1. C'est un champ scalaire surfacique discrète

↓ une seule donnée : z

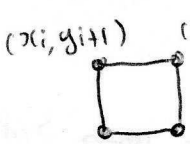
↓ on dépend de 2 variables (x, y)

↳ on a des points discrets (x_i, y_i)

$$z = f(x, y)$$

(voir cours intro Visualisation + cours 1)

2. On peut rapidement former une structure en quads avec ces coordonnées paramétrés par (x_i, y_i)



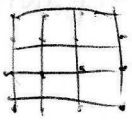
(voir TP. 7)

structures de données : vector <double> vertex
~~connectivity~~ vector <int> connectivity

construction de la connectivité → Pour $kx = 0 \dots N_x - 1$
Pour $ky = 0 \dots N_y - 1$

$$\text{connectivity} \leftarrow [\, kx + N_x ky \;,\; kx+1 + N_x ky \;,\; kx+1 + N_x (ky+1) \;,\; kx + N_x (ky+1) \,]$$

3.



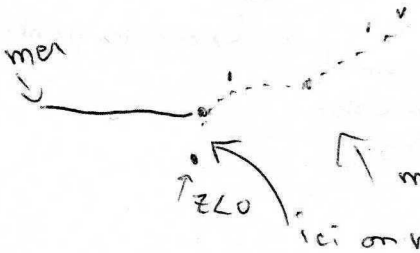
- B-Spline est possible car on a une structure en grille régulière
⇒ on peut définir des patchs paramétriques

(voir cours spline + révisions)

Avantage : montagnes lisses avec raccord C^2 ✓✓

- Bezier : Peu d'intérêt = Les raccords seront difficiles à réaliser ou bien il faut considérer tt les points de un même patch (degré trop élevé)

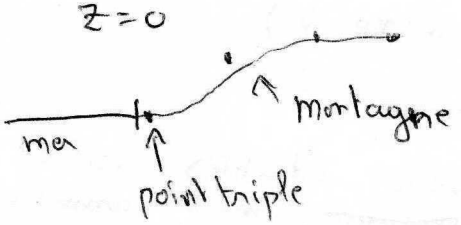
- Edition Hiérarchique : Tout à fait possible et approprié pour B-Spline !
↳ le contrôle est locale.



ici on ne commence pas au premier point.

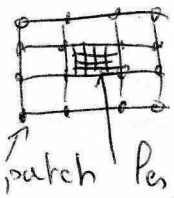
inconvénient : - IP faut calculer analytiquement ce z
- On aura une discontinuité de normale.

- Possibilité 2 : On double voir triple les points du bord et on fixe



(voir cours Spline + exo personel)

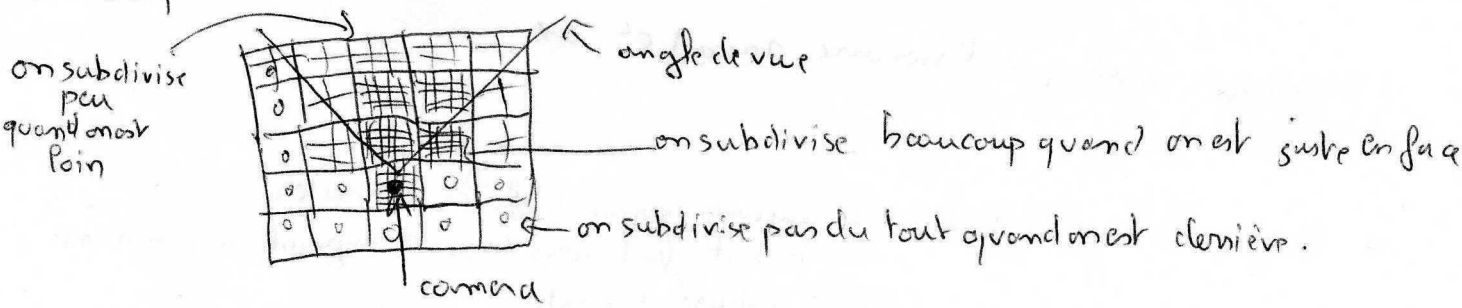
5. on va considérer des B-splines classiques au degré 3. (patch 4x4)



On subdivisera donc tous les "quads" sauf ceux du bord par $M_x \times M_y$ sommets.

on a donc au total $(N_x-2)(N_y-2)\,M_x M_y$ points.

