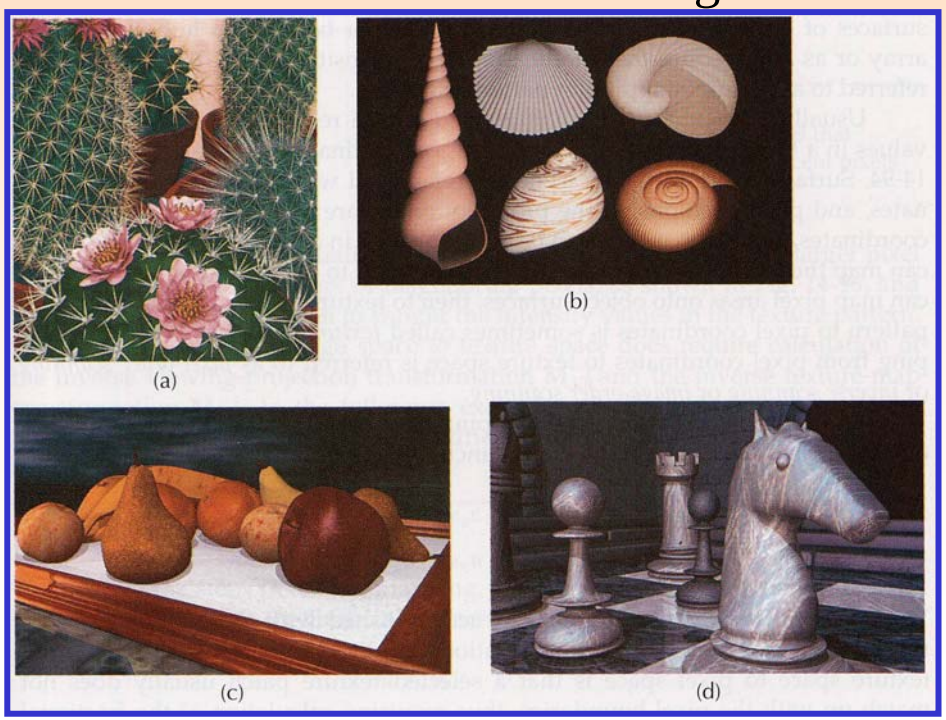
Modèles de génération de texture

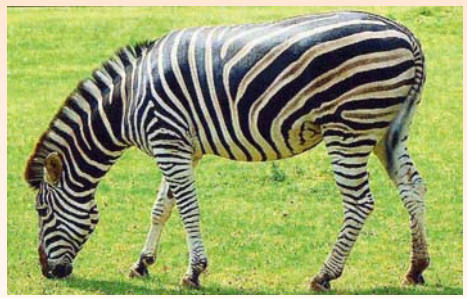
Modèles de génération de texture

- Éliminer le caractère plutôt synthétique des objets.

  Modéliser des surfaces telles que des murs de briques, des routes de gravier, des tapis au long poil, etc.

  Prendre en compte des motifs que l'on veut voir apparaître sur des surfaces : la surface d'un vase où l'on a reproduit une peinture célèbre, un terrain de soccer renfermant les lignes blanches règlementaires, etc.





Hearn & Baker, Computer graphics C version. Prentice Hall, 1997.

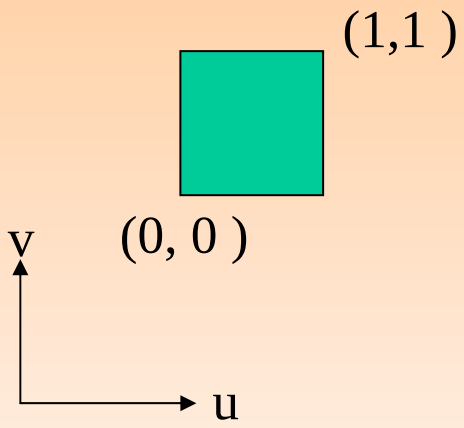
Différents modèles de génération de texture

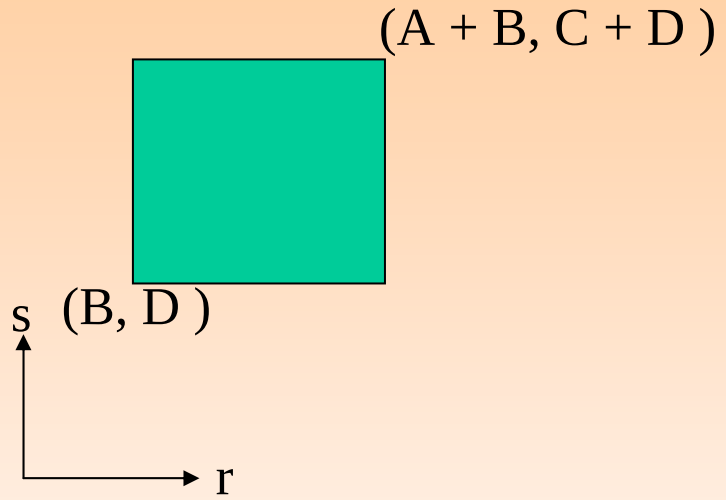
- Il s'agit d'appliquer à la surface d'un objet des variations inégales de couleurs ou des effets de rugosité.
- En infographie, une texture peut donc être vue comme :
  - (a) une image ou un tableau 2D de couleurs T,
    - Comment appliquer T à une surface quelconque ?
    - Comment générer T à l'aide de modèles mathématiques ou de techniques d'analyse de texture ?
  - (b) un tableau 3D de couleurs T,
    - Comment générer T ?
  - (c) une approche procédurale pour générer les variations de couleurs,
  - (d) des perturbations appliquées au vecteur normal à la surface.
    - Comment définir la rugosité de la surface ?

(A) une texture vue comme une image ou un tableau 2D de couleurs T

- Comment appliquer T à une surface quelconque ?

