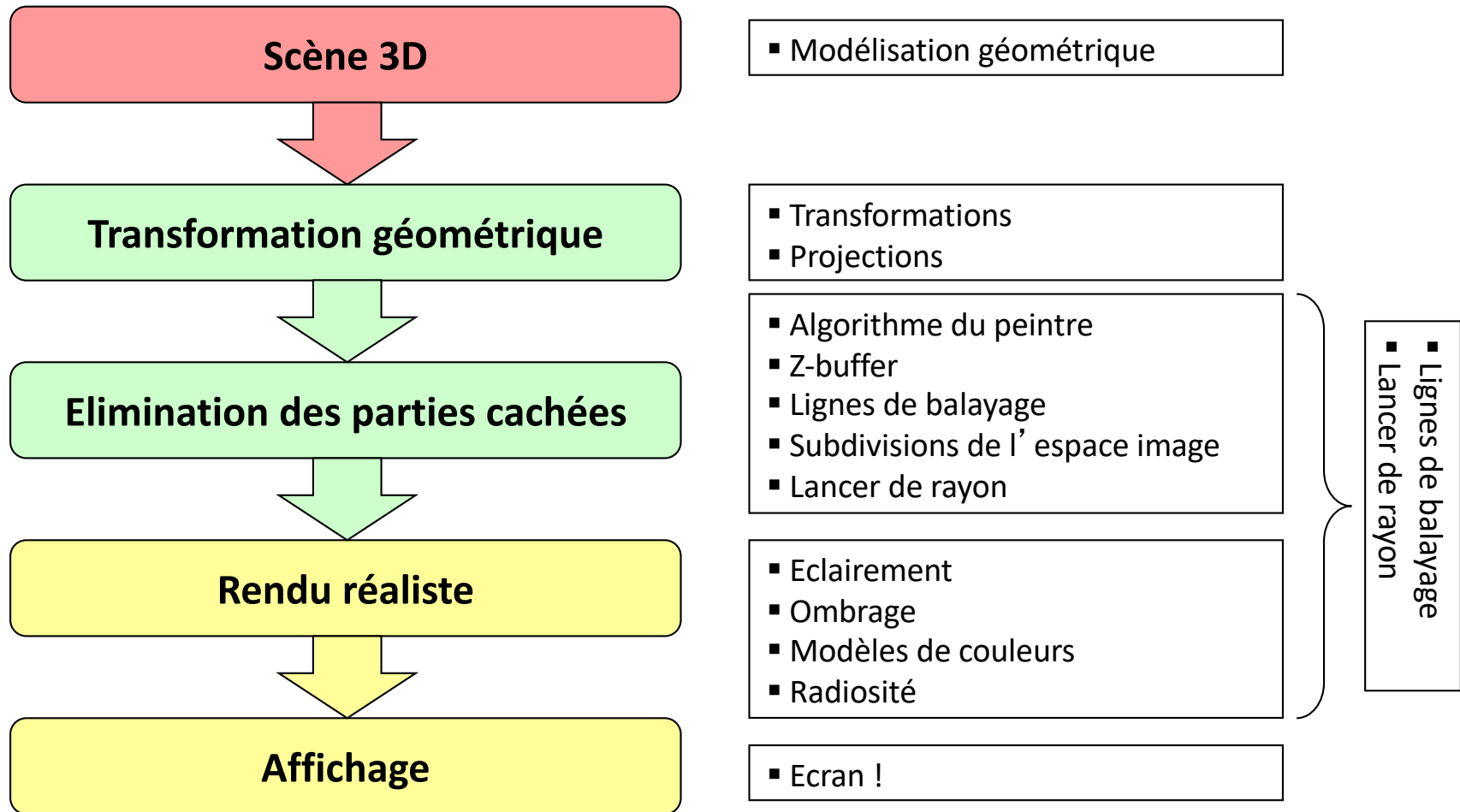
Pourquoi ?



Un tout petit peu d'histoire

- Modélisation surfacique (années 1960 – 70)
 - De Casteljau (Citroën)
 - Bézier (Renault)
 - Coons (General Motors)
 - Ferguson (Boeing)
- Modélisation volumique (années 1980)
 - Réquicha
 - Mäntylä
 - Hoffmann

Chaine de visualisation

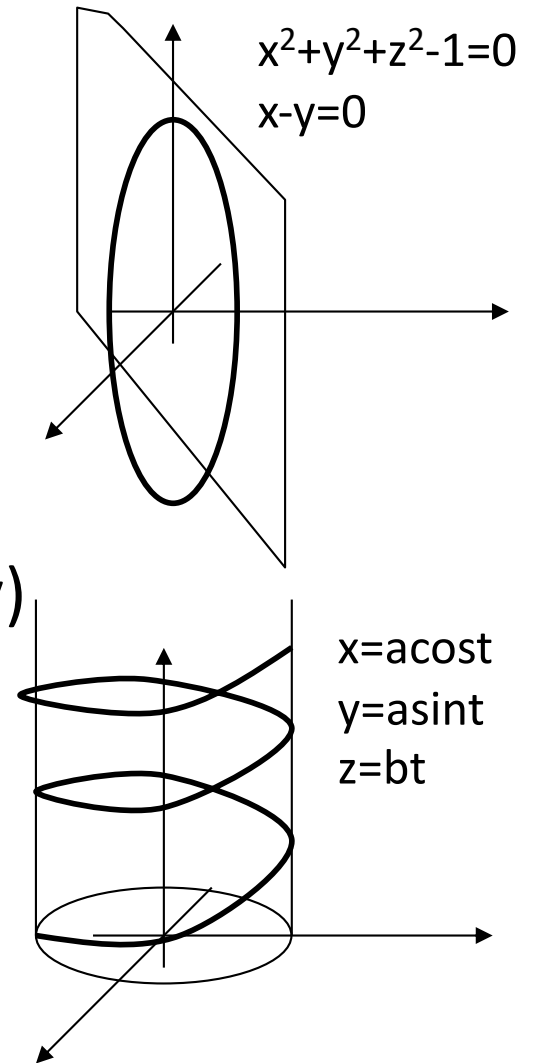


Première partie

Courbes et surfaces paramétriques

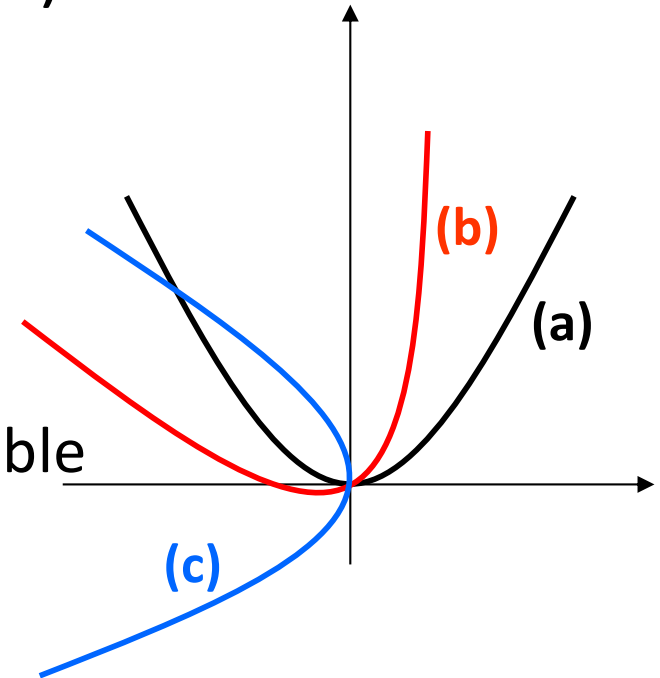
Représentation des courbes et surfaces

- Représentation implicite
 - Courbes : $f(x,y,z) = 0$ et $g(x,y,z) = 0$
 - Surfaces : $f(x,y,z) = 0$
- Représentation paramétrique
 - Courbes : $x = f(t)$, $y = g(t)$, $z = h(t)$
 - Surfaces : $x = f(u,v)$, $y = g(u,v)$, $z = h(u,v)$
 - Courbe cubique :
 - $x = axt^3 + bxt^2 + cxt + dx$
 - $y = ayt^3 + byt^2 + cyt + dy$
 - $z = azt^3 + bzt^2 + czt + dz$



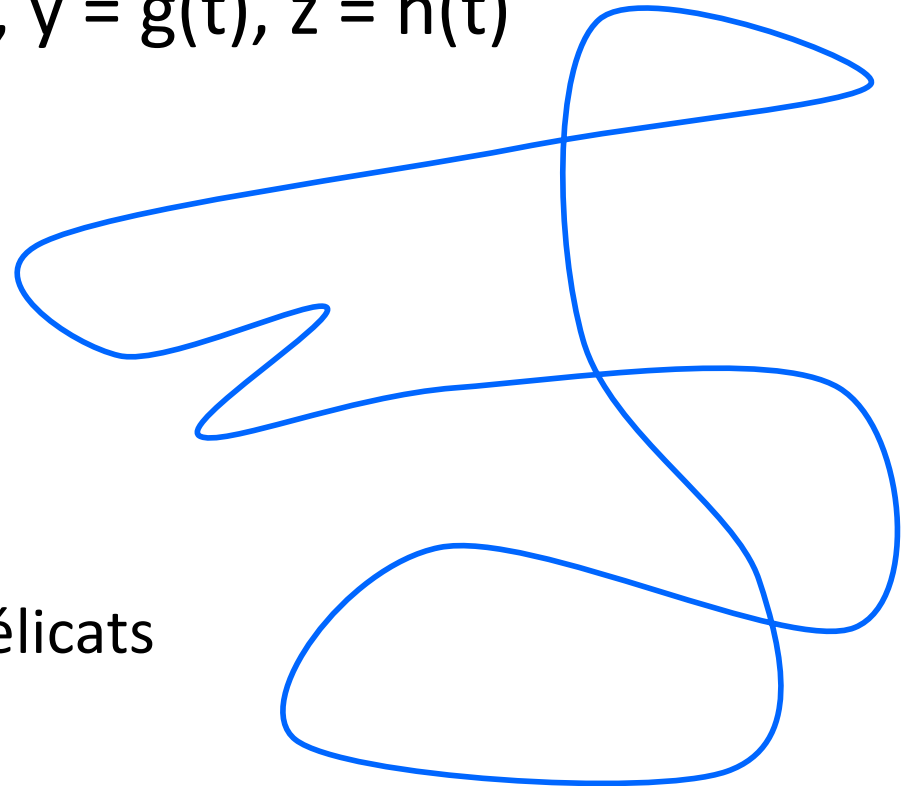
Représentation des courbes

- Représentation cartésienne
 - Courbes planes : $y = f(x)$
 - Courbes gauches : $y = f(x), z = g(x)$
 - Avantages :
 - simple
 - accès direct à y et z connaissant x
 - Inconvénients :
 - tangente verticale non représentable
 - rotation altère la courbe (b)
 - (c) non représentable



Représentation des courbes

- Représentation paramétrique
 - Courbes planes : $x = f(t)$, $y = g(t)$
 - Courbes gauches : $x = f(t)$, $y = g(t)$, $z = h(t)$
 - Avantages :
 - liberté sur les tangentes
 - forme quelconque
 - plein de propriétés
 - Inconvénients :
 - calculs parfois lourds ou délicats

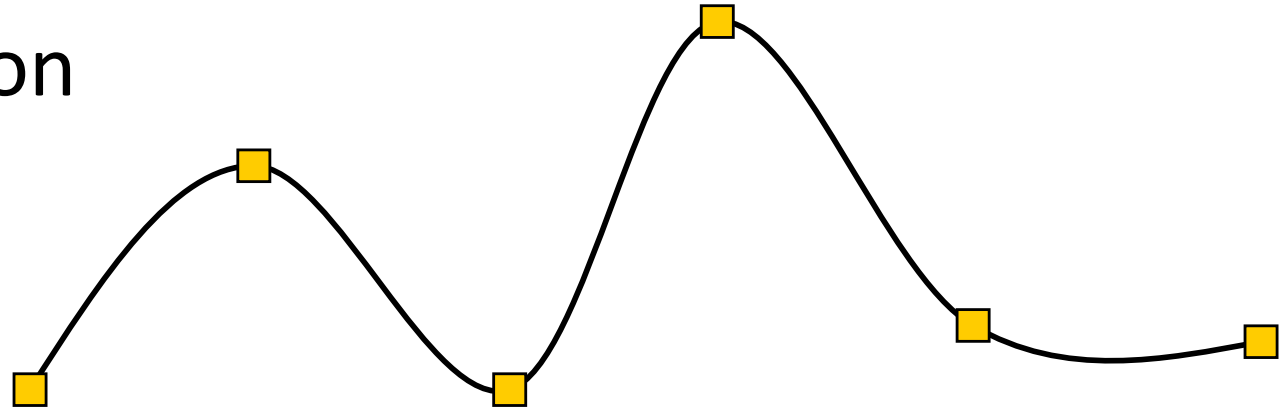


Historique

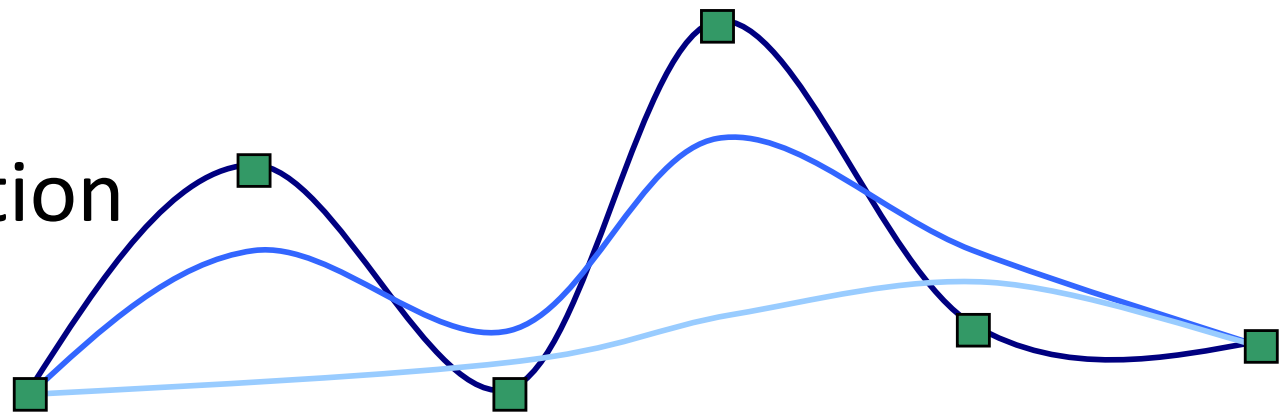
- Grands noms (années 60 – 70) :
 - Bézier (Renault)
 - De Casteljau (Citroën)
 - Coons (General Motors)
 - Ferguson (Boeing)
 - De Boor ← B-Splines
- Grands systèmes de CFAO (années 80) :
 - Euclid
 - Catia
- Fin du duel Bézier – B-Splines : 1990
- Problème de reconstruction : 1990
- Surfaces de subdivisions : 1995

Interpolation vs approximation

- Interpolation

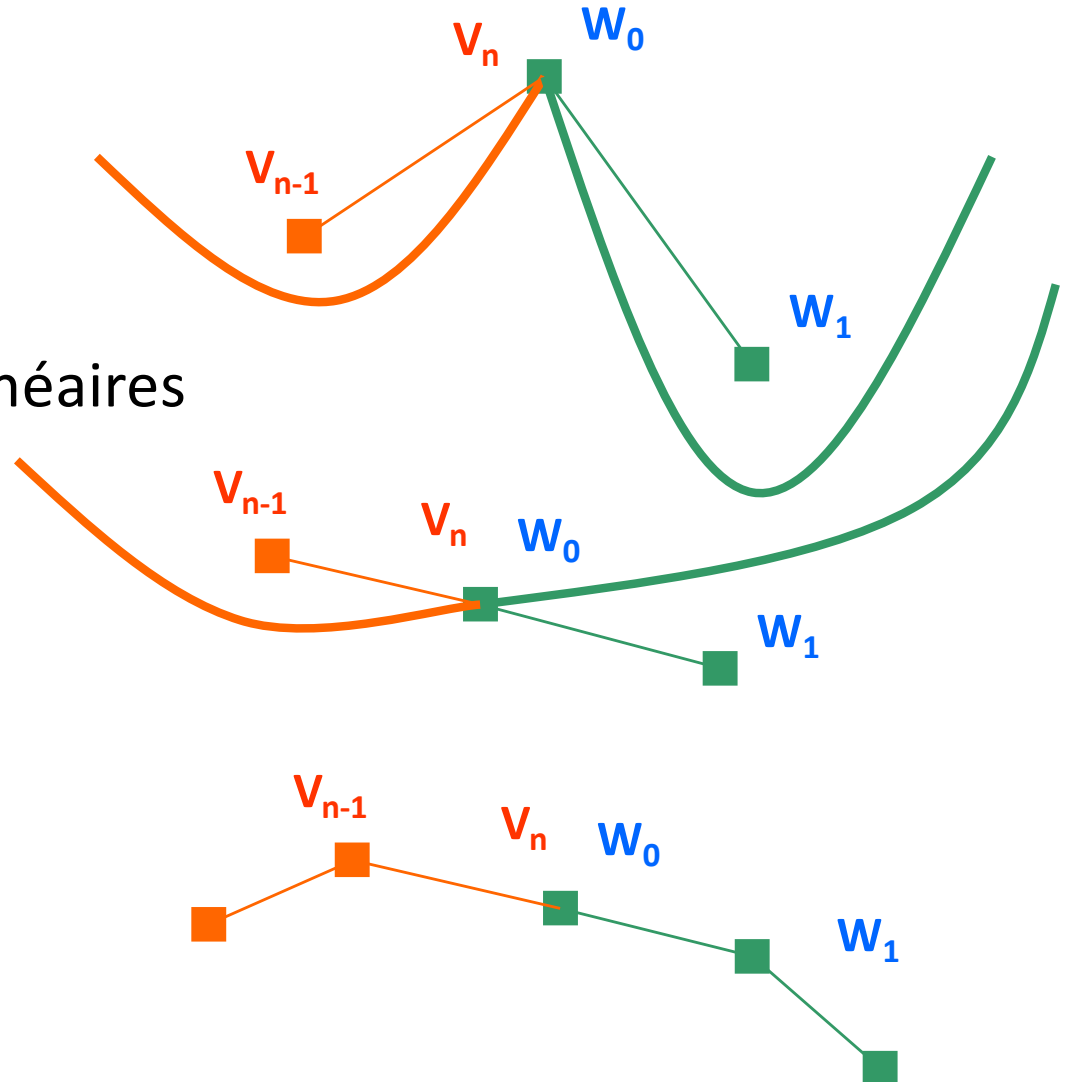


- Approximation



Continuité

- Continuité de position
 - $G^0 : V_n = W_0$
 - C^0 : pareil
- Continuité de tangence
 - $G^1 : V_{n-1}V_n$ et W_0W_1 colinéaires
 - C^1 : même norme
- Continuité de courbure
 - G^2
 - C^2
- Continuité de torsion
 - G^3
 - C^3



Interpolation de points

Interpolation polynomiale

- L'interpolation polynomiale est à la base de tous les concepts d'interpolation
- Soient $P_0, \dots, P_n$ et $t_0, \dots, t_n$ les valeurs du paramètre correspondant.
 - Interpolier consiste à trouver une courbe $P(t)$ passant par tous les points
 - On cherche un polynôme satisfaisant les contraintes d'interpolation : $P(t_i) = P_i$ ($i=0..n$)
- Théorème :

– Il existe un unique polynôme P de degré $\leq n$ tel que $P(t_i) = P_i$ ($i=0..n$) avec P_i et t_i fixés

Polynômes de Lagrange

- Forme explicite du polynôme d'interpolation P :

$$P(t) = \sum_{i=0}^n P_i L_i^n(t)$$

L_i^n : polynômes de Lagrange

avec $\sum_{i=0}^n L_i^n(t) = 1$

$$L_i^n(t) = \frac{\prod_{j=0, j \neq i}^n (t - t_j)}{\prod_{j=0, j \neq i}^n (t_i - t_j)}$$

$$L_i^n(t_j) = \delta_{i,j} = \begin{cases} 1 & \text{pour } i = j \\ 0 & \text{sinon} \end{cases}$$

Approche de Vandermonde

- On utilise la base monômiale

$$\begin{cases} P(t_0) = P_0 = a_0 + a_1 t_0 + \dots + a_n t_0^n \\ \dots \\ P(t_n) = P_n = a_0 + a_1 t_n + \dots + a_n t_n^n \end{cases}$$

$$P(t) = \sum_{i=0}^n a_i t^i$$

n+1 équations

$$\begin{bmatrix} P_0 \\ \vdots \\ P_n \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & \dots & t_0^n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & t_n & t_n^2 & \dots & t_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$

ou encore

$$P = TA \rightarrow A = T^{-1}P$$

Algorithme d'Aitkem

- Interpolation de deux points A et B

$$t \in [0,1]: \quad P(t) = (1-t)A + tB$$

$$u \in [a,b]: \quad P(u) = \frac{b-u}{b-a} A + \frac{u-a}{b-a} B$$

Algorithme d'Aitkem

- Evaluation récursive

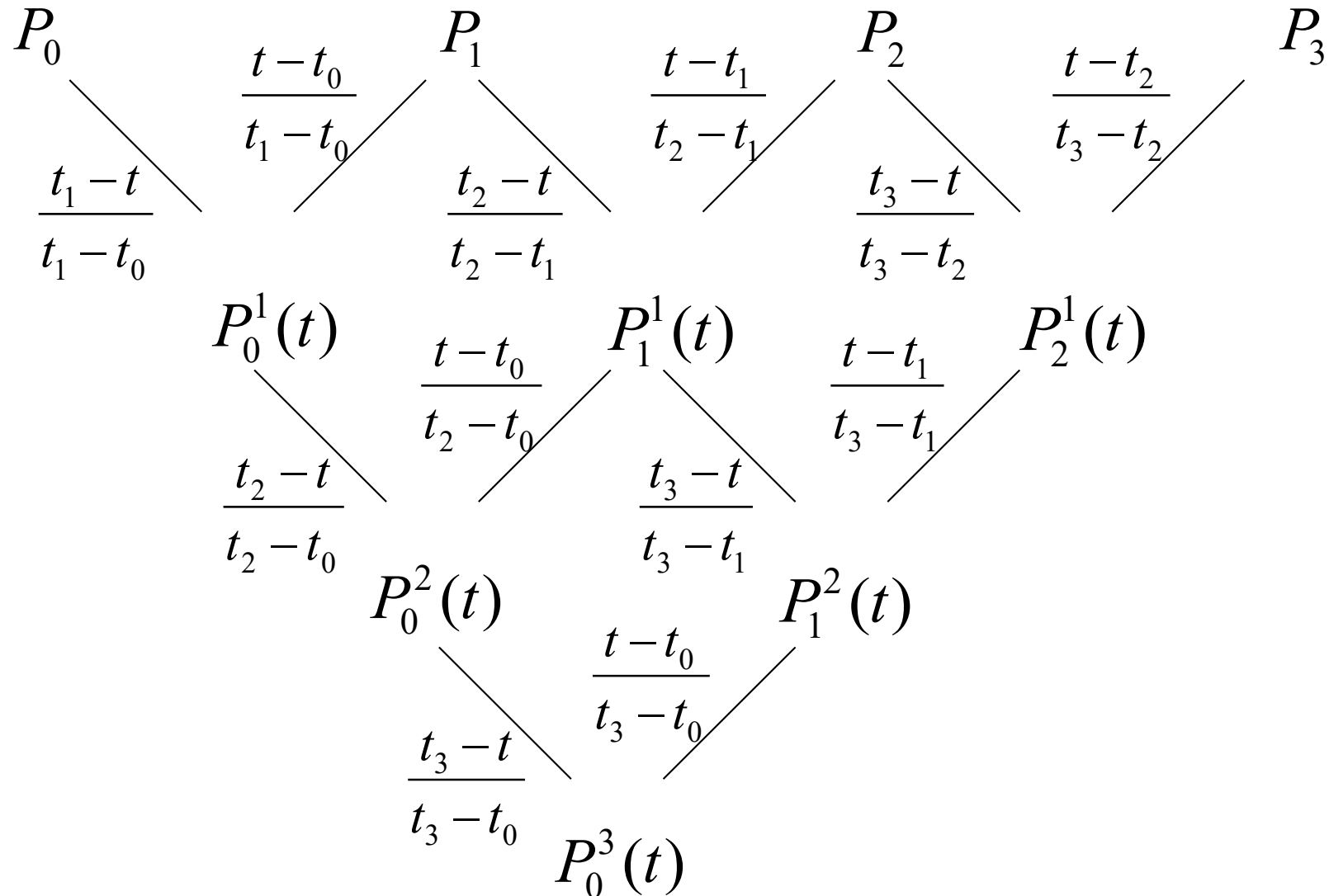
$$P_i^1(t) = \frac{t_{i+1} - t}{t_{i+1} - t_i} P_i + \frac{t - t_i}{t_{i+1} - t_i} P_{i+1} \quad i = 0 \dots n-1$$

$$P_0^{n-1}(t) \text{ interpole } P_0, P_1, \dots, P_{n-1} \quad P_0^{n-1}(t_i) = P_1^{n-1}(t_i) = P_i$$

$$P_1^{n-1}(t) \text{ interpole } P_1, P_2, \dots, P_n$$

$$\text{alors } P_0^n(t) = \frac{t_n - t}{t_n - t_0} P_0^{n-1}(t) + \frac{t - t_0}{t_n - t_{0i}} P_1^{n-1}(t) \quad P_0^n(t_i) = P_i$$

Algorithme d'Aitkem



Algorithme d'Aitkem

- Algorithme :

$$P_i^r(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i} P_i^{r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i} P_{i+1}^{r-1}(t) \quad \begin{array}{l} r = 1 \dots n \\ i = 0 \dots n - r \end{array}$$

Interpolation polynomiale

- Propriétés
 - Précision linéaire
 - Invariance par transformation affine
 - La courbe $P(t)$ n'est pas contenue dans l'enveloppe convexe des points interpolés
 - La propriété de diminution des variations de la courbe par rapport au polygone des points P_i n'est pas satisfaite
- Limitations
 - Interprétation géométrique non satisfaisante

Interpolation de points

- Interpolation de points et d'autres informations :
 - On ajoute les dérivées (tangentes)...
 - Soient les points P_0 , P_1 et les vecteurs tangents D_0 et D_1 . Trouver une courbe polynomiale cubique $P(t)$ interpolant ces données
 - Interpolants cubique et quintique d'Hermite
- Interpolation polynomiale par morceaux
- Splines
 - signifie « latte » (coques de bateaux)
 - la latte prend une position qui minimise la torsion

Courbes de Bézier

Polynômes de Bernstein

- Le $i^{\text{ème}}$ polynôme de Bernstein de degré n est :

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \text{avec} \quad \binom{n}{i} = \begin{cases} C_n^i = \frac{n!}{i!(n-i)!} & \text{si } 0 \leq i \leq n \\ 0 & \text{sinon} \end{cases}$$

- Exemple le plus courant : $n=3$

$$\begin{cases} B_0^3(t) = (1-t)^3 \\ B_1^3(t) = 3t(1-t)^2 \\ B_2^3(t) = 3t^2(1-t) \\ B_3^3(t) = t^3 \end{cases}$$

Polynômes de Bernstein

- Propriétés

- Stabilité :

- base optimale pour effectuer les principaux calculs
 - passage Bernstein – Base canonique ($P_n(t) = t^n$) connu

- Valeurs des polynômes

$$\begin{cases} B_0^n(0) = 1 \\ B_n^n(1) = 1 \end{cases} \quad \text{de plus :} \quad \begin{cases} \forall i \neq 0 \ B_i^n(0) = 0 \\ \forall i \neq n \ B_i^n(1) = 0 \end{cases}$$

- Positivité

$$\forall t \in [0,1] \ B_i^n(t) \geq 0$$

$$\forall t \in]0,1[\ B_i^n(t) > 0$$

Polynômes de Bernstein

- Propriétés

- Symétrie

$$\forall t \in [0,1] B_i^n(t) = B_{n-i}^n(1-t)$$

- Partition de l'unité

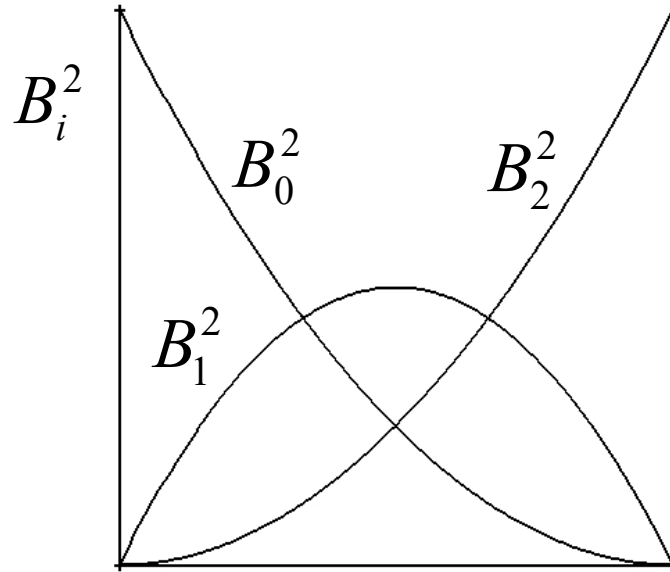
$$\forall t \in [0,1] \sum_{i=0}^n B_i^n(t) = 1$$

- Relation de récurrence

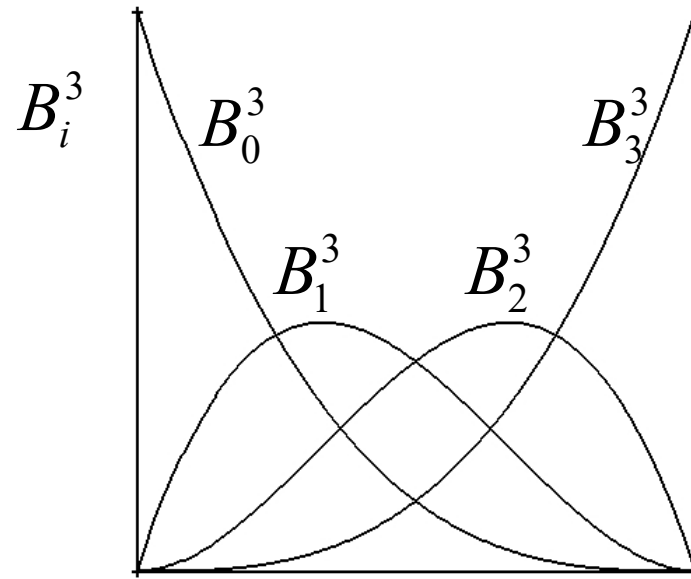
$$\begin{cases} B_0^0(t) = 1 \\ B_i^n(t) = (1-t)B_i^{n-1}(t) + t B_{i-1}^{n-1}(t) \end{cases} \quad \text{avec } B_j^n(t) = 0 \text{ pour } j \notin \{0, \dots, n\}$$

Polynômes de Bernstein

- Exemples

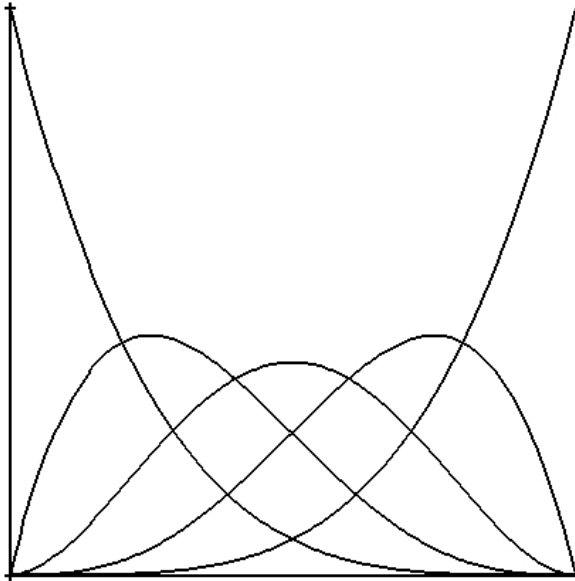


$$\begin{cases} B_0^2(t) = t^2 - 2t + 1 \\ B_1^2(t) = -2t^2 + 2t \\ B_2^2(t) = t^2 \end{cases}$$

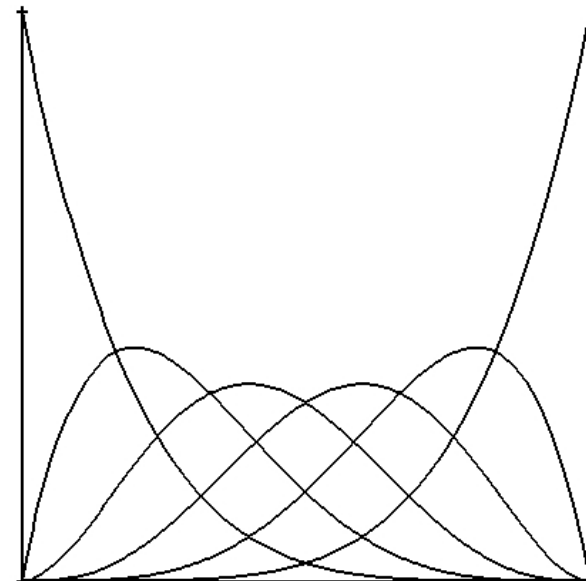


$$\begin{cases} B_0^3(t) = -t^3 + 3t^2 - 3t + 1 \\ B_1^3(t) = 3t^3 - 6t^2 + 3t \\ B_2^3(t) = -3t^3 + 3t^2 \\ B_3^3(t) = t^3 \end{cases}$$

Polynômes de Bernstein



$$\begin{cases} B_0^4(t) = t^4 - 4t^3 + 6t^2 - 4t + 1 \\ B_1^4(t) = -4t^4 + 12t^3 - 12t^2 + 4t \\ B_2^4(t) = 6t^4 - 12t^3 + 6t^2 \\ B_3^4(t) = -4t^4 + 4t^3 \\ B_4^4(t) = t^4 \end{cases}$$



$$\begin{cases} B_0^5(t) = -t^5 + 5t^4 - 10t^3 + 10t^2 - 5t + 1 \\ B_1^5(t) = 5t^5 - 20t^4 + 30t^3 - 20t^2 + 5t \\ B_2^5(t) = -10t^5 + 30t^4 - 30t^3 + 10t^2 \\ B_3^5(t) = 10t^5 - 20t^4 + 10t^3 \\ B_4^5(t) = -5t^5 + 5t^4 \\ B_5^5(t) = t^5 \end{cases}$$

Courbes de Bézier

- Définition

- Soient $(n+1)$ points P_i de $\mathbf{R}^2$ ou $\mathbf{R}^3$
- La courbe de Bézier associée à ces points est la courbe obtenue en prenant comme fonctions de poids les polynômes de Bernstein de degré n

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

- Les points P_i sont appelés points de contrôle
- La ligne polygonale $P_0P_1\dots P_n$ est appelée polygone caractéristique de la courbe, ou polygone de contrôle

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Exemples de courbes

- Linéaires (degré 1), deux points de contrôle
 - $P(t) = (1-t)P_0 + tP_1, t \in [0,1]$
- Quadratique (degré 2), trois points de contrôle
 - $P(t) = (1-t)^2P_0 + 2t(1-t)P_1 + t^2P_2, t \in [0,1]$
- Cubique (degré 3), quatre points de contrôle
 - $P(t) = (1-t)^3P_0 + 3t(1-t)^2P_1 + 3t^2(1-t)P_2 + t^3P_3, t \in [0,1]$
- Courbes de Bézier de degré supérieur à 3 :
 - rarement utilisées
 - raccordement de courbes cubiques
 - chevauchement de points de contrôle pour assurer la continuité

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Forme matricielle

$$P(t) = [B_0^n(t) \ B_1^n(t) \ \cdots \ B_n^n(t)] \begin{bmatrix} P_0 \\ \vdots \\ P_n \end{bmatrix}$$

$$P(t) = [t^n \ t^{n-1} \ \cdots \ 1] \begin{bmatrix} m_{0,0} & \cdots & m_{0,n} \\ \vdots & & \vdots \\ m_{n,0} & \cdots & m_{n,n} \end{bmatrix} \begin{bmatrix} P_0 \\ \vdots \\ P_n \end{bmatrix}$$

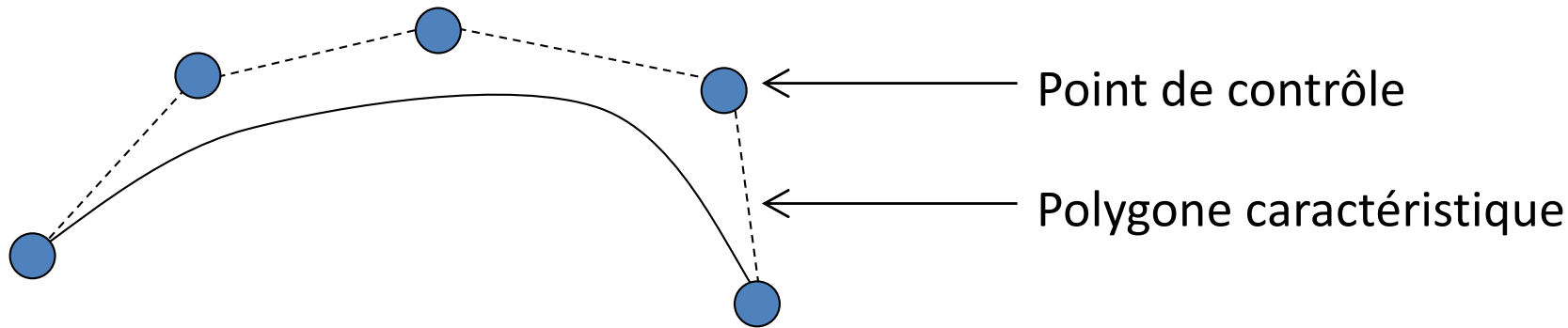
$$n=3: \quad M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{cases} B_0^3(t) = -t^3 + 3t^2 - 3t + 1 \\ B_1^3(t) = 3t^3 - 6t^2 + 3t \\ B_2^3(t) = -3t^3 + 3t^2 \\ B_3^3(t) = t^3 \end{cases}$$

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Caractéristiques

- Les points P_i contrôlent la forme de la courbe



- La courbe est obtenue par pondération des points de contrôles. Les points sont ordonnés.
- La courbe P est un polynôme de degré n exactement
- Autres fonctions de base $\rightarrow$ autre méthode

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Propriétés

- Précision linéaire :
 - Si les points de contrôle P_i sont alignés, alors la courbe de Bézier est sur un segment de droite
- La relation entre les points de contrôle est la courbe est invariante par transformation affine
- La relation entre les points de contrôle est la courbe n'est pas invariante par transformation projective

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

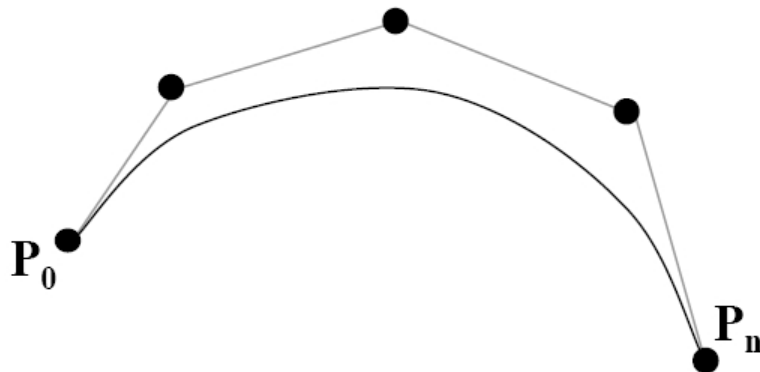
Propriétés

- Invariance par transformation affine de t :
 – $0 \leq t \leq 1 \rightarrow a \leq u \leq b$: $\sum_{i=0}^n P_i B_i^n(t) = \sum_{i=0}^n P_i B_i^n\left(\frac{u-a}{b-a}\right)$

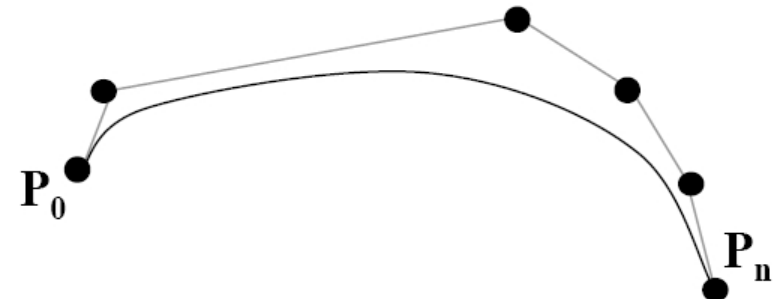
– Problème simplifié pour les Bézier

– Importance du paramétrage :

Courbe « bien » paramétrée



Courbe « mal » paramétrée



$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Propriétés

- La construction de la courbe de Bézier à partir du polygone de contrôle est invariante par combinaison barycentrique :

$$\alpha \sum_{i=0}^n P_i B_i^n(t) + \beta \sum_{i=0}^n V_i B_i^n(t) = \sum_{i=0}^n (\alpha P_i + \beta V_i) B_i^n(t)$$

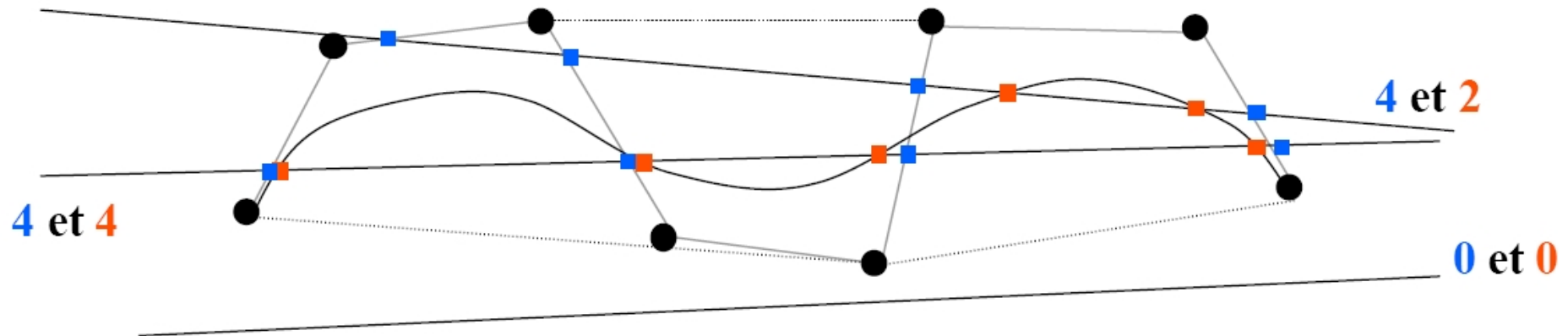
- La courbe de Bézier est contenue dans l'enveloppe convexe des points de contrôle
- Symétrie :
 - $P_0 \dots P_n$ visuellement identique à $P_n \dots P_0$

$$\sum_{i=0}^n P_i B_i^n(t) = \sum_{i=0}^n P_{n-i} B_i^n(1-t) \quad \text{car } B_i^n(t) = B_{n-i}^n(1-t)$$

$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

Propriétés

- La propriété de diminution des variations de la courbe par rapport au polygone de contrôle est satisfaite



$$P(t) = \sum_{i=0}^n P_i B_i^n(t) \quad \forall t \in [0,1]$$

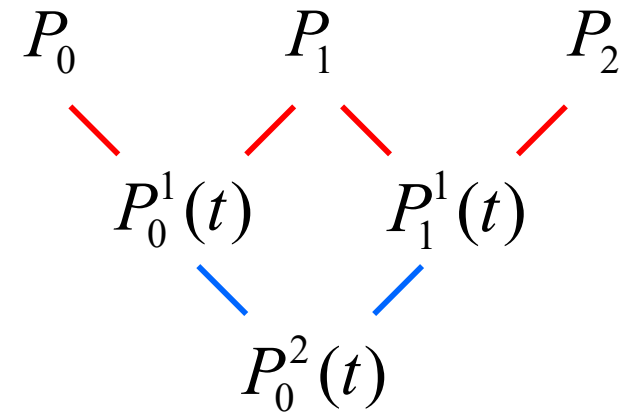
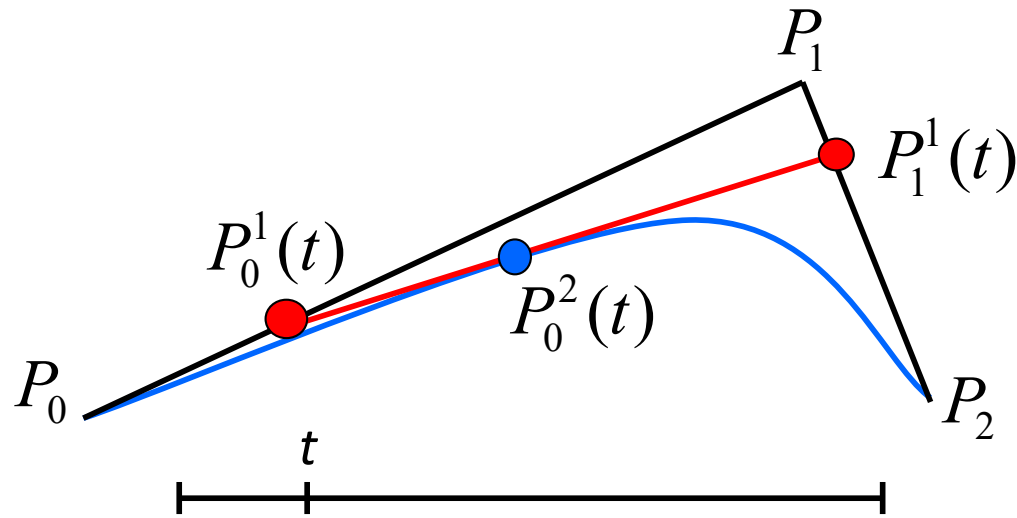
Propriétés