6. On peut subdiviser chaque patch indépendamment grâce aux splines.



=> Pour cela, on compare la position de la caméra au barycentre de chaque patch par exemple.

rem. si M ∈ matrice caméra
p ← position du centre d'un patch.

$M \cdot p = \tilde{p}$ ← position du patch dans la vu 2D ($\tilde{z}$ nous donne la profondeur).

(voir TP.6)

$z = z_2$ — c_2

7. $z = z_1$ — c_1 ← z quelconque. ⇒ il suffit d'interpoler la couleur par rapport à z.

$z = 0$ — c_0

Par exemple, on peut faire une interpolation linéaire.

$z \in [0, z_1]$ $\quad c(z) = \alpha_1 c_1 + (1-\alpha_1) c_2$ $\quad \alpha_1 = z/z_1$

$z \in [z_1, z_2]$ $\quad c(z) = \alpha_2 c_2 + (1-\alpha_2) c_1$ $\quad \alpha_2 = \dfrac{z - z_1}{z_2 - z_1}$

(voir TP. 7 interpolation de couleurs)

↑ composante à composante

Rem: On pourrait faire du degré 2 pour éviter la discontinuité de dérivé.

8. Pour appliquer les couleurs partout, il suffit d'utiliser la même interpolation que pour l'altitude z, mais sur les couleurs composante à composante :

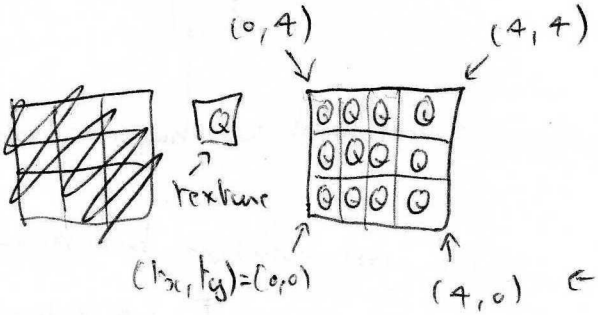
$$[R\ G\ B]^{(u,v)} = \sum_i \sum_j b_i(u) b_j(v)\ C_{ij}^{(u,v)}$$

(voir cours Splin

$C_{ij}^{(u,v)}$ ↑ $[R_i\ G_i\ B_i]$

9. La 2nd méthode semple mieux respecter les données.
- Cependant attention à l'interpolation sur les couleurs ! On va voir l'apparitions de fausses couleurs.

10.



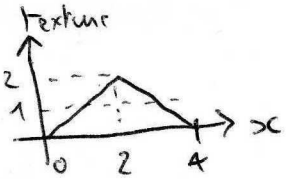
(voir TP7)

← ici on va répéter 4 fois la texture

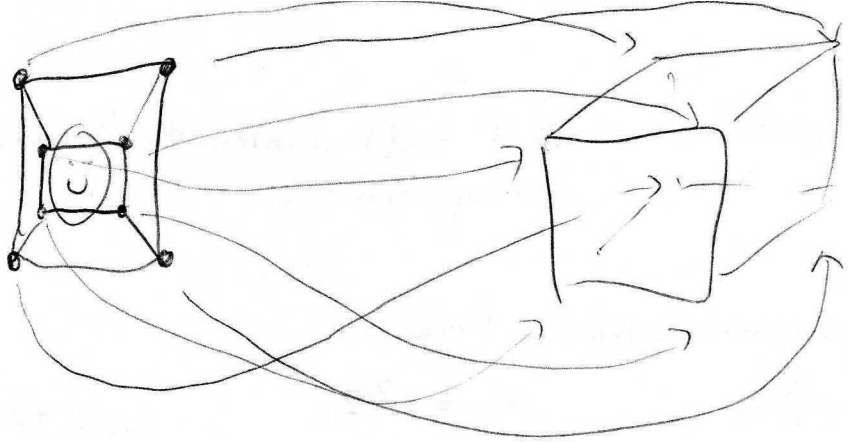
Attention au raccolement en passant d'un patch à l'autre (on va passer de (4,0) à (0,0) d'un coup sur le patch suivant).

⇒ Solution : - duplication du sommet, et coussure dans la connectivité
⇒ OpenGL n'interpolera pas.