Définir une fonction F d'application qui permet de passer d'un système à l'autre :





Surface S(u, v), u, v ∈ [0, 1]

texture

Étant donné un élément de la surface correspondant à (u, v), alors l'élément correspondant dans la texture est de la forme :

$$\begin{pmatrix} r \\ s \end{pmatrix} = F \begin{pmatrix} u \\ v \end{pmatrix}$$

Ex. I

$r = A\,u + B$

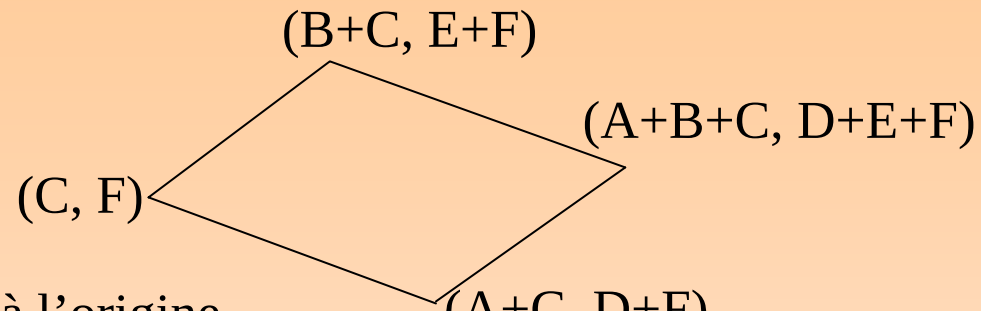
$s = C\,v + D$

Appliquer une texture à une surface

Ex. II

$r = A\,u + B\,v + C$

$s = D\,u + E\,v + F$



Ex. III

Sphère de rayon 1 centrée à l'origine

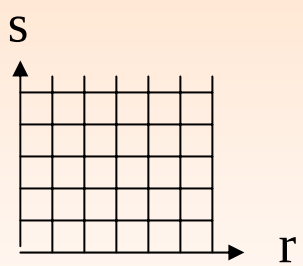
$x = \sin\theta \sin\phi$

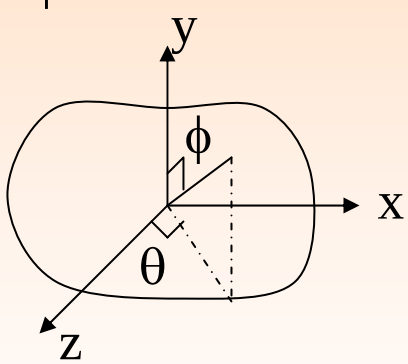
$y = \cos\phi$

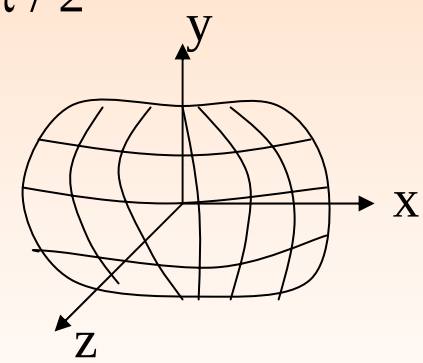
$z = \cos\theta \sin\phi$

$0 \le \theta \le \pi / 2$

$\pi / 4 \le \phi \le \pi / 2$

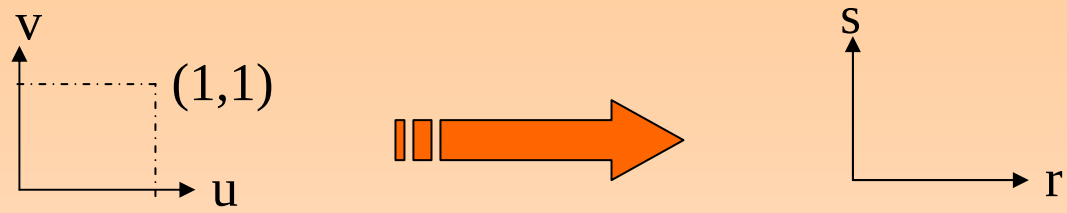






$$r = \frac{\theta}{\frac{1}{2}\pi} \qquad \text{et } s = \frac{\phi - \frac{1}{4}\pi}{\frac{1}{4}\pi}$$