- Condition aux limites :
 - La courbe de Bézier passe par P_0 et P_n : $P(0) = P_0$
 - Elle est tangente à $\overrightarrow{P_0 P_1}$ en P_0 et à $\overrightarrow{P_n P_{n-1}}$ en P_n
- Continuité :
 - Une courbe de Bézier de degré n est C^{n-1} continue
- Contrôle global
- La dérivée $n+1^{\text{ème}}$ d' une courbe de Bézier de degré n est nulle
 - conséquence : une courbe de Bézier de degré 2 est forcément plane

Calcul d'un point

- Méthode à proscrire :
 - Calcul direct des polynômes de Bernstein pour tout t
- Construction d'une parabole à partir de 3 points

$$\begin{aligned}
 P_0^1(t) &= (1-t)P_0 + tP_1 & P_0^2(t) &= (1-t)P_0^1(t) + tP_1^1(t) \\
 P_1^1(t) &= (1-t)P_1 + tP_2 & &= (1-t)^2 P_0 + 2t(1-t)P_1 + t^2 P_2
 \end{aligned}$$

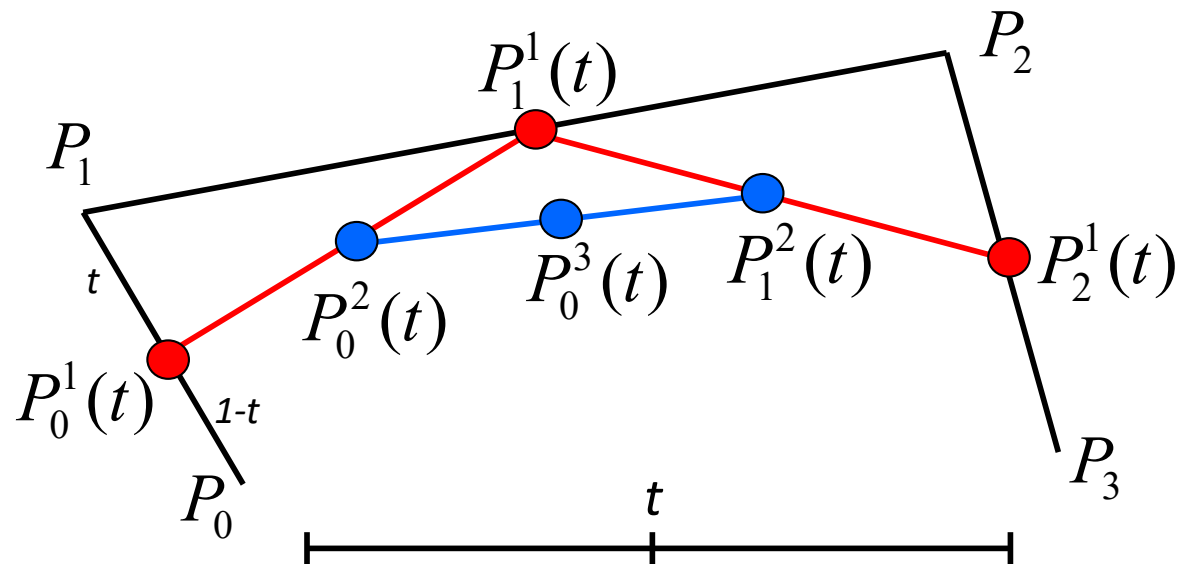


Calcul d'un point

- Algorithme de De Casteljau

$$\begin{cases} P_i^0(t) = P_i, i = 0..n \\ P_i^r(t) = (1-t)P_i^{r-1} + tP_{i+1}^{r-1}(t), r = 1..n, i = 0..n-r \end{cases}$$

- Alors $P_0^n(t)$ est le point de paramètre t sur la courbe de Bézier $P(t)$ de degré n



Algorithme de De Casteljau

$$\begin{array}{cccc}
 P_0 = P_0^0(t) & P_1 = P_1^0(t) & P_2 = P_2^0(t) & P_3 = P_3^0(t) \\
 \swarrow 1-t & \nearrow t \quad \searrow 1-t & \nearrow t \quad \searrow 1-t & \nearrow t
 \end{array}$$

$$\begin{array}{ccc}
 P_0^1(t) & P_1^1(t) & P_2^1(t) \\
 \swarrow 1-t & \nearrow t \quad \searrow 1-t & \nearrow t
 \end{array}$$

$$\begin{array}{cc}
 P_0^2(t) & P_1^2(t) \\
 \swarrow 1-t & \nearrow t
 \end{array}$$

$$P_0^3(t)$$

Algorithme de De Casteljau

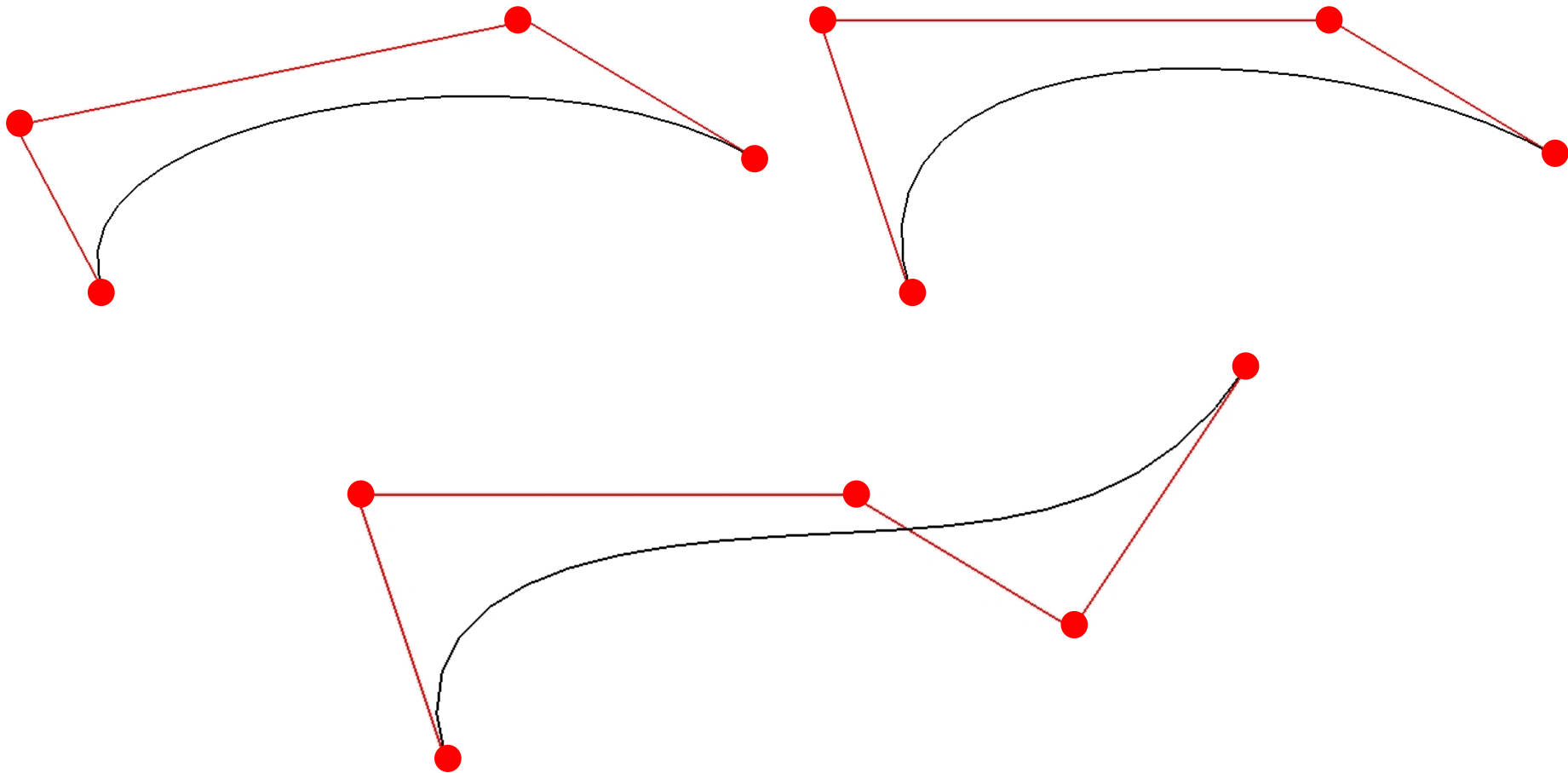
```
début
  pour i de 0 à n
    |
    |  $P_i^0 = P_i$ 
  finpour
  pour r de 1 à n
    |
    | pour i de 0 à n-r
      |
      |  $P_i^r = (1-t)P_i^{r-1} + tP_{i+1}^{r-1}$ 
    finpour
  finpour
fin
```

- Formulation récursive à proscrire
- Complexité :
 - $O(n^2)$ (n : degré) pour chaque valeur de t
 - Couteux mais stable
- Convergence :
 - Augmentation du nombre de points de contrôle
 - Quand le nombre de points de contrôle augmente, la courbe et le polygone de contrôle tendent à se confondre (si points « bien » répartis)

A la fin, $P(t) = P_0^n$

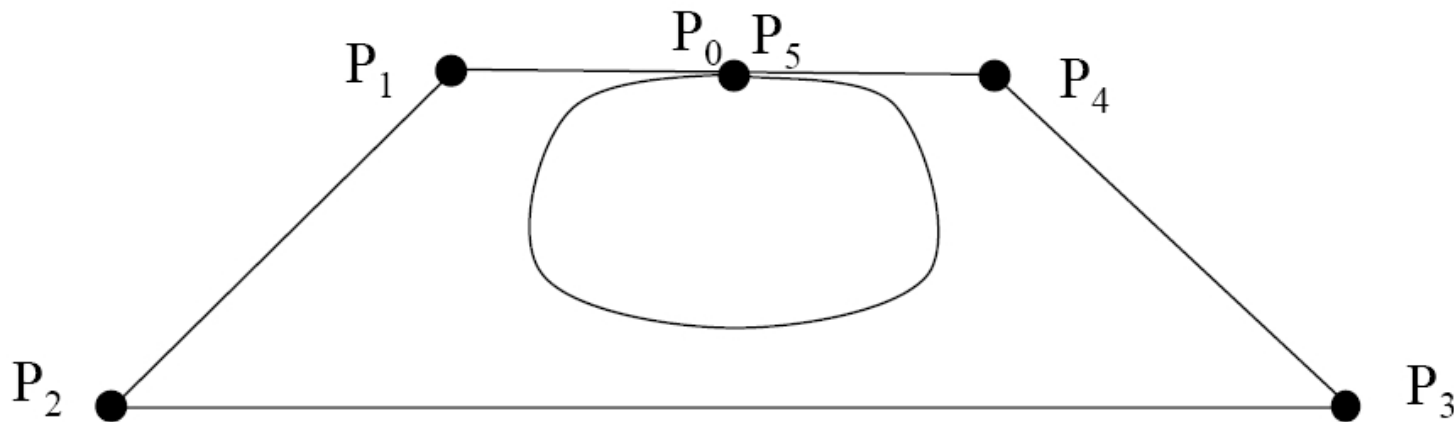
Algorithme de De Casteljau

- Exemples



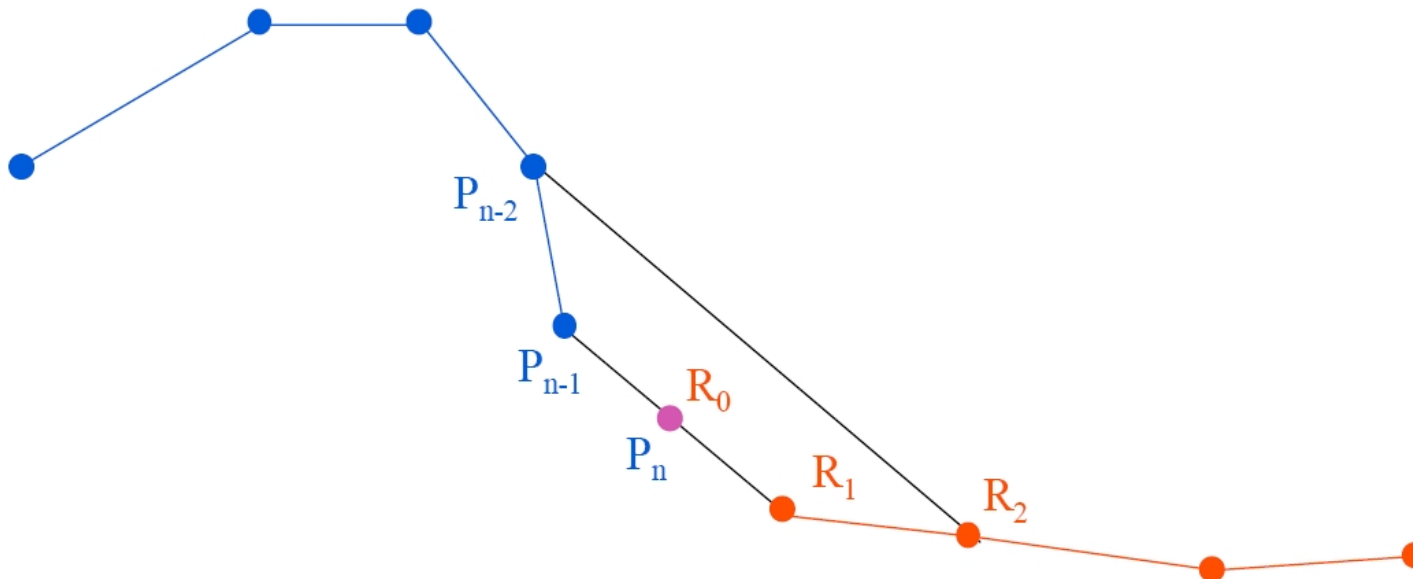
Manipulation de Béziérs

- Fermeture
 - Raccordement C^0 : $P_0 = P_5$
 - Raccordement C^1 : $P_1 - P_0 = P_5 - P_4$
 - Raccordement C^2 : $P_2 - 2P_1 + P_0 = P_5 - 2P_4 + P_3$



Manipulation de Béziens

- Raccordement C^2 de deux courbes de degrés n
 - Raccordement C^0 : $P_n = R_0$
 - Raccordement C^1 : $P_n - P_{n-1} = R_1 - R_0$
 - Raccordement C^2 : $P_n - 2P_{n-1} + P_{n-2} = R_2 - 2R_1 + R_0$



Conclusion sur les Béziens

- Modèle historique
 - simple,
 - expression de tout polynôme à coefficients vectoriels
 - contrôle global bien connu (lissage important)
 - peu de paramètres
- Mais
 - la complexité augmente avec le nombre de points
 - limitation obligatoire du degré (9 à 10 en pratique)
 - contrôle global
 - pas adapté pour l'interpolation directe d'un nuage de points

Courbes B-Splines

Fonctions B-Splines

$$B_i^k(u) = \begin{cases} (-1)^k (u_{i+k} - u_i) [u_i(k) : t] (u - t)_+^{k-1} & \text{pour } u_i \leq u < u_{i+k} \\ 0 & \text{pour } u < u_i \text{ et } u \geq u_{i+k} \end{cases}$$

$$\text{avec } \sum B_i^k(u) = 1, B_i^k(u) > 0 \text{ pour } u_i \leq u < u_{i+k}$$

Les différences divisées $[u_i(k) : t] (u - t)_+^{k-1} = [u_i, u_{i+1}, \dots, u_{i+k} : t] (u - t)_+^{k-1}$ sont définies par récurrence :

$$[u_i : t] (u - t)_+^{k-1} = (u - u_i)_+^{k-1}$$

$$[u_i, u_{i+1}, \dots, u_{i+k} : t] (u - t)_+^{k-1} = \begin{cases} \frac{[u_{i+1}, \dots, u_{i+k} : t] (u - t)_+^{k-1} - [u_{i+1}, \dots, u_{i+k-1} : t] (u - t)_+^{k-1}}{u_{i+k} - u_i} & \text{pour } u_{i+k} < u_i \\ \frac{1}{k!} D_t^{(k)} (u - t)_+^{k-1} \Big|_{t=u_i} & \text{pour } u_{i+k} < u_i \end{cases}$$

Fonctions B-Splines

$$B_i^k(u) = \begin{cases} (-1)^k (u_{i+k} - u_i) [u_i(k) : t] (u - t)_+^{k-1} & \text{pour } u_i \leq u < u_{i+k} \\ 0 & \text{pour } u < u_i \text{ et } u \geq u_{i+k} \end{cases}$$

$$\text{avec } \sum B_i^k(u) = 1, B_i^k(u) > 0 \text{ pour } u_i \leq u < u_{i+k}$$

Les puissances tronquées $(u - u_i)_+^{k-1}$ sont définies par :

$$(u - t)_+^0 = \begin{cases} 1 & \text{pour } t \leq u \\ 0 & \text{pour } t > u \end{cases}$$

$$(u - t)_+^r = \begin{cases} (u - t)^r & \text{pour } t \leq u \\ 0 & \text{pour } t > u \end{cases}$$

Fonctions B-Splines

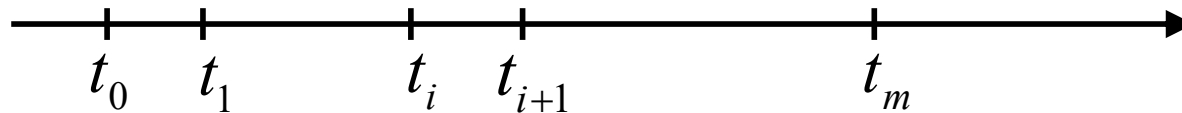
- Relation de récurrence

$$B_i^1(u) = \begin{cases} 1 & u_i \leq u \leq u_{i+1} \\ 0 & u < u_i \text{ et } u \geq u_{i+1} \end{cases}$$

$$B_i^k(u) = \begin{cases} \frac{u - u_i}{u_{i+k-1} - u_i} B_i^{k-1}(u) + \frac{u_{i+k} - u}{u_{i+k} - u_{i+1}} B_{i+1}^{k-1}(u) & u_i \leq u \leq u_{i+k} \\ 0 & u < u_i \text{ et } u \geq u_{i+k} \end{cases}$$

Fonctions B-Splines

- Vecteur de nœuds
 - point de départ : suite non décroissante de valeurs $t_0 \leq t_1 \leq \dots \leq t_m$
 - les t_i sont appelés les nœuds
 - la suite des t_i s'appelle vecteur de nœuds
 - les t_i peuvent être entiers ou réels



Fonctions B-Splines

- Fonctions B-Splines normalisées d'ordre k définies sur le vecteur de nœuds
 - soit k un entier fixé > 1
 - $(m-k+1)$ fonctions polynomiales par morceaux de degré $k-1$ notées vérifiant B_i^k

$$1) \sum_{i=0}^{m-k} B_i^k = 1 \quad \text{partition de l'unité}$$

$$2) \forall i, \forall k, B_i^k \geq 0 \quad \text{positivité}$$

$$3) B_i^k = 0 \text{ si } t \notin [t_i, t_{i+k}] \quad \text{support local}$$

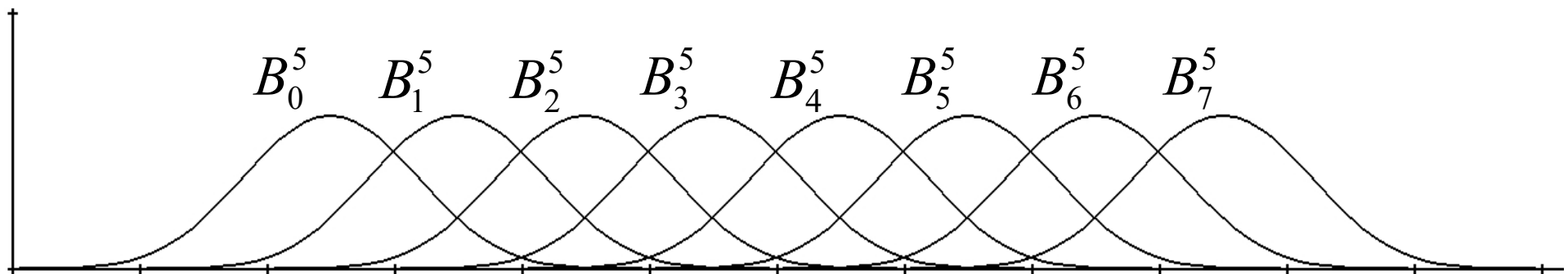
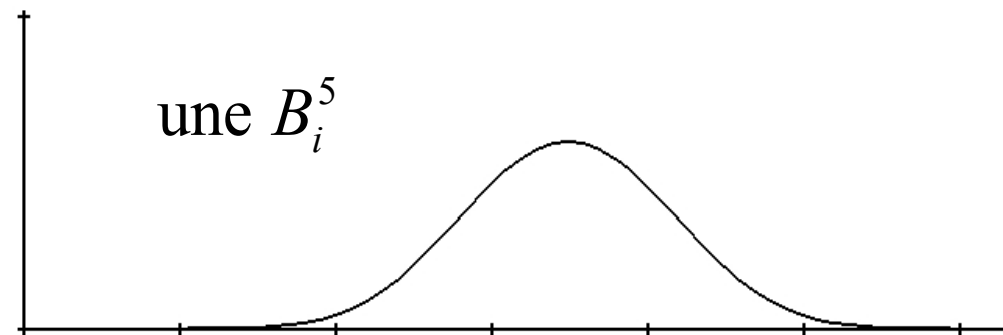
$$4) B_i^k \text{ est } C^{k-2} \text{ continue sur } [t_0, t_m]$$

Fonctions B-Splines

- Interprétation : la fonction B_i^k est
 - non nulle sur $[t_i, t_{i+k}]$
 - nulle en dehors
 - un polynôme de degré $k-1$ sur chaque $[t_j, t_{j+1}]$
($j=i, i+1, \dots, i+k-1$)
- De plus
 - les k polynômes se raccordent en C^{k-2}
 - la continuité implique : $B_i^k(t_i) = B_i^k(t_{i+k}) = 0$

Fonctions B-Splines

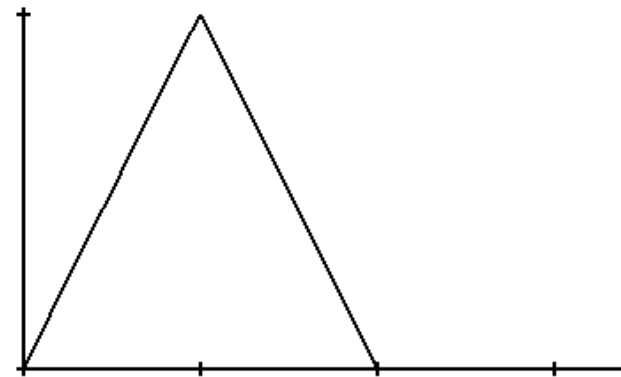
- Fonctions avec des nœuds équidistants :



Fonctions B-Splines

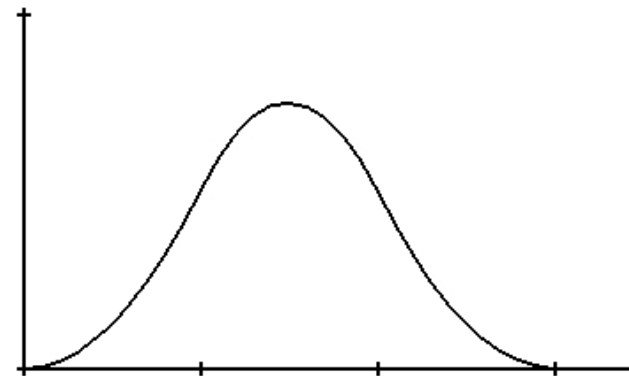
- Fonctions avec des nœuds équidistants :
 - ordre 2 ($k=2$, degré=1)

$$B_0^2(t) = \begin{cases} t & \text{si } 0 \leq t \leq 1 \\ (2-t) & \text{si } 1 \leq t \leq 2 \end{cases}$$



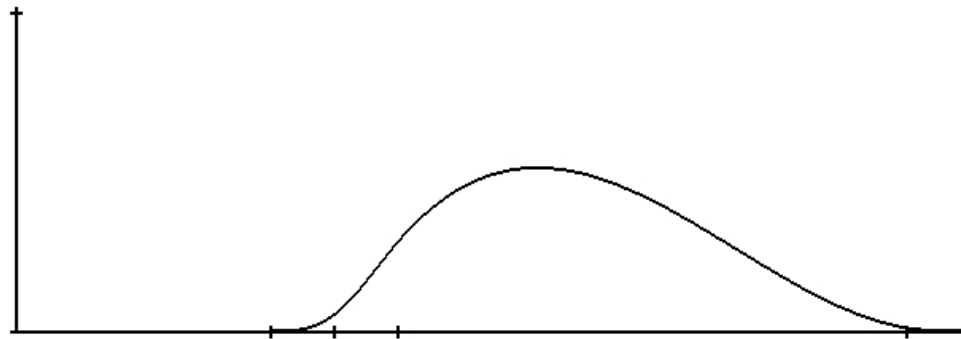
- ordre 3

$$B_0^3(t) = \begin{cases} \frac{t^2}{2} & \text{si } 0 \leq t \leq 1 \\ \frac{(-2t^2 + 6t - 3)}{2} & \text{si } 1 \leq t \leq 2 \\ \frac{(3-t)^2}{2} & \text{si } 2 \leq t \leq 3 \end{cases}$$

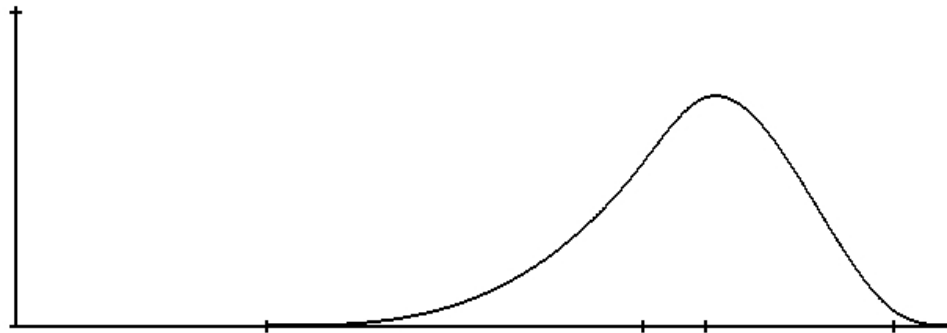


Fonctions B-Splines

- Influence de la répartition des nœuds (ordre 4)
 - vecteur des nœuds : 4, 5, 6, 14, 15



- vecteur des nœuds : 4, 10, 11, 14, 15

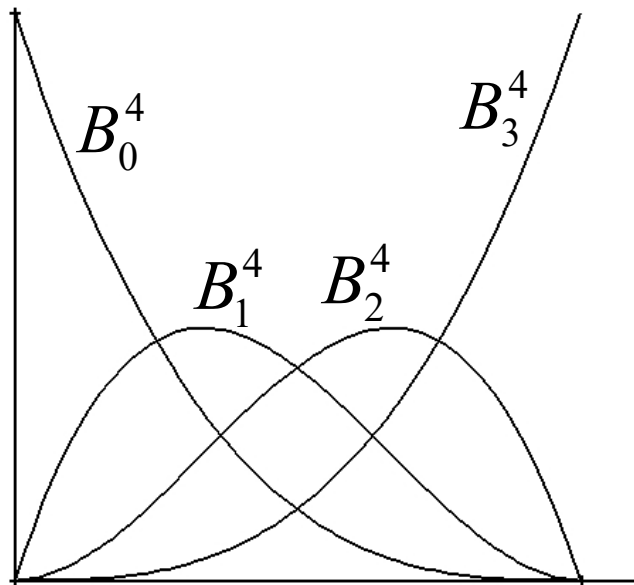


Fonctions B-Splines

- Superposition de nœuds (ordre 4)

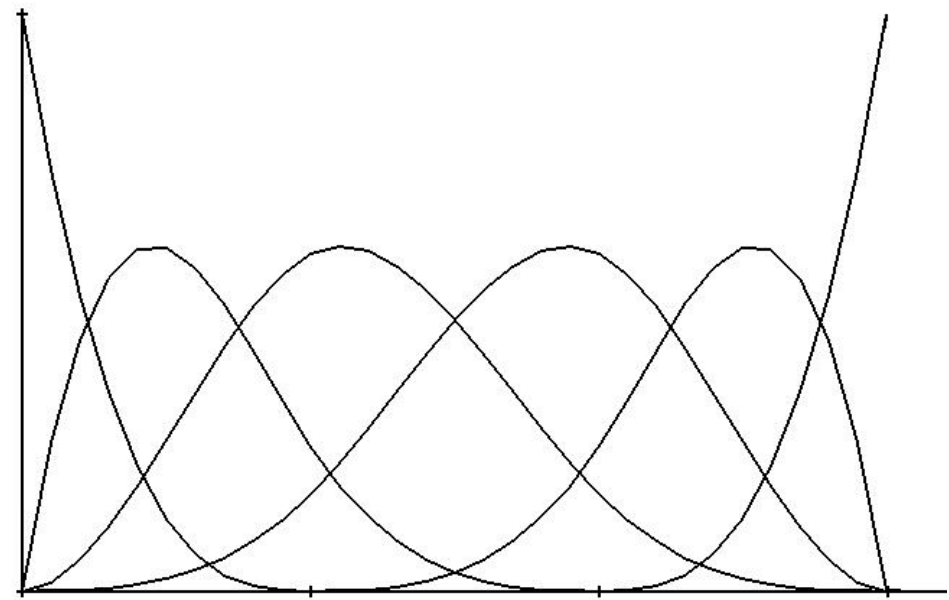
vecteur des nœuds :

0, 0, 0, 0, 1, 1, 1, 1



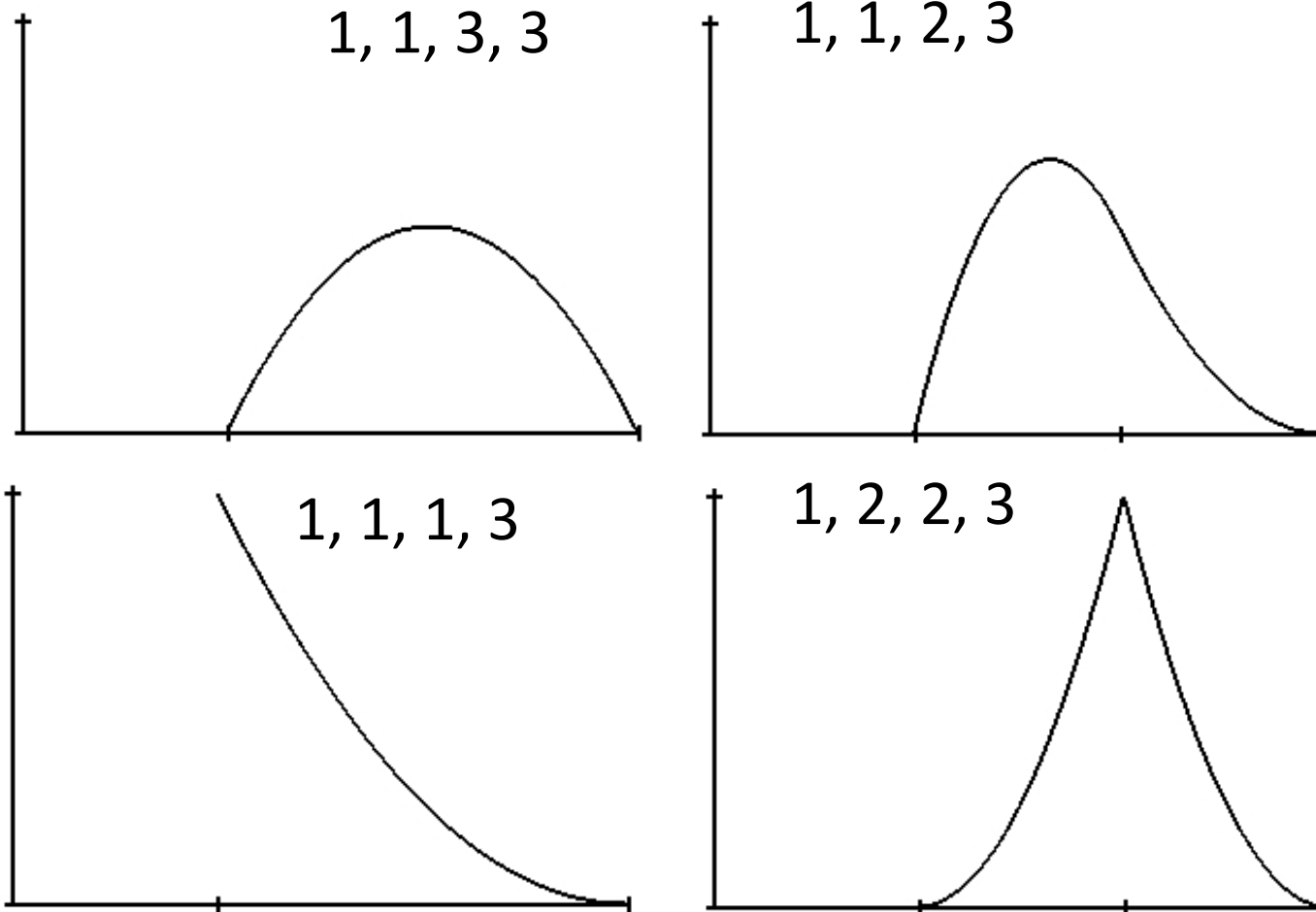
vecteur des nœuds :

0, 0, 0, 0, 1, 2, 3, 3, 3, 3



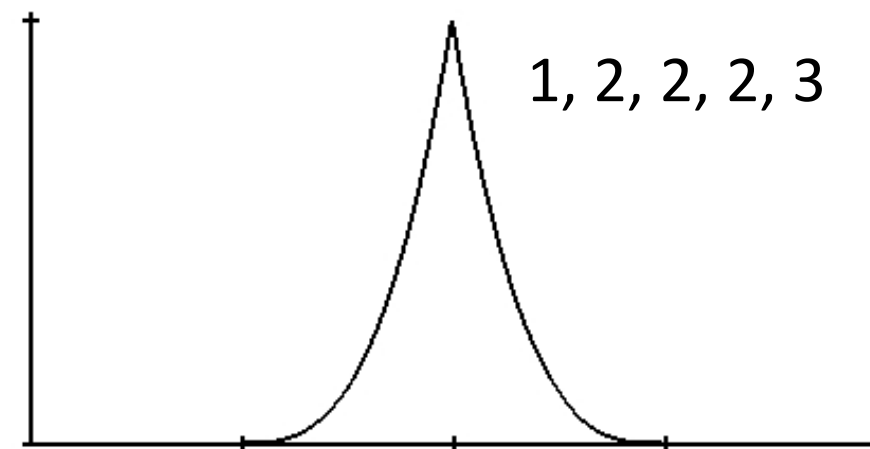
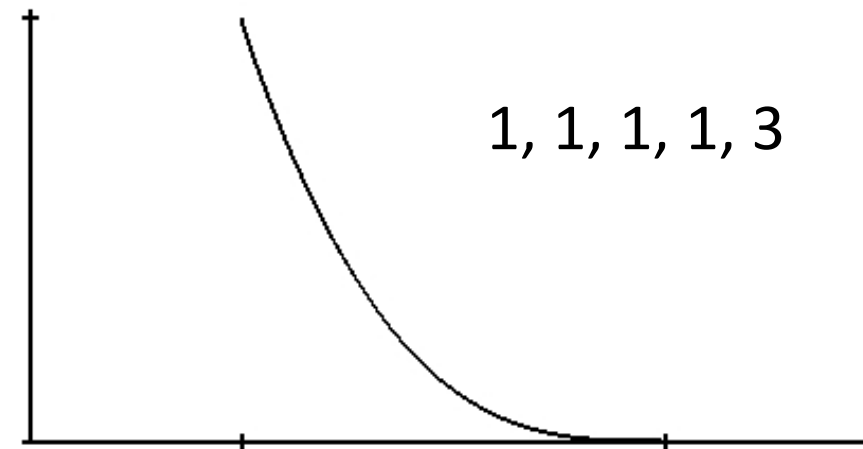
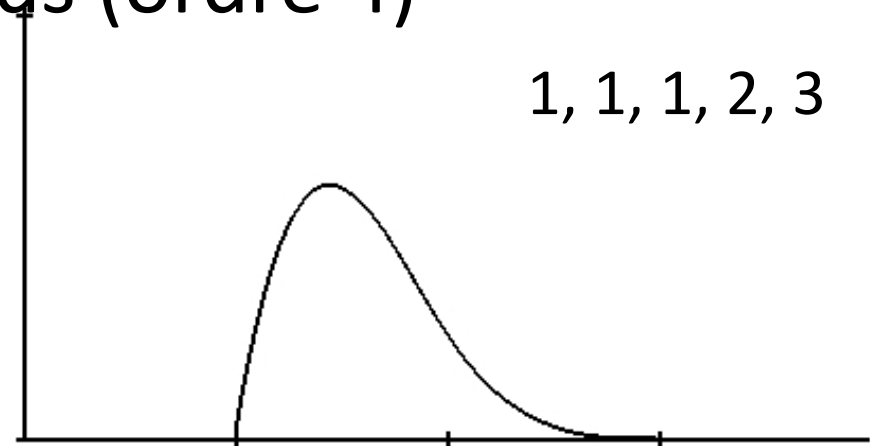
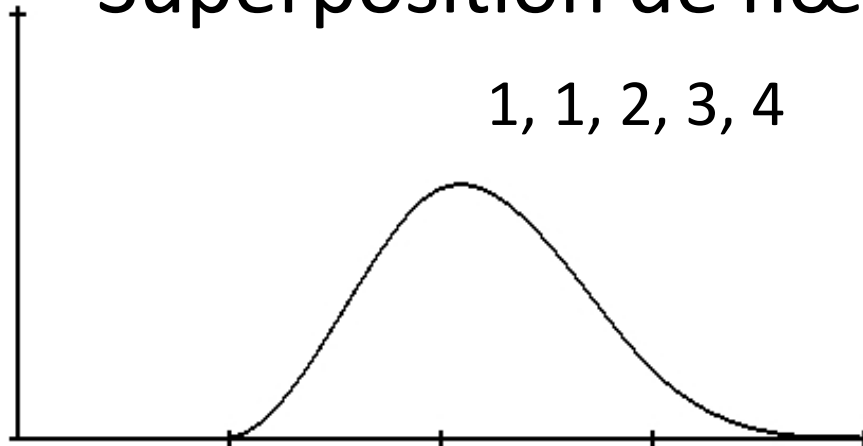
Fonctions B-Splines

- Superposition de nœuds (ordre 3)



Fonctions B-Splines

- Superposition de nœuds (ordre 4)



Courbes B-Splines

- Définition
 - Soient $(n+1)$ points P_i de $\mathbf{R}^2$ ou $\mathbf{R}^3$
 - La courbe B-Spline associée à ces points est la courbe obtenue en prenant comme fonctions de poids un ensemble de fonctions de base

$$P(t) = \sum_{i=0}^n P_i B_i^k(t)$$

- Les points P_i sont appelés points de contrôle
- La ligne polygonale $P_0 P_1 \dots P_n$ est appelée polygone caractéristique de la courbe, ou polygone de contrôle

Courbes B-Splines

- Il faut choisir un ordre k et un vecteur de nœuds
 - k choisi en premier
 - degré de continuité souhaité
 - pour avoir des polynômes de degré le plus bas possible
 - très souvent, $k = 4$
 - nombre de nœuds
 - nombre de nœuds : $n + k + 1$
 - vecteur de nœuds $T : t_0, t_1, \dots, t_{n+k}$
 - choix des nœuds
 - on prend en général $t_0 = t_1 = \dots = t_{k-1} < t_k < \dots < t_n < t_{n+1} = t_{n+2} = \dots = t_{n+k}$
 - l'intervalle de travail est toujours $[t_{k-1}, t_{n+1}]$
 - vaste problème

Courbes B-Splines

- Propriétés
 - $P(t)$ est le barycentre des points P_i affectés des masses B_i^k
 - Enveloppe convexe
 - Caractère local : $P(t)$ ne dépend que de k points de contrôle
 - Invariance par transformations affines
 - Passage par les points extrêmes
 - Propriété de diminution des variations
 - Propriété de reproduction de la linéarité
 - Si un point apparaît plusieurs fois, $P(t)$ est tirée vers ce point
 - La courbe est plus proche des points de contrôle qu'une courbe de Bézier
 - Les courbes de Bézier sont un cas particulier des B-Splines

Calcul d'un point

- Algorithme de De Boor – Cox (De Casteljau)

```
début  
  pour m de i-k+1 à i  
     $P_m^1 = P_m$   
  finpour  
  pour j de 1 à k-1  
    pour m de i-k+j+1 à i  
      
$$P_m^{j+1} = \frac{(t - t_m)P_m^j + (t_{m+k-j} - t)P_{m-1}^j}{t_{m+k-j} - t_m}$$
  
    finpour  
  finpour  
fin
```

$$k-1 \leq i \leq n$$

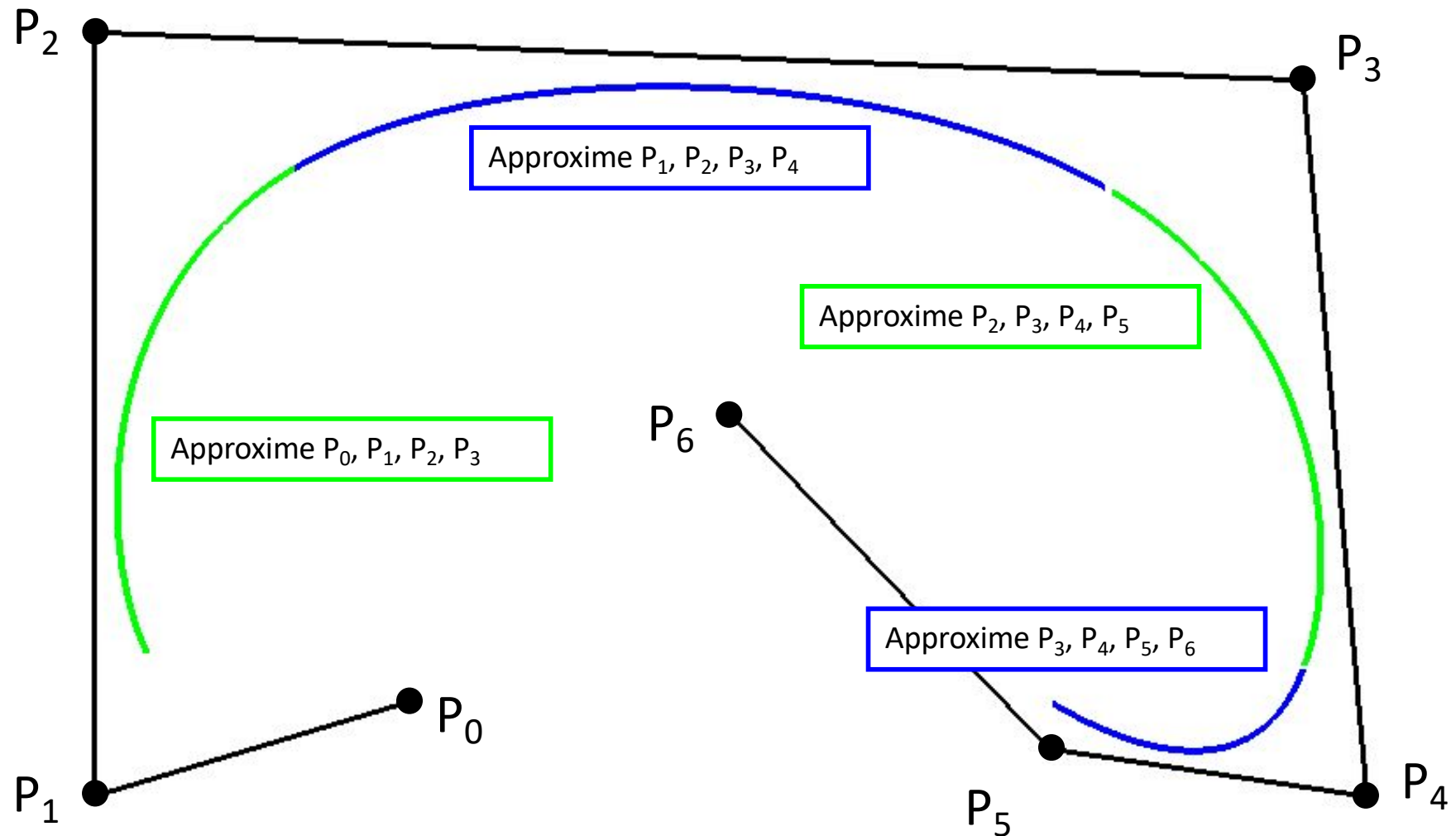
$$t_i \leq t \leq t_{i+1} \text{ avec } t_i < t_{i+1}$$