- utiliser une fonction périodique. Par ex

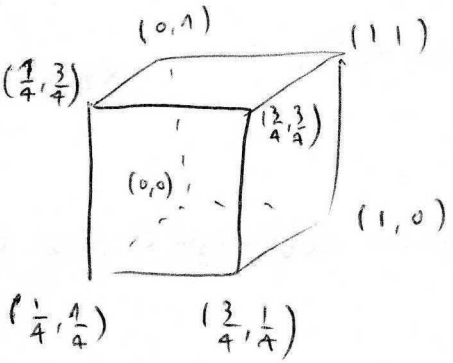


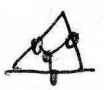
11

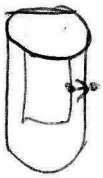


voir TP. 7

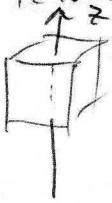
Les coordonnées sont ici



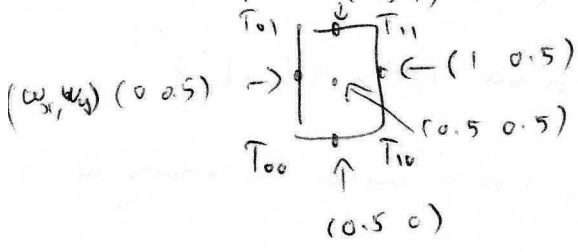
12. Subdivision de face simple  

=> On projete les nouveaux points sur le cylindre  (TP. 5)

avec $(x', y', z') = \frac{(x, y, z)}{\sqrt{x^2 + y^2}} \times R$

si on part d'un cube centré et de hauteur h le long de l'axe z 

- On interpole les coordonnées de textures directement sur chaque quels par exemple



-> ici interpolation bilinéaire. (TP. 7 + TP. 8)

$T = \omega_x \omega_y T_{11} + (1-\omega_x)\omega_y T_{01} + \omega_x(1-\omega_y) T_{10} + (1-\omega_x)(1-\omega_y) T_{00}$

- On différencie les coordonnées à ne pas projeter par marqueur par exemple

$N_f(k) = 4^k N_{f_0} = 4^k \cdot 6$ (TP).

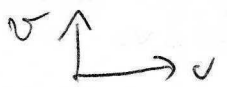
Nombre d'arêtes = $\frac{4}{2}(6 \cdot 4^k) = 2 \times (6 \cdot 4^k)$

(4 arêtes par faces ; ↑ chaque arête est partagé pour 2 faces.)

Nombre de Sommets $\Rightarrow 6 \cdot 4^k - 2 \cdot (6 \cdot 4^k) + 1 = 2$ (↙ pas de trous.)

$\Rightarrow s = 2 + 6 \cdot 4^k$.

13. B-Spline pas possible, on a pas une structure paramétrique



(Tous spline + Subdivision + révision).

→ on peut utiliser Catmull Clark par exemple.

14. vertex < >

connectivity < 1 2 3 / 4 5 6 7 / 9 10 12 16 22 33 7 / 8 9 5 >

size-poly < 0 3 7 14 > (TP)

accès-poly < 3 4 7 3 >

↳ on accède par : vertex [3 connectivity [accès-poly [p] + s] + k-dim]

(0 pour x, 1 pour y, 2 pour z.)

15 : À la seconde subdivision :

$N_{f_2} = N_{st} N_f + N_e$ ← donné par doo-Sabin.

(↑ sommet, ↑ face, ↑ edge)

On $N_{f_2} - N_{e_2} + N_{s_2} = 2 \Rightarrow N_{s_2} = N_f - 2$ car $N_{e_2} = \frac{4}{2} N_{s_2}$

Après on augmente le nombre de sommet par 4 à chaque étapes... (~~connectiv~~ adjacente de 4).

16. Il faut utiliser Butterfly car on doit interpoler les sommet.
On ne peut pas les déplacer ici !) (cours subdiv + révision

17. Ici c'est un champ vectoriel volumique discret.
↳ (v_x, v_y, v_z) ↳ dépend de (x, y, z)

$\vec{v} = f(\vec{x})$
↑ (v_x, v_y, v_z) ↑ (x, y, z) (cours visualisation)

18. On peut faire un rendu volumique. Par exemple l'MIP.

Mieux : On atténue beaucoup pour $\|v\| > v_0$ (cours Rendu volumique)
et pas pour $\|v\| < v_0$
⇒ On travail sur le champ scalaire $\|v\|$

19 : On a 4 sommets : 16 possibilités ... (voir marching cube)