Appliquer une texture (tableau de couleurs) à une surface courbe quelconque



Surface définie dans un espace paramétrique

texture

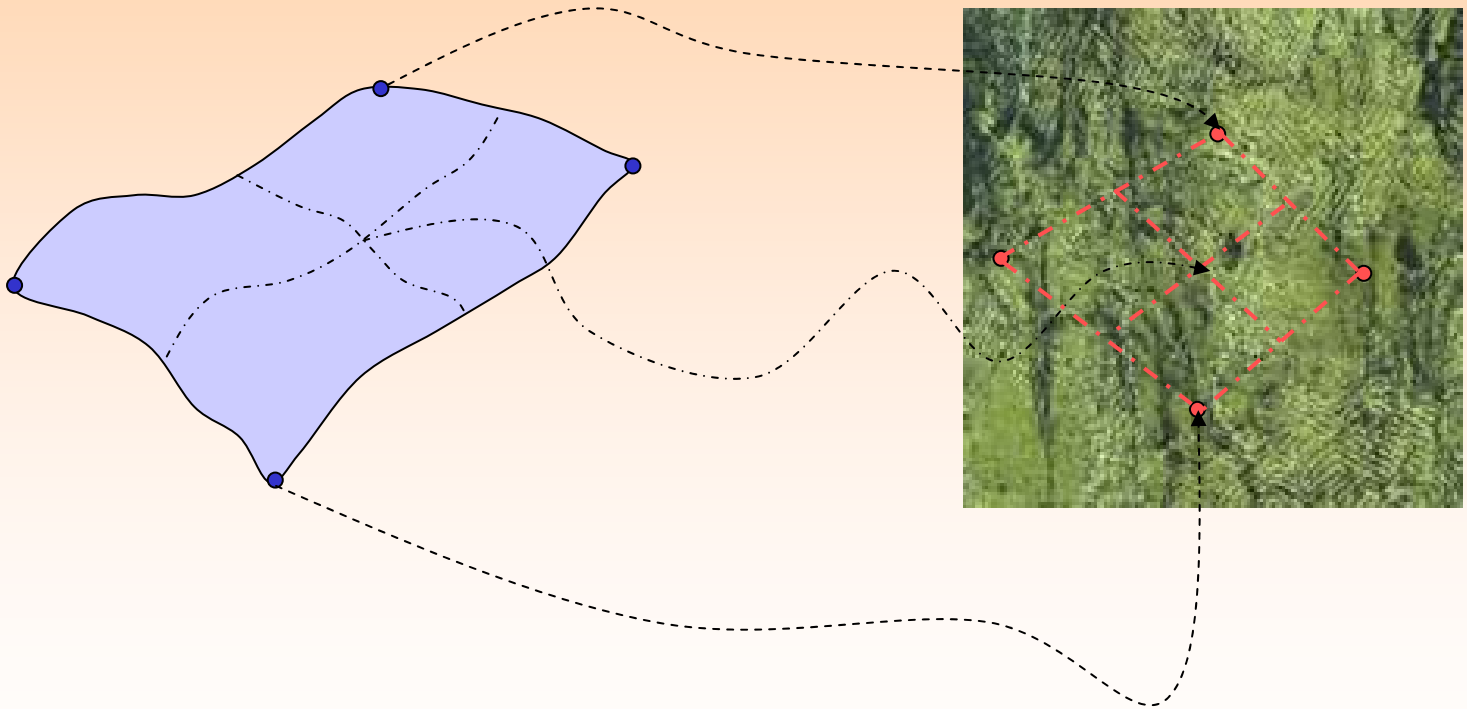
Pour appliquer une texture à une surface courbe quelconque, on peut établir une relation entre l'espace normalisé u x v où la surface est définie et l'espace r x s de la texture; cela donne lieu à des déformations de la texture.

- Pour pallier à ces déformations, on considère un algorithme de subdivision lequel est applicable aux techniques de points de contrôle. Ex. : surfaces de Bézier.
- Il s'agit d'associer aux 4 points S(0, 0), S(0, 1), S(1, 0) et S(1, 1) une position dans la texture. Puis, lorsque S est subdivisée en 4 « sous-surfaces » de Bézier, on fait de même pour la texture.
- Le processus prend fin lorsque les « sous-surfaces » sont « quasi-planes ». À ce moment-là, on applique la texture résultante à l'approximation plane.

Appliquer une texture (tableau de couleurs) à une surface courbe quelconque

Surface de Bézier

texture



(A) une texture vue comme une image ou un tableau 2D de couleurs T

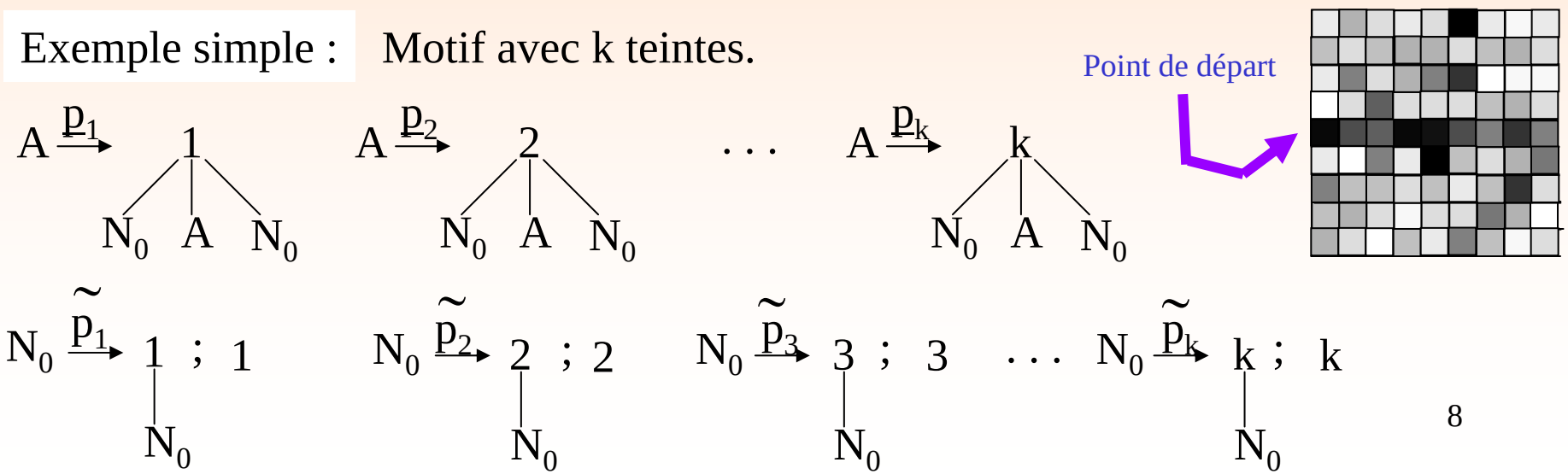
- Comment générer T à l'aide de modèles mathématiques ou de techniques d'analyse de texture ?

1. Approche syntaxique

- Chaque texture renferme un ou plusieurs motifs. Chaque motif de dimension L x L (texture) est défini à partir des règles de production d'une grammaire où chaque symbole terminal représente la couleur d'un pixel (motif).

- Pour éviter de reproduire systématiquement la même texture, la notion de grammaire stochastique est introduite pour tenir compte par ex. du bruit et de la distorsion.

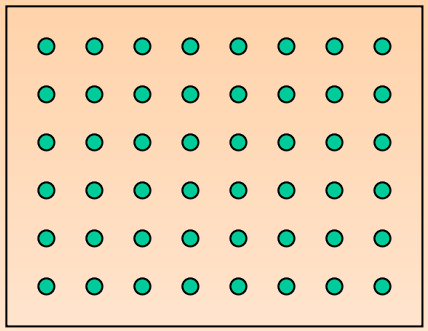
Exemple simple : Motif avec k teintes.