$[t_i, t_{i+1}]$ vrai intervalle)

$$\text{A la fin, } P(t) = P_i^k$$

Dessin d'une courbe

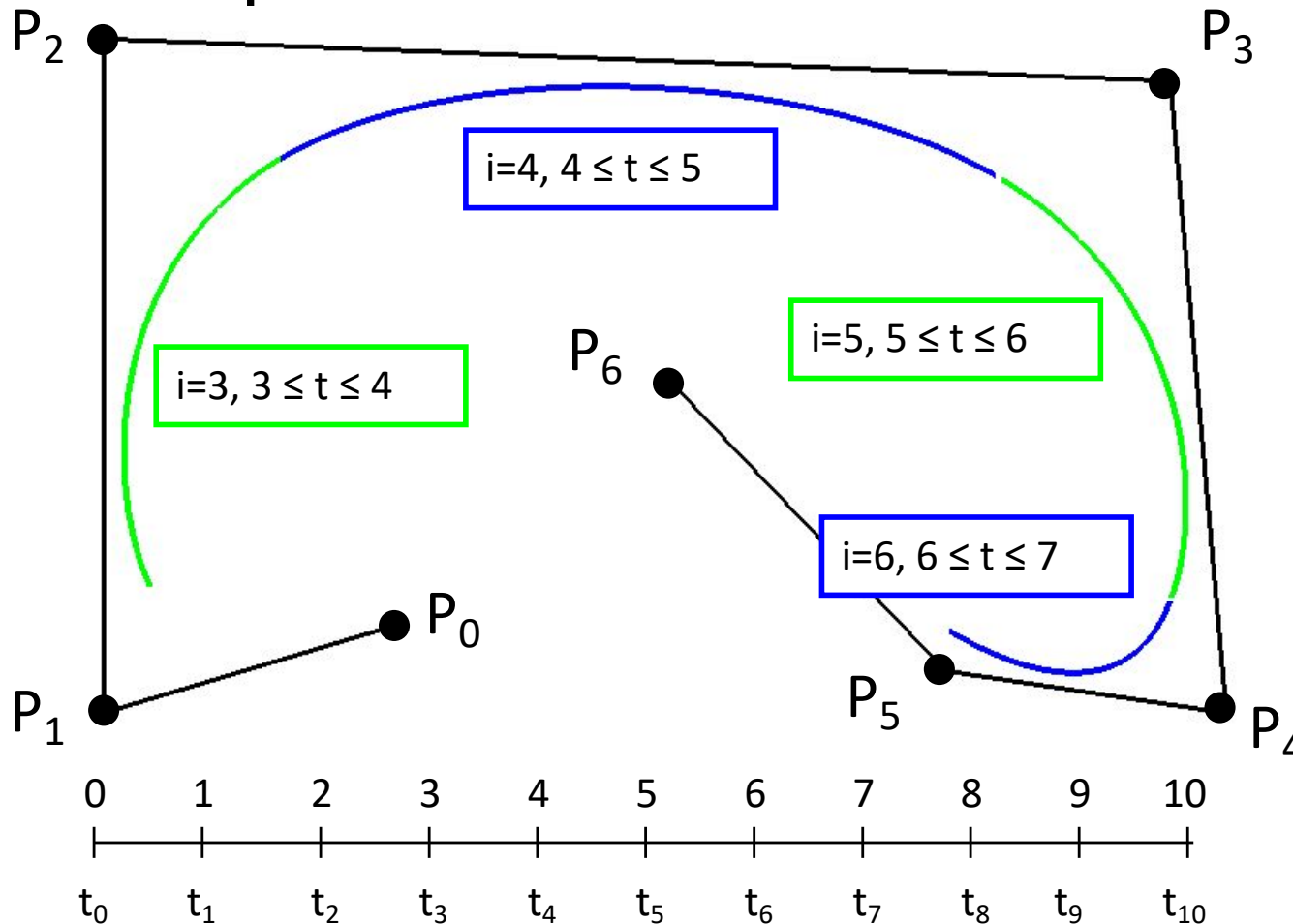
Exemple : $n=6$, $k=4$, 11 nœuds



Choix du vecteur de nœuds

Exemple : $n=6$, $k=4$, 11 nœuds

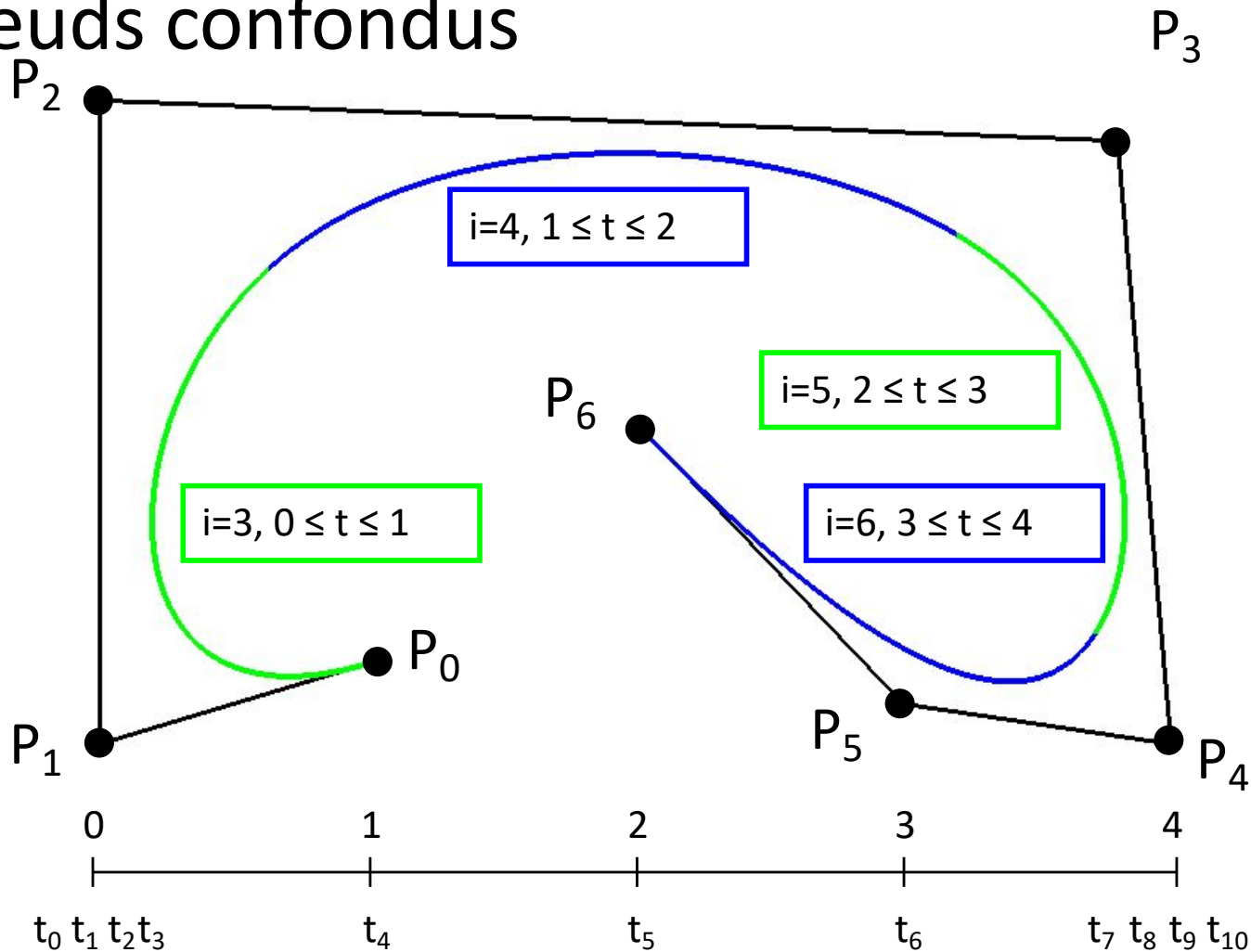
- Nœuds répartis uniformément



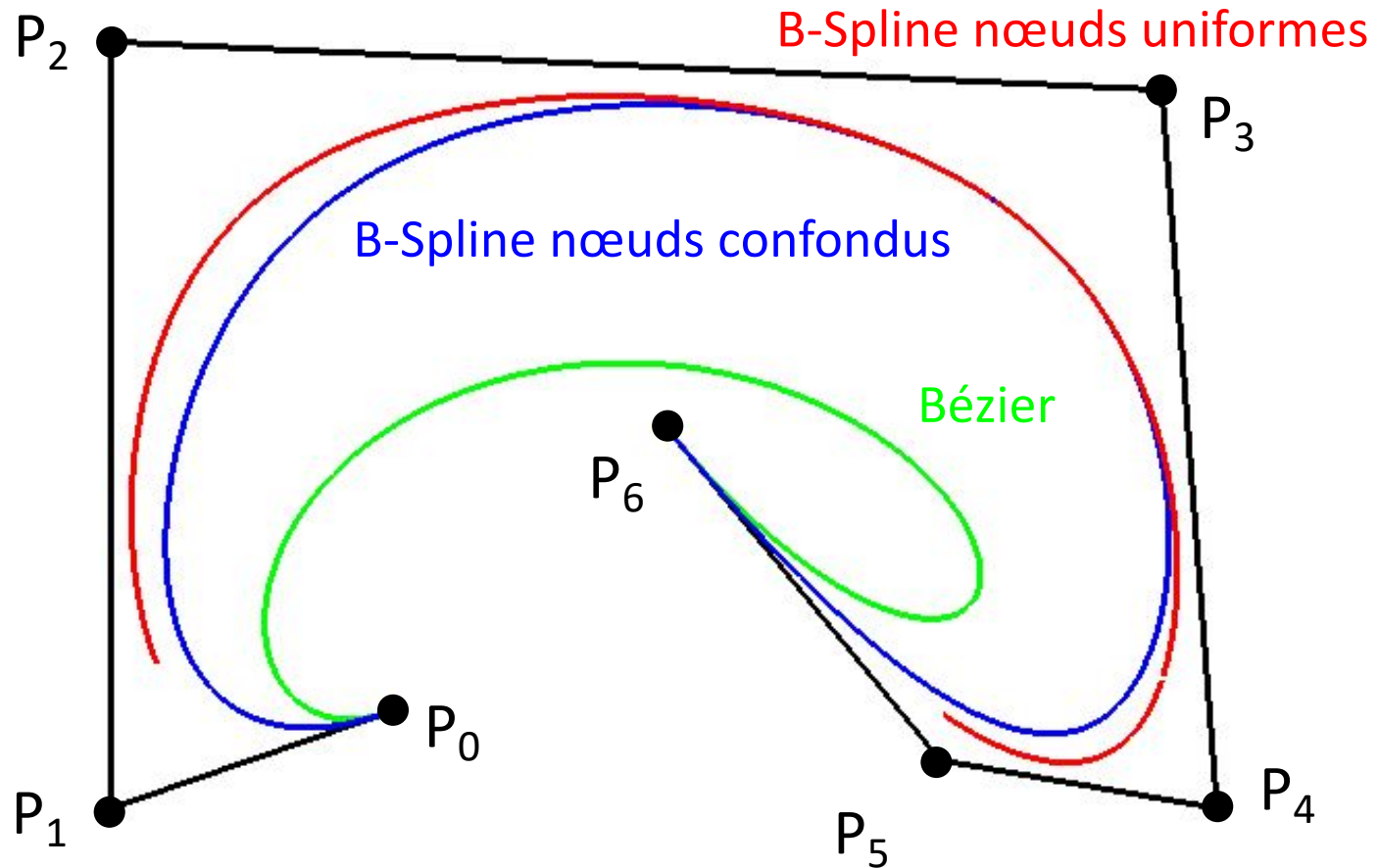
Choix du vecteur de nœuds

Exemple : $n=6$, $k=4$, 11 nœuds

- Nœuds confondus

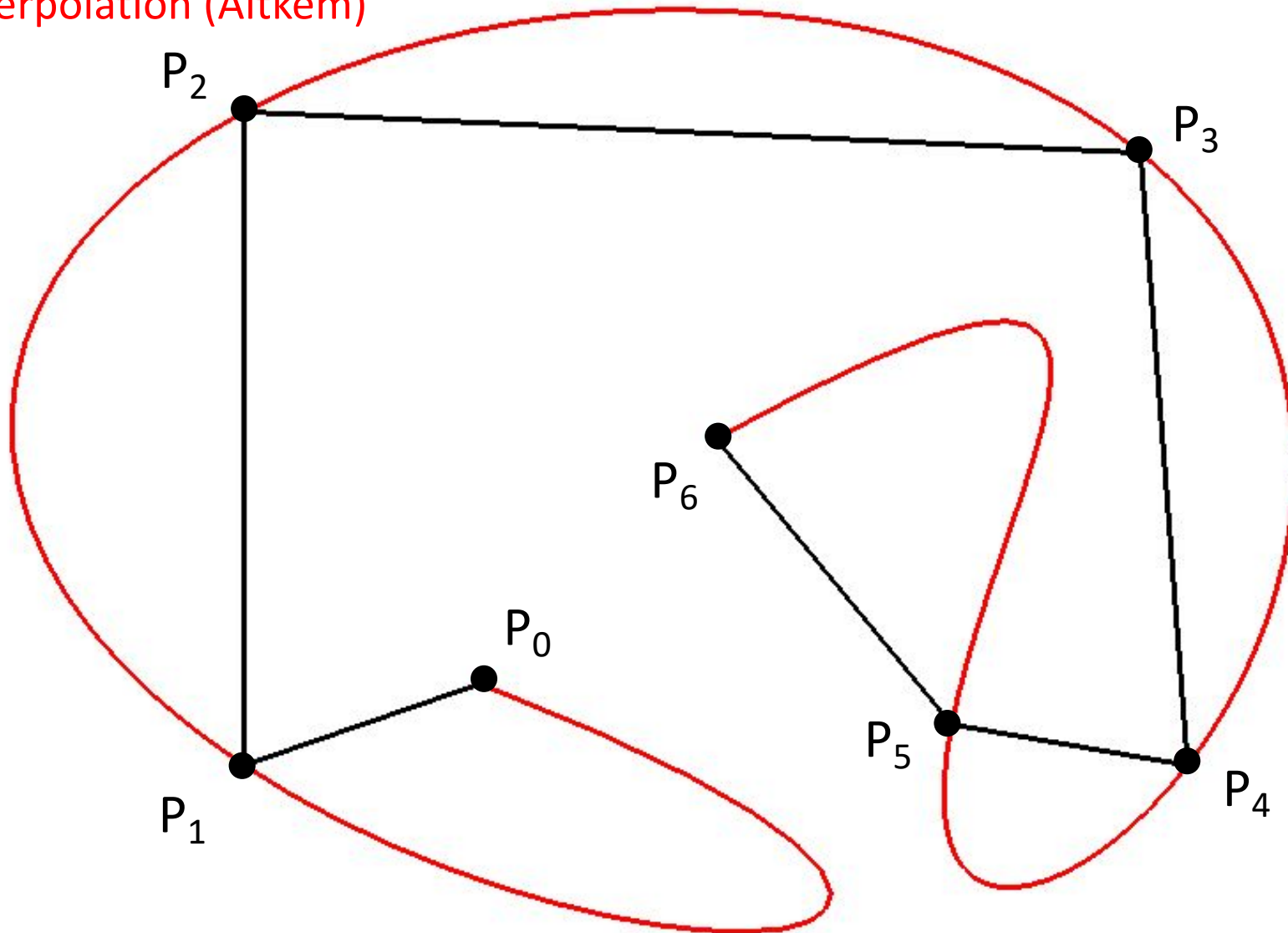


Lien avec les Béziers



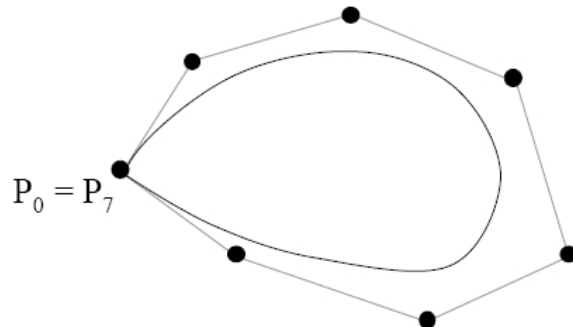
Rappel : interpolation

Interpolation (Aitkem)

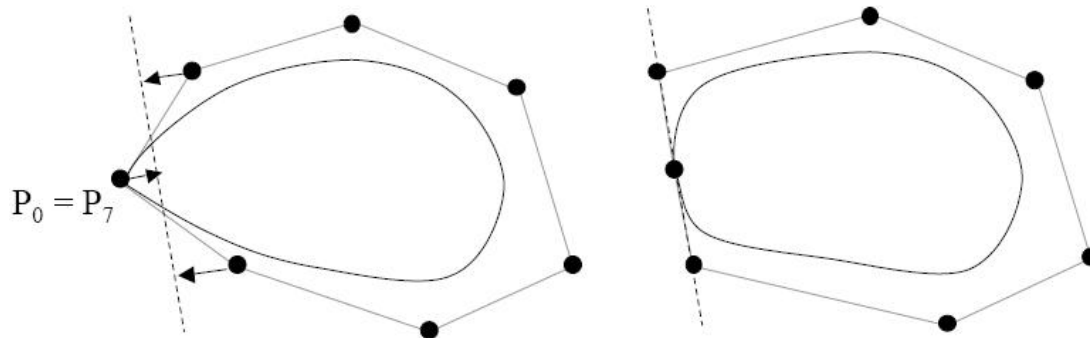


Fermeture d'une B-Spline

- Fermeture C^0
 - Premier point de contrôle = dernier point de contrôle

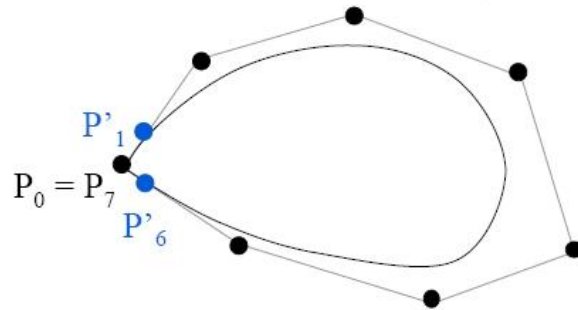


- Fermeture C^1 / G^1 (méthode 1)
 - Alignement de P_0, P_1, P_6

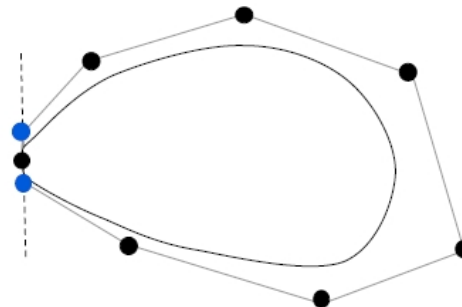
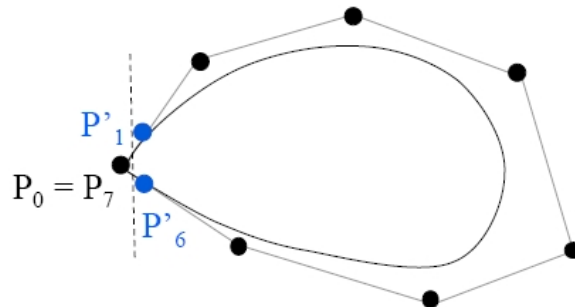


Fermeture d'une B-Spline

- Fermeture C^1 / G^1 (méthode 2)
 - Ajout de 2 nœuds $\rightarrow$ deux points de plus



- Alignement de P_0, P'_1, P'_6



Conclusion sur les B-Splines

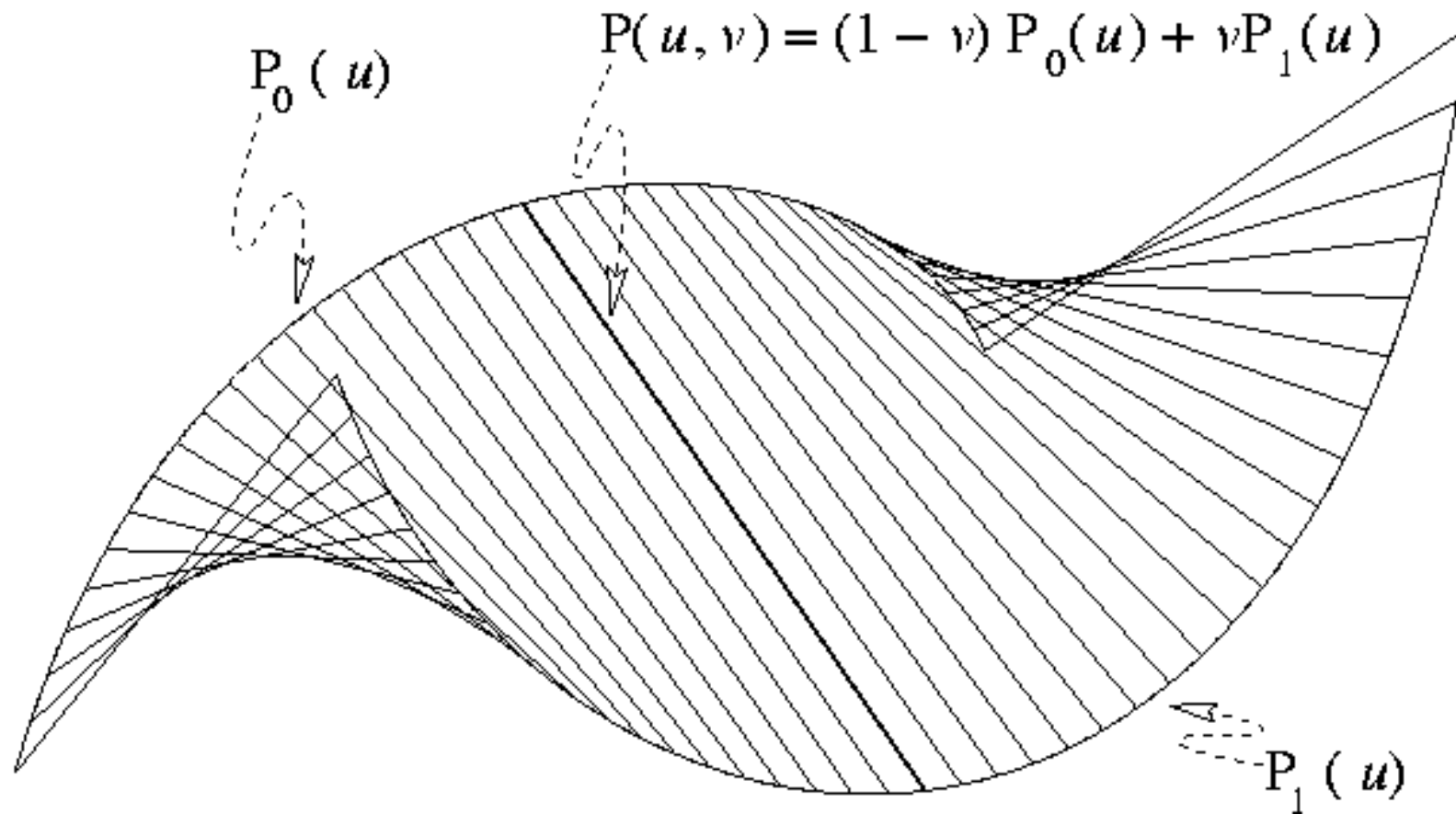
- Béziers : cas particulier des B-Splines
- B-Splines : Modèle opérationnel actuellement
 - Plus souple que Bézier
 - contrôle local
 - degré
 - vecteur de points
- Inconvénients
 - complexité algorithmique plus importante
 - difficile de maîtriser le vecteur de nœuds

Surfaces paramétriques

Surface réglée

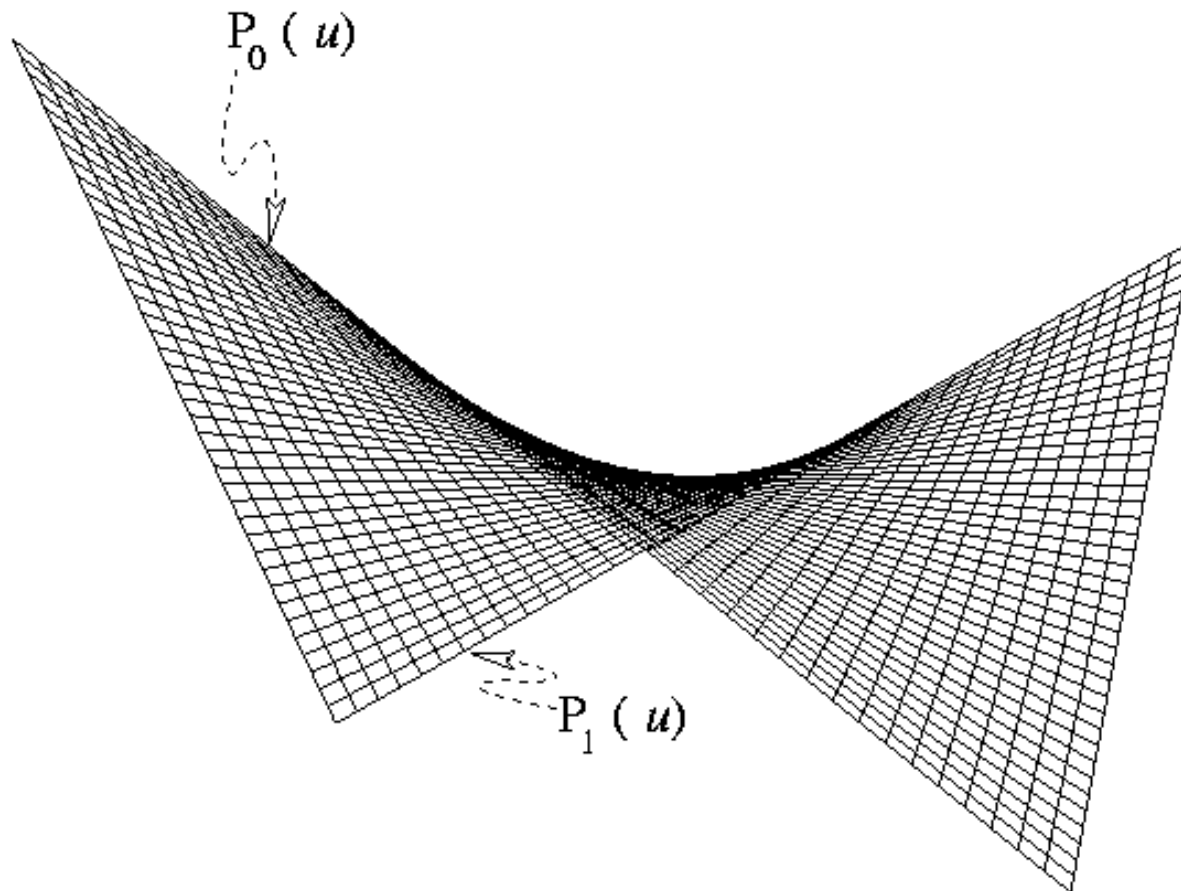
- Une surface est **réglée** si et seulement si par tout point il passe au moins une droite entièrement contenue dans la surface.
- On définit une surface réglée par deux courbes $P1(u)$ et $P2(u)$ sur lesquelles s'appuient les droites de la surface.
- Exemples :
 - Le **cône** est une surface réglée dont toutes les droites (appelées **directrices**) sont concourantes au sommet.
Les directrices s'appuient sur le sommet et sur un cercle non coplanaire avec le sommet.
 - Le **cylindre** est une surface réglée dont toutes les droites sont parallèles à l'axe.
- Un **patch réglé** est un morceau de surface réglée obtenu en limitant les valeurs de u et v à des intervalles finis.

Surface réglée



Patch bilinéaire

- Un **patch bilinéaire** est un morceau de surface réglée telle que $P_0(u)$ et $P_1(u)$ sont des segments de droites.

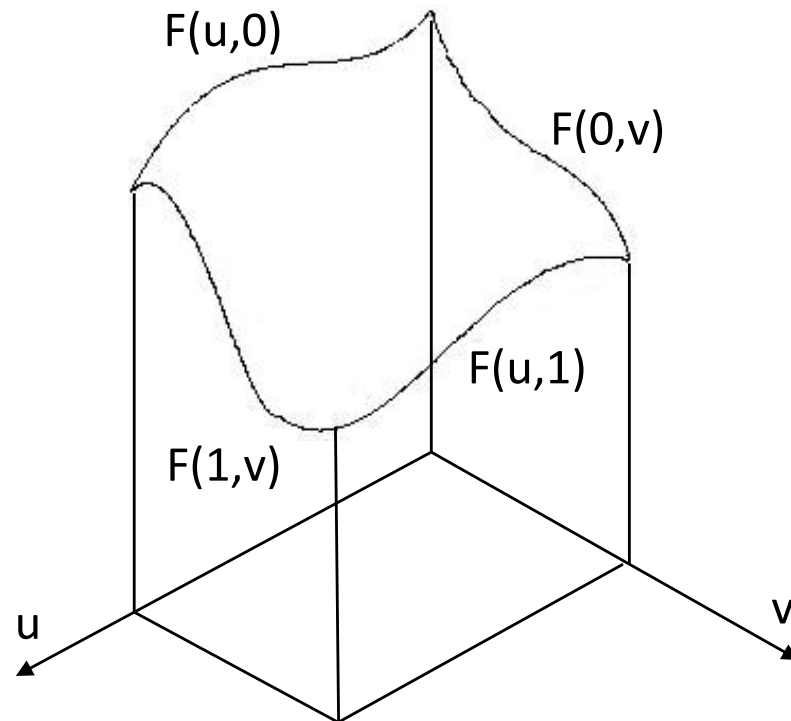


Carreaux de Coons bilinéaires

- Interpolation de courbes frontières à l'aide d'opérateurs booléens
- On dispose de 4 courbes frontière à interpoler

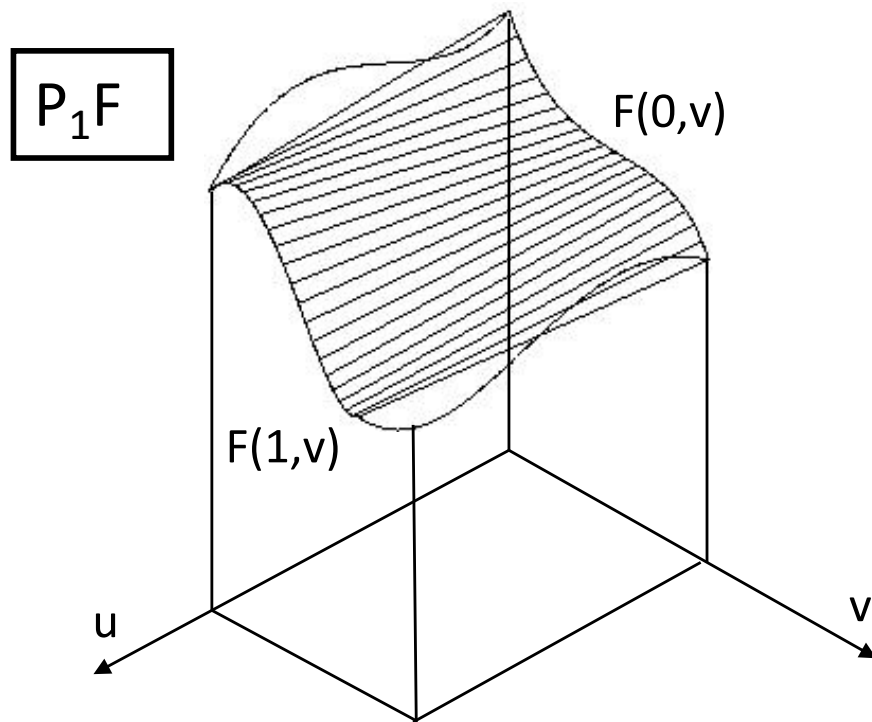
:

- $F(u,0)$
- $F(u,1)$
- $F(0,v)$
- $F(1,v)$

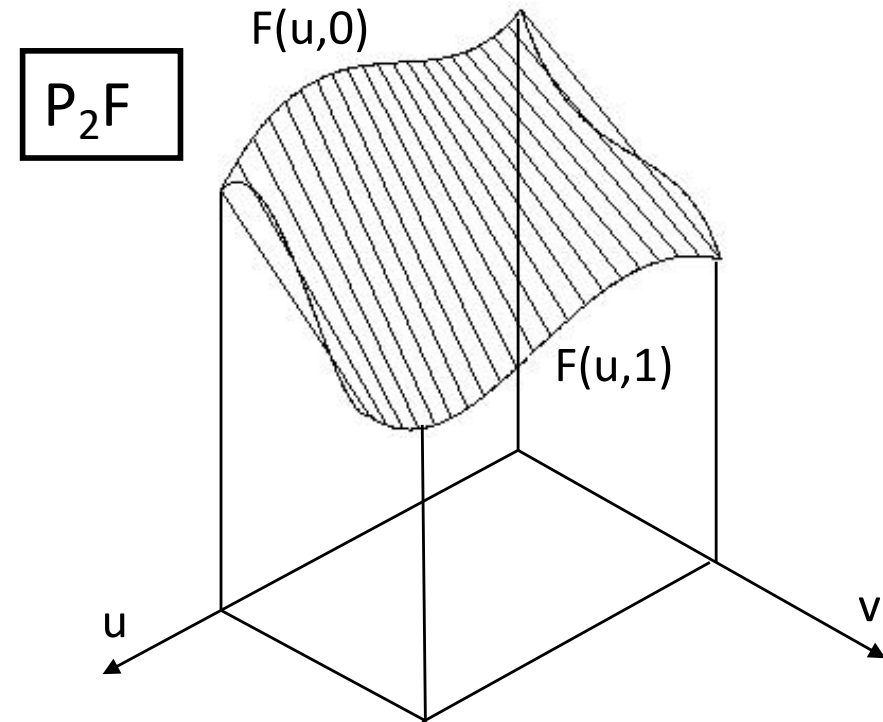


Carreaux de Coons bilinéaires

- Surfaces réglées s'appuyant sur les courbes frontières :



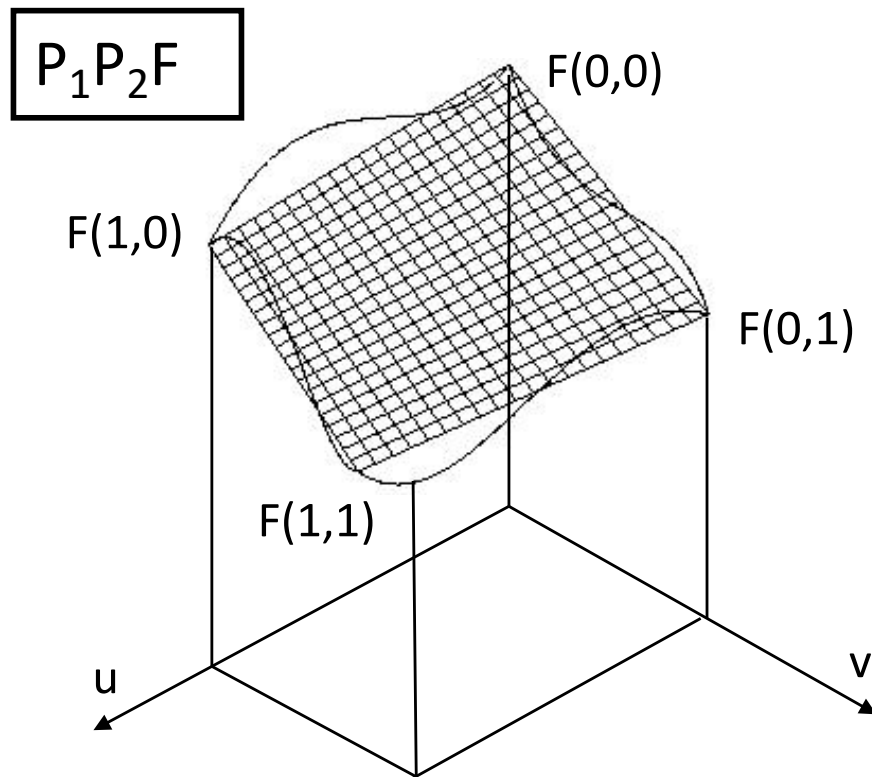
$$P_1F = (1-u) F(0,v) + u F(1,v)$$



$$P_2F = (1-v) F(u,0) + v F(u,1)$$

Carreaux de Coons bilinéaires

- Patch bilinéaire s'appuyant sur les points extrêmes :

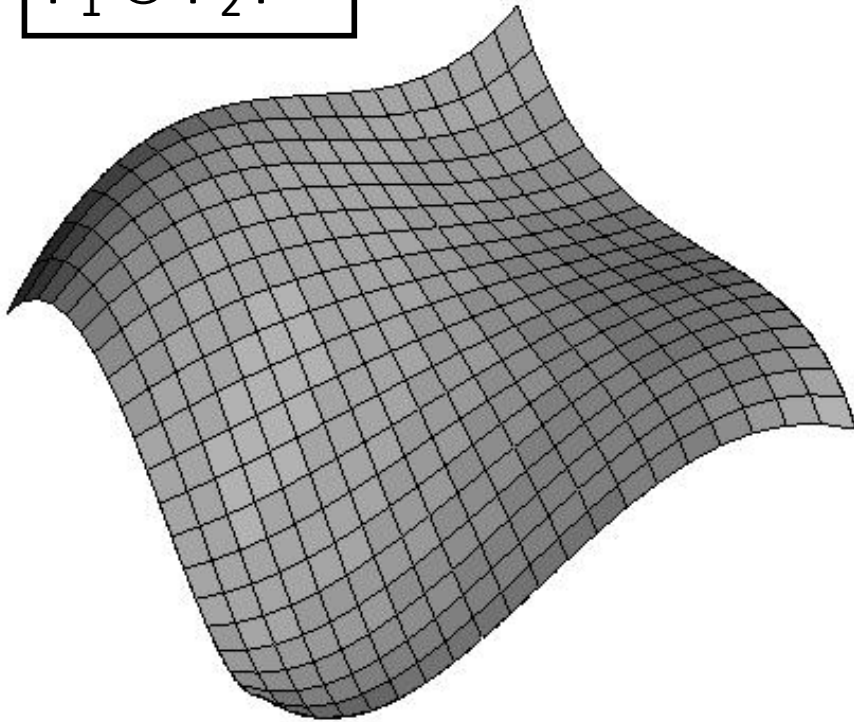


$$\begin{aligned} P_1 P_2 F = & (1-u) (1-v) F(0,0) \\ & + u (1-v) F(1,0) \\ & + (1-u) v F(0,1) \\ & + u v F(1,1) \end{aligned}$$

Carreaux de Coons bilinéaires

- Le patch de Coons est obtenu en ajoutant les surfaces réglées et en retranchant le patch bilinéaire

$$P_1 \oplus P_2 F$$



$$P_1 \oplus P_2 F = P_1 F + P_2 F - P_1 P_2 F$$

Carreau de Coons bicubiques

- En plus des 4 courbes frontière, on utilise les dérivées le long de ces courbes :
 - $F_v(u,0)$
 - $F_v(u,1)$
 - $F_u(0,v)$
 - $F_u(1,v)$
- On remplace les fonctions de base $(1-u, u, 1-v, v)$ par les polygones cubiques d' Hermite :
 - $h_0^3(t) = 2t^3-3t^2+1$
 - $h_0^2(t) = -2t^3+3t^2$
 - $h_0^1(t) = t^3-2t^2+t$
 - $h_0^0(t) = t^3-t^2$

Carreau de Coons bicubiques

$$P_1 F(u, v) = \begin{bmatrix} h_0^3(u) & h_1^3(u) & h_2^3(u) & h_3^3(u) \end{bmatrix} \begin{bmatrix} F(0, v) \\ F(1, v) \\ F_u(0, v) \\ F_u(1, v) \end{bmatrix}$$

$$P_2 F(u, v) = \begin{bmatrix} F(u, 0) & F(u, 1) & F_v(u, 0) & F_v(u, 1) \end{bmatrix} \begin{bmatrix} h_0^3(v) \\ h_1^3(v) \\ h_2^3(v) \\ h_3^3(v) \end{bmatrix}$$

$$P_1 P_2 F(u, v) = \begin{bmatrix} h_0^3(u) & h_1^3(u) & h_2^3(u) & h_3^3(u) \end{bmatrix} \begin{bmatrix} F(0,0) & F(0,1) & F_v(0,0) & F_v(0,1) \\ F(1,0) & F(1,1) & F_v(1,0) & F_v(1,1) \\ F_u(0,0) & F_u(0,1) & F_{uv}(0,0) & F_{uv}(0,1) \\ F_u(1,0) & F_u(1,1) & F_{uv}(1,0) & F_{uv}(1,1) \end{bmatrix} \begin{bmatrix} h_0^3(v) \\ h_1^3(v) \\ h_2^3(v) \\ h_3^3(v) \end{bmatrix}$$

$$P_1 \oplus P_2 F = P_1 F + P_2 F - P_1 P_2 F$$

Éléments de torsion
(carré de Gregory)



Carreau de Coons bicubiques

- Éléments de torsion : carré de Grégory

$$F_{uv}(0,0) = \frac{u \frac{\partial}{\partial v} F_u(0,0) + v \frac{\partial}{\partial u} F_v(0,0)}{u + v}$$

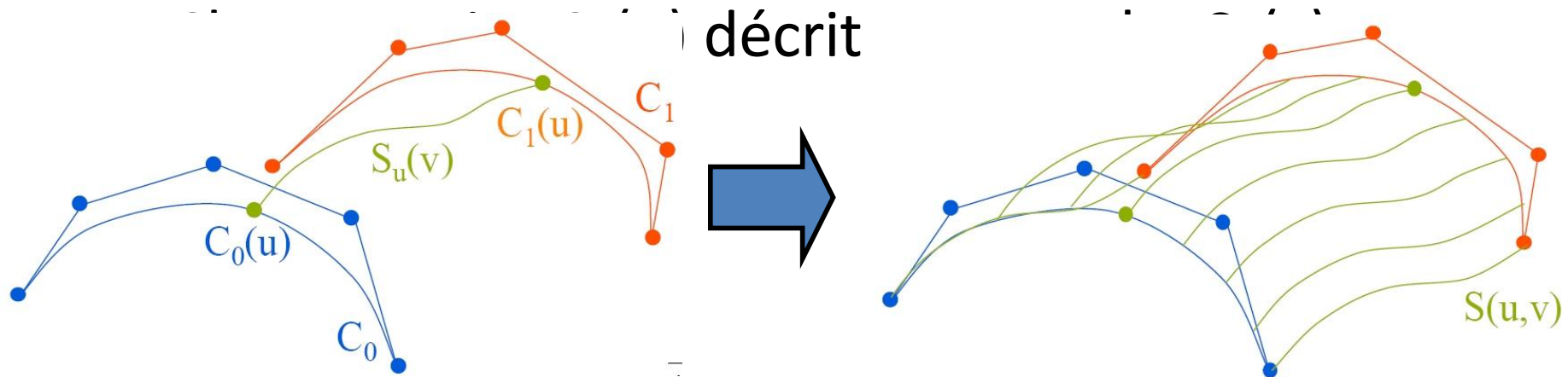
$$F_{uv}(0,1) = \frac{u \frac{\partial}{\partial v} F_u(0,1) + (1-v) \frac{\partial}{\partial u} F_v(0,1)}{u + (1-v)}$$

$$F_{uv}(1,0) = \frac{(1-u) \frac{\partial}{\partial v} F_u(1,0) + v \frac{\partial}{\partial u} F_v(1,0)}{(1-u) + v}$$

$$F_{uv}(1,1) = \frac{(1-u) \frac{\partial}{\partial v} F_u(1,1) + (1-v) \frac{\partial}{\partial u} F_v(1,1)}{(1-u) + (1-v)}$$

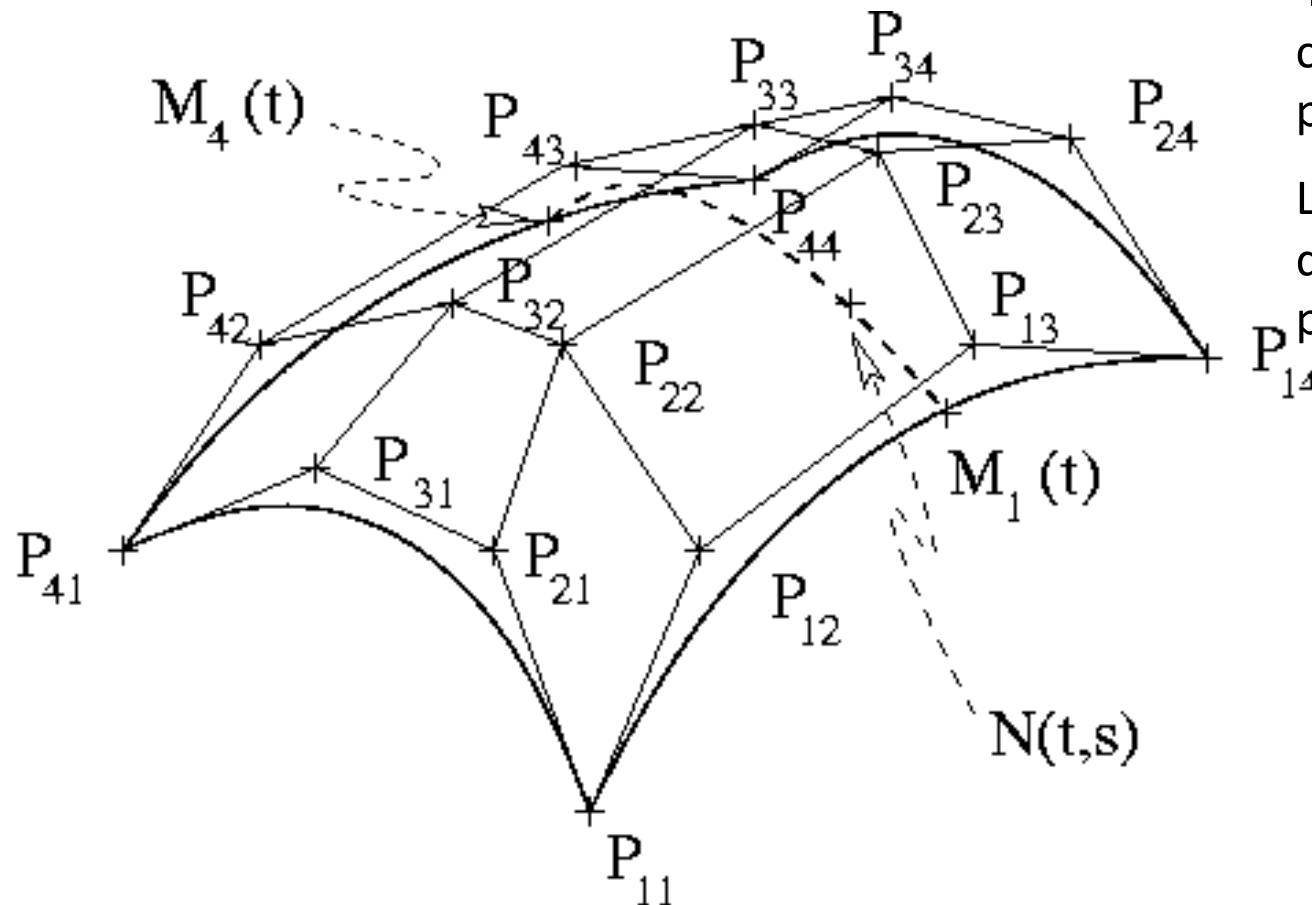
Carreaux rectangulaires de Bézier

- Principe
 - Soit une courbe C_0
 - Déformons C_0 vers une nouvelle position C_1
 - Supposons que la déformation soit continue