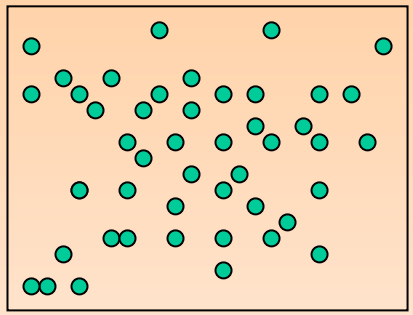


2. Modèle de croissance (Yokoyama & Haralick[78])

Ce modèle simule la croissance des cellules dans la nature.

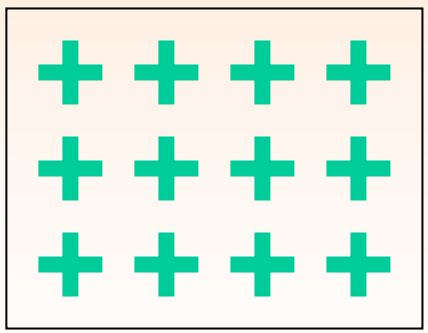
1. Création de cellules souches de façon déterministe ou non.

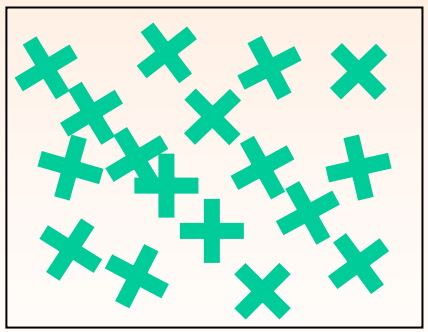


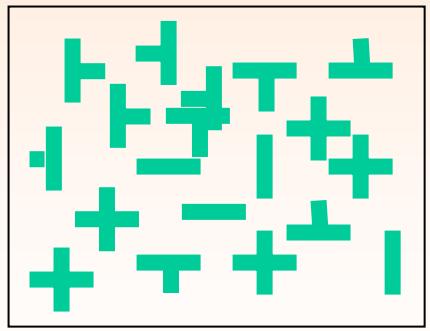


2. Développement des cellules souches en motifs réguliers ou aléatoires.

A) déterministe B) stochastique







3. Évolution de chaque cellule (croissance)
   toutes les cellules peuvent se reproduire.

4. Affichage : plusieurs types de cellules

3. Modèle de séries chronologiques

- Une texture est définie à l'aide d'un processus stochastique. Il s'agit d'une série de teintes où chaque nouvelle teinte est générée à partir des M dernières teintes générées.

- Soit X_j, la $j^{ième}$ teinte de la texture, celle-ci est une combinaison linéaire convexe des M dernières teintes générées auxquelles on additionne un bruit blanc.

$$X_j = \phi_1 X_{j-1} + \phi_2 X_{j-2} + \ldots + \phi_M X_{j-M} + e$$

où $\phi_1, \phi_2, \ldots \phi_M$ sont des coefficients à estimer à partir de textures originales, e est un bruit blanc i.e. une variable aléatoire obéissant à une loi normale $N(0, \sigma^2)$.

- Le bruit blanc e permet d'introduire une distorsion, un caractère flou à la texture générée. Plus σ^2 est élevé, plus la distorsion est importante.

- Pour générer les M premières teintes de la texture, l'usager doit fournir le vecteur de probabilités suivant :

probabilité de générer la $j^{ième}$ teinte $\equiv$ Prob(T = j), j = 1, 2, …, k

sachant qu'il existe k teintes au total.

4. Modèle basé sur des processus de Markov

- Pour générer la première teinte de la texture, l'usager doit fournir le vecteur de probabilités P suivant :

  probabilité de générer la $j^{ième}$ teinte $\equiv P[j] = Prob(T = j), j = 1, 2, \ldots, k$

  sachant qu'il existe k teintes au total.

- Pour générer les teintes suivantes, nous avons besoin d'une matrice de probabilités de transitions M (passage d'une couleur à l'autre) :

  probabilité de choisir la $i^{ième}$ teinte sachant qu'au pixel précédent, la $j^{ième}$ teinte a été choisie $\equiv M[i, j] = Prob(T = i \mid T = j)$, $i, j = 1, 2, \ldots, k$

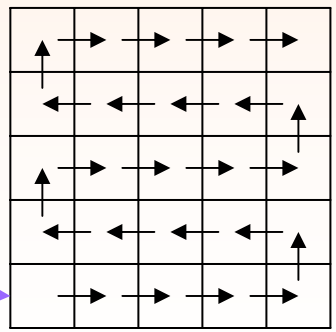
- **Algorithme :** Soient $t_1, t_2, \ldots, t_n$ les n couleurs à générer dans la texture,

  1. $t_1 \leftarrow$ Generer_teinte(P).
  2. Pour i allant de 2 à n faire $t_i \leftarrow$ Generer_teinte($M[., t_{i-1}]$).

```
entier entre 1 et k Generer_teinte(vecteur Q à k composantes)
{
        choisir un nombre H au hasard entre 0 et 1.
        Etat ← 1.
        Acc ← Q[Etat].
        Tant que Acc < H faire      Etat ← Etat + 1
                                    Acc ← Acc + Q[Etat].
        Retourner la valeur de Etat.
}
```

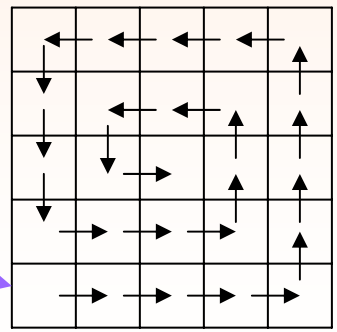
- Dans les modèles de séries chronologiques et ceux basés sur les processus de Markov, on doit non seulement générer les différentes teintes de la texture mais aussi choisir le mode de parcours des pixels.