Carreaux rectangulaires de Bézier

- Surface bicubique de Bézier



16 points de contrôle P_{ij}

4 points $M_k(t)$ sur les courbes cubiques de Bézier définies par P_{kj}

Le point $N(t,s)$ sur la courbe de Bézier définie par les 4 points $M_k(t)$ précédents

Carreaux rectangulaires de Bézier

- Produit tensoriel à partir des polygones de Bernstein

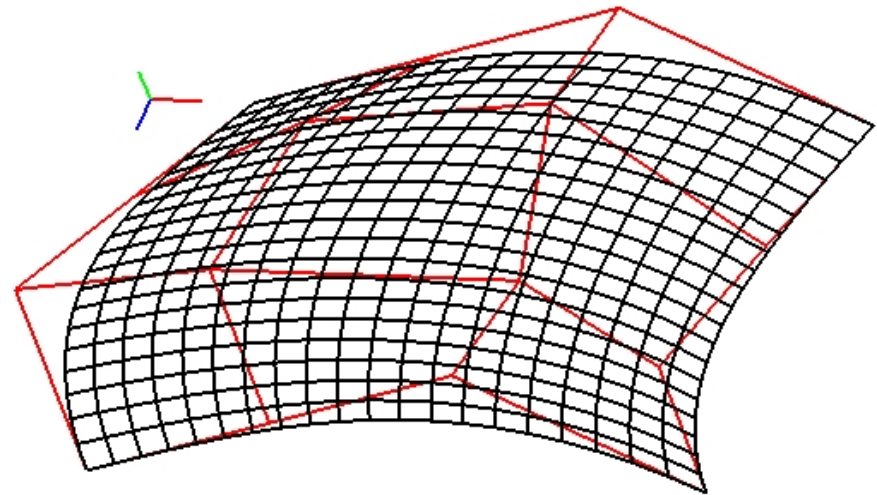
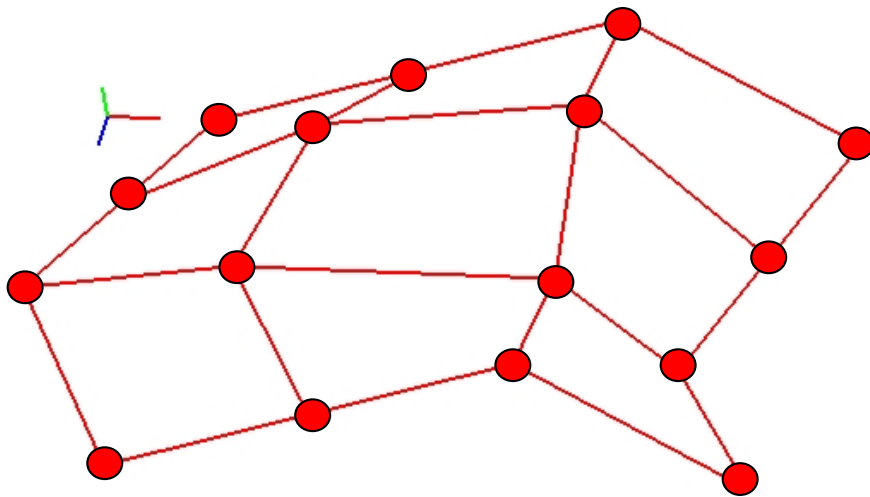
- Forme explicite :
$$P(u, v) = \sum_{i=0}^m \sum_{j=0}^n P_{i,j} B_i^m(u) B_j^n(v)$$

- Forme matricielle :

$$P(u, v) = \begin{bmatrix} B_0^m(u) & \cdots & B_m^m(u) \end{bmatrix} \underbrace{\begin{bmatrix} P_{00} & \cdots & P_{0n} \\ \vdots & & \vdots \\ P_{m0} & \cdots & P_{mn} \end{bmatrix}}_{\text{maillage des points de contrôle de Bézier}} \begin{bmatrix} B_0^n(v) \\ \vdots \\ B_n^n(v) \end{bmatrix}$$

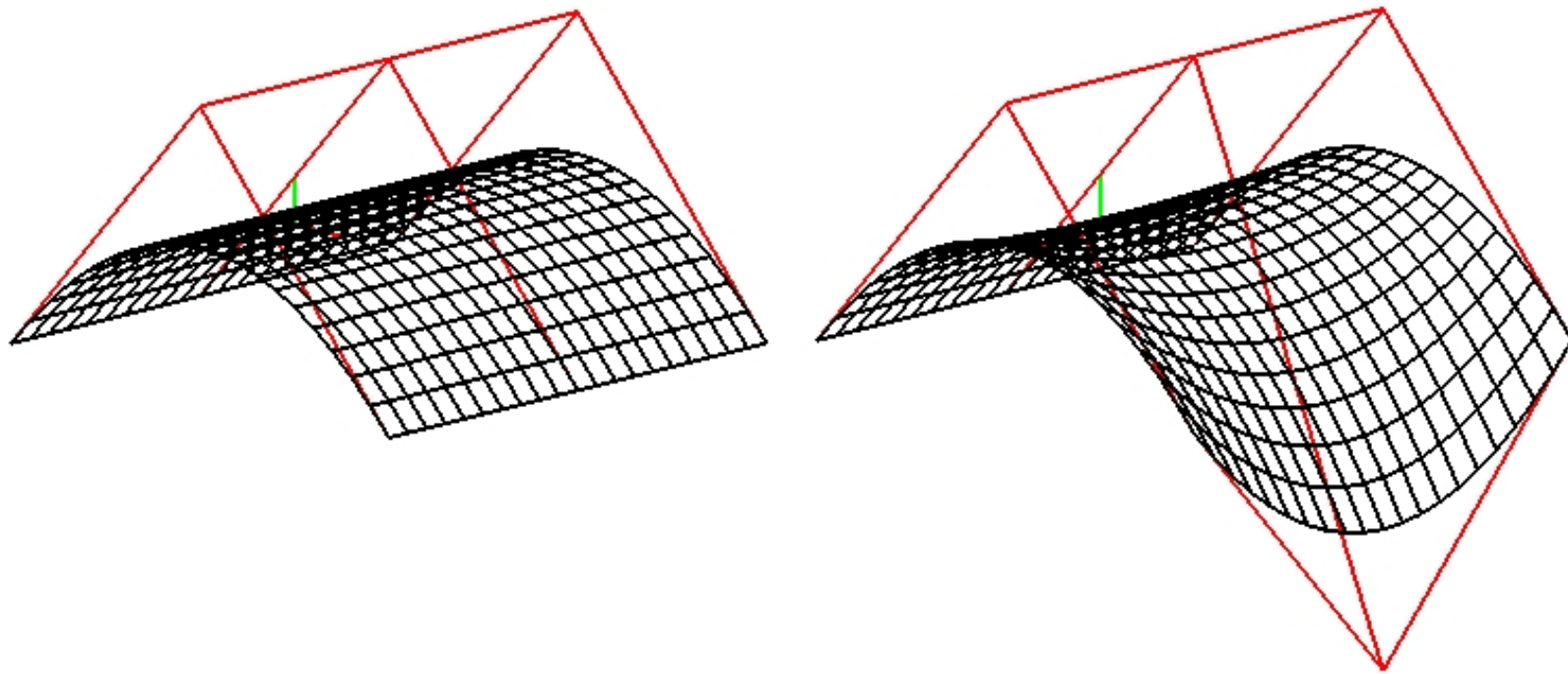
Carreaux rectangulaires de Bézier

- Exemple avec 16 points de contrôle



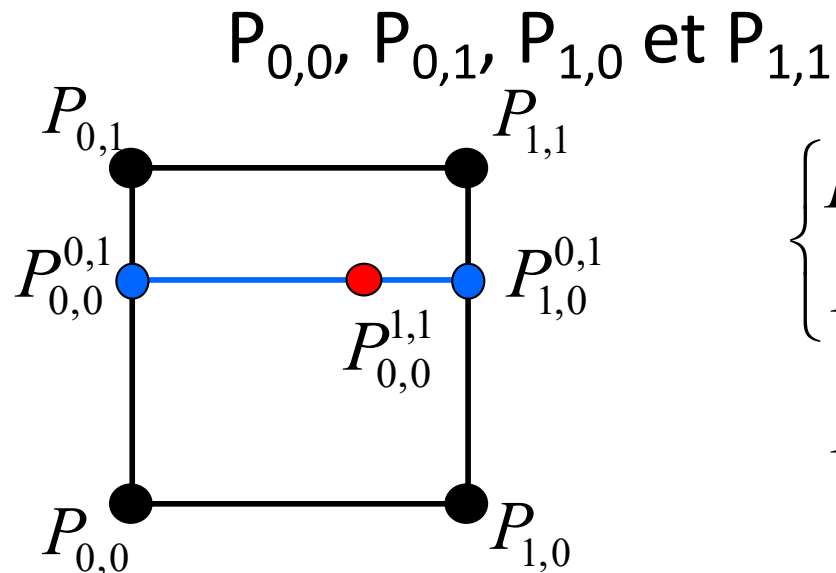
Carreaux rectangulaires de Bézier

- Propriétés : les mêmes que pour les courbes
- Perte de convexité :



Calcul d'un point

- Interpolation bilinéaire de 4 points



$$\begin{cases} P_{0,0}^{0,1} = (1-v)P_{0,0} + vP_{0,1} \\ P_{1,0}^{0,1} = (1-v)P_{1,0} + vP_{1,1} \end{cases}$$

$$P_{0,0}^{1,1} = (1-u)vP_{0,0}^{0,1} + uvP_{1,0}^{0,1}$$

$$P_{0,0}^{1,1} = (1-u)(1-v)P_{0,0} + (1-u)vP_{0,1} + u(1-v)P_{1,0} + uvP_{1,1}$$

$$P(u, v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} P_{0,0} & P_{0,1} \\ P_{1,0} & P_{1,1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix}$$

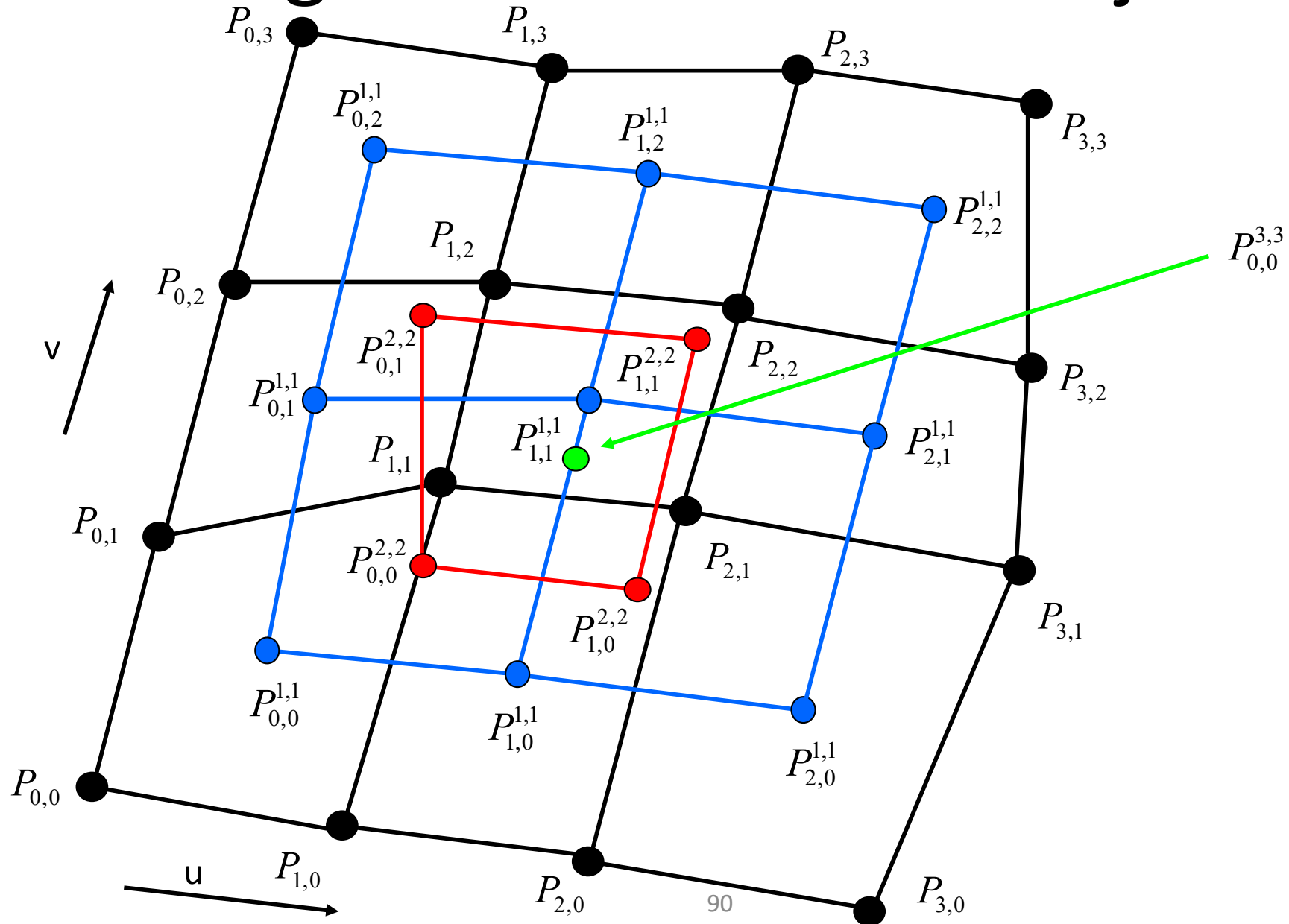
Algorithme de De Casteljau

- Pour $m=n$

$$\begin{cases} P_{i,j}^{0,0}(u,v) = P_{i,j} \\ P_{i,j}^{r,r}(u,v) = \begin{bmatrix} 1-u & u \end{bmatrix} \begin{bmatrix} P_{i,j}^{r-1,r-1} & P_{i,j+1}^{r-1,r-1} \\ P_{i+1,j}^{r-1,r-1} & P_{i+1,j+1}^{r-1,r-1} \end{bmatrix} \begin{bmatrix} 1-v \\ v \end{bmatrix} \end{cases}$$

Alors $P_{i,j}^{n,n}(u,v)$ est le point de paramètre (u,v) sur la surface

Algorithme de De Casteljau



Algorithme de De Casteljau

début

| pour i de 0 à n

| | pour j de 0 à n

| | | $P_{i,j}^{0,0} = P_{i,j}$

| | finpour

| finpour

| pour r de 1 à n

| | pour i de 0 à n-r

| | | pour j de 0 à n-r

$$P_{i,j}^{r,r} = (1-u)(1-v)P_{i,j}^{r-1,r-1} + (1-u)vP_{i,j+1}^{r-1,r-1}$$

$$+ u(1-v)P_{i+1,j}^{r-1,r-1} + uvP_{i+1,j+1}^{r-1,r-1}$$

| | | finpour

| | finpour

| finpour

fin

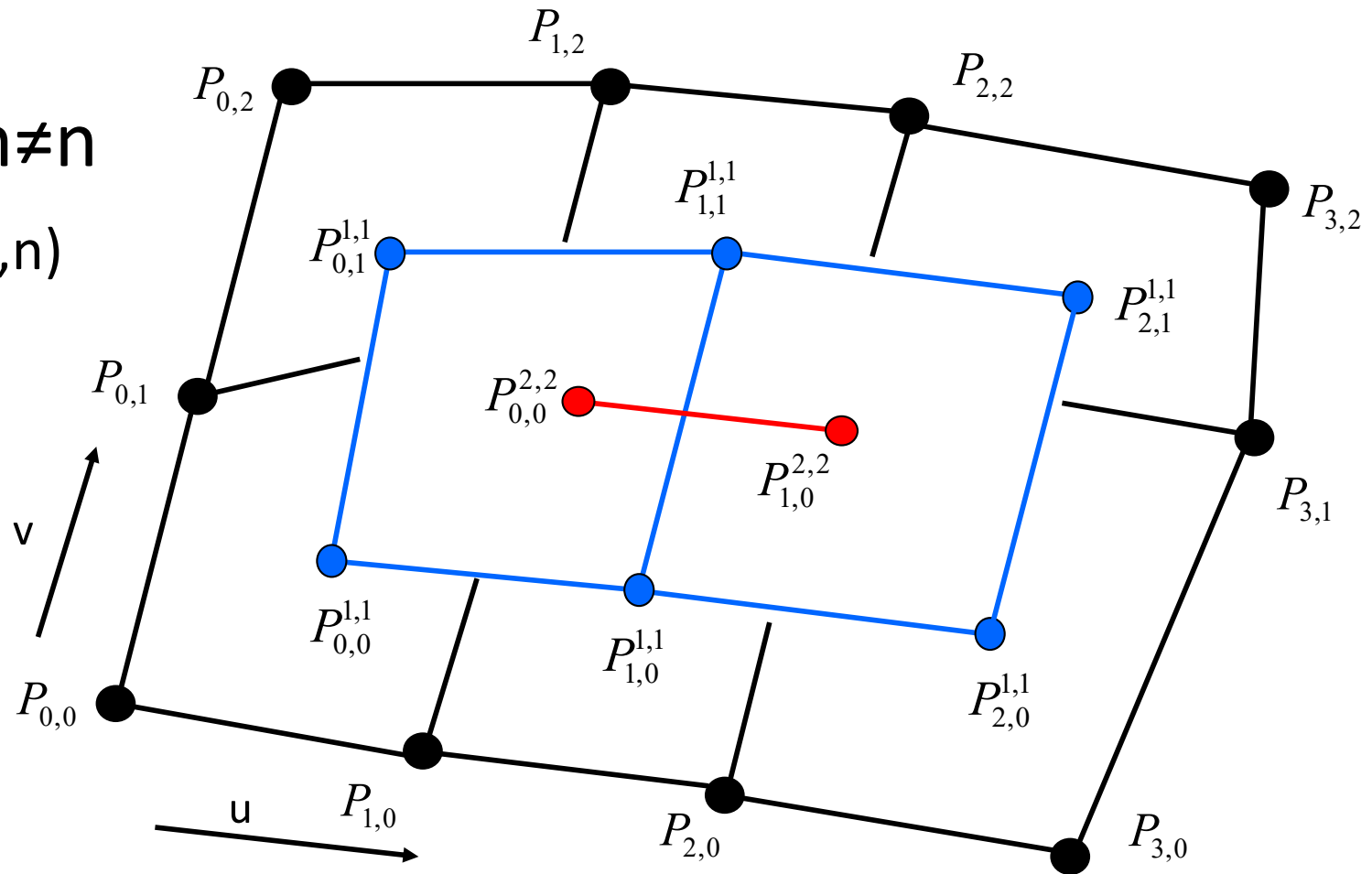
A la fin,

$$P(u, v) = P_{0,0}^{n,n}$$

Algorithme de De Casteljau

- Pour $m \neq n$

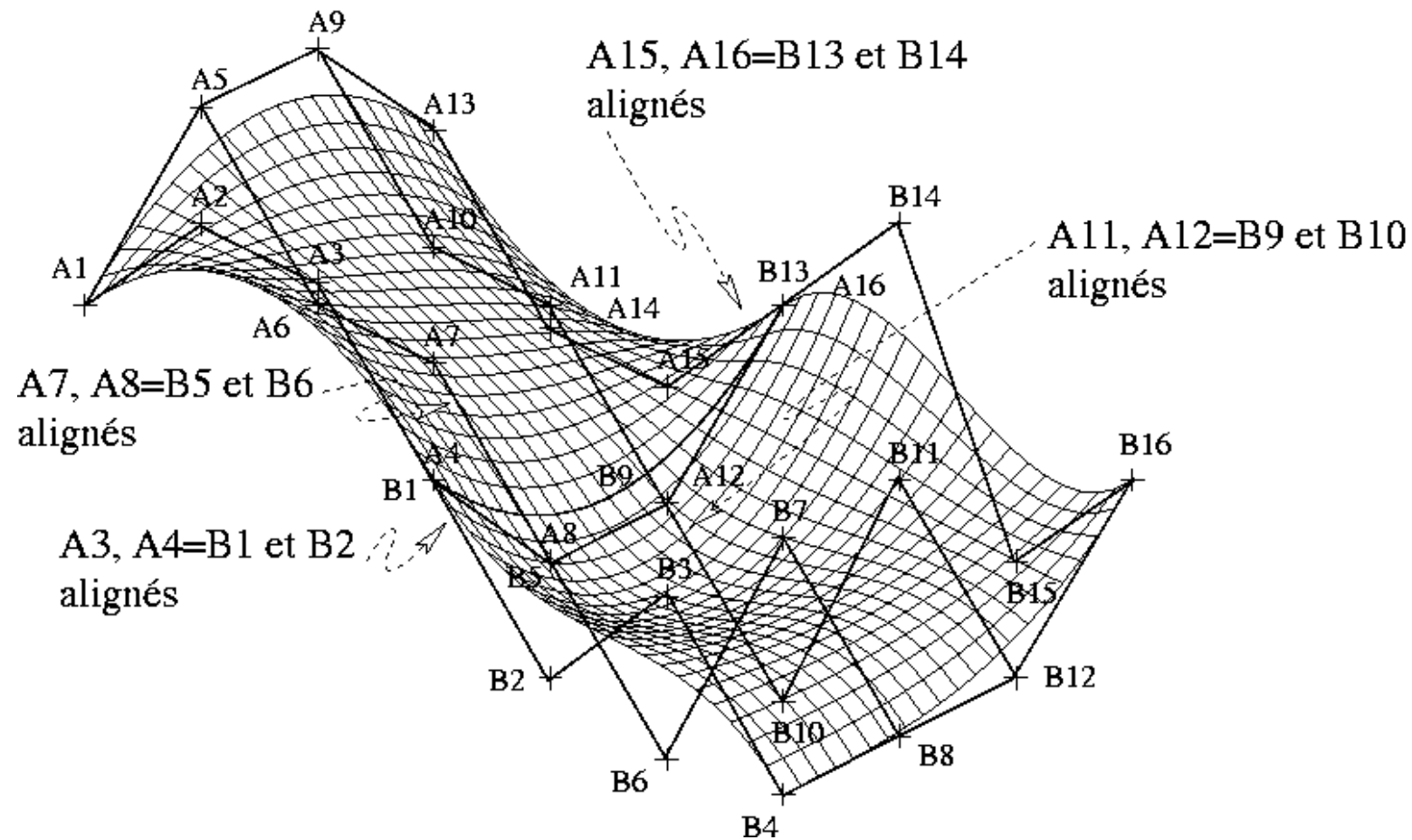
$$k = \min(m, n)$$



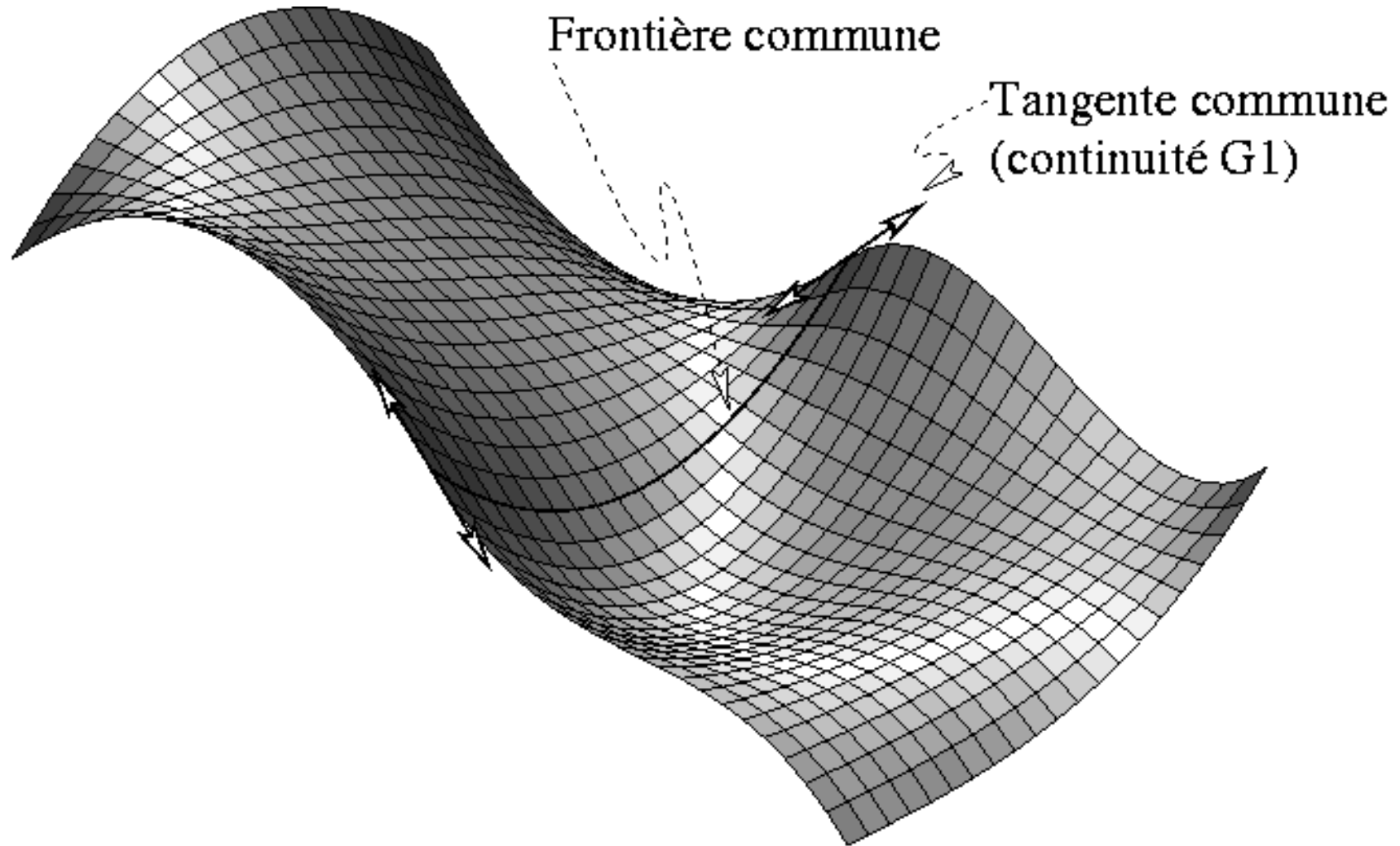
$P_{i,j}^{k,k}$ forme un polygone de contrôle d'une courbe de Bézier

On applique l'algorithme à une variable de De Casteljau

Raccordement de carreaux



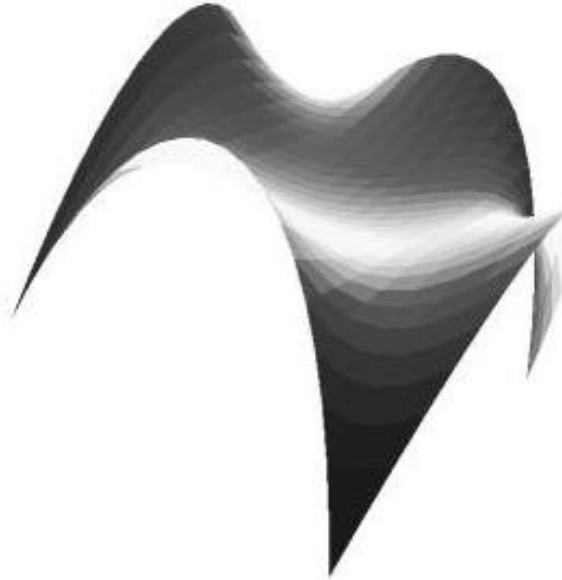
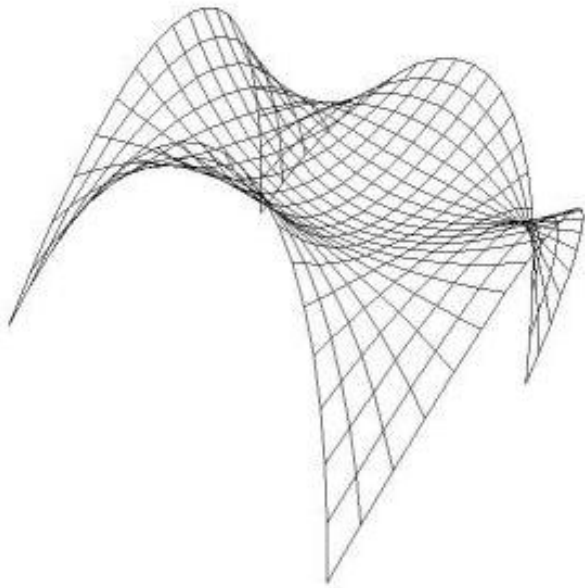
Raccordement de carreaux



NURBS

- Non Uniform Rational B-Splines
- Objectif : représenter sous un même formalisme
 - les courbes et surfaces de forme libre
 - des primitives :
 - cercles
 - coniques (paraboles, hyperboles...)
 - quadriques (cylindres, sphères, cônes...)
 - ...
- NURBS : implémentation de courbes et surfaces paramétriques OpenGL

NURBS



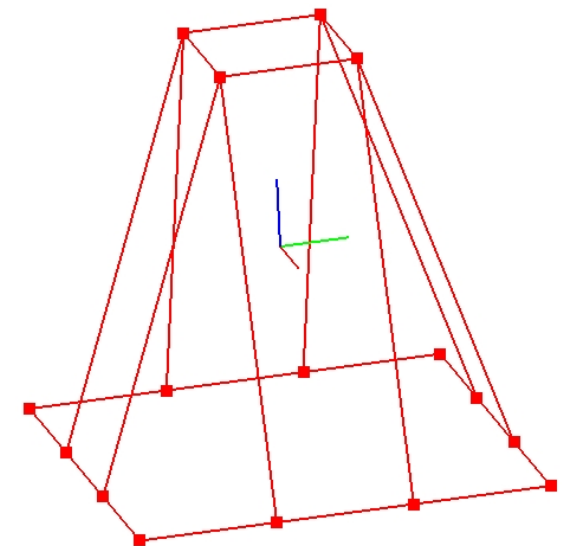
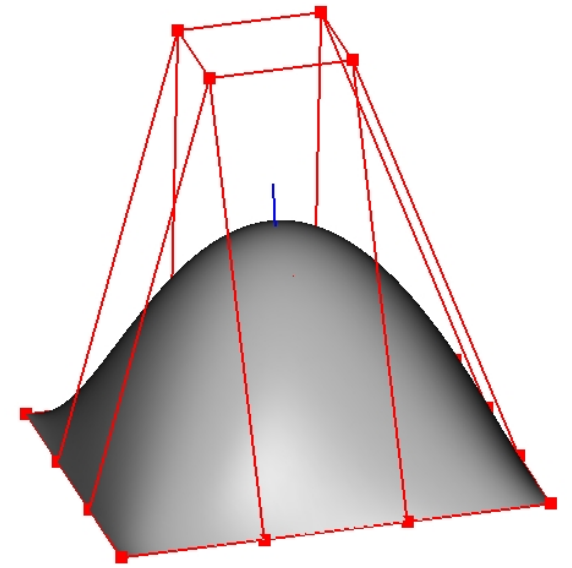
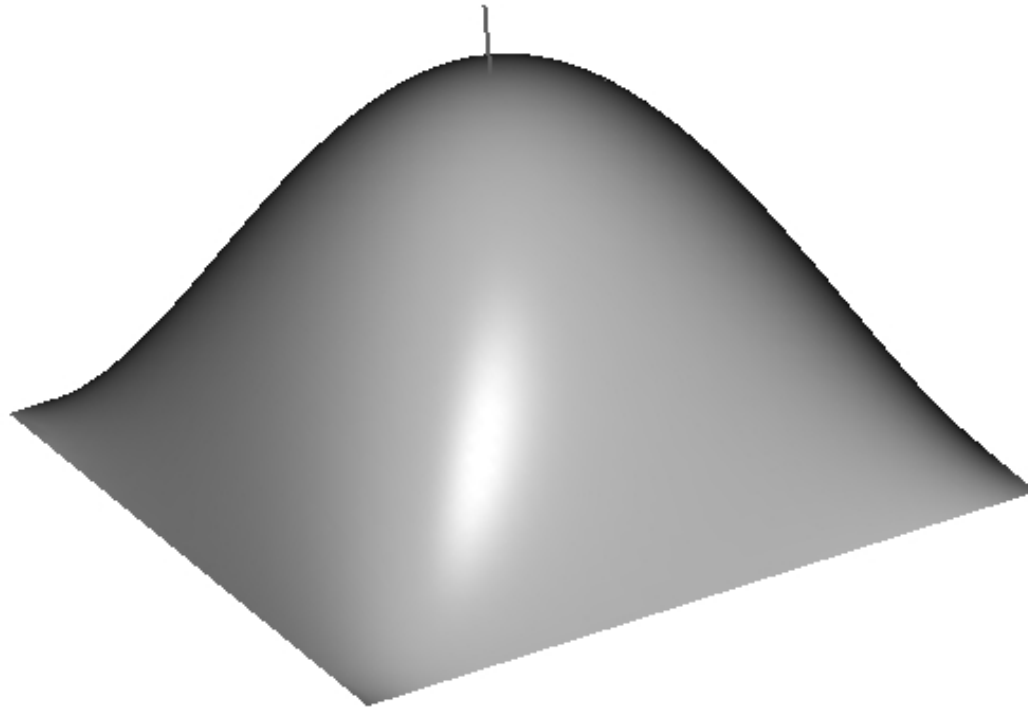
NURBS

- Les surfaces **NURBS** s'obtiennent par extension de la définition des courbes NURBS :
 - deux vecteurs de nœuds T et S , un pour chacune des variables t et s ,
 - $(m+1) \times (n+1)$ points de contrôle $P_{i,k}$
 - $(m+1) \times (n+1)$ fonctions de pondération $R_{m,n,i,k}$ déduites des fonctions de pondération des B-splines $S_{m,k}$ au moyen de $(m+1) \times (n+1)$ **poids** $w_{i,k}$:

$$R_{m,n,i,k}(t) = \frac{w_{i,k} S_{m,i}(t) S_{n,k}(s)}{\sum_{j,l} w_{j,l} S_{m,j}(t) S_{n,l}(s)} \quad N(t,s) = \sum_{i,k} R_{m,n,i,k}(t) P_{i,k}$$

NURBS

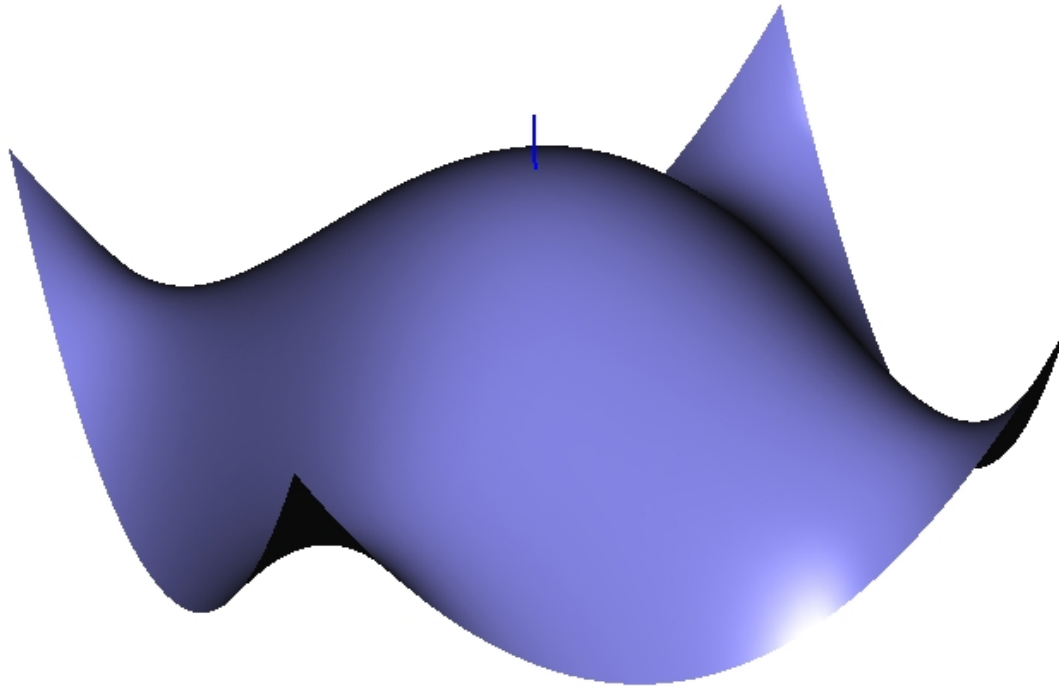
- Exemple OpenGL / GLU



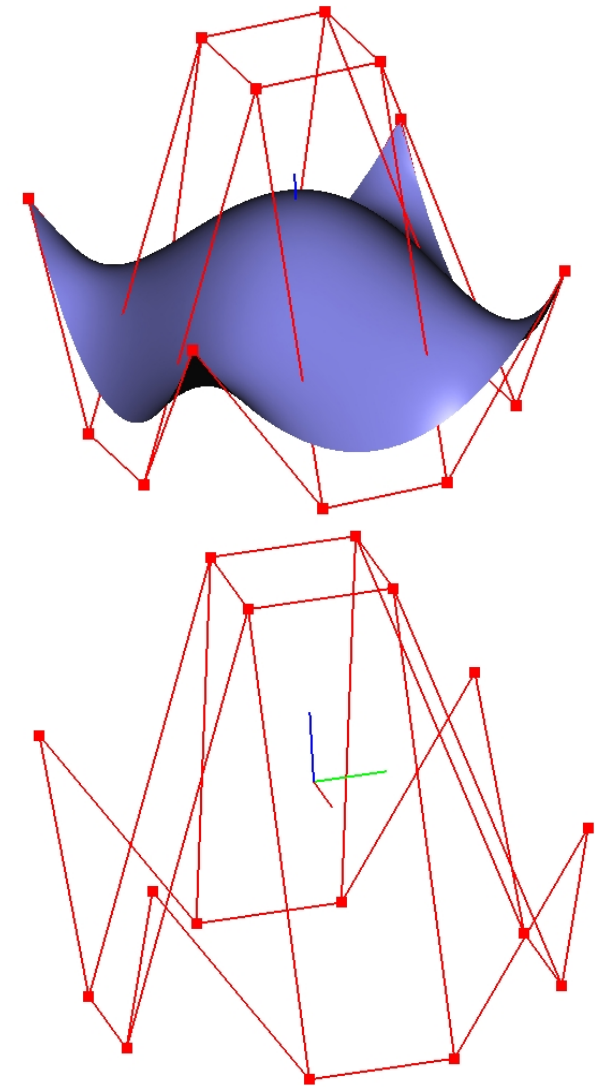
Source du code : The OpenGL Red Book
http://www.opengl.org/documentation/red_book/

NURBS

- Exemple OpenGL / GLU



Source du code : The OpenGL Red Book
http://www.opengl.org/documentation/red_book/



PN triangles

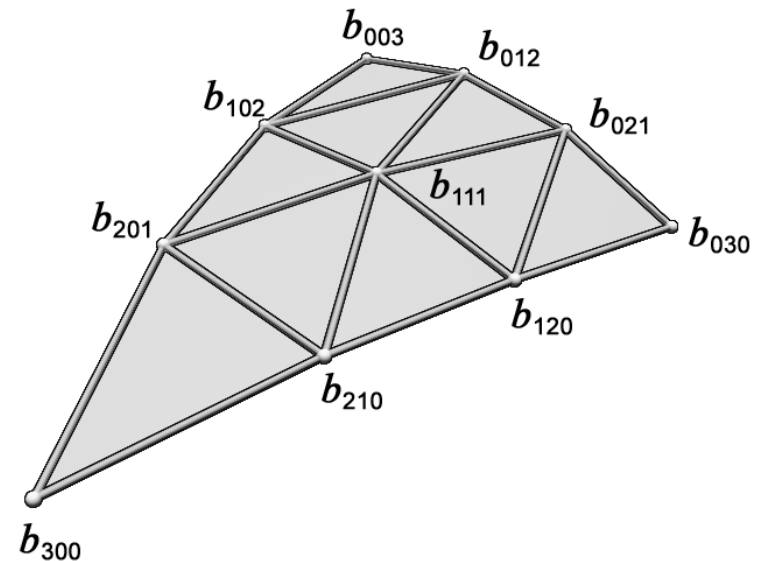
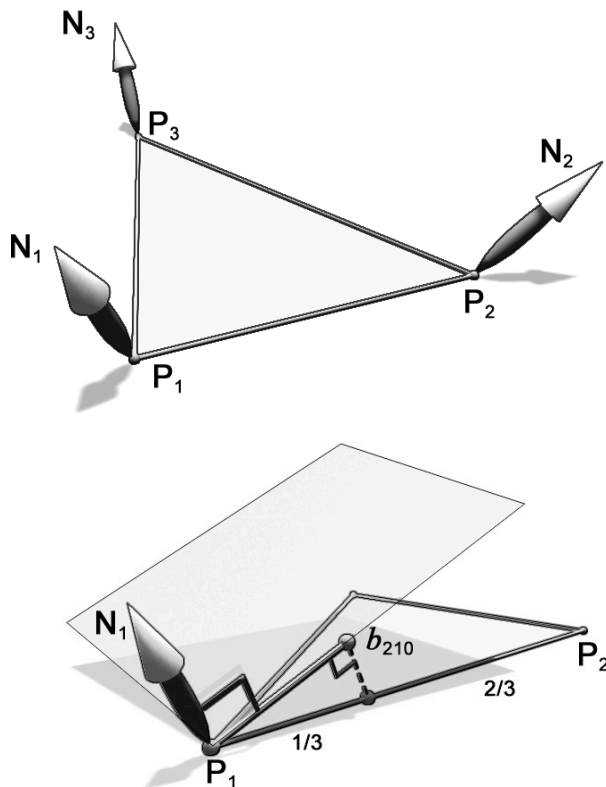
- Principe
 - Une surface construite à partir de 3 sommets et de leurs 3 normales
 - La surface « la plus simple » : pas de données supplémentaires (triangles voisins...)
 - Résultat : on « tire » la surface vers les plans tangents
- Détails : *Curved PN triangles*, A. Vlachos, J. Peters, C. Boyd, J.-L. Mitchell, 2001

PN triangles

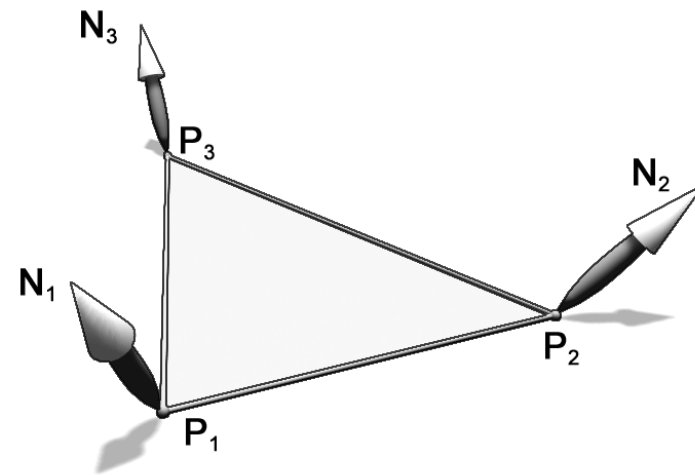
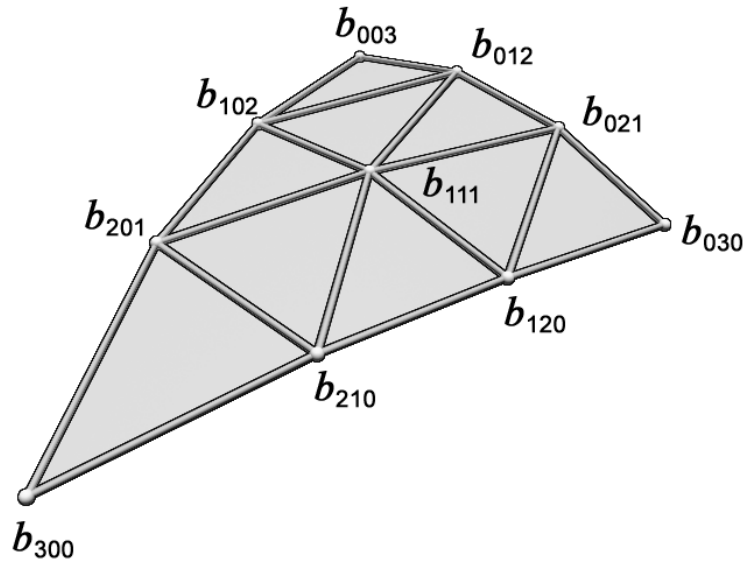
- Principe
 - Construction de 9 points de contrôle à partir des 3 sommets et de leurs normales
 - La surface est définie par un polynôme de degré 3 s'appuyant sur les points de contrôle
 - L'évaluation d'un point de la surface (pour une valeur des paramètres u , v et w) fournit un point de la surface