Balayage horizontal

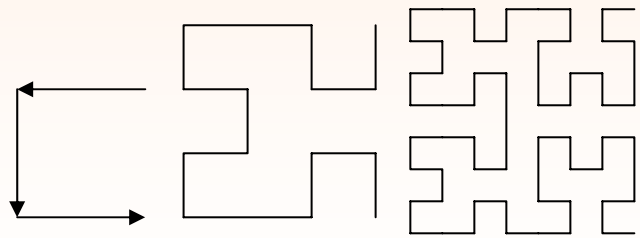


Point de départ

Courbes en spirale



Courbes de Hilbert



5. Synthèse d ’une texture ayant les caractéristiques de la « texture-mère »

$P(i, j) \equiv$ Probabilité que 2 cellules voisines aient respectivement i et j comme teinte dans l’image mère.

$Q(i, j) \equiv$ Probabilité de choisir une teinte j dans la nouvelle texture étant donné que la teinte correspondante dans la « texture-mère » est i.

$$\sum_j Q(i, j) = 1, \quad \forall i$$
$$Q(i, j) \geq 0, \quad \forall i, j$$

$r(k, h) \equiv$ Probabilité que 2 cellules voisines aient respectivement k et h comme teinte dans l’image synthétisée à partir de l’image-mère ».

$$r(k, h) \equiv \sum_i \sum_j Q(i, k)\, Q(j, h)\, P(i, j)$$

$$R = Q^t P Q$$

$f(k) \equiv$ Probabilité de rencontrer la $k^{\text{ième}}$ teinte dans une cellule quelconque de l’image synthétisée.

$$f(k) \equiv \sum_h r(k, h)$$

Utiliser « Generer_teinte » avec ce vecteur de probabilités pour générer les différentes teintes.

6. Modèle basé sur la théorie fractale

Les règles de construction (les propriétés statistiques) d'une texture sont les mêmes peu importe l'échelle où nous sommes.

Soit une texture T représentée à l'aide d'une grille de pixels d'ordre n où n est habituellement une puissance de 2,

les couleurs aux 4 extrémités de la grille sont fixées :

T(0,0), T(n, 0), T(0, n) et T(n, n).

- Programme principal :

Initialiser les couleurs suivantes : T(0,0), T(n, 0), T(0, n) et T(n, n);
T[0, 0].rouge = N (T[0, 0].rouge, σ_0);
T[0, 0].vert = N(T[0, 0].vert, σ_0);
T[0, 0].bleu = N(T[0, 0].bleu, σ_0);

Loi normale

. . .

Calcul_texture(T, 0, n, 0, n, 1);

- int Calcul_couleur(int Niveau_de_recurrence, couleur Q[], int p_1, int p_2)
{
    calculer $c_{milieu} = (Q[p_1] + Q[p_2]) / 2$;
    calculer $p_{milieu} = (p_1 + p_2) / 2$;
    $Q[p_{milieu}].rouge = N(c_{milieu}.rouge, \sigma_0\ 2^{-\ Niveau\_de\_recurrence\ *\ h})$
    $Q[p_{milieu}].vert = N(c_{milieu}.vert, \sigma_0\ 2^{-\ Niveau\_de\_recurrence\ *\ h})$
    $Q[p_{milieu}].bleu = N(c_{milieu}.bleu, \sigma_0\ 2^{-\ Niveau\_de\_recurrence\ *\ h})$
    Retourne la position p_{milieu}.
}

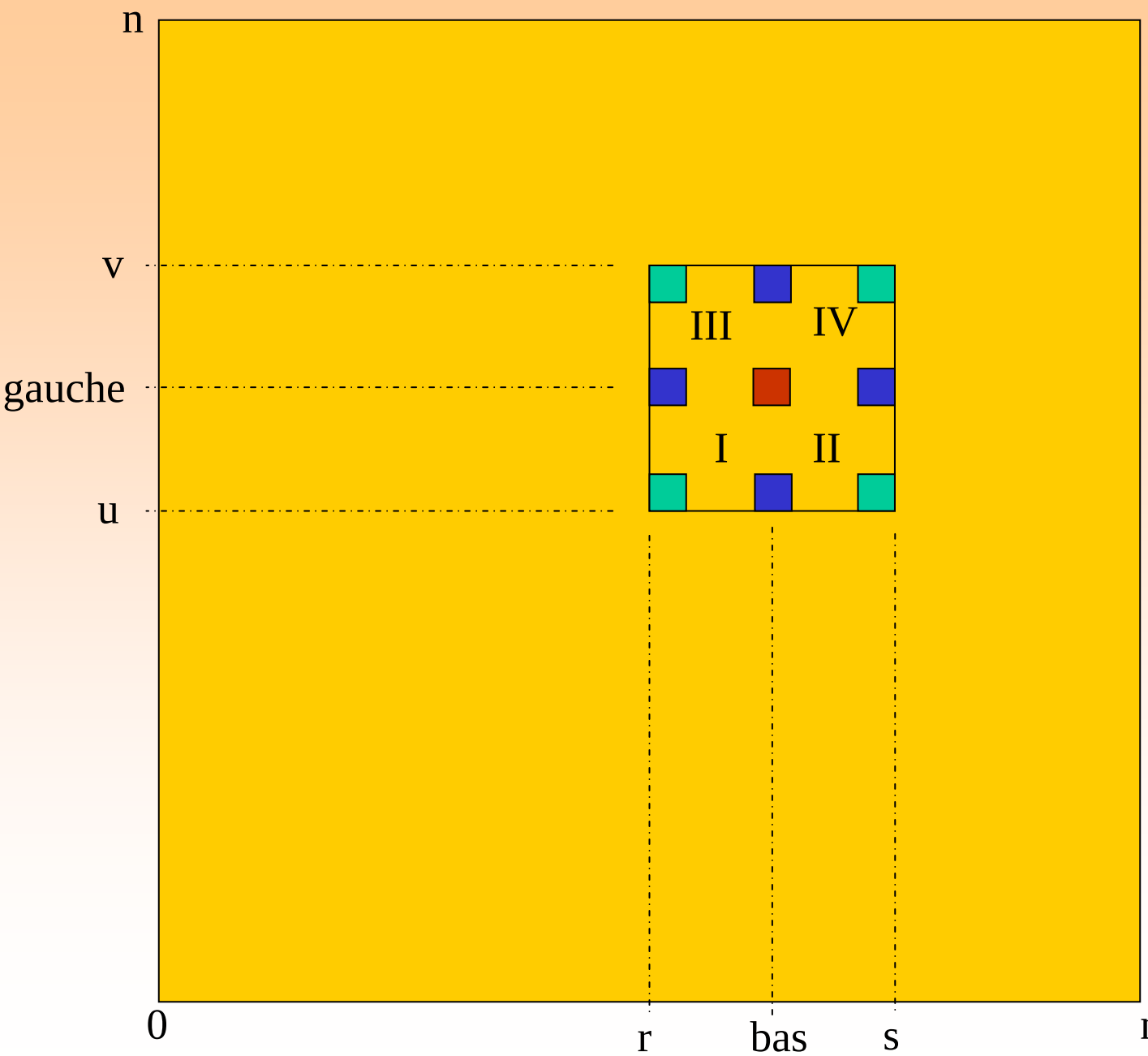
Extension : $h \longmapsto h(n)$ — h petit : surface très rugueuse ; h élevé : surface unie.

```
void Calcul_texture(couleur T[][], int r, int s, int u, int v, int Niveau)
{
        bas = Calcul_couleur(Niveau, T[u, .], r, s);
        bas = Calcul_couleur(Niveau, T[v, .], r, s);
        gauche = Calcul_couleur(Niveau, T[., r], u, v);
        gauche = Calcul_couleur(Niveau, T[., s], u, v);

        bas = Calcul_couleur(Niveau, T[gauche, .], r, s);
        Couleur_du_centre = T[gauche, bas];
        gauche = Calcul_couleur(Niveau, T[., bas], u, v);
        T[gauche, bas] = ½ (T[gauche, bas] + Couleur_du_centre);

        Si bas est différent de r + 1 alors
                Calcul_texture(T, r, bas, u, gauche, Niveau + 1);
                Calcul_texture(T, bas, s, u, gauche, Niveau + 1);
                Calcul_texture(T, r, bas, gauche, v, Niveau + 1);
                Calcul_texture(T, bas, s, gauche, v, Niveau + 1);
}
```