PN triangles

- Construction des points de contrôle
 - À partir d'un sommet, de sa normale et d'une arête
 - Projette un point de l'arête sur le plan tangent



PN triangles



- Evaluation de polynôme
 - $p(u,v)_{w=1-u-v} = b_{300}w^3 + b_{030}u^3 + b_{003}v^3$
 $+ b_{201}3w^2u + b_{120}3wu^2 + b_{201}3w^2v$
 $+ b_{021}3u^2v + b_{102}3wv^2 + b_{012}3uv^2$
 $+ b_{111}6wuv$

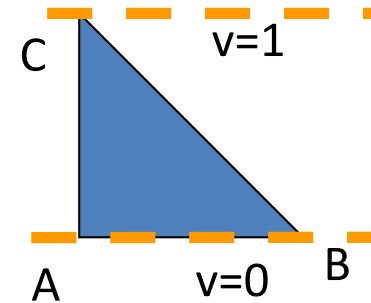
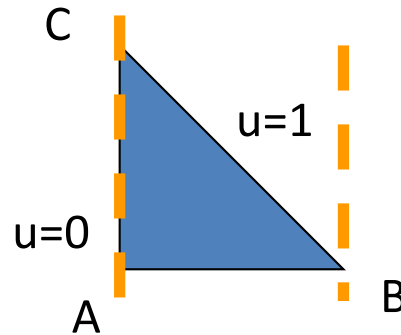
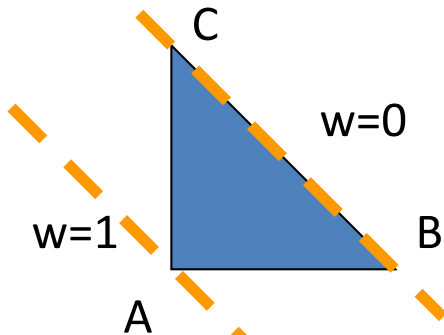
PN triangles

- Evaluation des points de la surface
 - Domaine paramétrique (u, v, w)
 - Contraintes :
 - $u + v + w = 1$
 - $0 \leq u \leq 1$ et $0 \leq v \leq 1$ et $w = 1 - u - v$
- Rappel
 - On peut évaluer les points d'un triangle abc de la même manière : $p(u, v)_{w=1-u-v} = wA + uB + vC$

PN triangles

- Domaine paramétrique

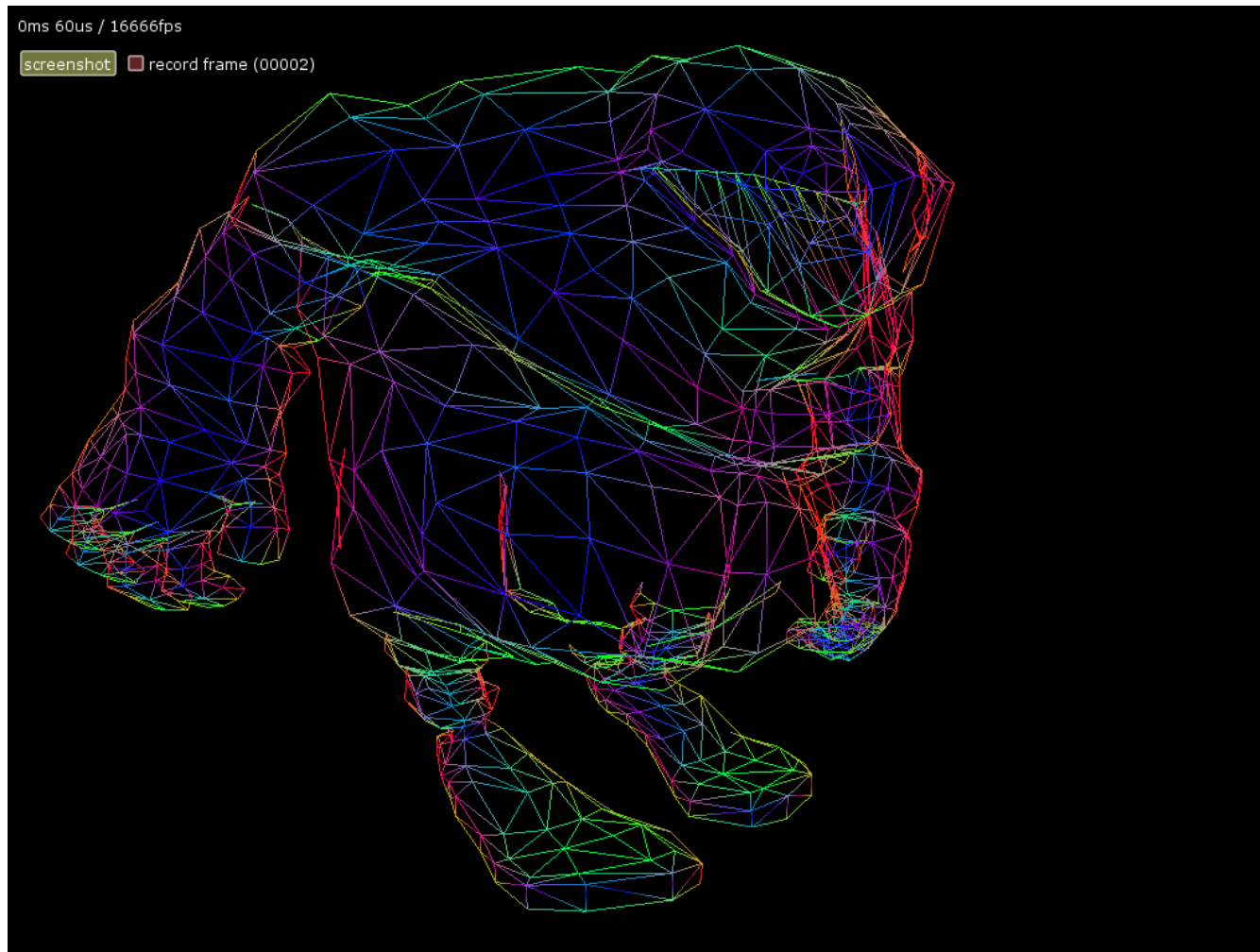
- $p(0,0,1) = A$
- $p(1,0,0) = B$
- $p(0,1,0) = C$



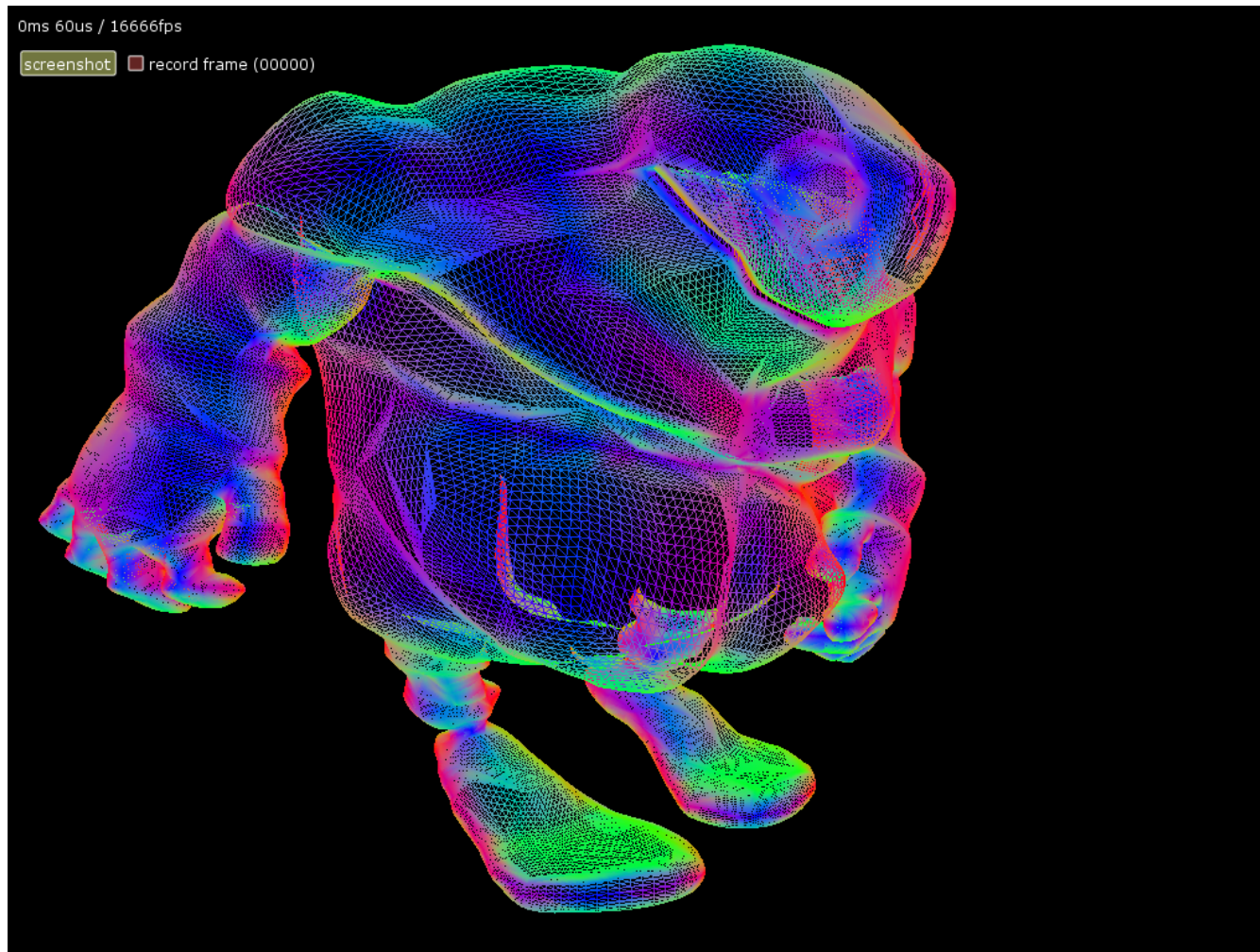
- Exercices

- Quelles sont les valeurs de u et v pour le milieu du triangle ?
- Quelle position (x, y, z) correspond au milieu du triangle ?
- Mêmes questions pour le milieu de chaque arête du triangle

PN triangles



PN triangles

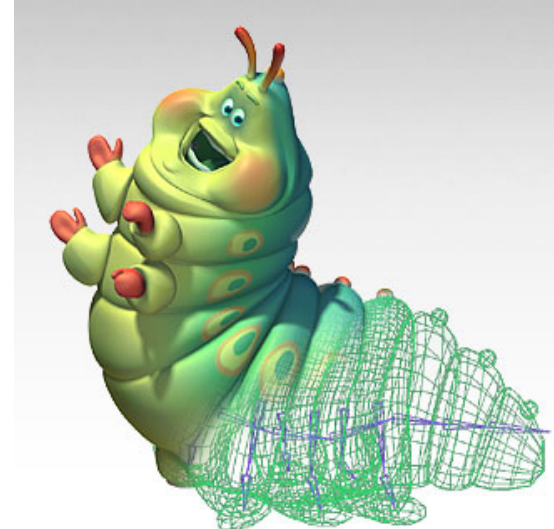
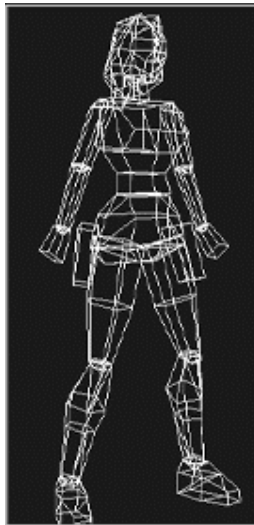


Seconde partie

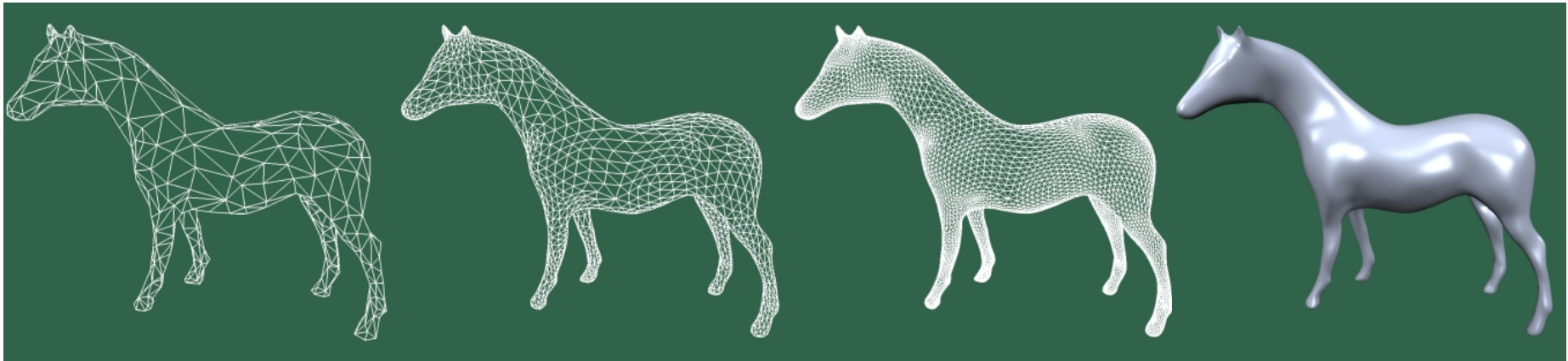
Courbes et surfaces de subdivision

Surfaces paramétriques

- Carreaux de Bézier, B-splines, NURBS...
- Avantages
 - Modélisation par carreau type couture
 - Chaque carreau est défini par un polygone de contrôle
- Difficultés
 - contrôle des connexions entre carreaux
 - ajuster la surface à un polygone de contrôle quelconque



Surfaces de subdivision

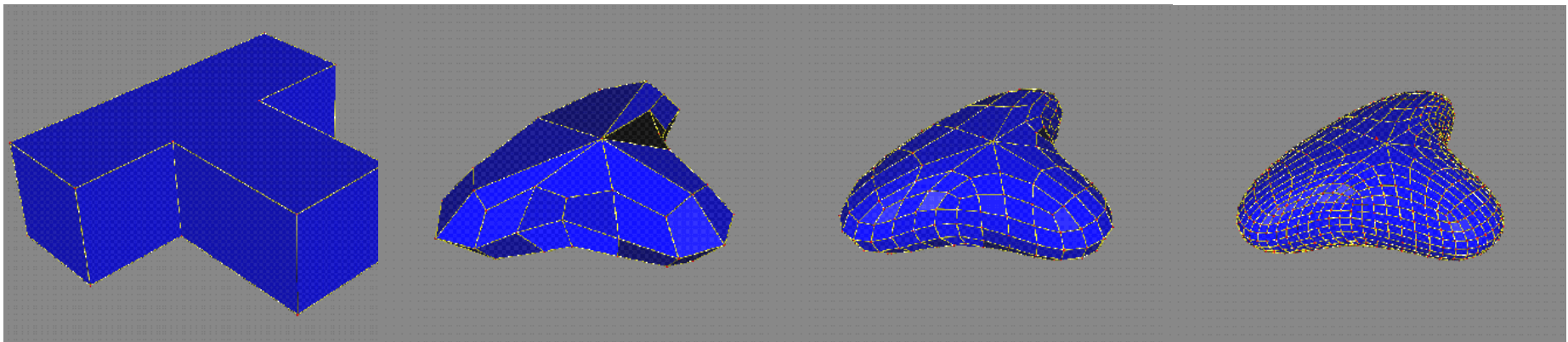


maillage initial

après 1 pas de
subdivision

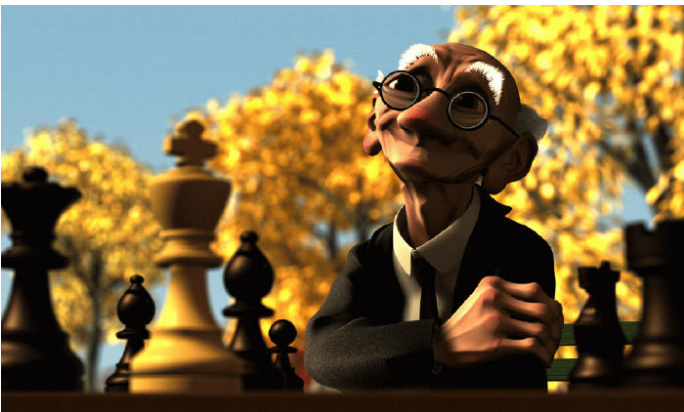
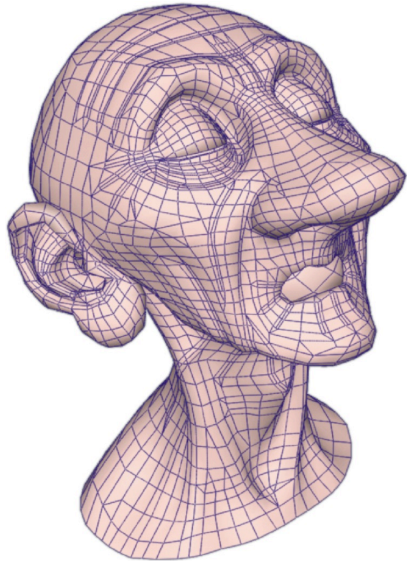
après 2 pas de
subdivision

surface limite

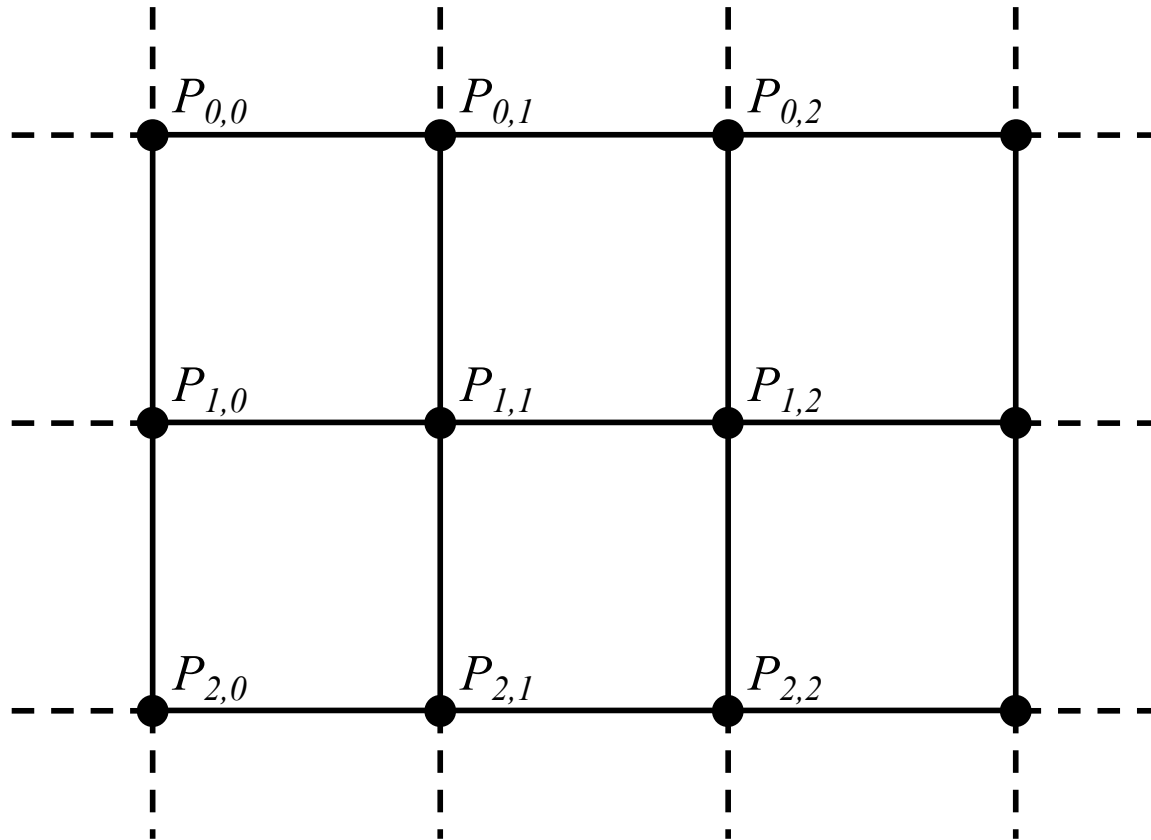


1^{ère} utilisation

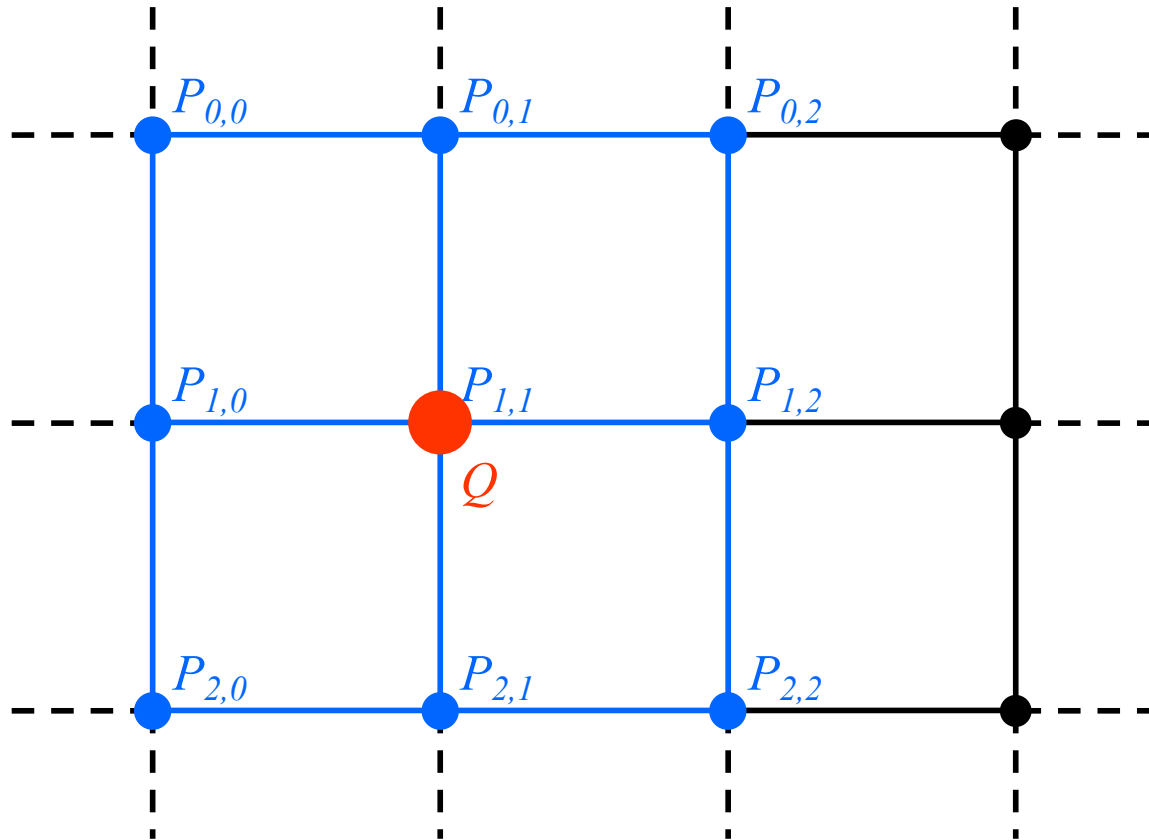
- Geri's Game (1998) : Pixar Animation Studios



Principe



Principe



$$\begin{array}{ccccc}
 1 & \text{---} & 6 & \text{---} & 1 \\
 | & & | & & | \\
 6 & \text{---} & 36 & \text{---} & 6 \\
 | & & | & & | \\
 1 & \text{---} & 6 & \text{---} & 1
 \end{array}$$

$$\begin{aligned}
 Q = & \frac{1}{64} P_{0,0} + \frac{6}{64} P_{0,1} + \frac{1}{64} P_{0,2} \\
 & + \frac{6}{64} P_{1,0} + \frac{36}{64} P_{1,1} + \frac{6}{64} P_{1,2} \\
 & + \frac{1}{64} P_{2,0} + \frac{6}{64} P_{2,1} + \frac{1}{64} P_{2,2}
 \end{aligned}$$

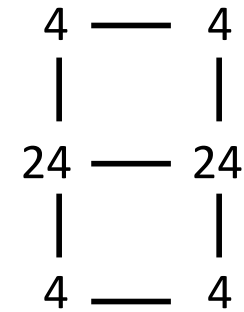
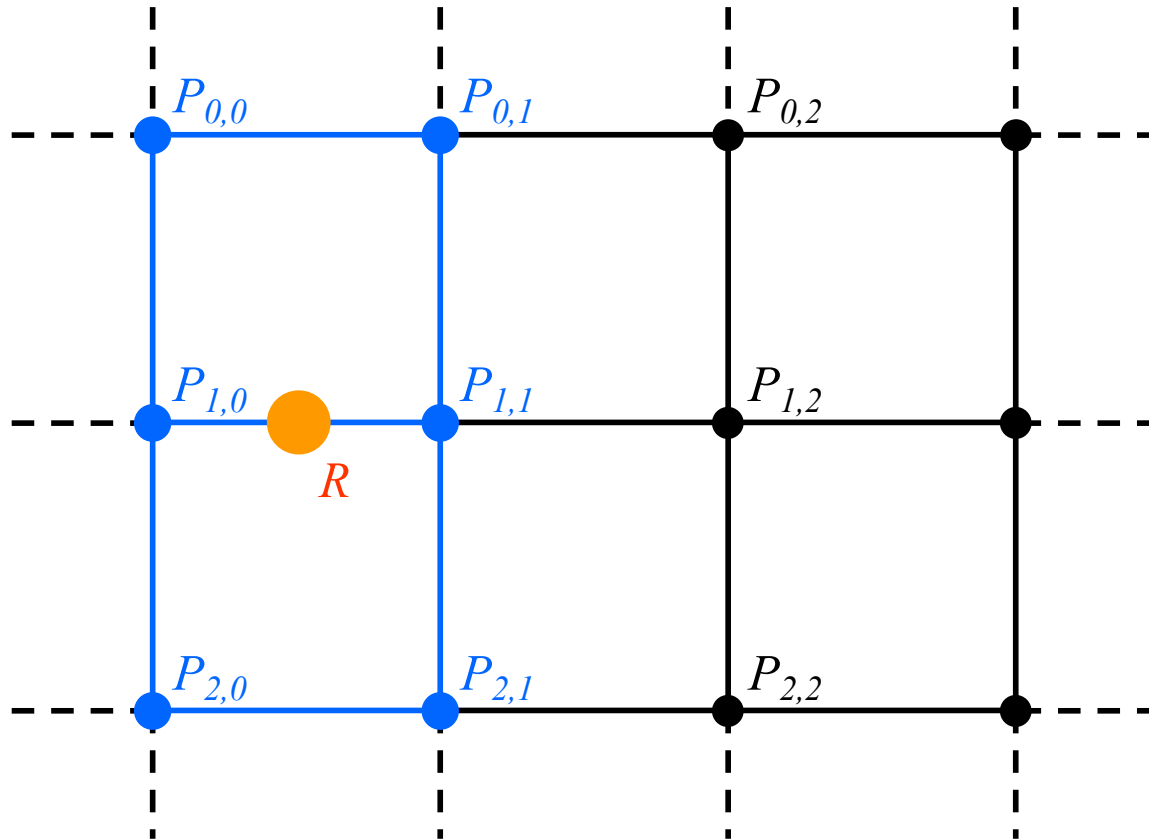
1. Chaque sommet $P_{i,j}$ est **déplacé**

La nouvelle position Q dépend

- de $P_{i,j}$
- de ses voisins

➔ **notion de masque**

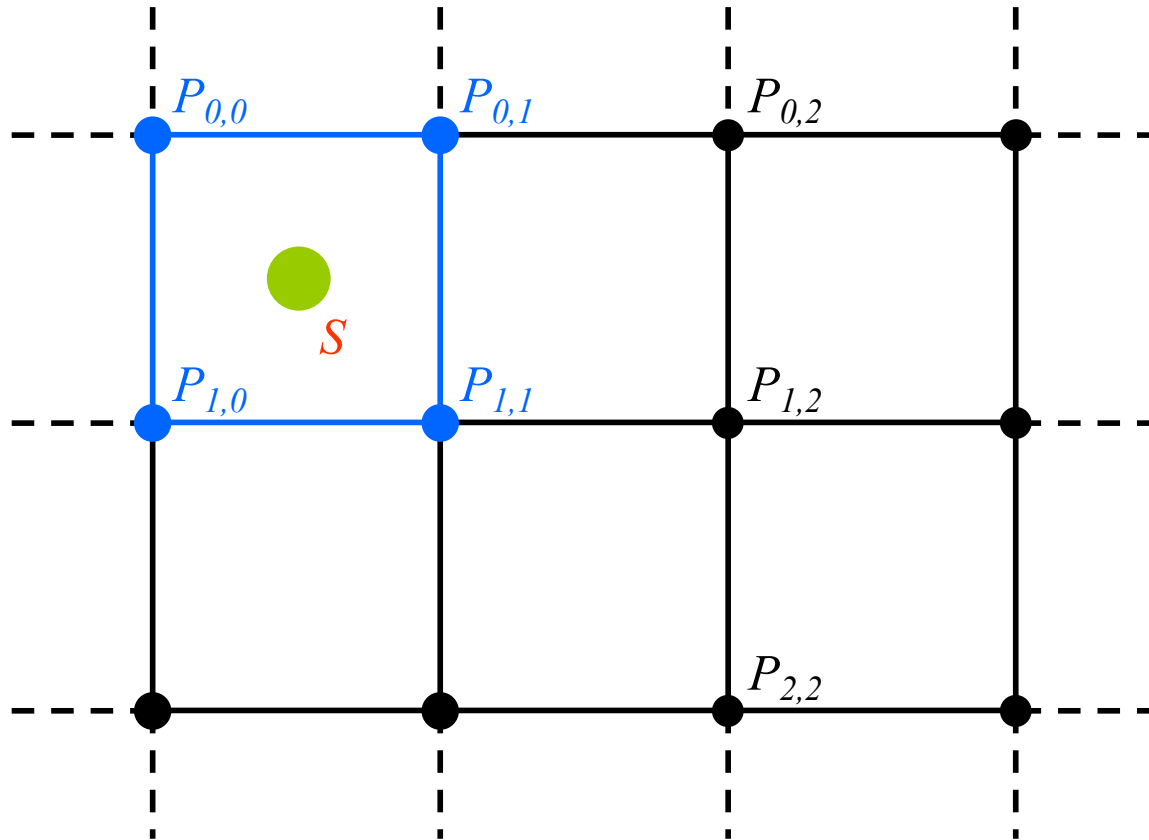
Principe



$$R = \frac{4}{64} P_{0,0} + \frac{4}{64} P_{0,1} + \frac{24}{64} P_{1,0} + \frac{24}{64} P_{1,1} + \frac{4}{64} P_{2,0} + \frac{4}{64} P_{2,1}$$

2. De nouveaux points sont **insérés**
 - a. sur chaque **arête**

Principe

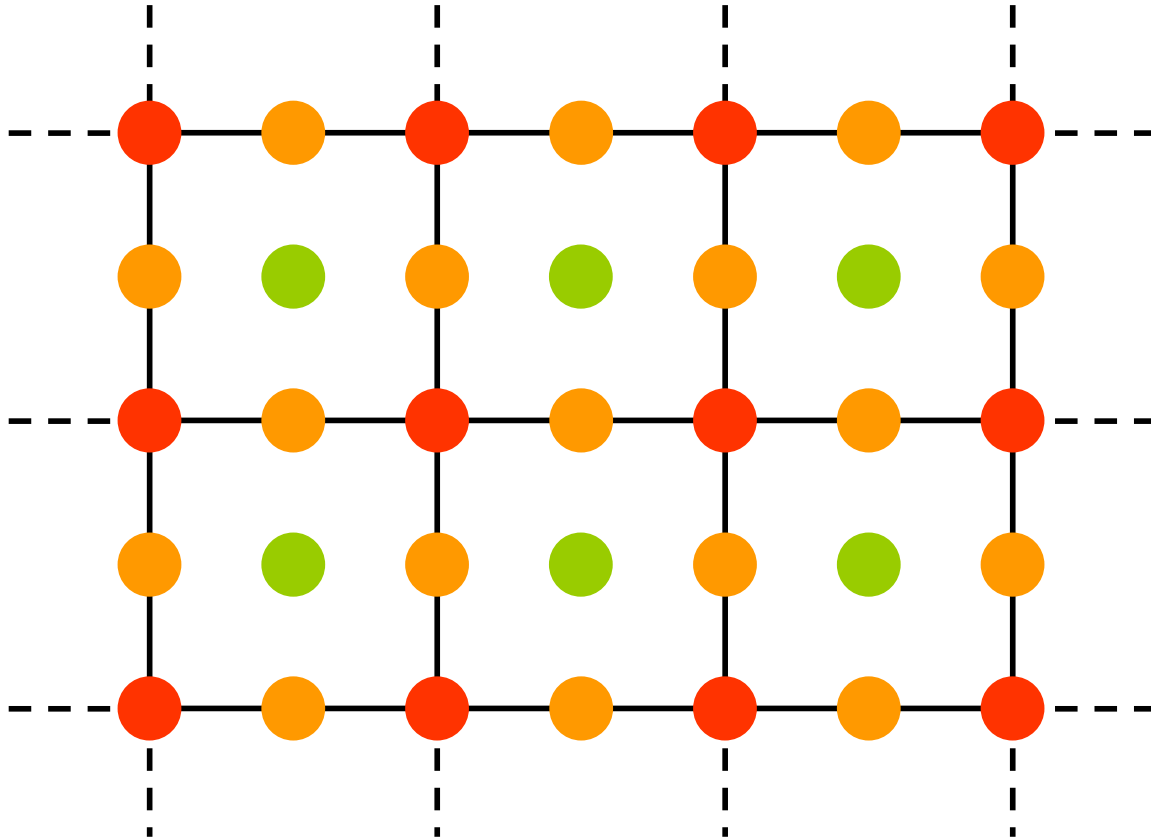


$$\begin{array}{cc} 16 & \text{---} & 16 \\ | & & | \\ 16 & \text{---} & 16 \end{array}$$

$$S = \frac{16}{64} P_{0,0} + \frac{16}{64} P_{0,1} + \frac{16}{64} P_{1,0} + \frac{16}{64} P_{1,1}$$

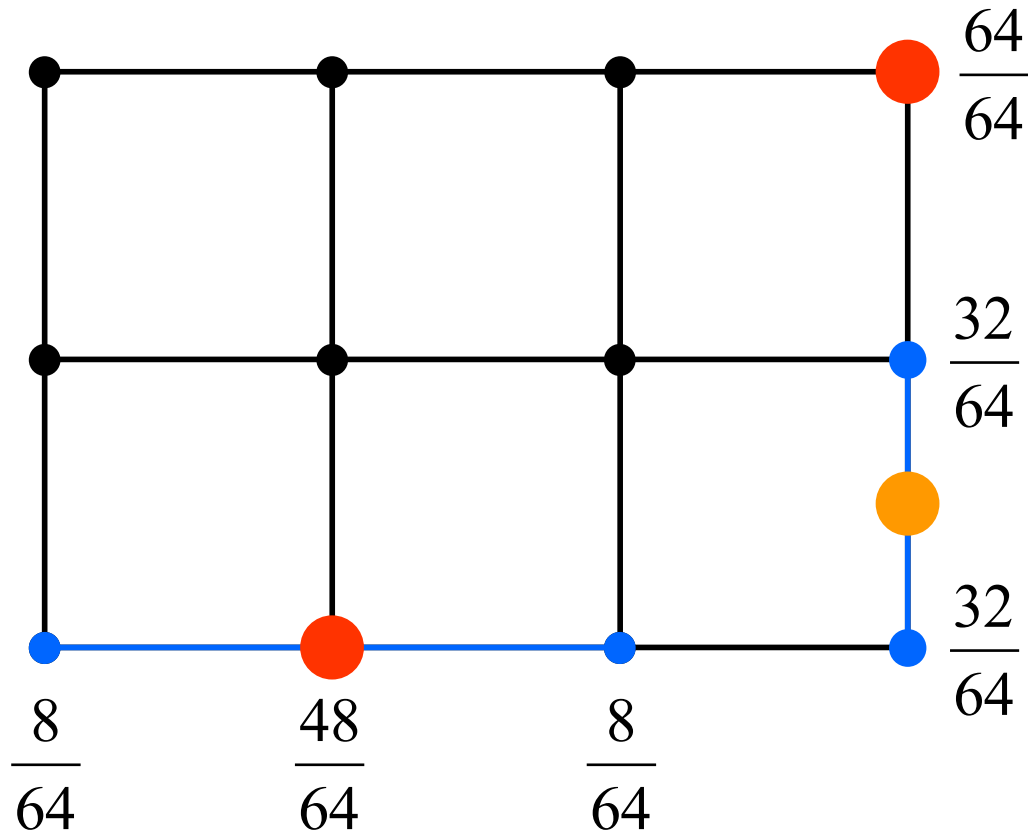
2. De nouveaux points sont **insérés**
 - b. sur chaque **face** (éventuellement)

Principe



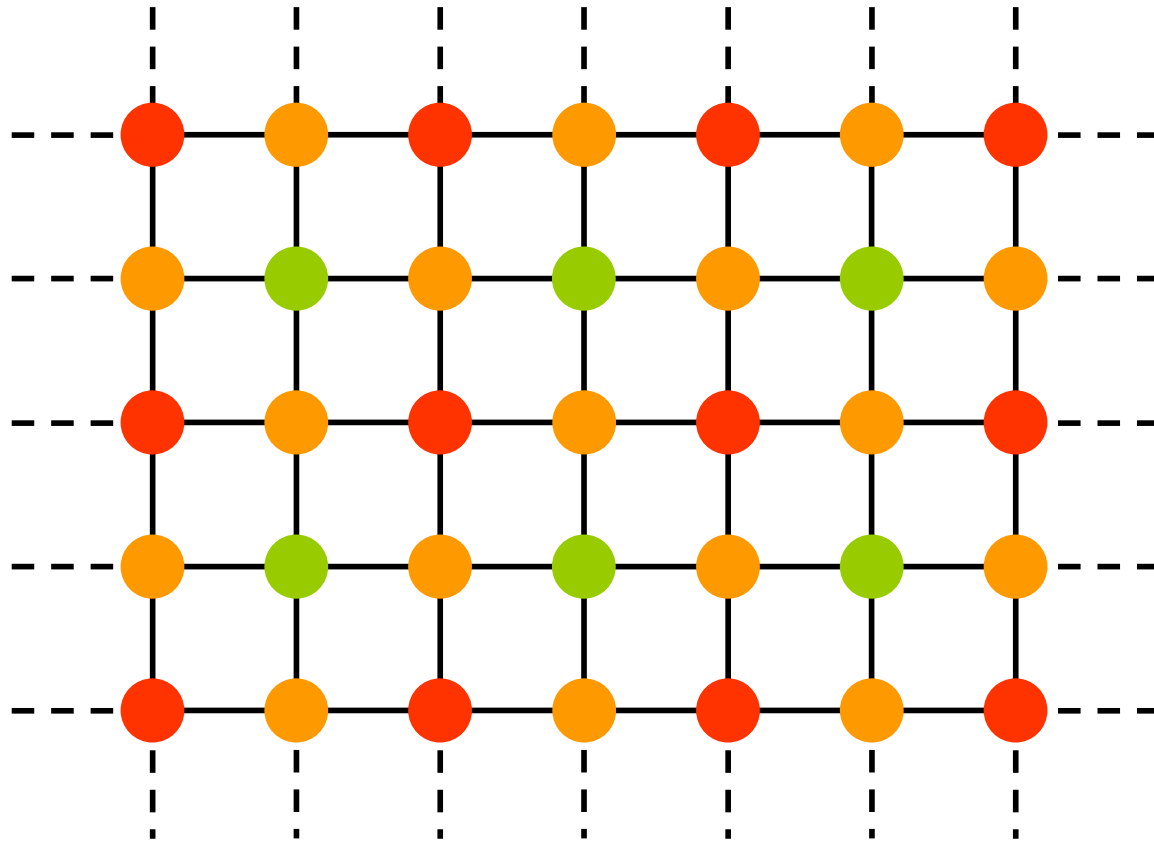
Les étapes 1 et 2 sont répétées sur tout le maillage initial

Principe



Masques spécifiques pour les bords

Principe



Les nouveaux points sont maillés

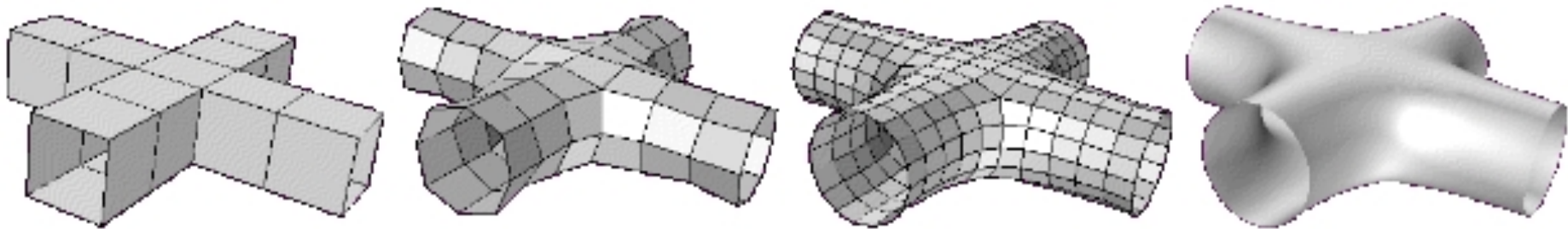
- ➔ tout ceci correspond à **un** pas de subdivision
- ➔ à réitérer jusqu'à la surface limite

Catmull-Clark
(grille rectangulaire)



Principe

- Relativement facile avec un maillage régulier fermé
- Autres masques pour
 - maillages quelconques
 - bords
- Disposer d'une structure de données permettant
 - "pour chaque sommet, arête, face...",
 - "couper une face en deux..."



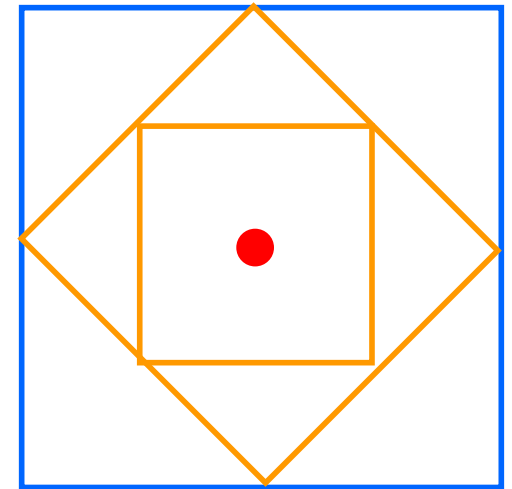
Courbes de subdivision

Principe

- Courbe limite à partir d'un carré

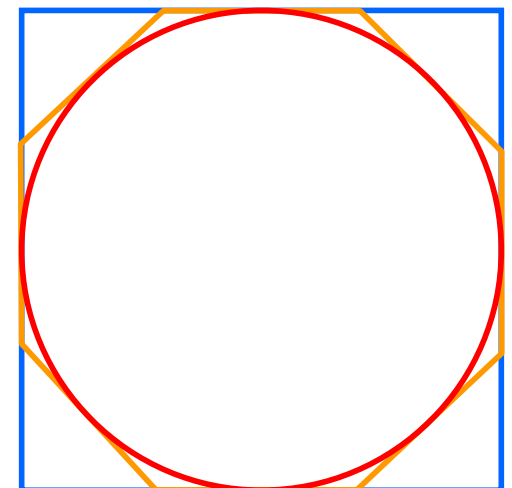
Chaque arête est coupée en 2

Courbe limite



Chaque arête est coupée en 3

Courbe limite



B-Splines uniformes (rappel)

- Courbes B-Splines uniformes

$$P(t) = [t^n \ t^{n-1} \ \dots \ 1] M \begin{bmatrix} P_0 \\ \vdots \\ P_n \end{bmatrix}$$

- Quadratique (n=2)

$$M = \begin{bmatrix} 1 & -2 & 2 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

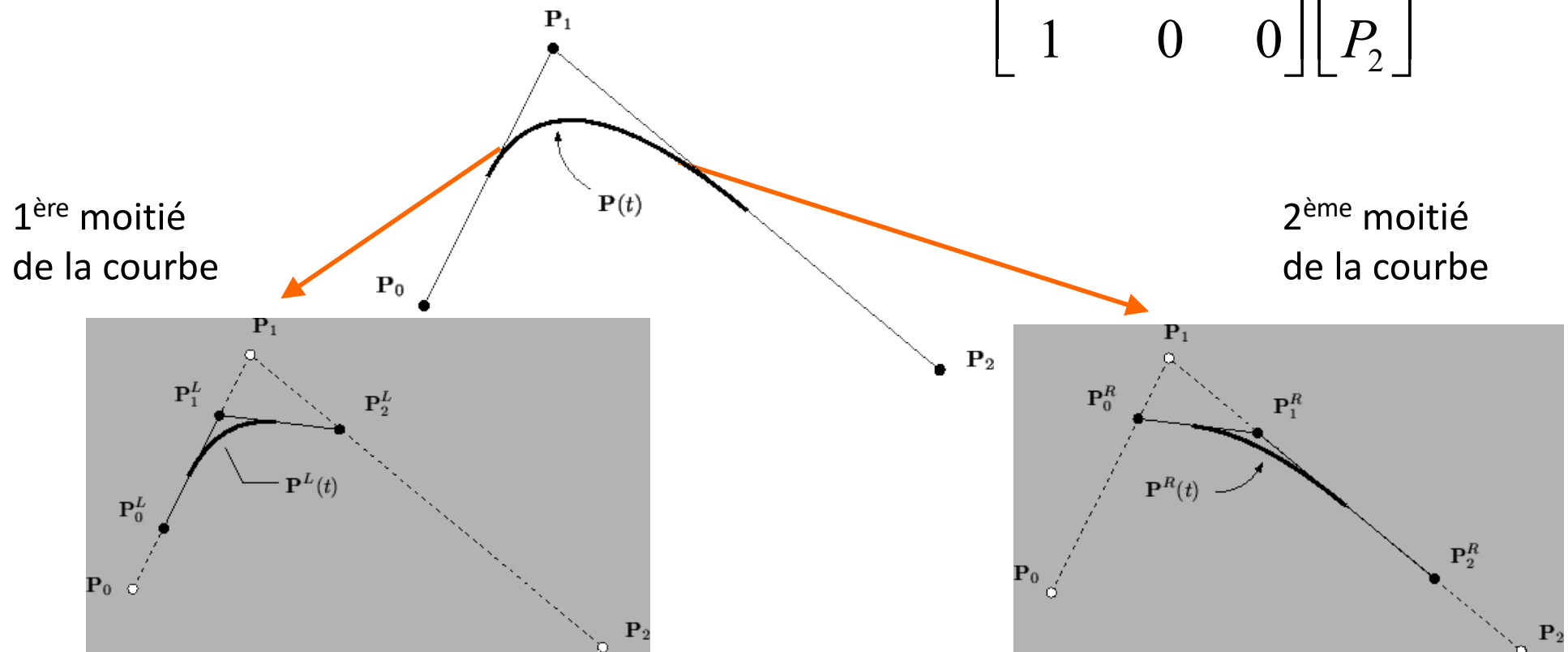
- Quintique (n=3)

$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

B-Splines uniformes quadratiques

- Courbes B-Splines uniformes

$$P(t) = \begin{bmatrix} t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$



B-Splines uniformes quadratiques

- 1^{ère} moitié de la courbe

$$P_{[0,1/2]}(t) = P\left(\frac{t}{2}\right) = \left[\left(\frac{t}{2}\right)^2 \quad \frac{t}{2} \quad 1\right] \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

$$= [t^2 \ t \ 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} M \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

$$\text{avec } M = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

$$= [t^2 \ t \ 1] M M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} M \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

$$= [t^2 \ t \ 1] M \left(M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} M \right) \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

$$= [t^2 \ t \ 1] M \begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \end{bmatrix}$$

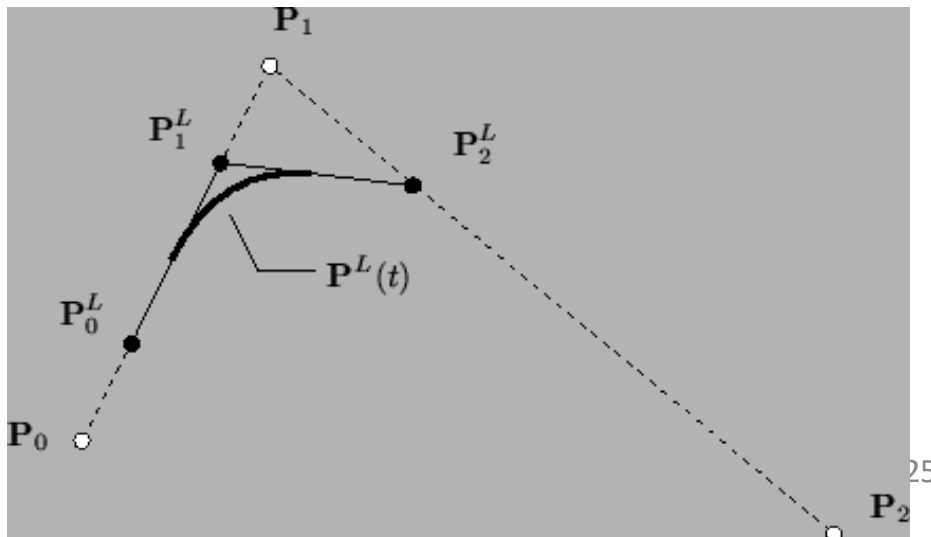
$$\text{avec } \begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \end{bmatrix} = \left(M^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/4 \end{bmatrix} M \right) \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3P_0 + P_1 \\ P_0 + 3P_1 \\ 3P_1 + P_2 \end{bmatrix}$$

B-Splines uniformes quadratiques

- 1^{ère} moitié de la courbe

$$P^L(t) = [t^2 \ t \ 1]M \begin{bmatrix} P_0^L \\ P_1^L \\ P_2^L \end{bmatrix}$$

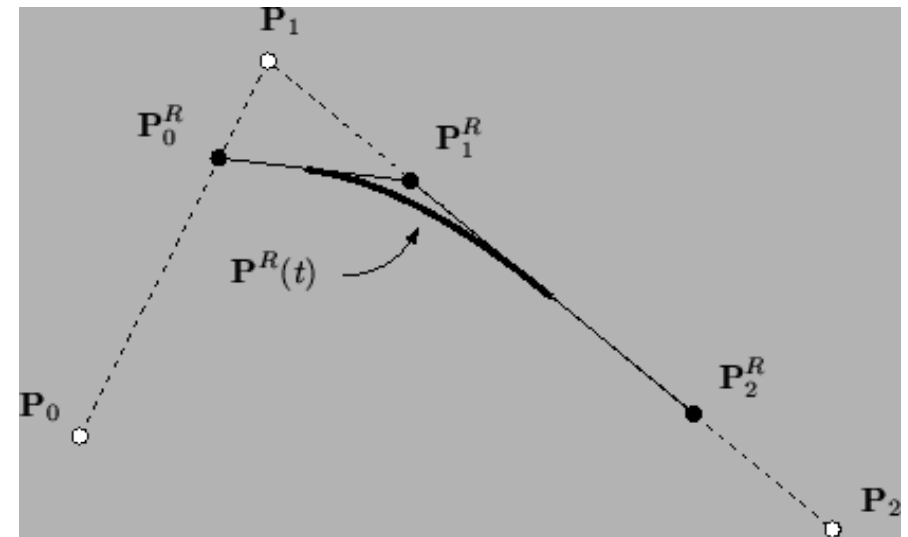
$$\begin{bmatrix} P_0^L \\ P_1^L \\ P_2^L \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3P_0 + P_1 \\ P_0 + 3P_1 \\ 3P_1 + P_2 \end{bmatrix}$$



- 2^{ème} moitié de la courbe

$$P^R(t) = [t^2 \ t \ 1]M \begin{bmatrix} P_0^R \\ P_1^R \\ P_2^R \end{bmatrix}$$

$$\begin{bmatrix} P_0^R \\ P_1^R \\ P_2^R \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} P_0 + 3P_1 \\ 3P_1 + P_2 \\ P_1 + 3P_2 \end{bmatrix}$$



B-Splines uniformes quadratiques

- On regroupe
$$\begin{bmatrix} P_0^L \\ P_1^L \\ P_2^L \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} \quad \begin{bmatrix} P_0^R \\ P_1^R \\ P_2^R \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix}$$

→ algorithme de Chaikin

$$\begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3P_0 + P_1 \\ P_0 + 3P_1 \\ 3P_1 + P_2 \\ P_1 + 3P_2 \end{bmatrix}$$

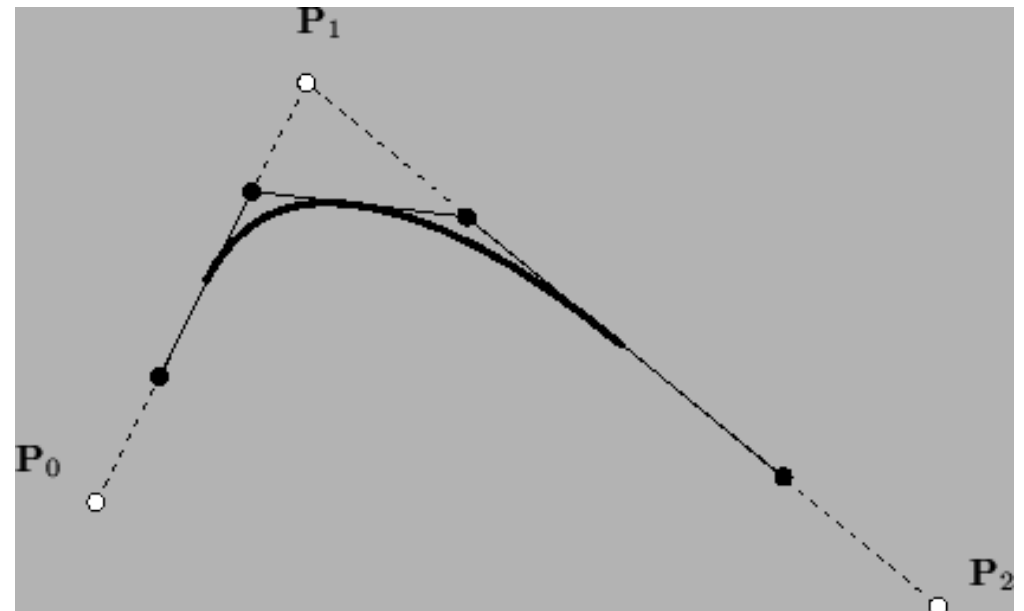


Schéma de Chaikin

- Encore appelé corner-cutting

$$\begin{bmatrix} Q_0 \\ Q_1 \\ Q_2 \\ Q_3 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \\ 0 & 1 & 3 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \end{bmatrix} = \frac{1}{4} \begin{bmatrix} 3P_0 + P_1 \\ P_0 + 3P_1 \\ 3P_1 + P_2 \\ P_1 + 3P_2 \end{bmatrix} \Rightarrow \begin{bmatrix} Q_{2i} \\ Q_{2i+1} \end{bmatrix} = \begin{bmatrix} \frac{3}{4}P_i + \frac{1}{4}P_{i+1} \\ \frac{1}{4}P_i + \frac{3}{4}P_{i+1} \end{bmatrix}$$

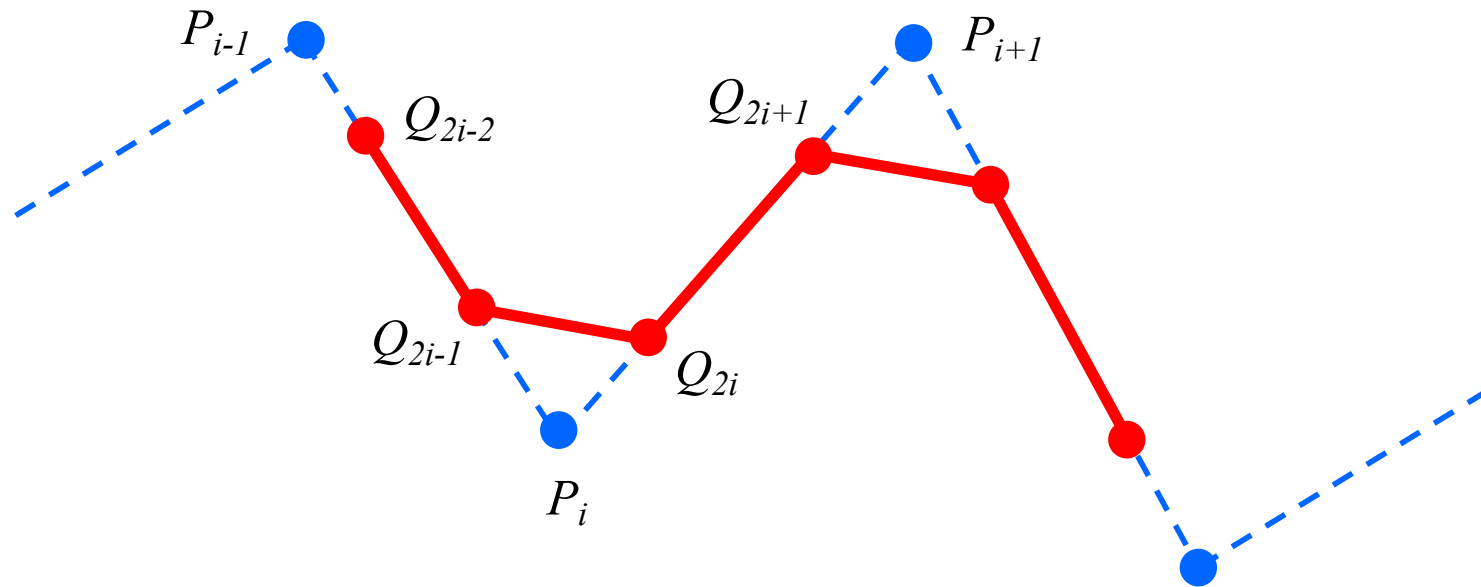


Schéma de Chaikin

$$(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-3}$$

- Schéma régulier **approximant**

$$\text{Pour } 0 \leq i \leq n-2 \begin{cases} Q_{2i} = aP_i + (1-a)P_{i+1} \\ Q_{2i+1} = bP_i + (1-b)P_{i+1} \end{cases} \text{ avec } a = \frac{3}{4} \text{ et } b = \frac{1}{4}$$

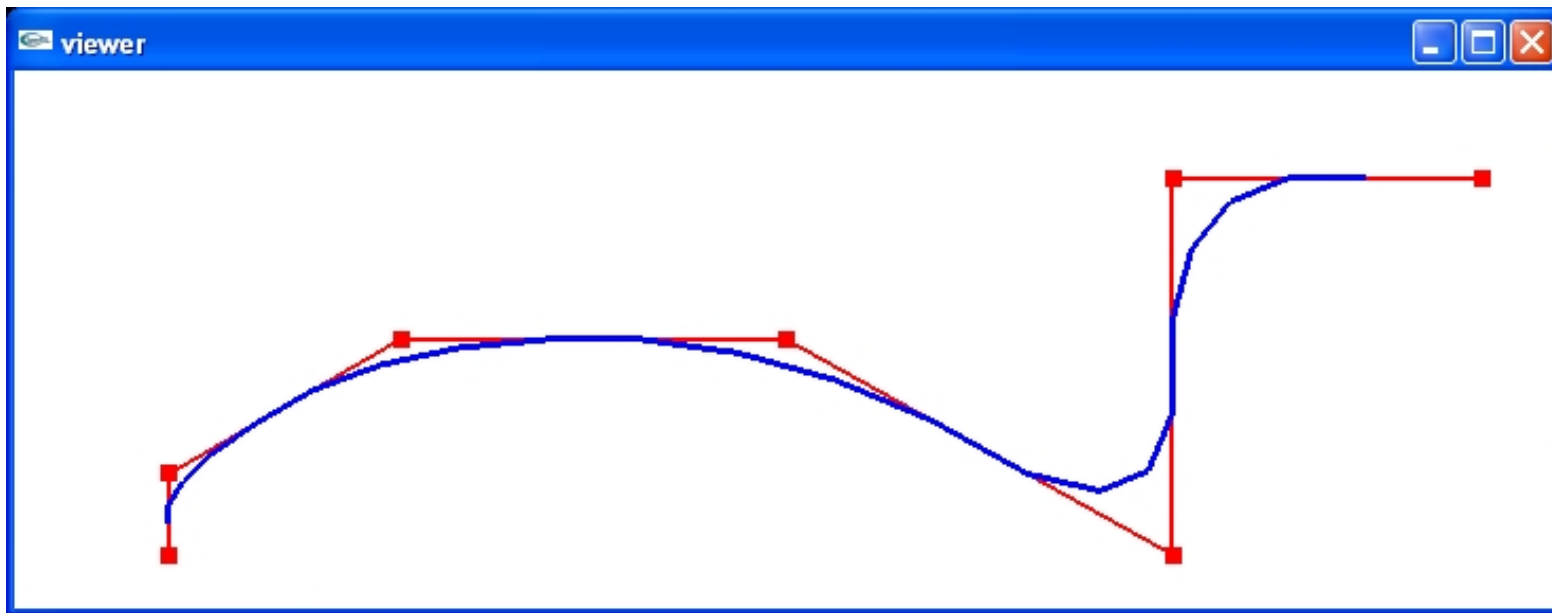


Schéma de Catmull-Clark

- Schéma régulier **approximant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-4}$
- $$\left\{ \begin{array}{l} \text{Pour } 0 \leq i \leq n-1 \quad Q_{2i} = \frac{1}{2}P_i + \frac{1}{2}P_{i+1} \\ \text{Pour } 0 \leq i \leq n-2 \quad Q_{2i+1} = \frac{1}{8}P_i + \frac{6}{8}P_{i+1} + \frac{1}{8}P_{i+2} \end{array} \right.$$

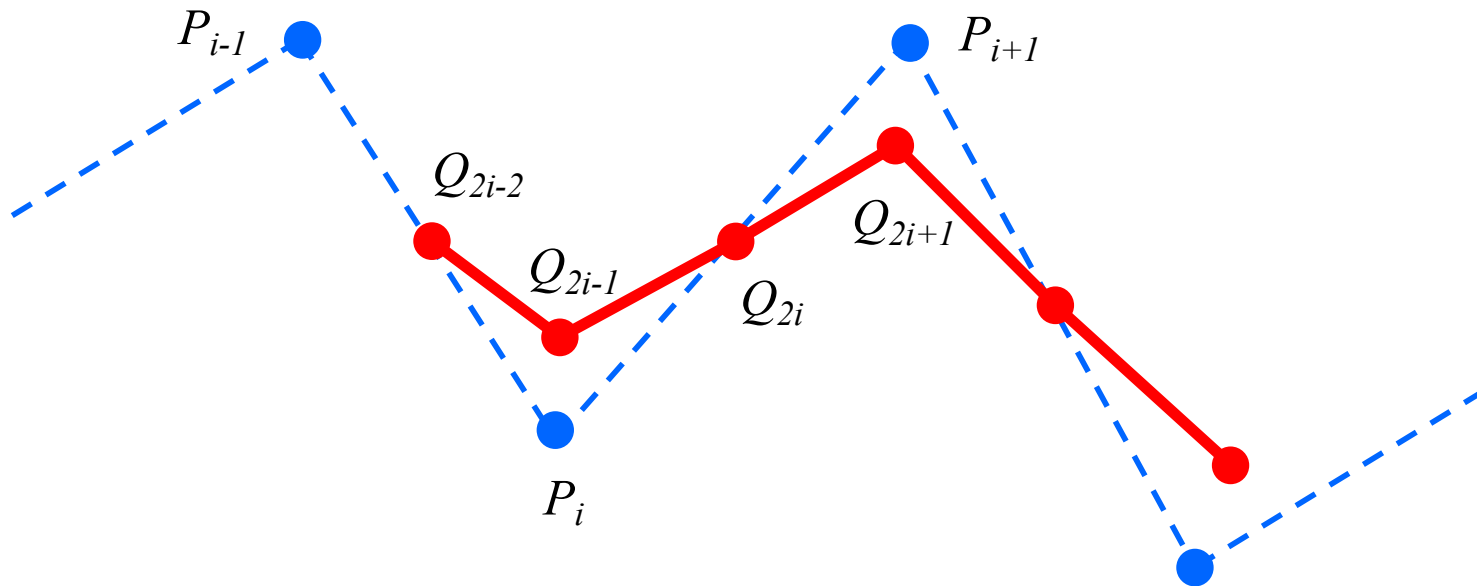
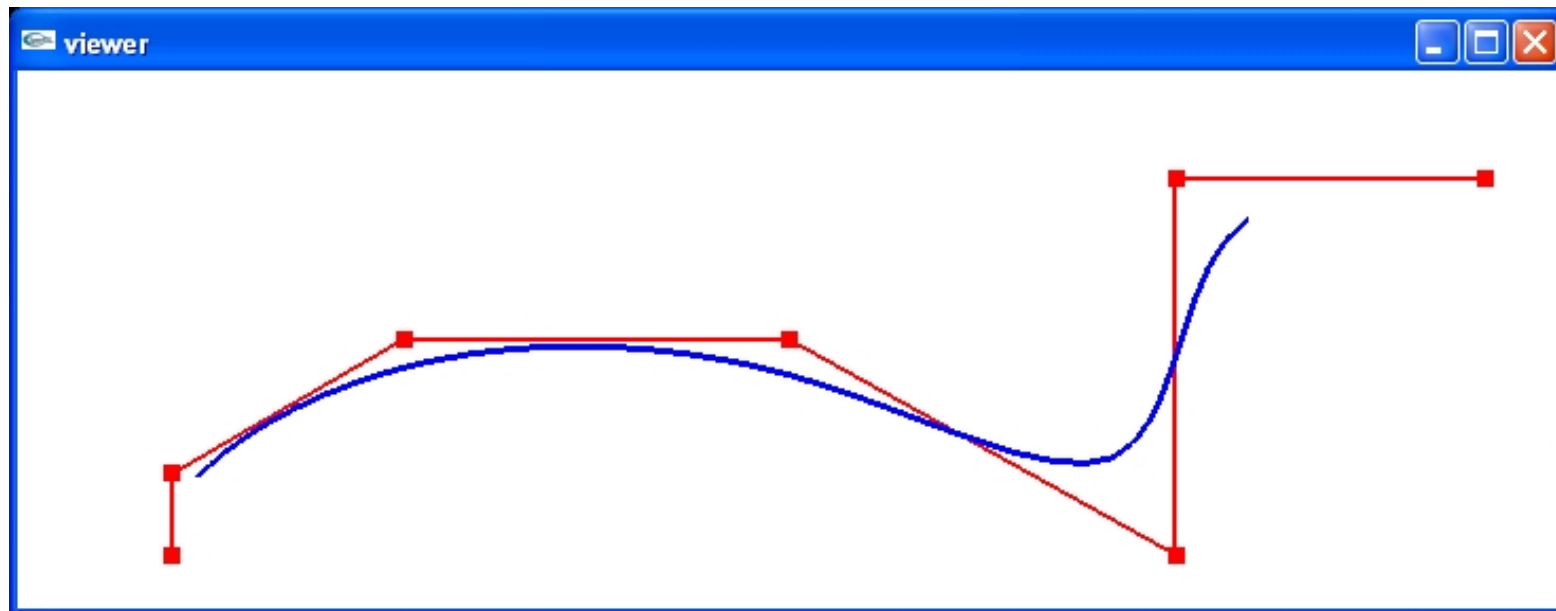


Schéma de Catmull-Clark

- Schéma régulier **approximant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-4}$

$$\left\{ \begin{array}{l} \text{Pour } 0 \leq i \leq n-1 \quad Q_{2i} = \frac{1}{2}P_i + \frac{1}{2}P_{i+1} \\ \text{Pour } 0 \leq i \leq n-2 \quad Q_{2i+1} = \frac{1}{8}P_i + \frac{6}{8}P_{i+1} + \frac{1}{8}P_{i+2} \end{array} \right.$$



Four points scheme

- Schéma régulier **interpolant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-2}$

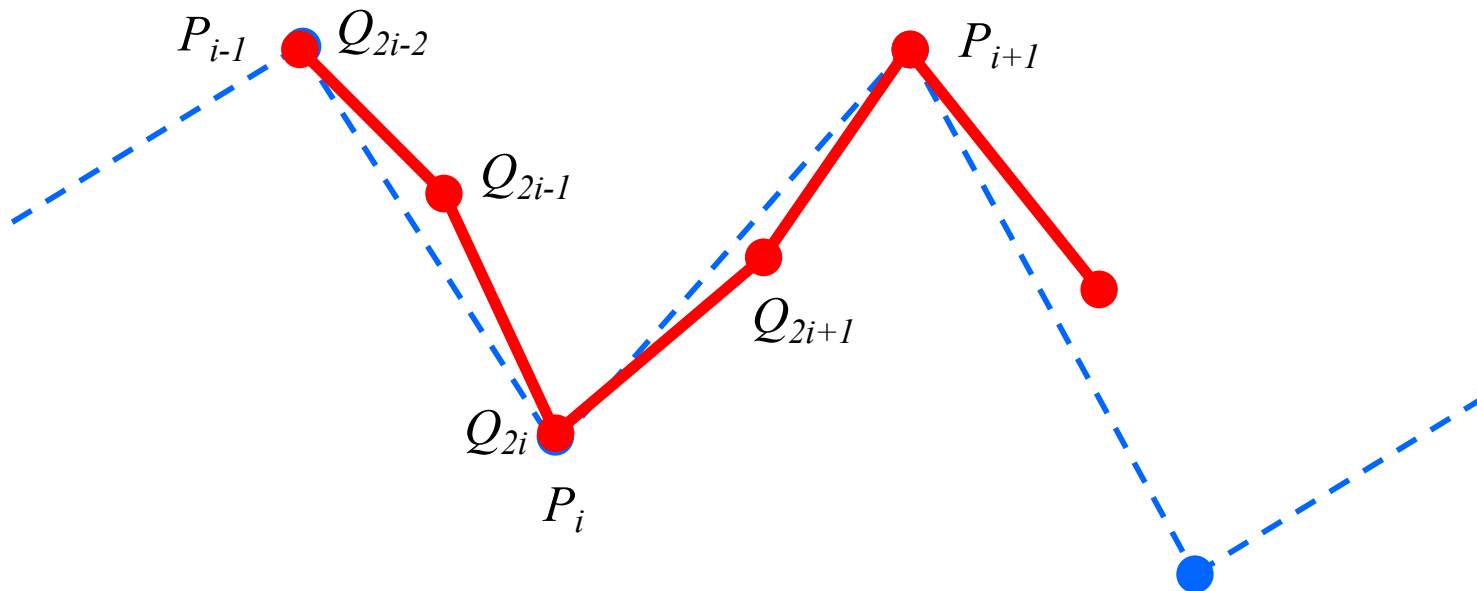
$$\left\{ \begin{array}{ll} \text{Pour } 0 \leq i \leq n-1 & Q_{2i} = P_i \\ \text{Pour } 1 \leq i \leq n-3 & Q_{2i+1} = -\frac{1}{16}P_{i-1} + \frac{9}{16}P_i + \frac{9}{16}P_{i+1} - \frac{1}{16}P_{i+2} \end{array} \right.$$

$$\text{Conditions aux bords} \left\{ \begin{array}{l} Q_1 = \frac{3}{8}P_0 + \frac{6}{8}P_1 - \frac{1}{8}P_2 \\ Q_{2n-3} = -\frac{1}{8}P_{n-3} + \frac{6}{8}P_{n-2} + \frac{3}{8}P_{n-1} \end{array} \right.$$

Four points scheme

- Schéma régulier **interpolant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-2}$

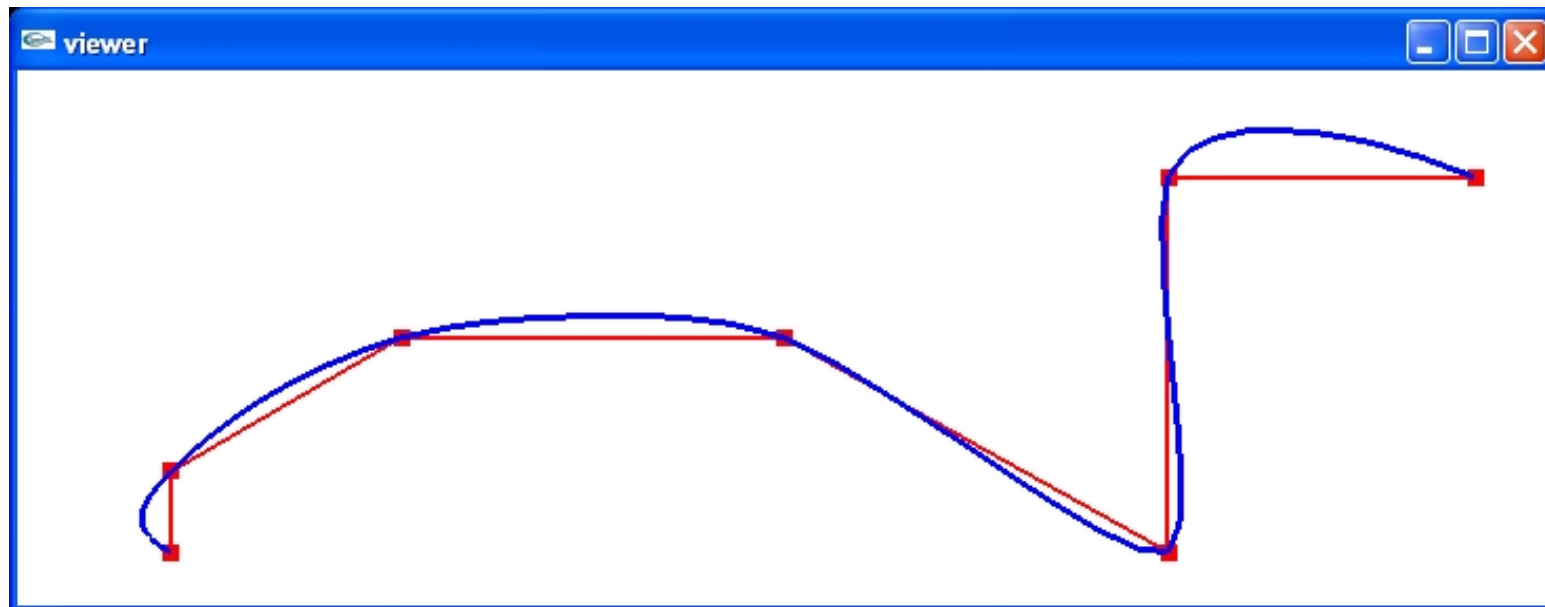
$$\begin{cases} \text{Pour } 0 \leq i \leq n-1 & Q_{2i} = P_i \\ \text{Pour } 1 \leq i \leq n-3 & Q_{2i+1} = -\frac{1}{16}P_{i-1} + \frac{9}{16}P_i + \frac{9}{16}P_{i+1} - \frac{1}{16}P_{i+2} \end{cases}$$



Four points scheme

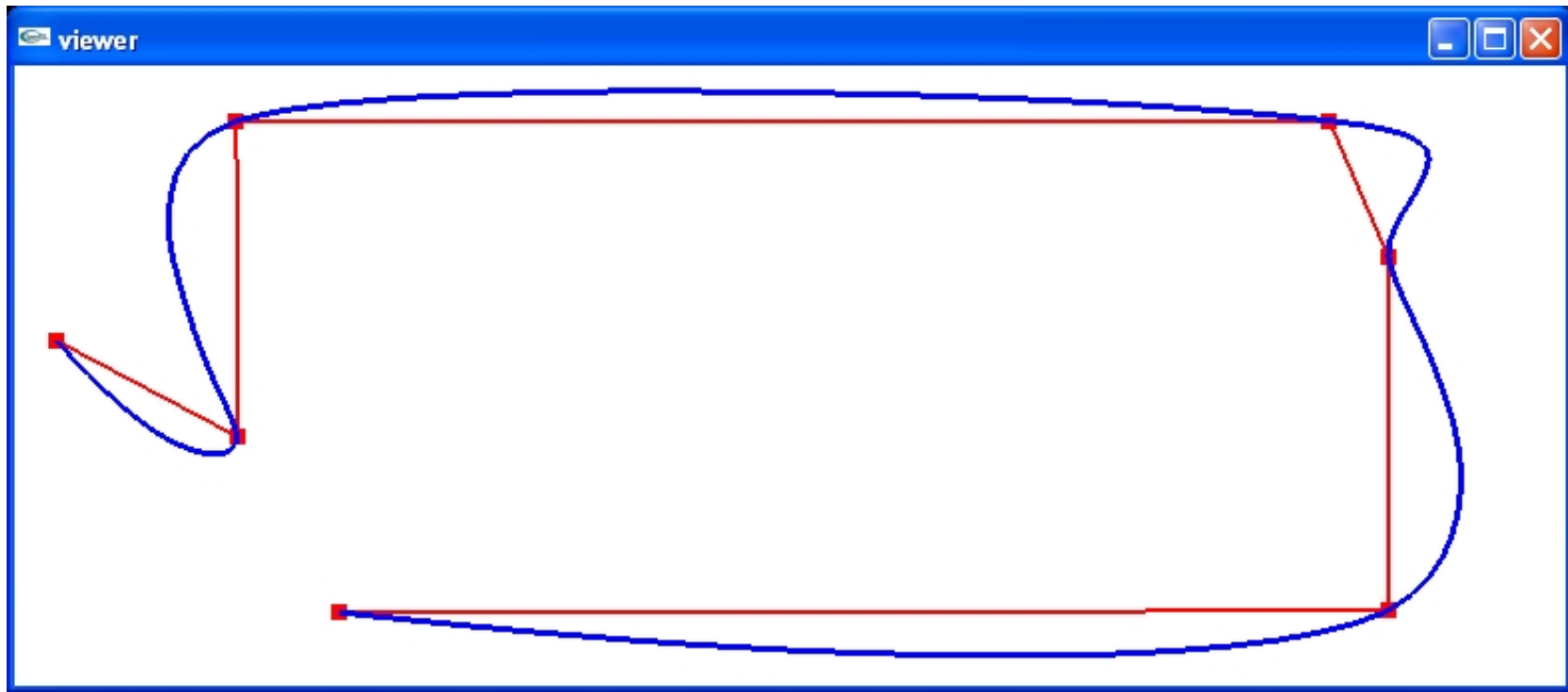
- Schéma régulier **interpolant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-2}$

$$\begin{cases} \text{Pour } 0 \leq i \leq n-1 & Q_{2i} = P_i \\ \text{Pour } 1 \leq i \leq n-3 & Q_{2i+1} = -\frac{1}{16}P_{i-1} + \frac{9}{16}P_i + \frac{9}{16}P_{i+1} - \frac{1}{16}P_{i+2} \end{cases}$$



Four points scheme

- Problèmes d'oscillation



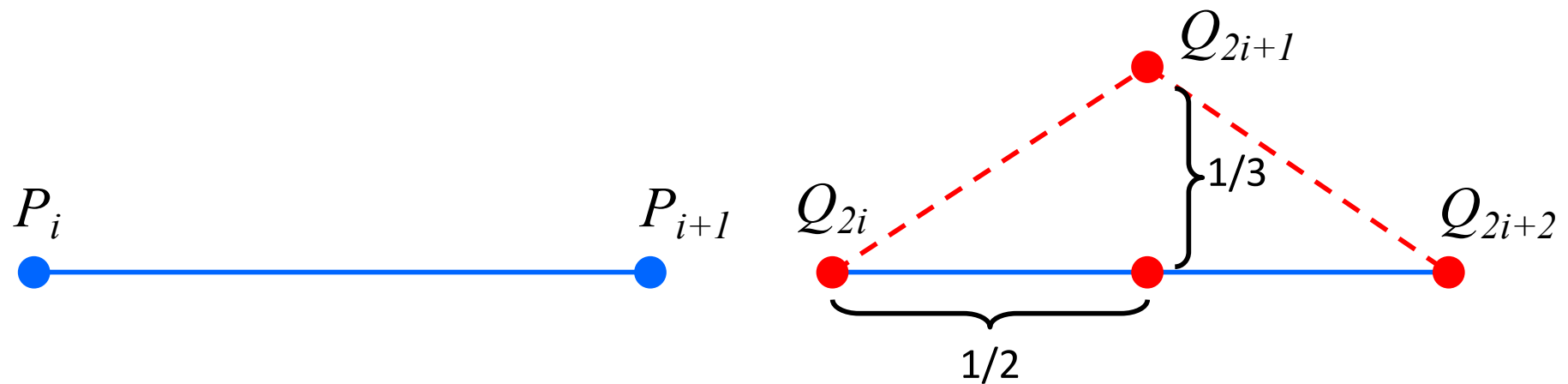
La convexité du polygone de contrôle n'est pas toujours respectée dans la courbe limite

Fractale chou fleur

- Schéma fractal **interpolant** $(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 2n-2}$

$$\begin{cases} \text{Pour } 0 \leq i \leq n-1 & Q_{2i} = P_i \\ \text{Pour } 0 \leq i \leq n-2 & Q_{2i+1} = \frac{1}{2}(P_i + P_{i+1}) + \frac{1}{3}N(P_{i+1} - P_i) \end{cases}$$

avec pour un point $P = (x, y)$, $N(P) = (-y, x)$



Fractale

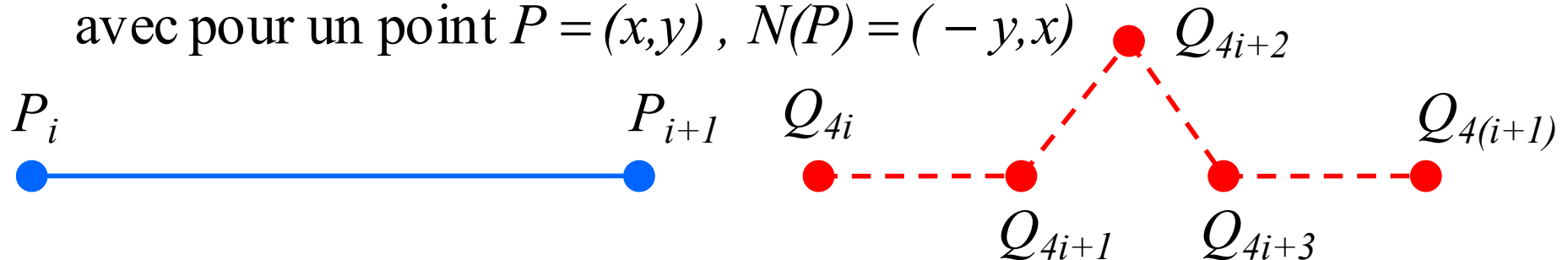
flocon de Von Koch

$$(P_i)_{0 \leq i \leq n-1} \rightarrow (Q_j)_{0 \leq j \leq 4n-4}$$

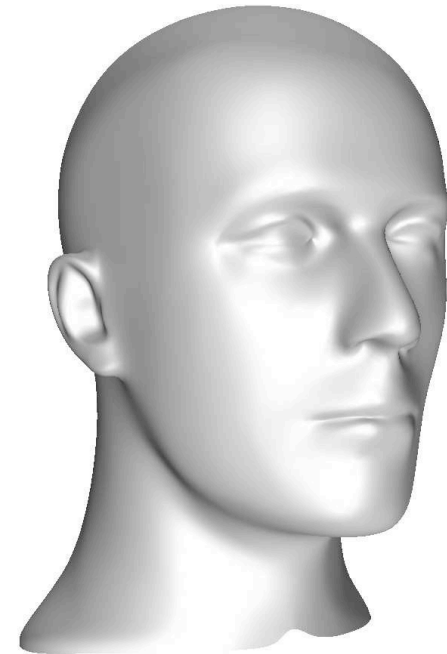
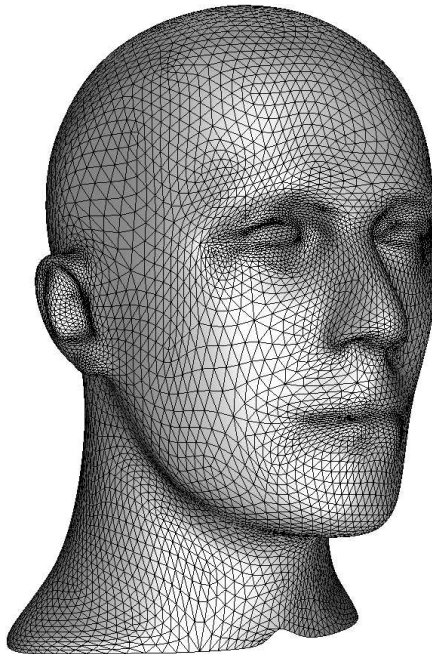
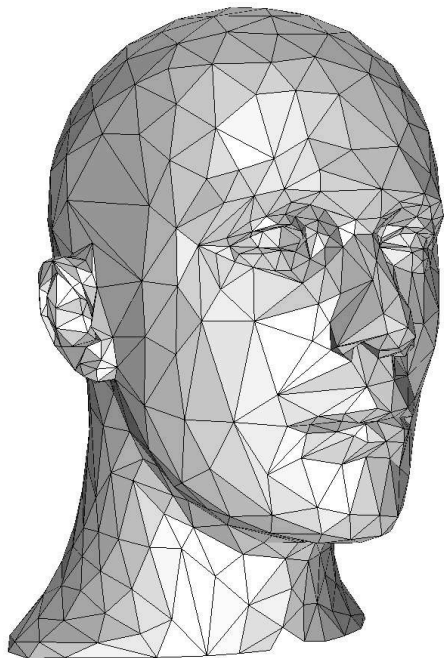
- Schéma fractal **interpolant**

$$\left\{ \begin{array}{l} \text{Pour } 0 \leq i \leq n-1 \quad Q_{4i} = P_i \\ \text{Pour } 0 \leq i \leq n-2 \quad \left\{ \begin{array}{l} Q_{4i+1} = \frac{1}{3}(2P_i + P_{i+1}) \\ Q_{4i+2} = \frac{1}{2}(P_i + P_{i+1}) + \frac{\sqrt{5}}{6}N(P_{i+1} - P_i) \\ Q_{4i+3} = \frac{1}{3}(P_i + 2P_{i+1}) \end{array} \right. \end{array} \right.$$

avec pour un point $P = (x, y)$, $N(P) = (-y, x)$



Surfaces de subdivision



(loop)

Schémas principaux

Doo-Sabin

$\Leftrightarrow$ Surface B-Spline quadratique uniforme

(équivalent aux courbes de Chaikin)

approximant

maillages de quadrilatères ou quelconques

Catmull-Clark

$\Leftrightarrow$ Surface B-Spline cubique uniforme

approximant

maillages de quadrilatères ou quelconques

Loop

$\Leftrightarrow$ Surface Box-Spline quadrique uniforme

approximant

maillages triangulaires

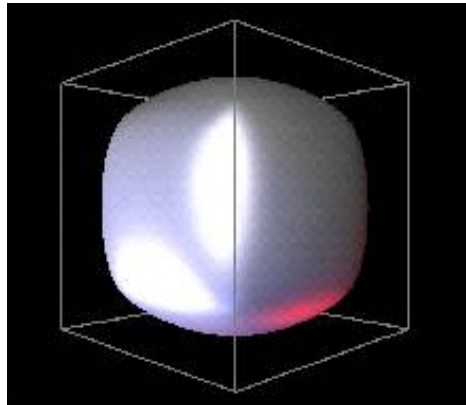
Butterfly

interpolant

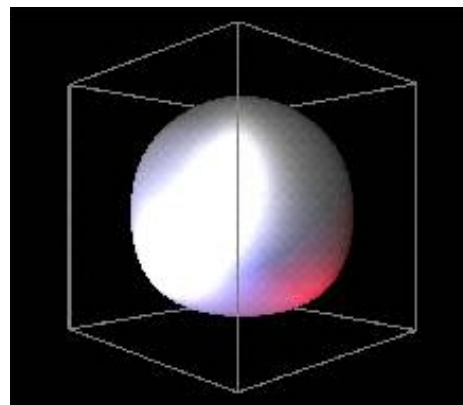
maillages triangulaires

Schémas principaux

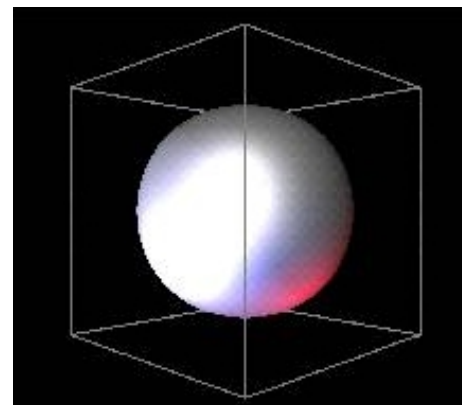
Doo-Sabin



Catmull-Clark



Loop



Butterfly

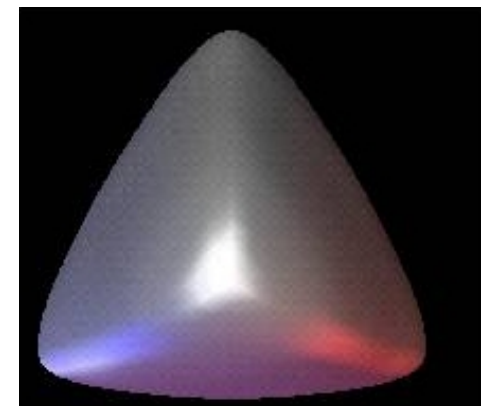
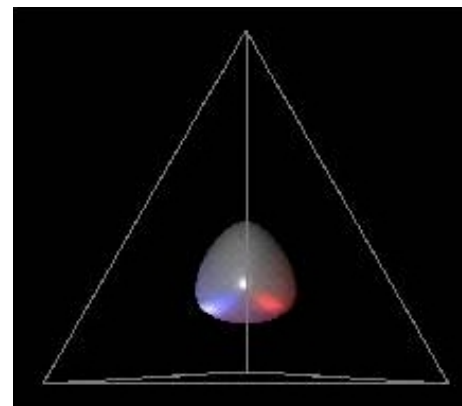
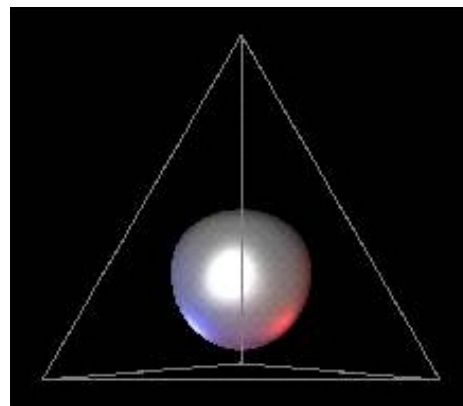
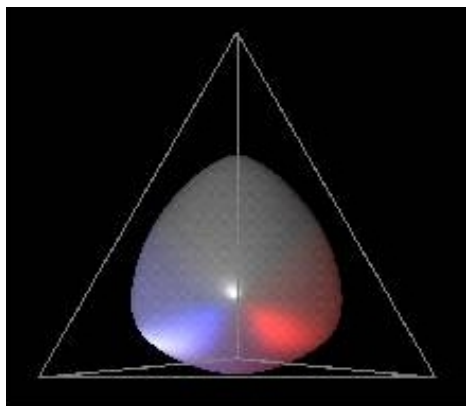
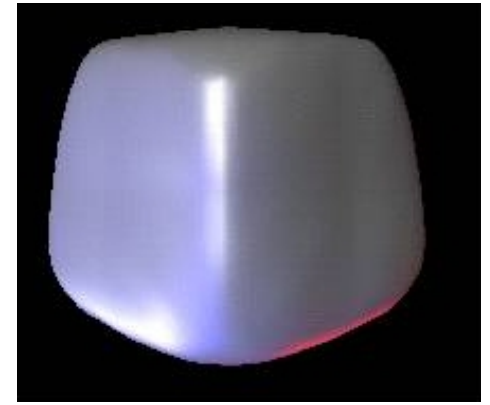