I
II
III
IV


n
v
gauche
u
III
IV
I
II
0
r
bas
s
n

Désavantages rencontrés en utilisant un tableau 2D T de couleurs :

- Difficulté d'appliquer T à la surface.
- Difficulté de générer une texture T adéquate.
- Difficulté de copier des textures l'une à côté de l'autre sachant que le tableau est de taille réduite p/r à la surface.

(B) un tableau 3D de couleurs T

Pour chaque point (x, y, z) d'une surface, il doit exister une application F qui retourne la position d'une couleur dans le tableau T.

Note : On considère les coordonnées des points de la surface dans l'espace usager au lieu d'un espace paramétrique ce qui évite des distorsions.

Le tableau T exige beaucoup d'espace mémoire.

- Comment générer T ?

On peut adapter la plupart des modèles précédents pour générer T.

(C) une approche procédurale pour générer les variations de couleurs

- Prend en entrée les coordonnées (x, y, z) d'un point de la surface et retourne la couleur (R, V, B) à ce point

  ou encore,

  prend en entrée les coordonnées (u, v) d'un point défini dans un espace normalisé et retourne la couleur (R, V, B) à ce point.

Ex. I :

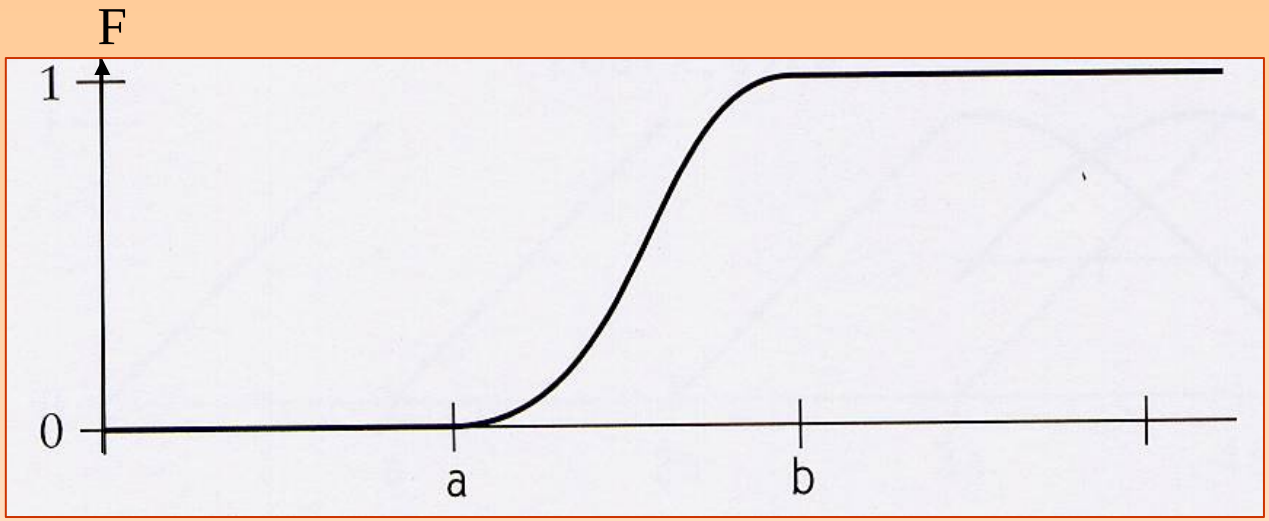
Imaginons une surface de Bézier S où les points de contrôle sont : P_i, i = 0, 1, 2, …, N-1.