Schéma de Doo-Sabin

- Malcolm A. Sabin (11/2008)

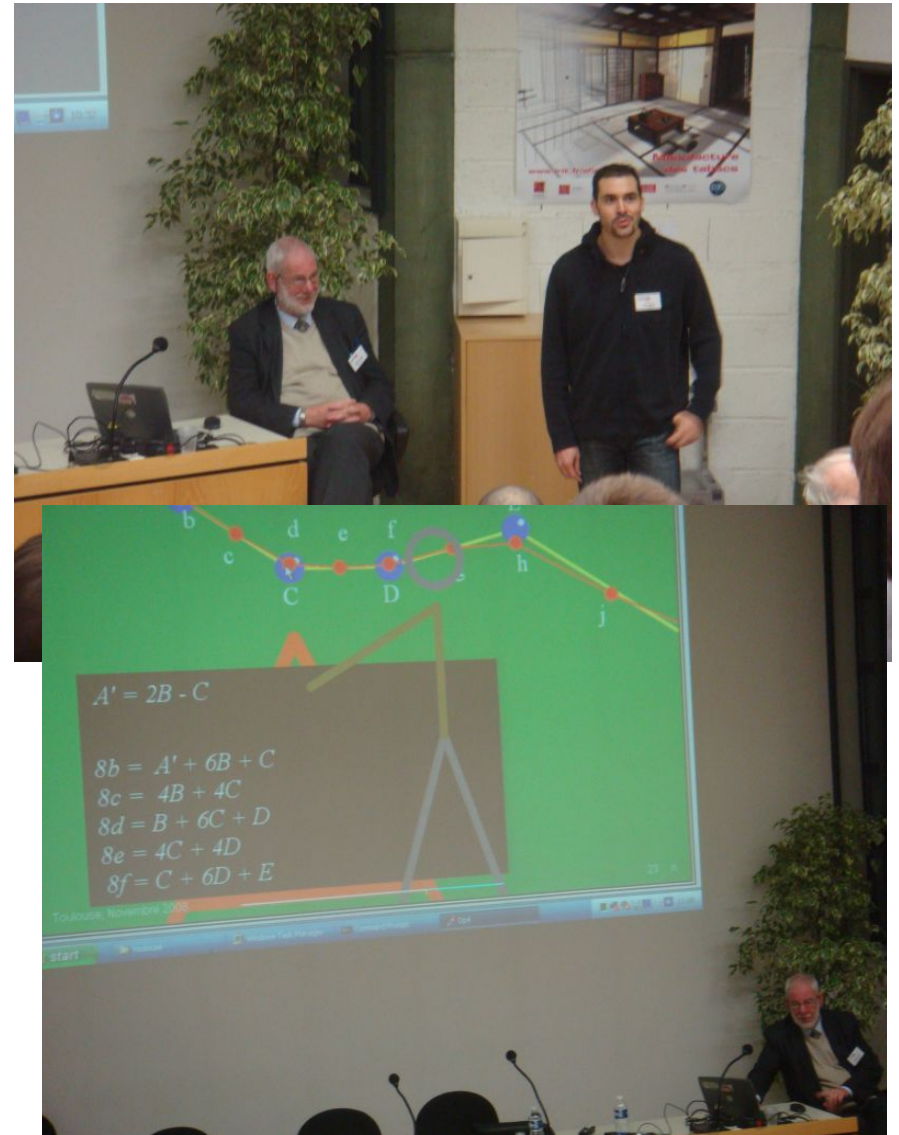
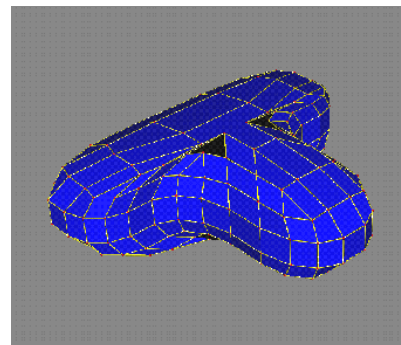
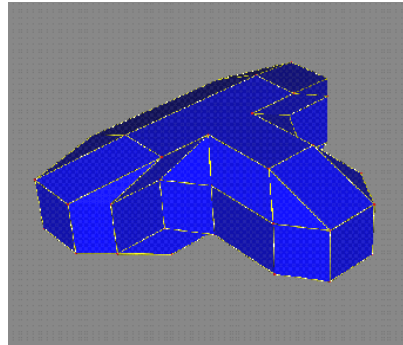
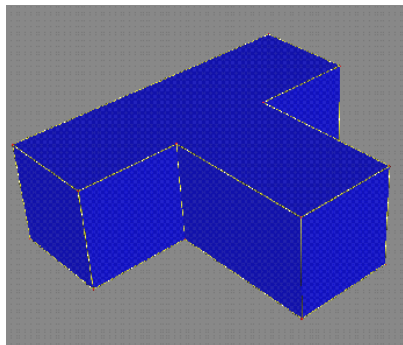
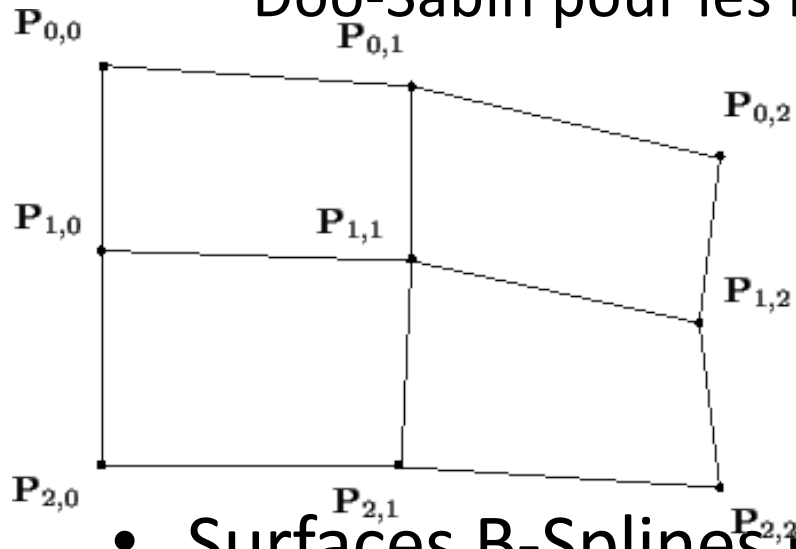


Schéma de Doo-Sabin

- Principe du *corner cutting* (Chaikin)
- Schéma **approximant** (ne passe pas par les sommets originaux)
- Fonctionne sur des **maillages quelconques**
- Surface B-Spline Quadratique
- C^1 continue
- Après 1 subdivision :
 - tous les sommets sont de valence 4



Doo-Sabin pour les maillages quadrangulaires réguliers



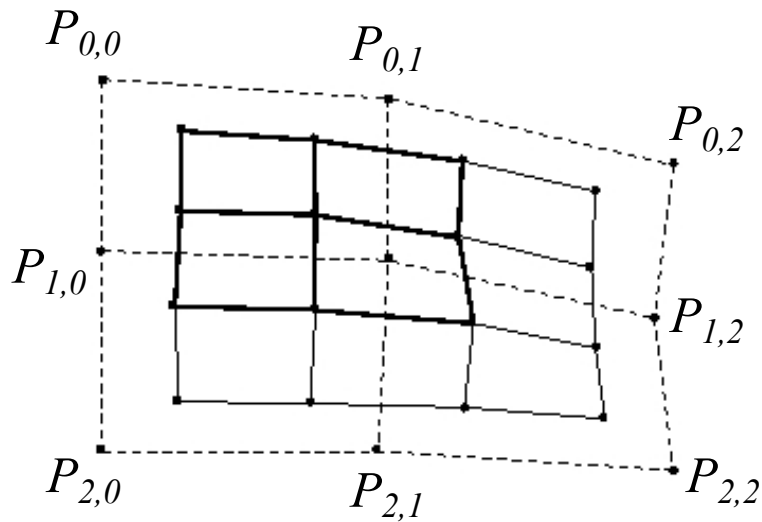
- Surfaces B-Splines uniformes quadriques

$$P(u, v) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} \\ P_{1,0} & P_{1,1} & P_{1,2} \\ P_{2,0} & P_{2,1} & P_{2,2} \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{bmatrix}^T \begin{bmatrix} v^2 \\ v \\ 1 \end{bmatrix}$$

- 1^{er} quart de la surface :

$$P_{[0,1/2]}(u, v) = P\left(\frac{u}{2}, \frac{v}{2}\right) = \begin{bmatrix} u^2 & u & 1 \end{bmatrix} \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix} \begin{bmatrix} P_{0,0} & P_{0,1} & P_{0,2} \\ P_{1,0} & P_{1,1} & P_{1,2} \\ P_{2,0} & P_{2,1} & P_{2,2} \end{bmatrix} \frac{1}{4} \begin{bmatrix} 3 & 1 & 0 \\ 1 & 3 & 0 \\ 0 & 3 & 1 \end{bmatrix}^T \begin{bmatrix} v^2 \\ v \\ 1 \end{bmatrix}$$

Doo-Sabin pour les maillages quadrangulaires réguliers



$$P'_{0,0} = \frac{1}{16} (9P_{0,0} + 3P_{0,1} + 3P_{1,0} + P_{1,1})$$

$$P'_{0,1} = \frac{1}{16} (3P_{0,0} + 9P_{0,1} + P_{1,0} + 3P_{1,1})$$

$$P'_{1,0} = \frac{1}{16} (3P_{0,0} + P_{0,1} + 9P_{1,0} + 3P_{1,1})$$

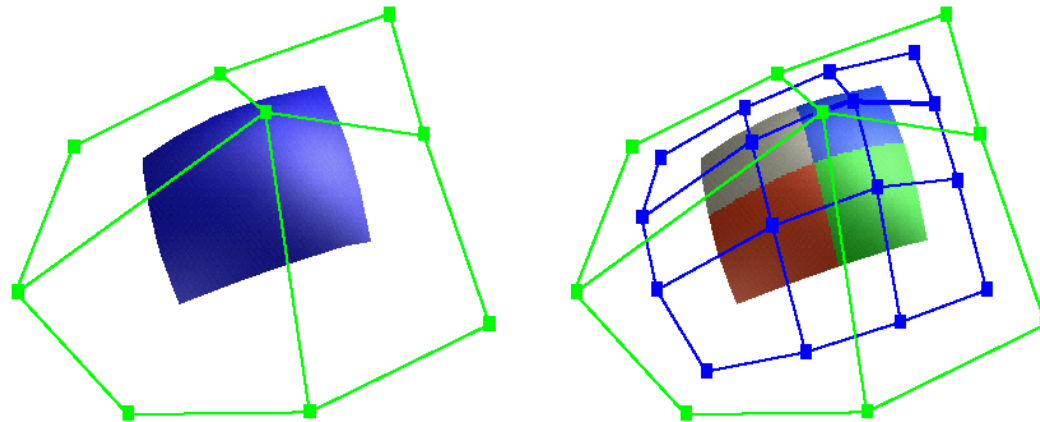
$$P'_{1,1} = \frac{1}{16} (P_{0,0} + 3P_{0,1} + 3P_{1,0} + 9P_{1,1})$$

- Pour chaque face
 - pour chaque sommet de cette face
 - créer un sommet à partir du masque

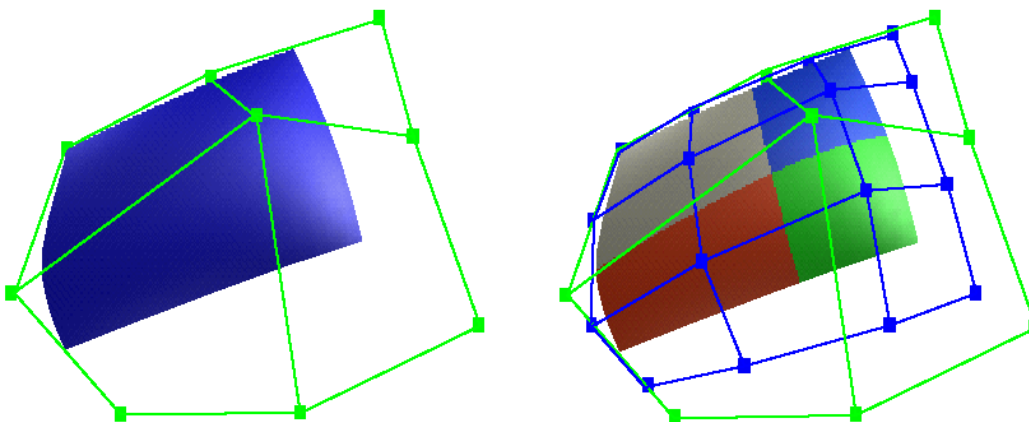
$$\begin{array}{|c|} \hline 9 \\ \hline 3 \end{array} \begin{array}{|c|} \hline 3 \\ \hline 1 \end{array}$$

Doo-Sabin pour les maillages quadrangulaires réguliers

- Après une subdivision



- Masques spécifiques pour les bords (pour "coller" au bord)



Doo-Sabin pour les maillages quelconques

- Autre formulation

$$E_0 = \frac{1}{2}(P_{0,0} + P_{1,0})$$

$$E_1 = \frac{1}{2}(P_{0,0} + P_{0,1})$$

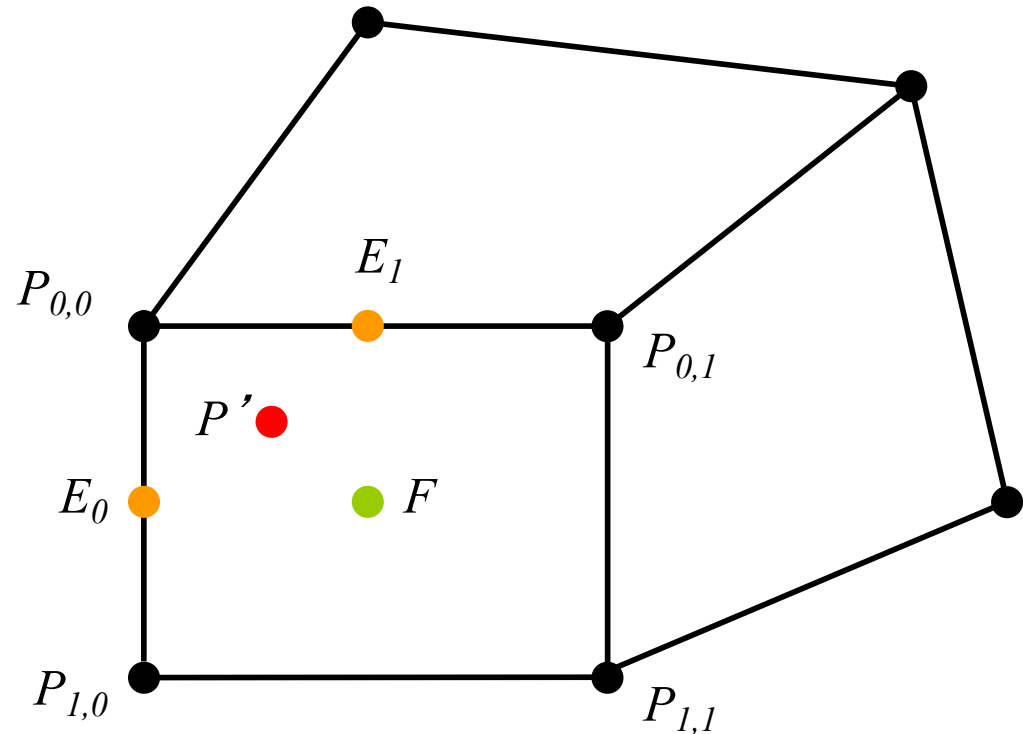
$$F = \frac{1}{4}(P_{0,0} + P_{0,1} + P_{1,0} + P_{1,1})$$

$$P' = \frac{1}{4}(P_{0,0} + E_0 + E_1 + F)$$

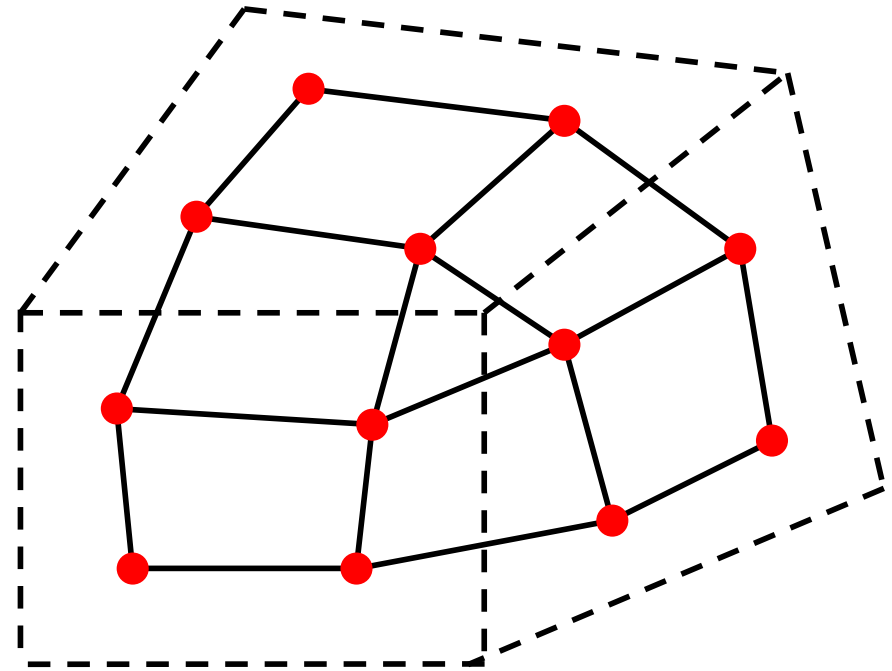
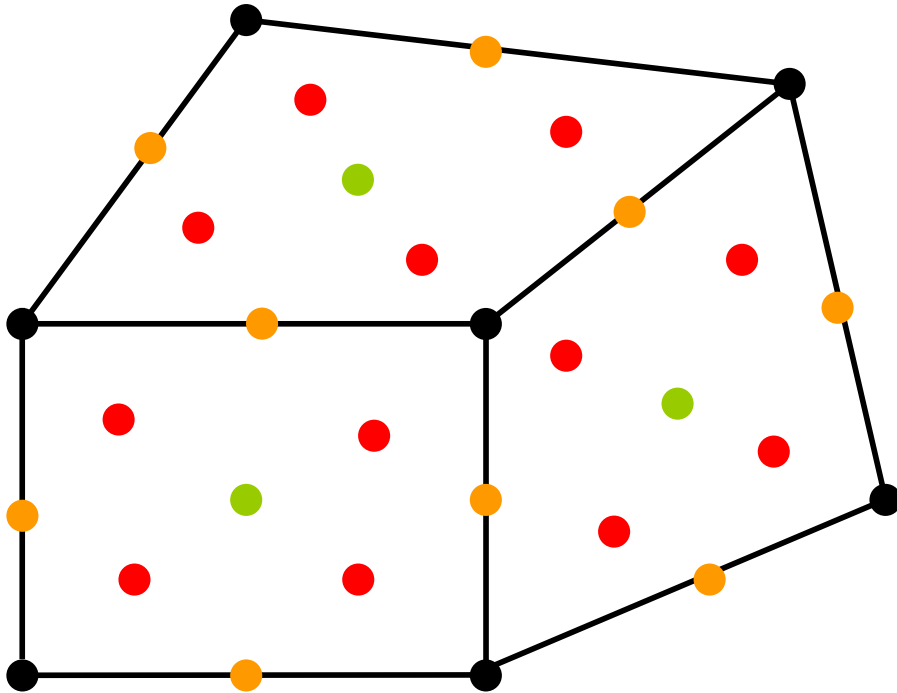
$$= \frac{1}{4}(P_{0,0} + \frac{1}{2}(P_{0,0} + P_{1,0}) + \frac{1}{2}(P_{0,0} + P_{0,1}) + \frac{1}{4}(P_{0,0} + P_{0,1} + P_{1,0} + P_{1,1}))$$

$$= \frac{1}{4}(\frac{4}{4}P_{0,0} + \frac{2}{4}P_{0,0} + \frac{2}{4}P_{1,0} + \frac{2}{4}P_{0,0} + \frac{2}{4}P_{0,1} + \frac{1}{4}P_{0,0} + \frac{1}{4}P_{0,1} + \frac{1}{4}P_{1,0} + \frac{1}{4}P_{1,1})$$

$$= \frac{1}{16}(9P_{0,0} + 3P_{1,0} + 3P_{0,1} + P_{1,1})$$



Doo-Sabin pour les maillages quelconques



Un *edge-point* est le milieu d'une arête

Un *face-point* est le centre d'une face

Un **nouveau point** est la moyenne

- d'un ancien point
- d'un *face-point* d'une face incidente à l'ancien point
- des deux *edge-points* incidents à l'ancien point et à la face

Les **nouveaux points** sont reliés entre eux

Schéma de Catmull-Clark

- Schéma **approximant**
- Fonctionne sur les **maillages quadrangulaires ou quelconques**
- Surface B-Spline cubique
- G^1 continue (sauf points extraordinaires)
- Après une subdivision :
 - **toutes les faces sont des quadrilatères**

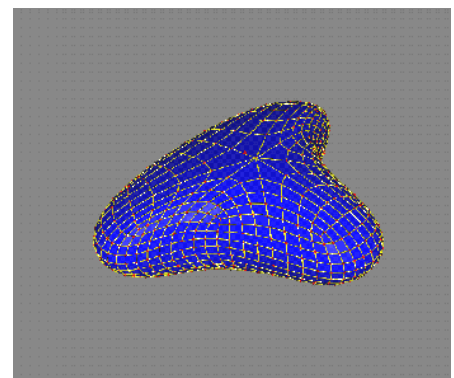
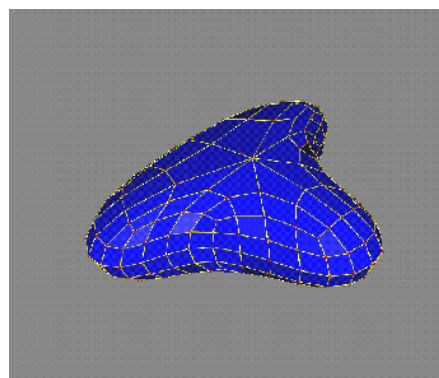
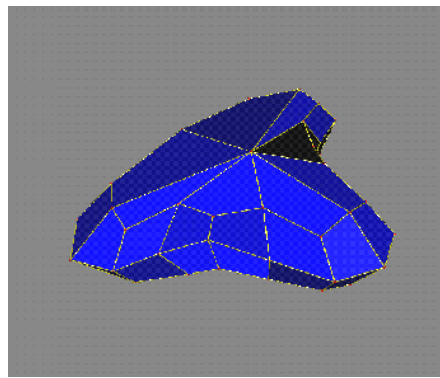
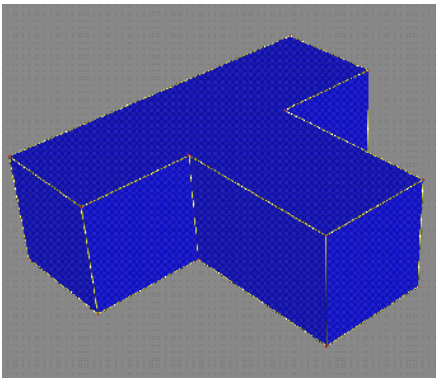


Schéma de Catmull-Clark



Face



Arête

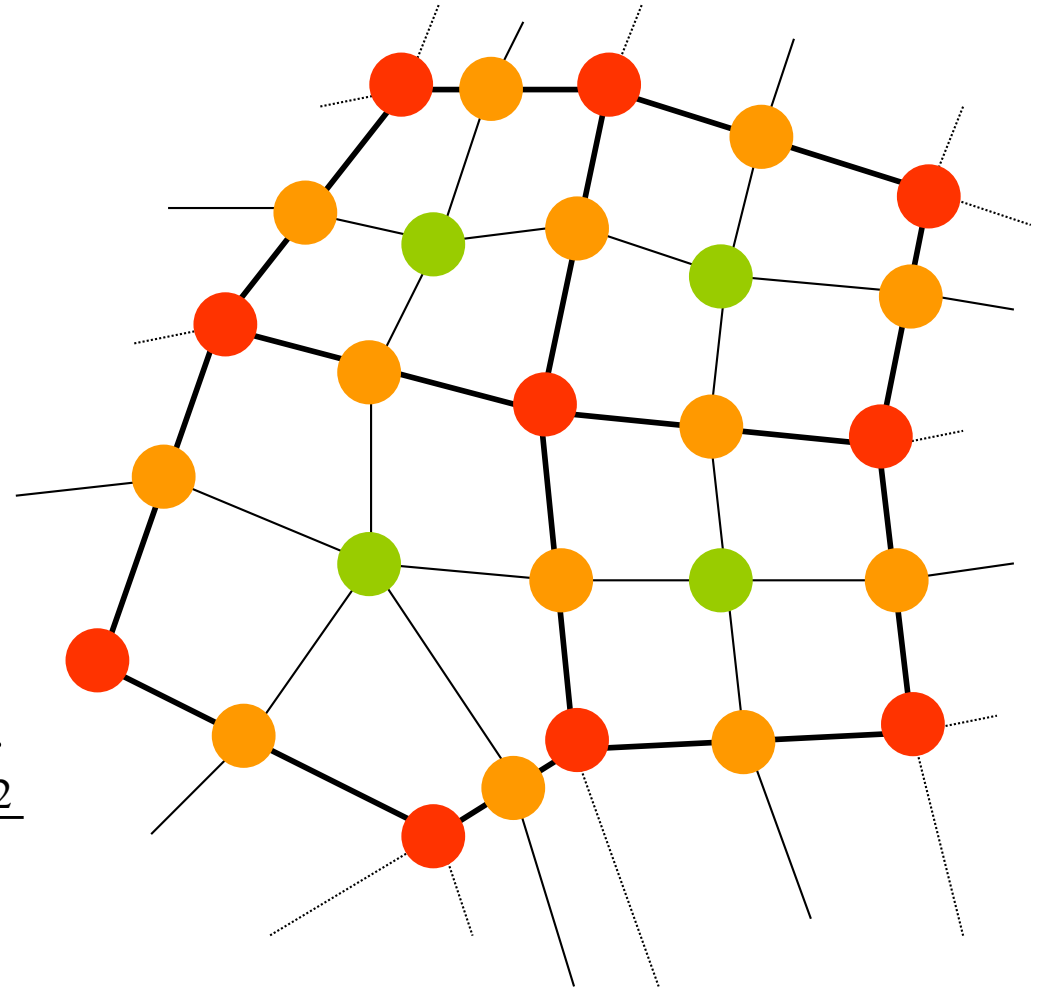


Sommet

$$f = \frac{1}{n} \sum_1^n v_i$$

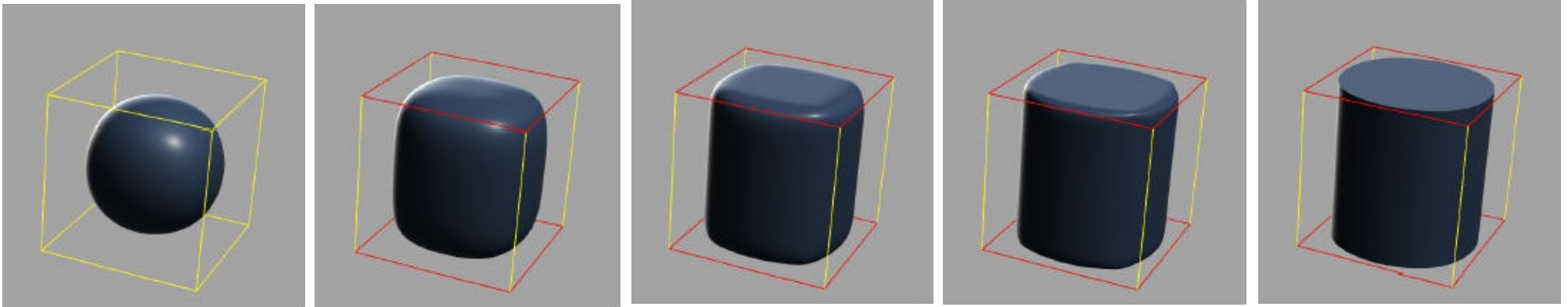
$$e = \frac{v_1 + v_2 + f_1 + f_2}{4}$$

$$v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2} \sum_j e_j + \frac{1}{n^2} \sum_j f_j$$



Catmull-Clark

Gestion des arêtes vives



1. Marquer chaque arête comme **vive** ou **non vive**

- $n = 0 \rightarrow$ **non vive**
- $n > 0 \rightarrow$ **vive**

Pendant la subdivision :

2. Si une arête est **vive**, alors on utilise les nouvelles règles.

Les nouvelles arêtes créées sont associées à $n-1$.

3. Si une arête est **non vive**, alors on utilise les règles de subdivision normales.

Catmull-Clark

Gestion des arêtes vives

● Face (inchangé)

$$f = \frac{1}{n} \sum_1^n v_i$$

● Arête

$$e = \frac{v_1 + v_2}{2}$$

● Sommet

Nombre d'arêtes
vives adjacentes

pointe >2

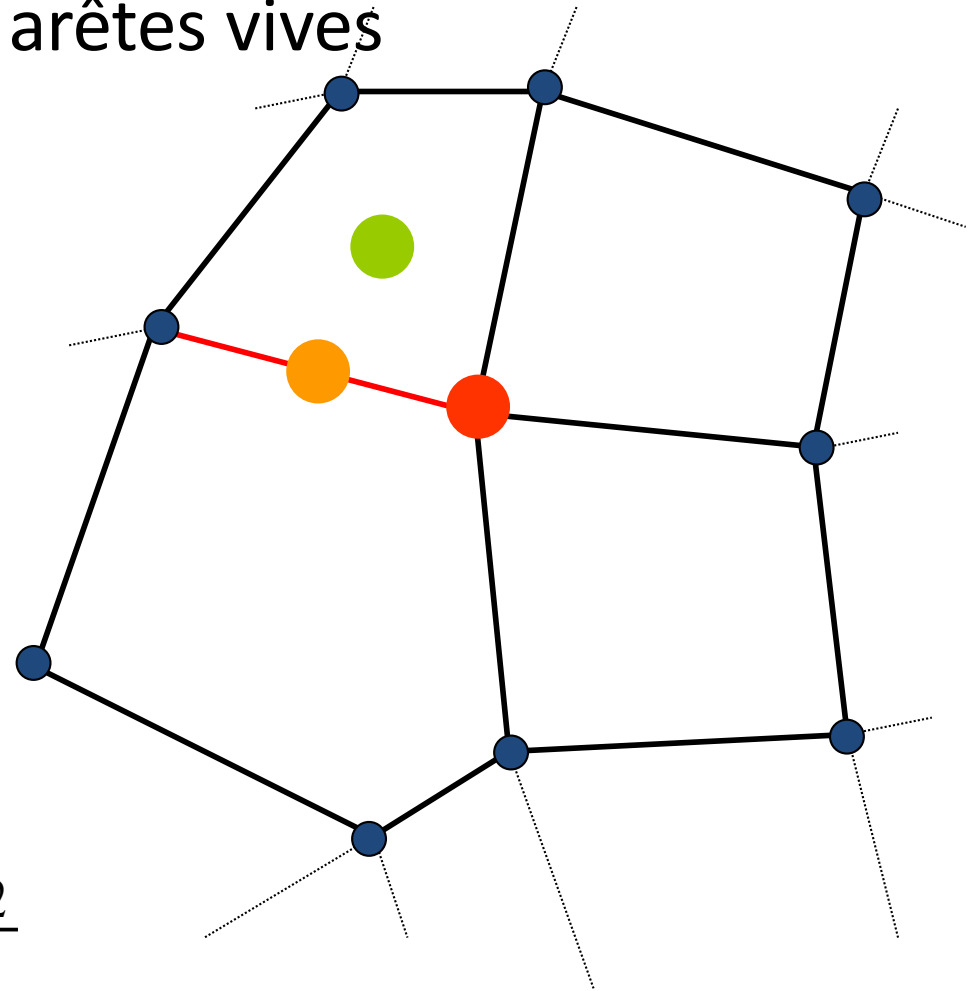
$$v_{i+1} = v_i$$

angle =2

$$v_{i+1} = \frac{e_1 + 6v_i + e_2}{8}$$

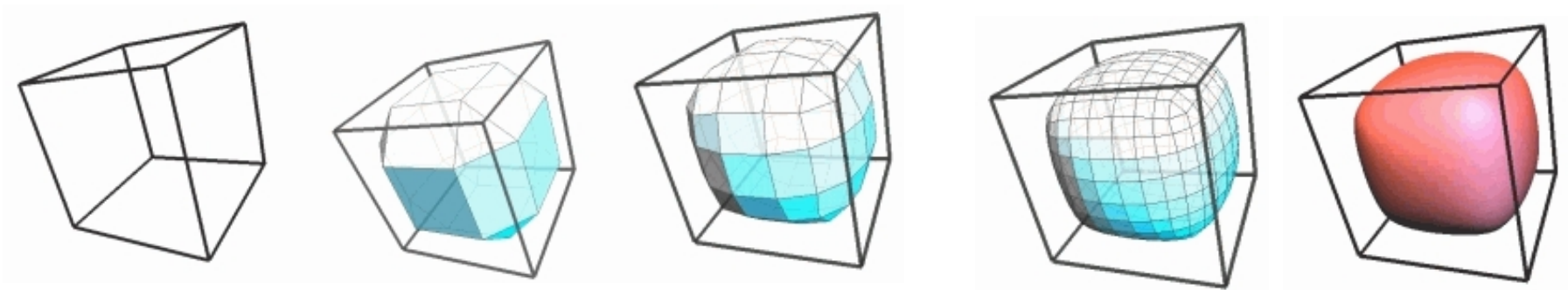
coin 0,1

$$v_{i+1} = \frac{n-2}{n} v_i + \frac{1}{n^2} \sum_j e_j + \frac{1}{n^2} \sum_j f_j$$

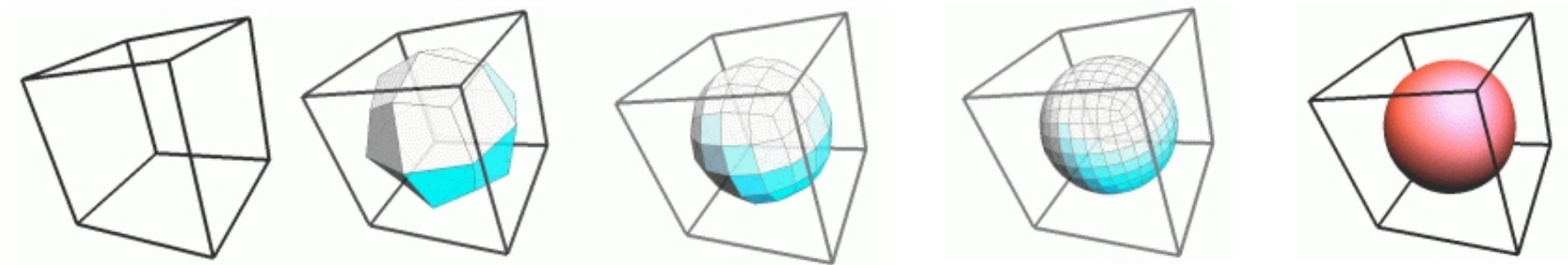


Doo-Sabin vs Catmull-Clark

Doo-Sabin

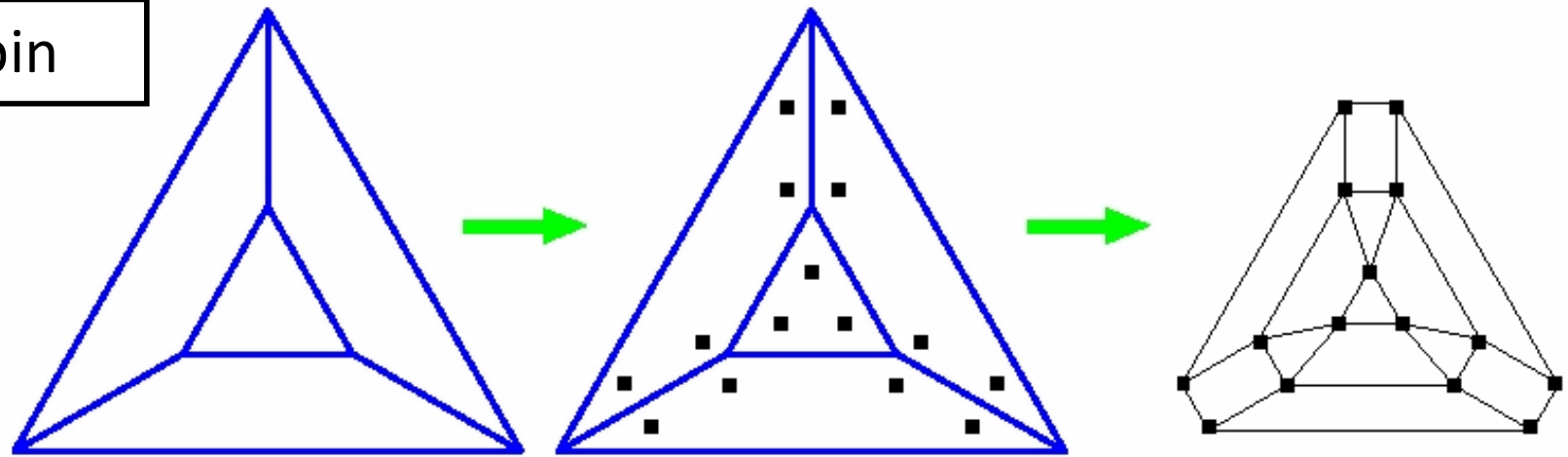


Catmull-Clark



Doo-Sabin vs Catmull-Clark

Doo-Sabin



Catmull-Clark

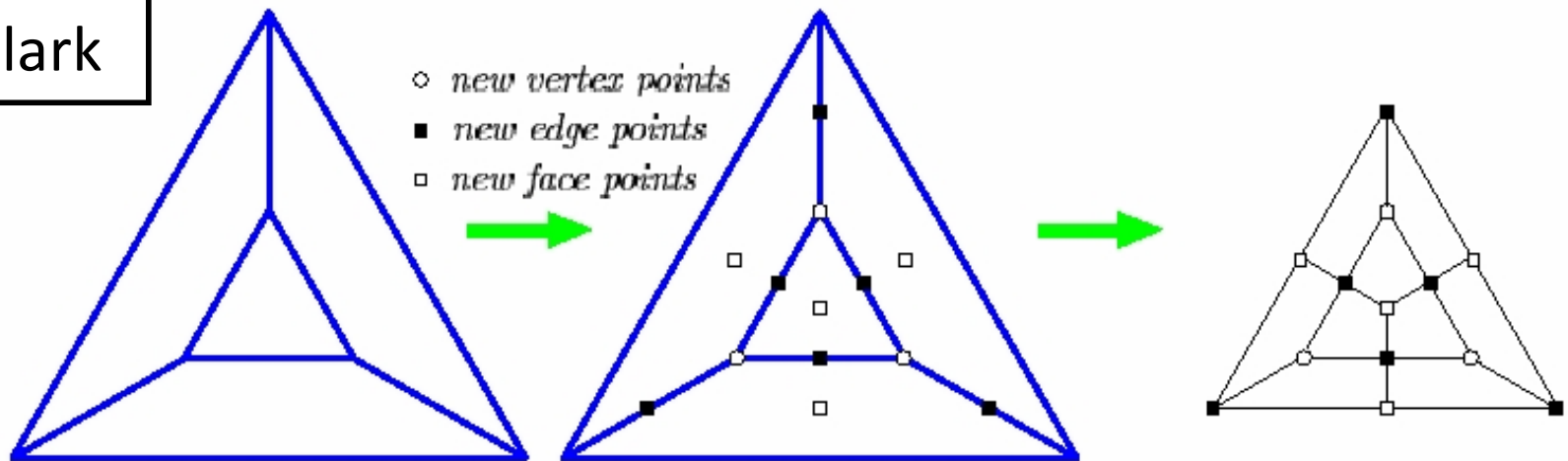
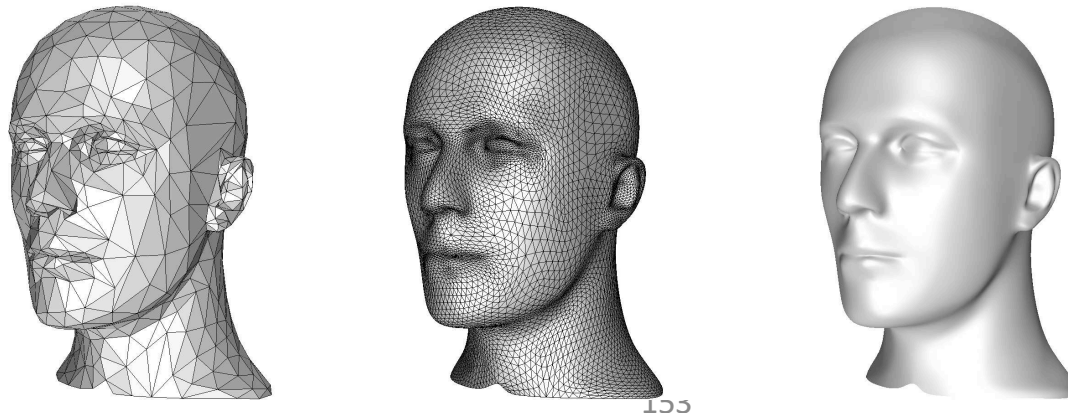