On peut associer à chaque point P_i une teinte T_i. Ainsi, à chaque point S(u, v) correspond une couleur T(u, v), où T est une surface de Bézier où les T_i jouent le rôle de points de contrôle.

Ex. II :

À chaque point (x, y, z) correspond une couleur de la forme
$(F(a_1, a_2, x), F(b_1, b_2, y), F(c_1, c_2, z))$
où $a_1 \le x \le a_2$, $b_1 \le y \le b_2$, $c_1 \le z \le c_2$
représente le domaine où la surface est définie.

```
float F(float a, float b, float w)
{
        if (w < a) return 0;
        if (w >= b) return 1;
        w = (w – a) / (b – a);
        return (x * x * (3 – 2 * x));
}
```



Ex. III : Trouvez 3 fonctions f_R, f_V et f_B dont le domaine est celui de la surface S et l'image est dans [0, 1].

- Il peut s'agir aussi d'une fonction de $\Re^3 \rightarrow \mathbb{N}^+$; à chaque point (x, y, z) correspond l'indice d'une teinte.

Avantages de l’approche procédurale :

- Exige peu d’espace mémoire.
- Aucune limite de résolution n’est imposée.
- La surface où la texture doit être appliquée n’est pas fixée.
- Permet de modéliser une famille de textures en ajoutant des paramètres à la procédure.

Désavantages de l’approche procédurale :

- Difficile à construire pour représenter le réalisme voulu.
- Difficile de prévoir la texture que nous obtiendrons.
- Peut exiger des temps de calculs importants.

(D) des perturbations appliquées au vecteur normal à la surface

◦ Comment définir la rugosité de la surface ? Grâce à une fonction de perturbation.

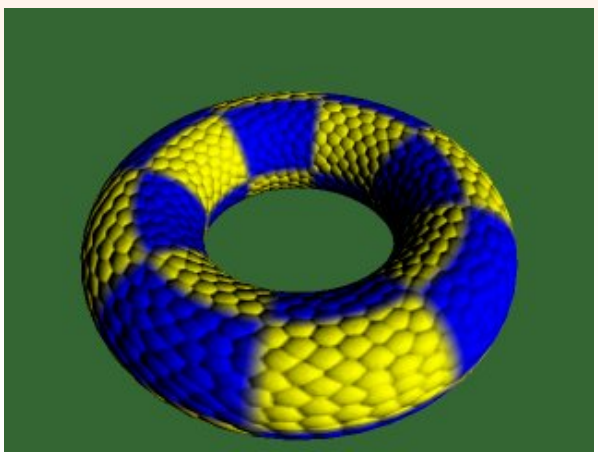
Il faut perturber la normale à la surface et utiliser la normale perturbée
dans les modèles d'illumination.

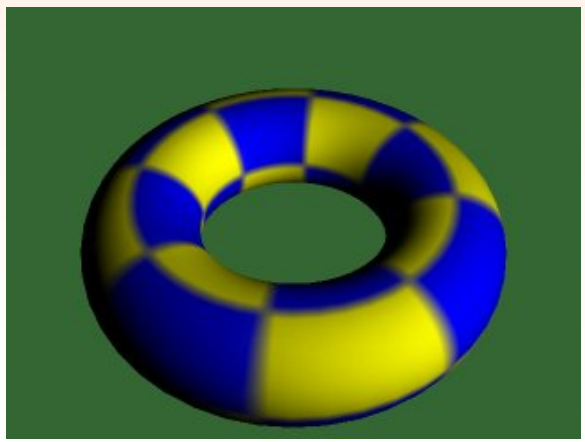
⟹ perturbation de la direction des rayons lumineux.

Soient $Q(u, v)$: l'équation d'une surface,
$Q_u(u, v)$: $\partial Q(u, v) / \partial u$
$Q_v(u, v)$: $\partial Q(u, v) / \partial v$
$N(u, v)$: $Q_u \times Q_v$: la normale à la surface Q
$P(u, v)$: fonction de perturbation de $[0, 1] \times [0, 1] \rightarrow \Re$

alors la nouvelle surface est :

$Q'(u, v) = Q(u, v) + P(u, v)\, N / \|N\|$





Calcul de la normale de la surface perturbée

$N'(u, v)$: $Q'_u \times Q'_v$: la normale à la surface Q'

$Q_u + P_u N / ||N|| + P(N / ||N||)_u \approx Q_u + P_u N / ||N||$

$Q_v + P_v N / ||N|| + P(N / ||N||)_v \approx Q_v + P_v N / ||N||$

$$N' \cong Q_u \times Q_v + P_v (Q_u \times N) / ||N|| + P_u (N \times Q_v) / ||N|| + P_u P_v (N \times N) / ||N||^2$$

$P_u P_v (N \times N) / ||N||^2 \rightarrow 0$

$$N' \cong N + P_v (Q_u \times N) / ||N|| + P_u (N \times Q_v) / ||N||$$

Note :

$Q(u, v) \longrightarrow 2\,Q(u, v)$

$\Rightarrow N(u, v) \longrightarrow 4\,N(u, v)$

$\Rightarrow P_v (Q_u \times N) / ||N|| + P_u (N \times Q_v) / ||N|| \longrightarrow 2\,(\ldots\ldots)$

$\therefore$ Réduction de la rugosité lorsque la surface augmente de taille.

◦ Comment perturber la normale à la surface ? En se basant sur la théorie fractale.

- Programme principal : soit une surface paramétrique S(u, v), u, v $\in$ [0, 1],

Calculer les normales à la surface S aux 4 extrémités : n_{00}, n_{10}, n_{01} et n_{11};
Calculer les vecteurs de perturbation aux normales :

$p_{00}.x = N(0, \sigma_0)$; $p_{00}.y = N(0, \sigma_0)$; $p_{00}.z = N(0, \sigma_0)$;

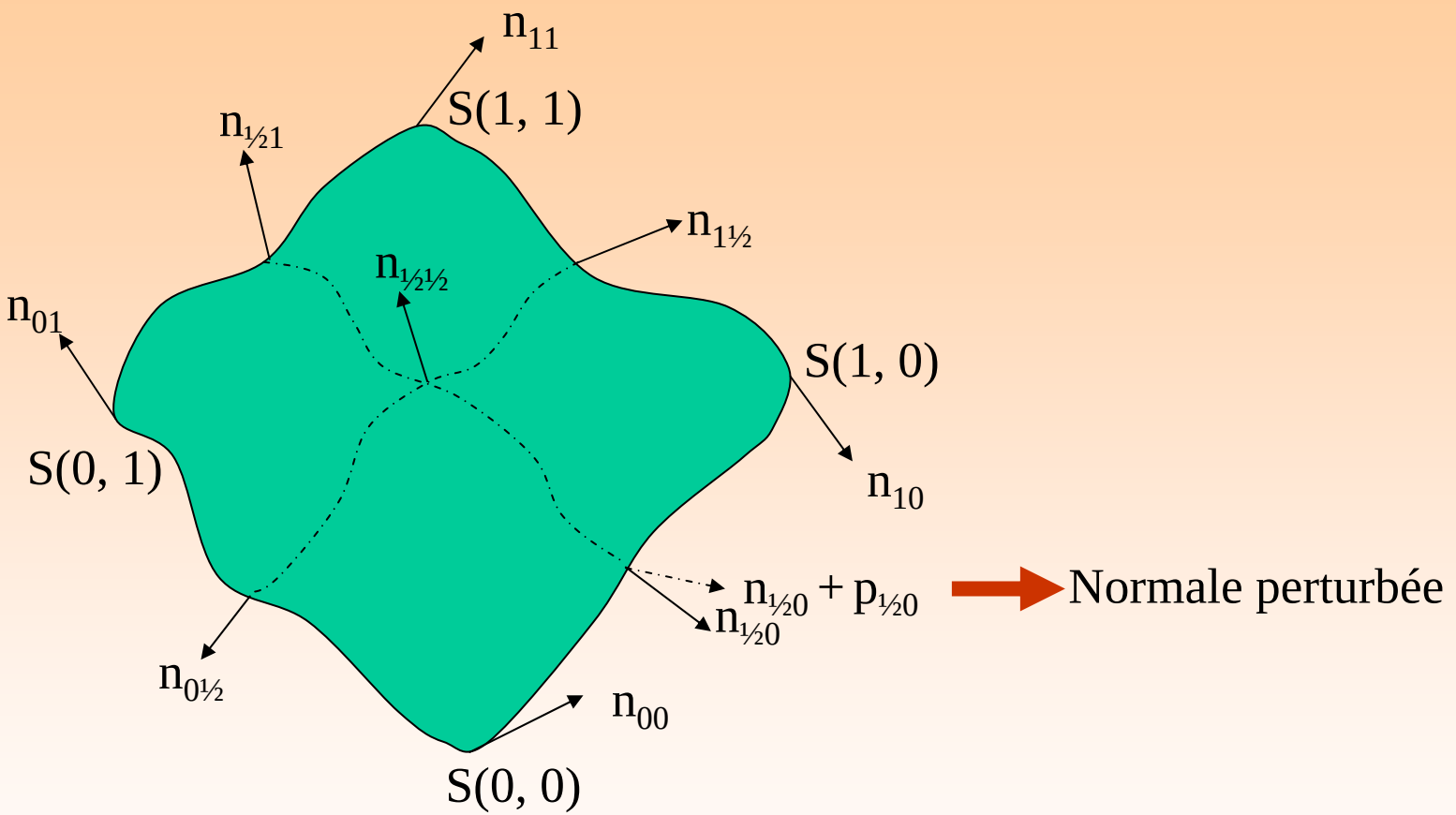
. . .

Ranger dans une liste L la définition de la surface S avec n_{00}, n_{10}, n_{01} et n_{11} et p_{00}, p_{10}, p_{01} et p_{11} avec le niveau 1 de récursivité.
Initialiser M à une liste vide.
Subdivision_surface(L, M);

-

```
vecteur Calcul_perturbation(int Niveau_de_recurrence, vecteur p, vecteur q)
{
        calculer p_milieu = (p + q) / 2;
        p_milieu.x = N(p_milieu.x, σ_0 2^- Niveau_de_recurrence * h )
        p_milieu.y = N(p_milieu.y, σ_0 2^- Niveau_de_recurrence * h )
        p_milieu.z = N(p_milieu.z, σ_0 2^- Niveau_de_recurrence * h )
        Retourne le vecteur p_milieu.
}
```

```
void Subdivision_surface(liste L, liste M)
{
    Tant que la liste L n'est pas vide
        Enlever de la liste L la définition d'une surface S avec n00, n10, n01 et
        n11 les normales aux 4 extrémités et p00, p10, p01 et p11 les perturbations
        aux 4 extrémités avec le niveau m de récursivité.
        Décomposez S en 4 sous-surfaces S1, S2, S3 et S4.
        Calculer n½0, n0½, n1½, n½1 et n½½ les normales de S à (0, ½), (½, 0),
        (½, 1), (1, ½) et (½, ½).
         p½0 = Calcul_perturbation(m, p00, p10);
         p0½ = Calcul_perturbation(m, p00, p01);
         p½1 = Calcul_perturbation(m, p01, p11);
         p1½ = Calcul_perturbation(m, p10, p11);
         p½½ = ½(Calcul_perturbation(m, p0½, p1½)+Calcul_perturbation(m, p½0, p½1));
        Si la sous-surface S1 est « quasi-plane »
        alors    ranger S1 dans la liste M avec n00, n½0, n0½ et n½½ les normales
                 aux 4 extrémités et p00, p½0, p0½ et p½½ les perturbations aux 4
                 extrémités avec le niveau m+1 de récursivité.
        sinon    ranger les mêmes informations dans la liste L.
        . . .
}
```



On utilise une technique de subdivision.