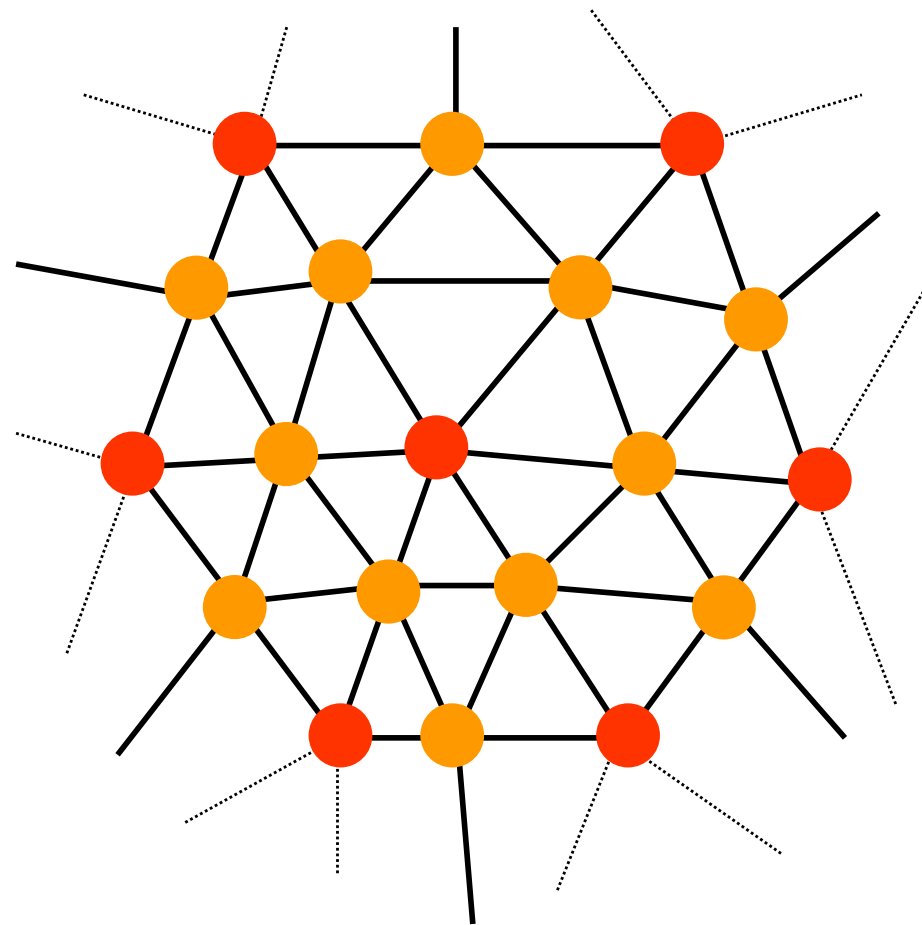
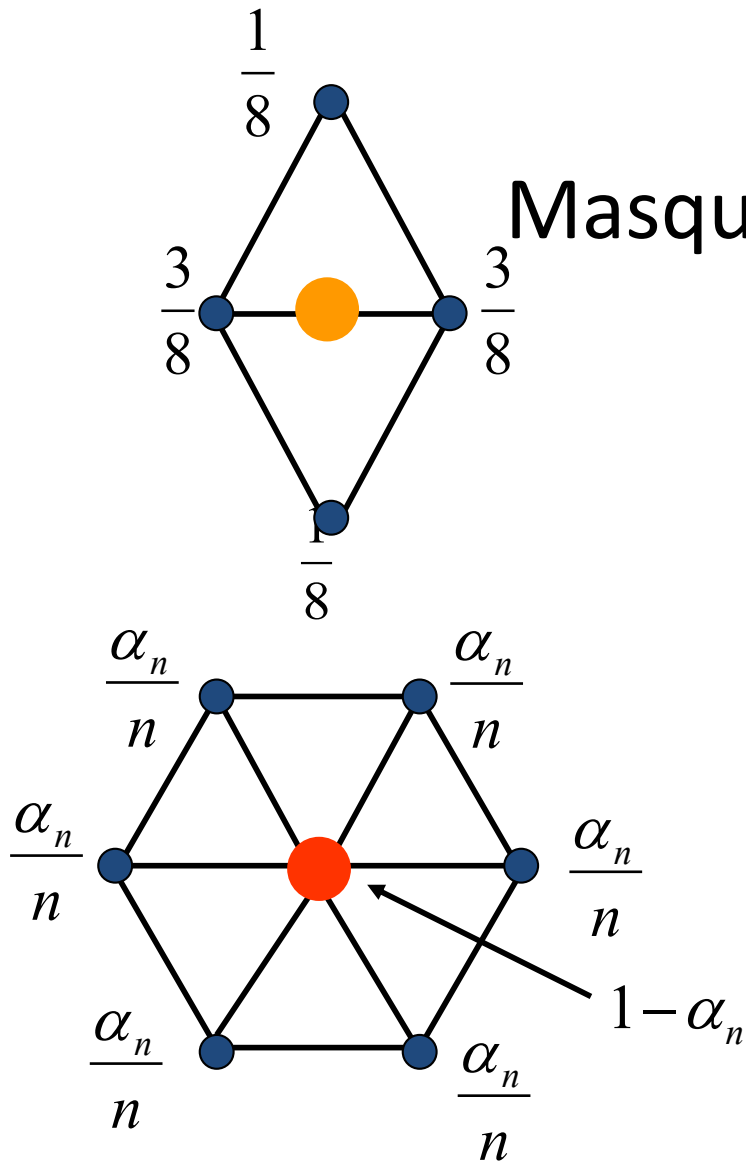
Schéma de Loop

- Schéma **approximant**
- Fonctionne sur **maillages triangulaires quelconques**
- Surface Box-Spline quadrique
- G^1 continuité, sauf aux points extraordinaires
- Simple à mettre on œuvre
 - 2 masques pour les sommets internes
 - 2 masques pour les sommets du bord



Loop

Masques de subdivision

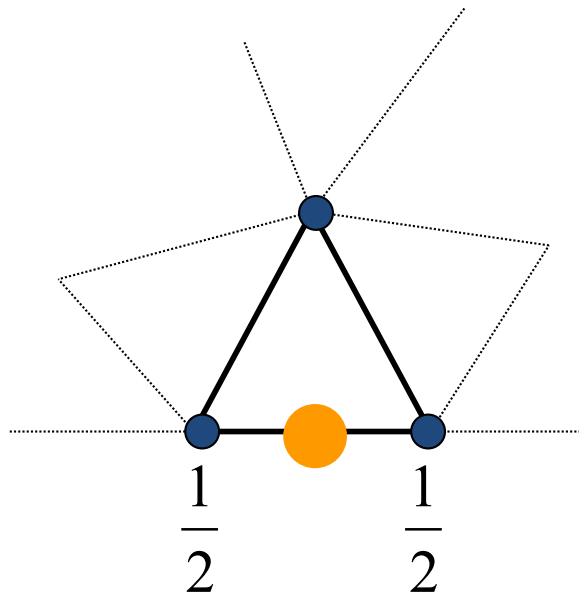


$$\alpha_n = \frac{1}{64} \left(40 - \left(3 + 2 \cos \left(\frac{2\pi}{n} \right) \right)^2 \right)$$

Loop

Masques pour les bords

Loi d'arête



Loi de sommet

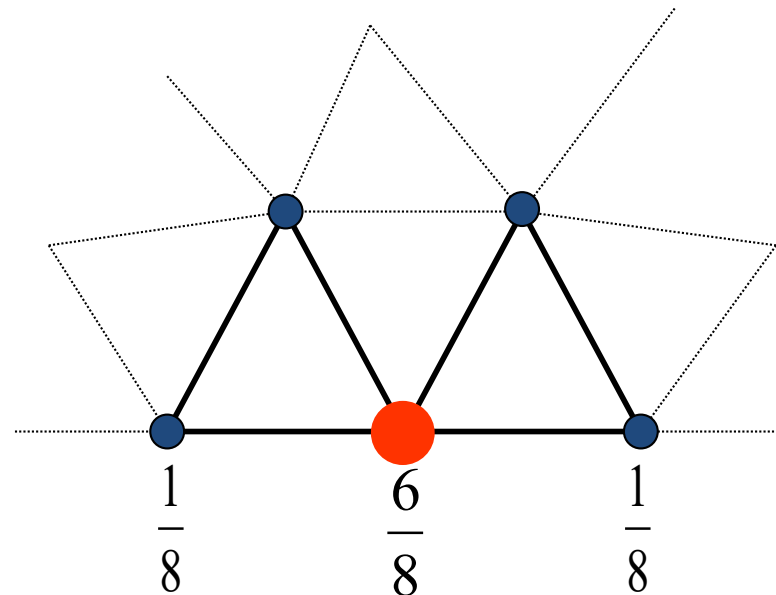
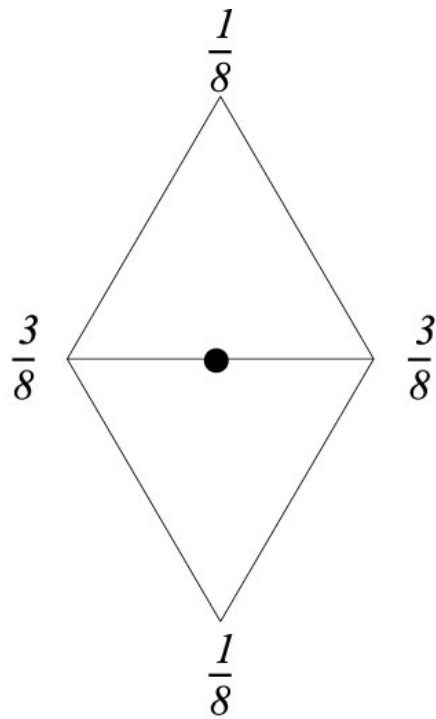
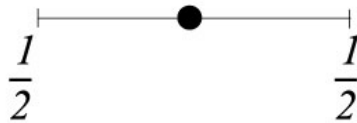
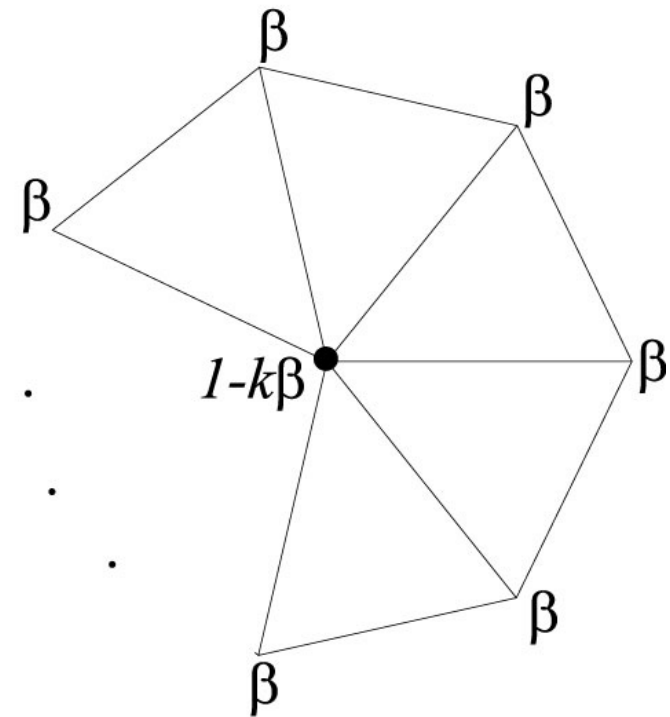


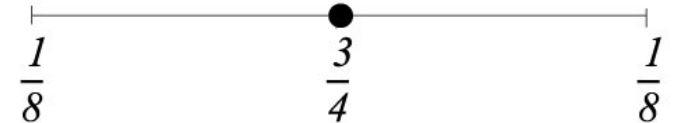
Schéma de Loop



Interior



Crease and boundary



a. Masks for odd vertices

b. Masks for even vertices

Schéma Butterfly

- Schéma **interpolant**
- Fonctionne sur **maillages triangulaires quelconques**
- Simple à mettre en œuvre
 - 3 masques

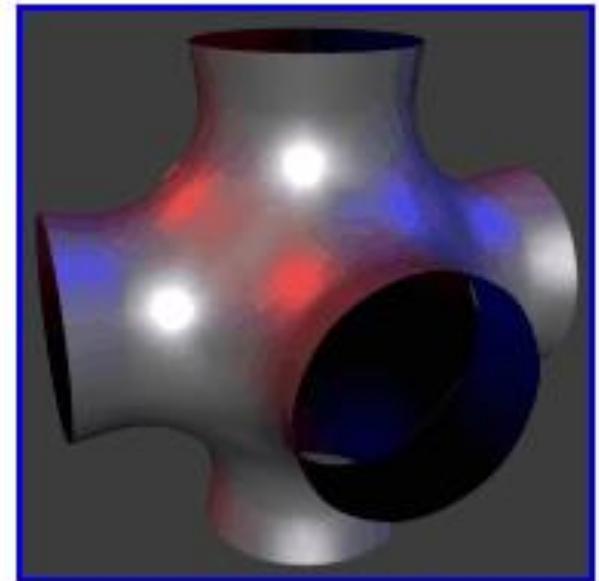
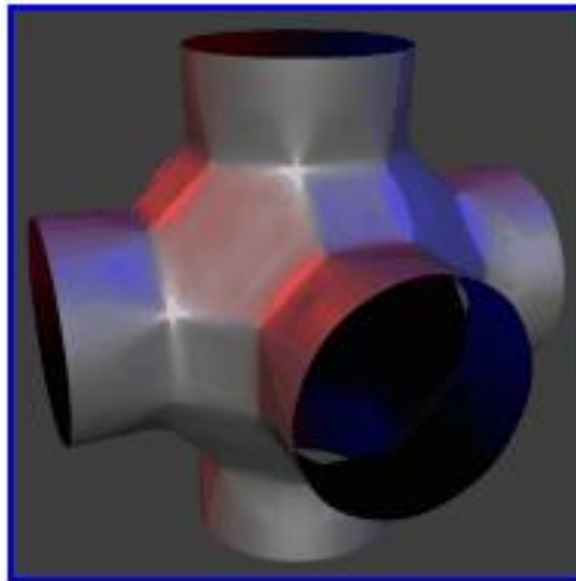
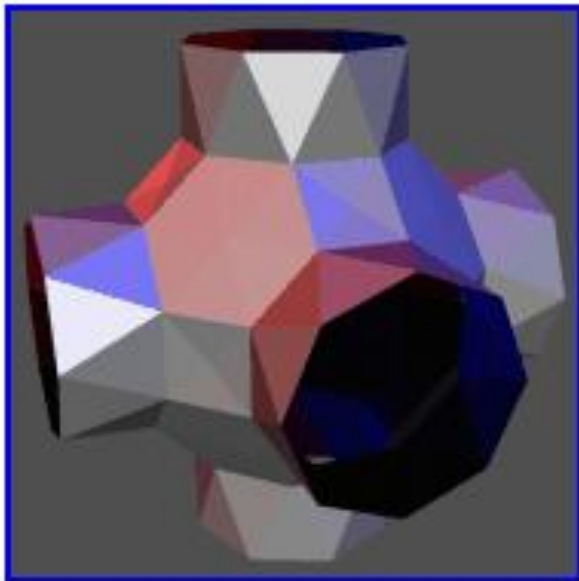
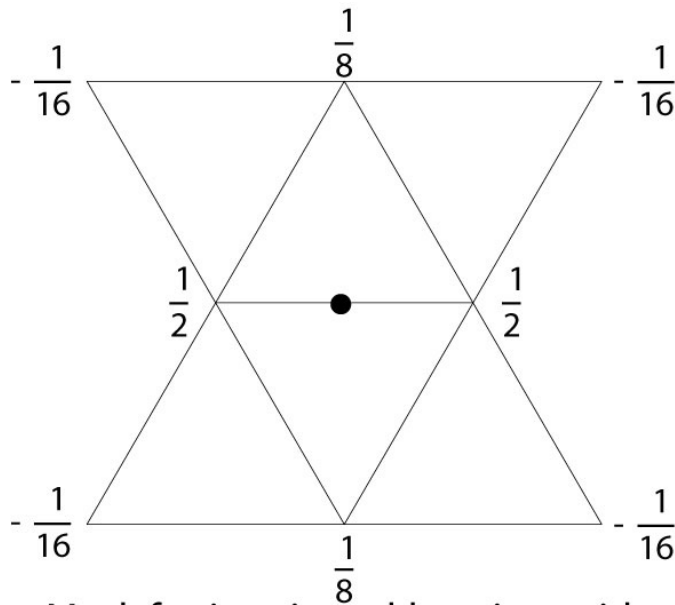


Schéma Butterfly

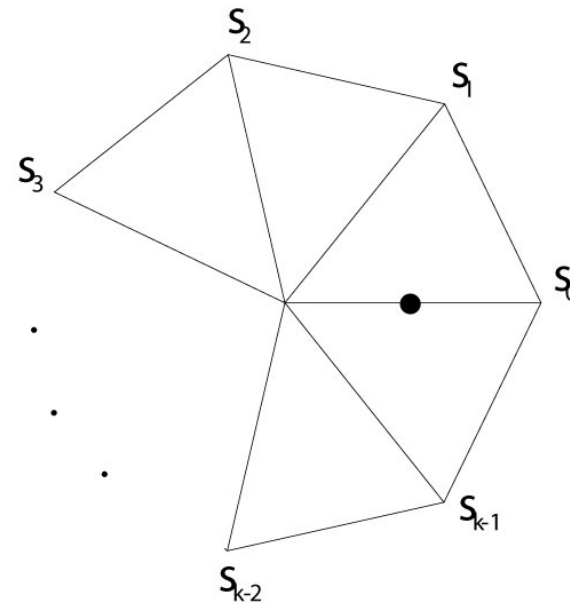


Mask for interior odd vertices with regular neighbors



Mask for crease and boundary vertices

a. Masks for odd vertices



b. Mask for odd vertices adjacent to an extraordinary vertex

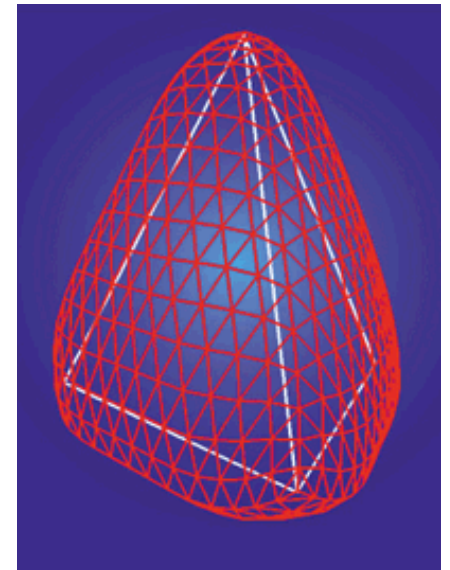
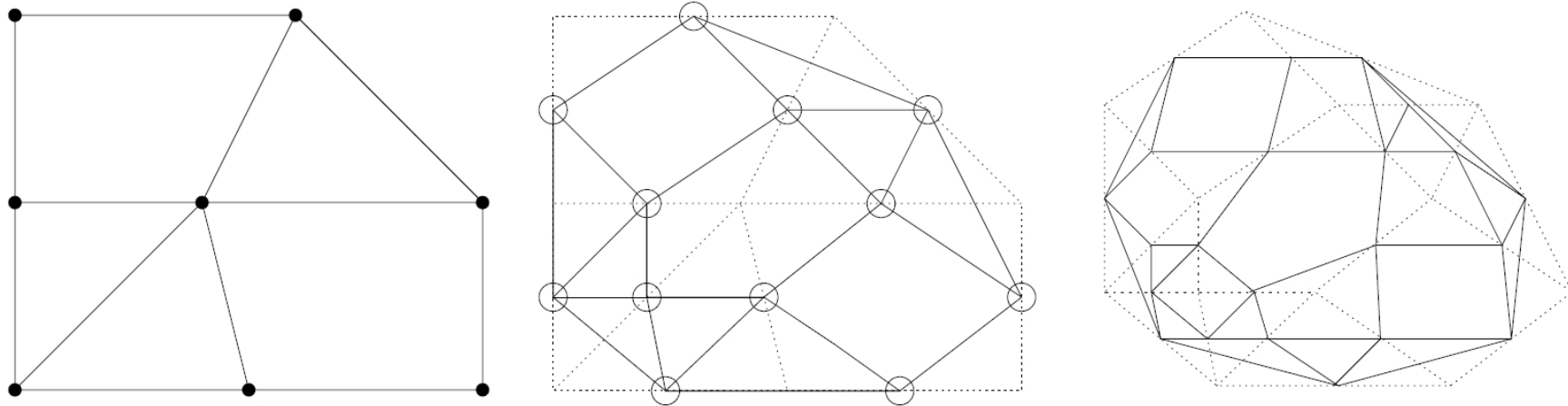
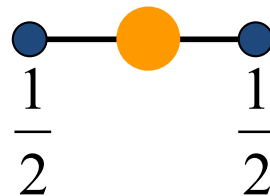


Schéma Mid-edge



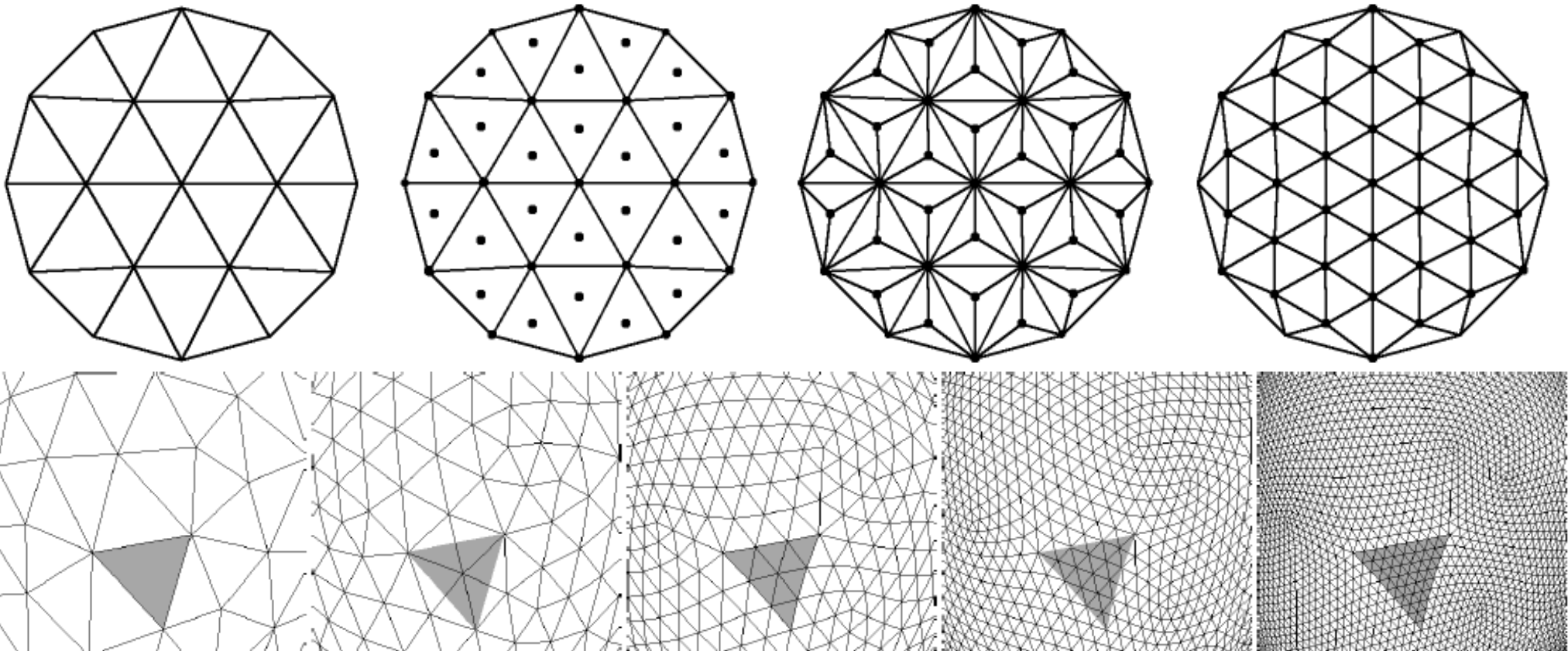
Masque d'arête



Puis relier entre eux chaque paire de nouveaux sommets adjacents à un même ancien sommet

Schéma Racine de trois

- un sommet est ajouté au centre de chaque triangle
- chaque triangle est coupé en 3
- les anciennes arêtes internes sont "basculées"



Seconde partie

Surfaces implicites et CSG

Surfaces implicites

- Rappels

- Définies par une équation implicite

$$S = \{ P(x,y,z) \ / \ f(x,y,z) = iso \}$$

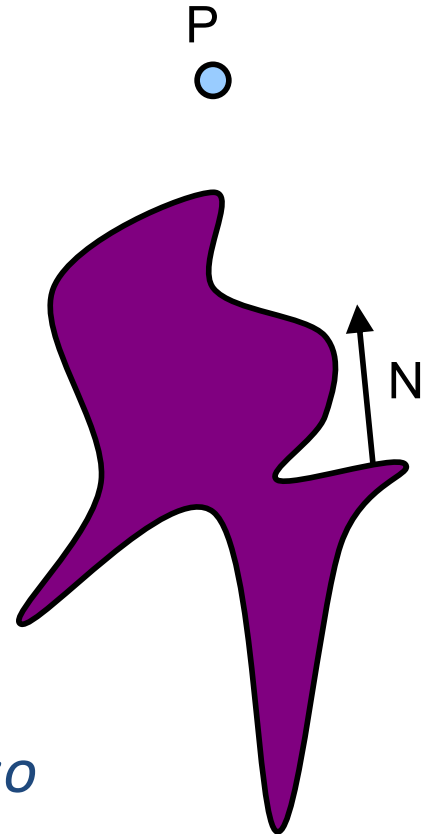
- f est la « *fonction potentiel* »

- Normale à la surface : $N = - \nabla f$ (df/dx, df/dy, df/dz)

- Modélisation volumique : *intérieur* $f(x,y,z) > iso$

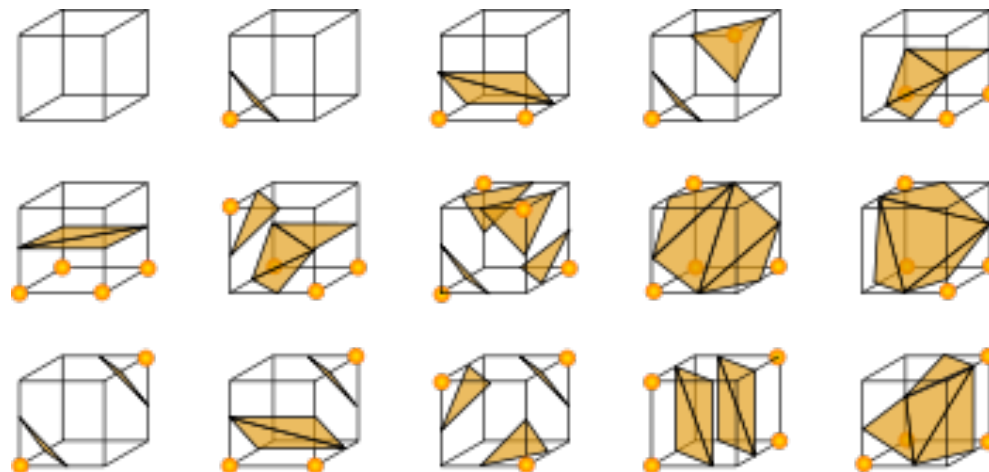
- Test d'appartenance d'un point

(détections de collisions, lancer de rayons)



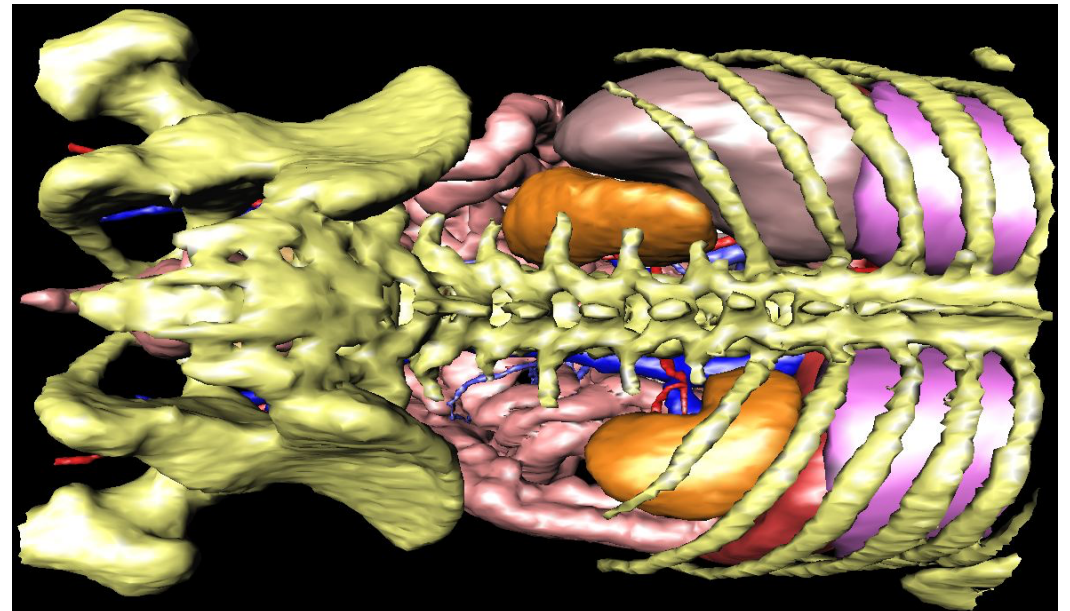
Surfaces implicites

- Visualisation par marching cubes
 - intersection de la surface implicite avec un cube
 - on obtient 0, 1 ou plusieurs polygones
 - 256 configurations de polygones possibles
 - 14 cas possibles à symétrie et réflexivité près



Surfaces implicites

- Visualisation par marching cubes
 - On échantillonne l'espace $\rightarrow$ voxels
 - On calcule l'intersection de chaque voxel avec la surface
 - On assemble les polygones obtenus



Surfaces implicites

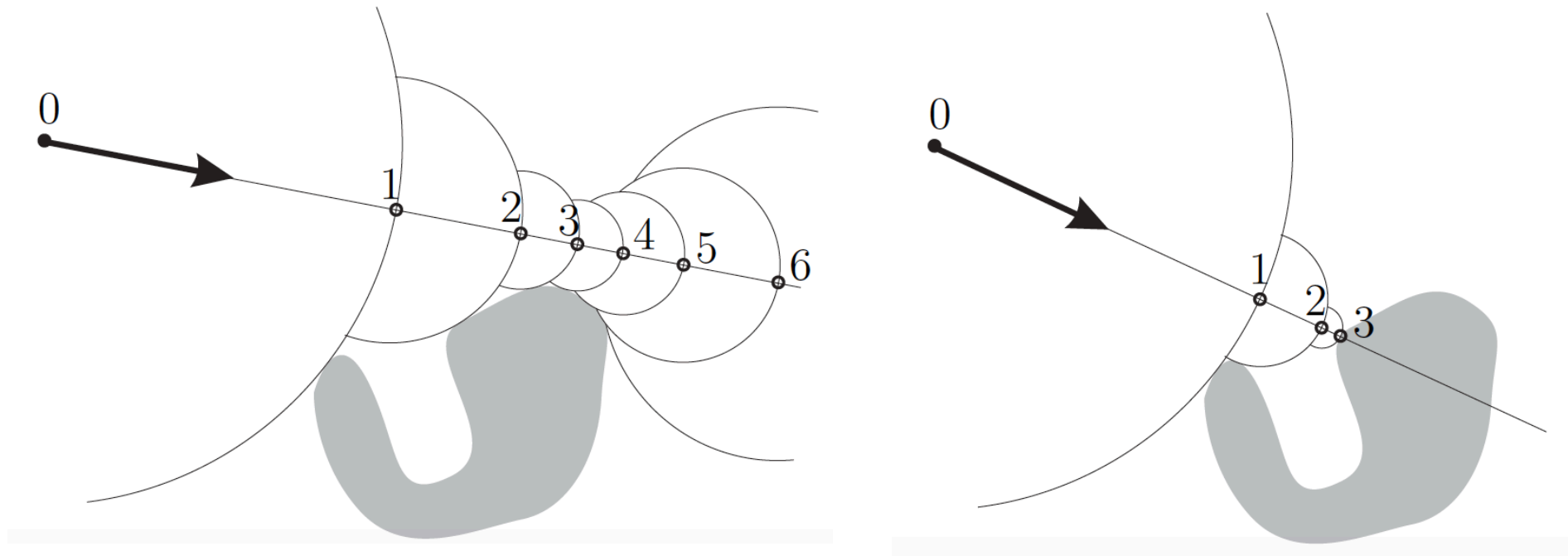
- Visualisation par lancer de rayon : intersection rayon/surface
 - analytique
 - si on sait résoudre l'équation de l'intersection rayon / surface
 - dichotomique
 - nécessité d'échantillonner le rayon
 - lancer de sphères
 - nécessité de connaître la distance à la surface

Surfaces implicites

- Visualisation par lancer de sphères
 - *Sphere tracing: a geometric method for the antialiased ray tracing of implicit surfaces*, John C. Hart, The Visual Computer, 1996
 - On doit connaître
 - Pour une surface A , la fonction $f_A(P)$ qui donne la distance minimale entre tout point P et A
 - Pour une surface A , la valeur D_A qui est la distance de traversement maximale depuis l'origine du rayon

Surfaces implicites

- Visualisation par lancer de sphères



Illustrations : Pascal Mignot, Université de Reims

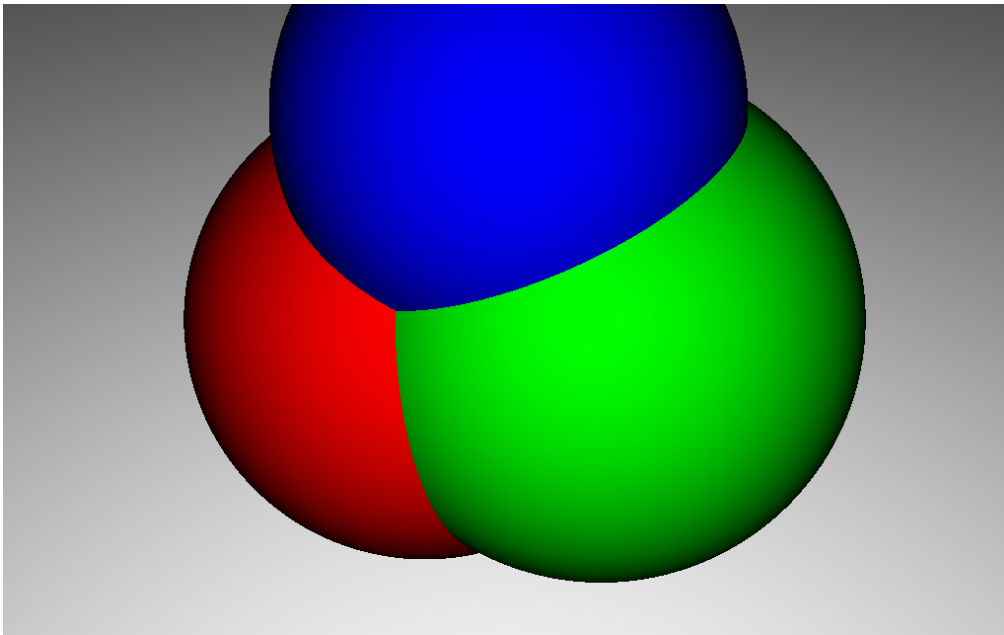
<http://mathinfo.univ-reims.fr/image/siRendu/Documents/2004-Chap6-RayTracing.pdf>

Surfaces implicites

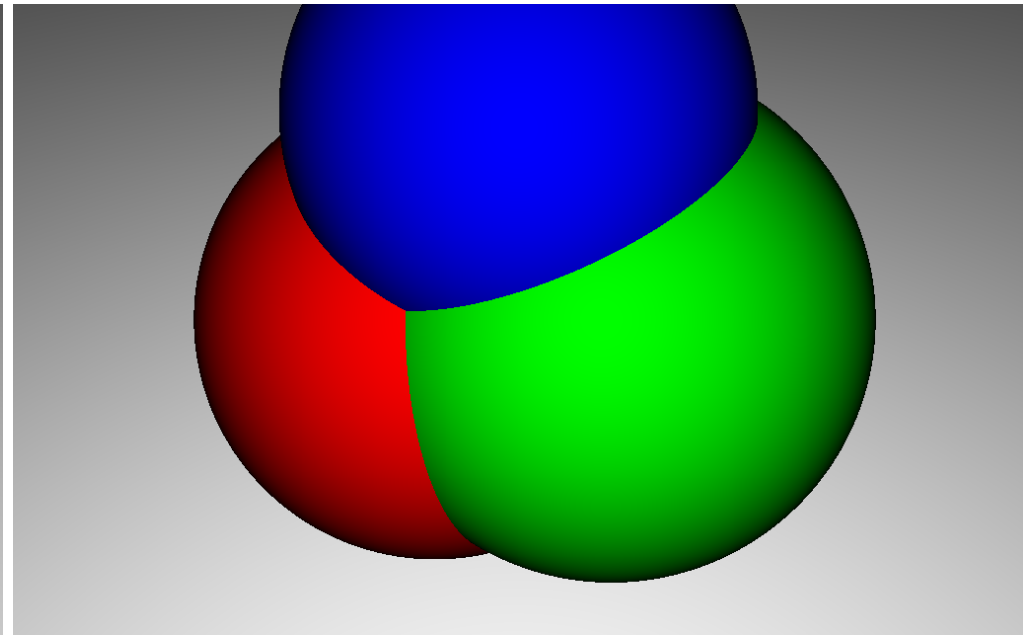
- Visualisation par lancer de sphères
 - Les fonctions de distances sont connues pour les surfaces classiques
 - Avec une liste d'objets
 - La distance à cette liste et la plus petite des distance à chaque objet
 - On lance donc un seul rayon par pixel, et on itère en progressant de cette distance

Surfaces implicites

- Visualisation par lancer de sphères



Lancer de rayon
~ 0,5 seconde



Lancer de sphère
~ 1 seconde ...

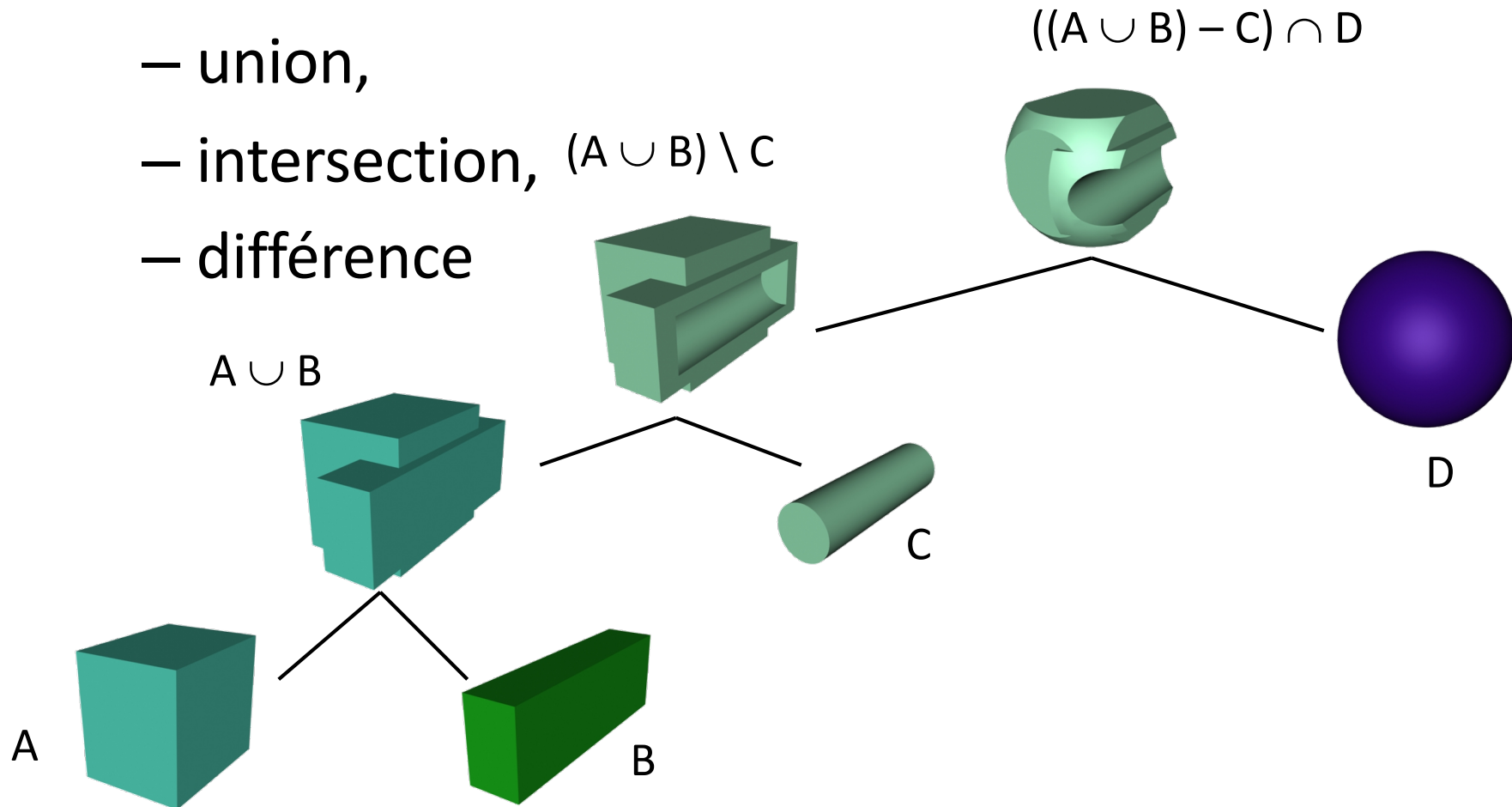
Objets construits par opérations booléennes

- Opérations booléennes :

- union,

- intersection, $(A \cup B) \setminus C$

- différence



Géométrie constructive CSG

- Représentation des objets par un arbre de construction
 - les feuilles :
 - représentation volumique d'objets simples ou paramétrés
 - les noeuds :
 - opérations booléennes (régularisées)
 - transformations affines
 - opérations d'extrusion
- Passage CSG $\Rightarrow$ B-Rep
 - évaluation du bord des objets
- Passage B-Rep $\Rightarrow$ CSG
 - opérations booléennes sur des objets définis par leur bord

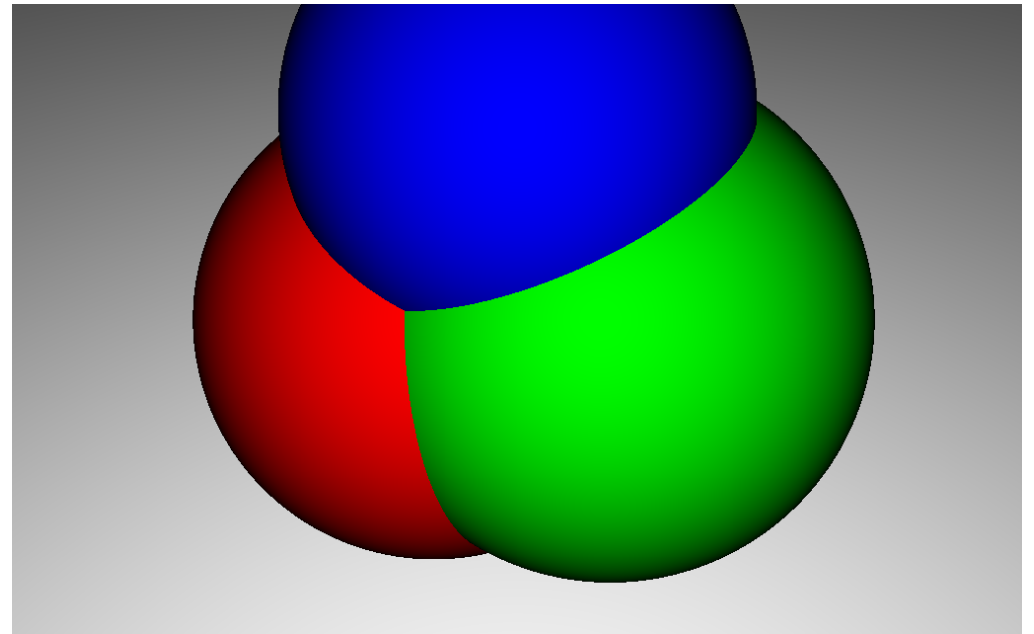
CSG

- Lancer de sphères
 - Soient A et B deux surfaces implicites et f_A et f_B les fonctions de distance **signées**
 - La distance est négative si le point P est à l'**intérieur** de l'objet
 - Les objets sont donc forcément fermés et orientables
 - On détermine les fonctions booléennes en combinant les fonctions de distance

CSG

- Lancer de sphères
 - L'**union** de A et B est calculée par lancer de sphère avec la fonction de distance

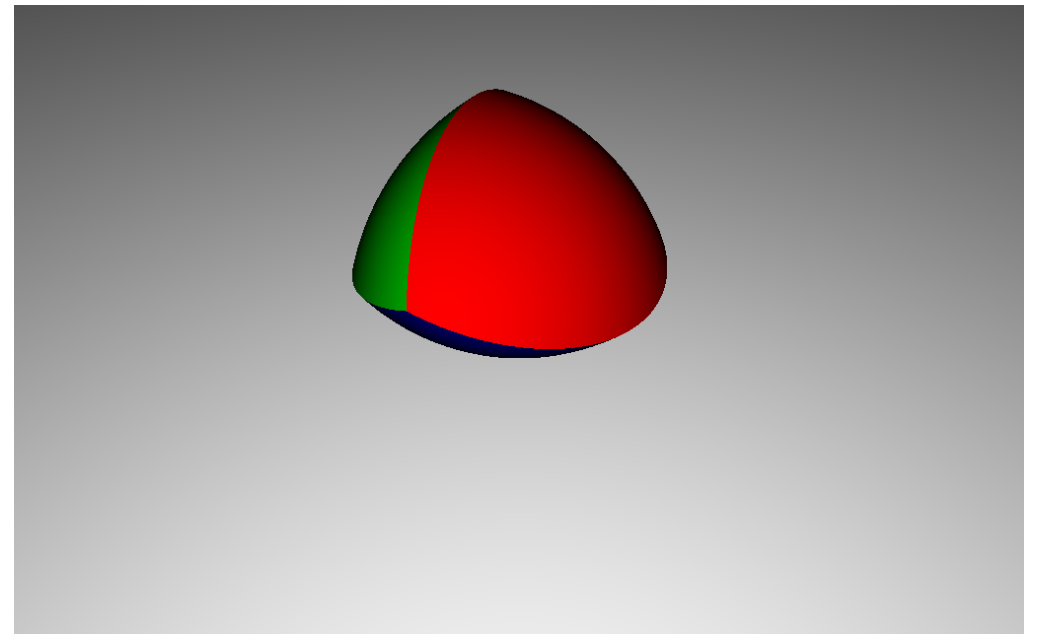
$$d(P, A \cup B) = \min(f_A(P), f_B(P))$$



CSG

- Lancer de sphères
 - L'**intersection** de A et B est calculée par lancer de sphère avec la fonction de distance

$$d(P, A \cap B) = \max(f_A(P), f_B(P))$$



CSG

- Lancer de sphères
 - Le **complémentaire** de A dans $\mathbf{R}^3$ est calculé par lancer de sphère avec la fonction de distance

$$d(P, \mathbf{R}^3 \setminus A) = -f_A(P)$$

CSG

- Lancer de sphères
 - La **différence** de A et B est calculée par lancer de sphère avec la fonction de distance

$$d(P, A - B) = d(A \cap \mathbf{R}^3 \setminus B) = \max(f_A(P), -f_B(P))$$

