Remplissage

Le remplissage d'une zone est le « coloriage » de cette zone ou région.

Les zones peuvent être définies au niveau des pixels ou au niveau géométrique.

a) Au niveau des pixels : La zone est décrite

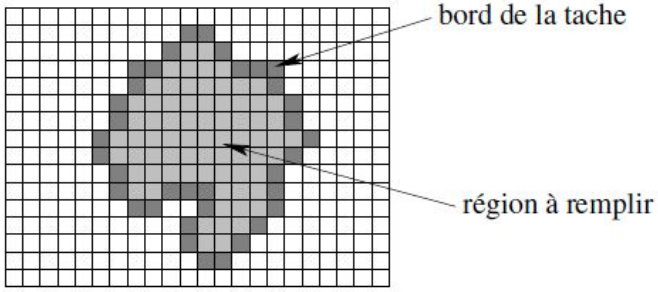
- soit pat l'ensemble des pixels qui la composent = zone surfacique.

  Algorithme de remplissage = algorithme de coloriage

- Soit par les pixels qui la bordent = la région est un contour

b) Au niveau géométrique : Une zone est décrite en termes d'objets, segments, polygones, circonférences.

Remplissage de régions



Principe :

Ce type d'algorithme traite les régions définies par une frontière.

A partir d'un pixel (germe) intérieur à la région, on propage récursivement la couleur de remplissage au voisinage de ce pixel jusqu'à atteindre la frontière.

Remplissage d'une région donnée par son contour

Problème

On souhaite remplir avec la couleur de remplissage (CR) une région dont le contour est formé par des points jointifs de couleur contour (CC).

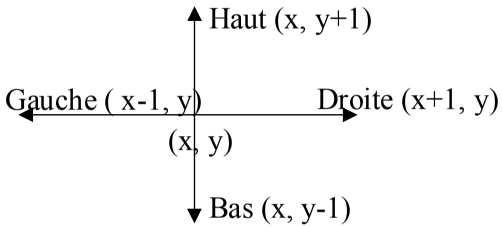
Si des points de la surface sont de couleur CC ou CR, ils ne seront donc pas modifiés.

Cette propriété permet de définir des régions avec trous à l'aide de contours intérieurs de la région.

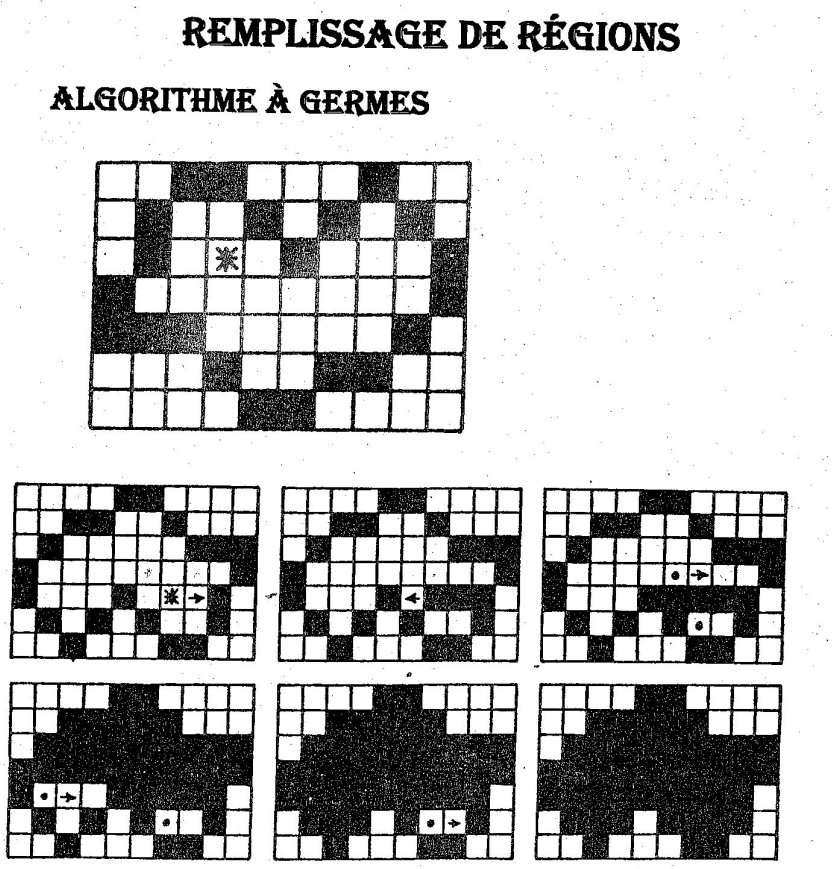
Algorithme de remplissage d'une région de connectivité 4

Connexité : Régions (resp. Frontières) de connexité 4 :

Deux pixels de la région peuvent être joints par une séquence de déplacement dans les 4 directions :



Exemple



RemplissageRégion_4connectivité(x, y, CC, CR)

CC : couleur du contour
CR : couleur de remplissage (couleur de la région)
x, y : coordonnée du pixel initial = germe appartient à la région

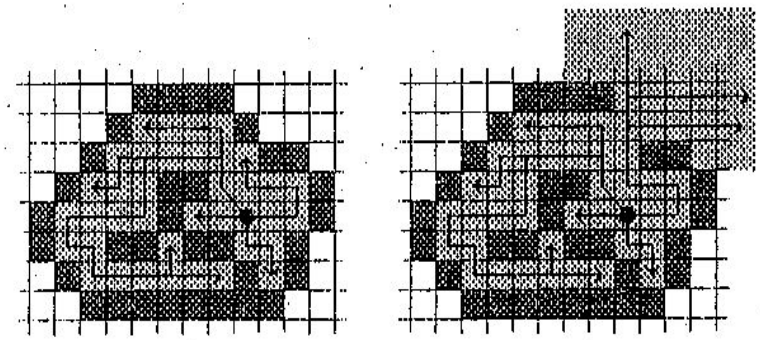
Début

```
/* Couleur du pixel avant remplissage */
CP ← CouleurPixel(x, y)
Si (CP != CC et CP != CR) alors
    AffichePixel(x, y, CR)
    RemplissageRégion_4connectivité(x,y+1, CC, CR)
    RemplissageRégion_4connectivité(x,y-1, CC, CR)
    RemplissageRégion_4connectivité(x+1,y, CC, CR)
    RemplissageRégion_4connectivité(x-1,y, CC, CR)
Fsi
```

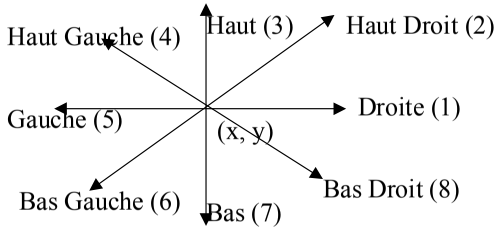
Fin

Remarques

- l'utilisation de la récursivité entraîne très vite des problèmes de dépassements de capacité de la pile si les zones remplies sont trop grandes
- débordement si le contour n'est pas fermé.



- Avec 8 connexités on effectue 8 appels récursifs :



X + 1, y _____ 1

X + 1, y + 1 _____ 2

X , y+1 _____ 3

X – 1, y + 1 _____ 4

X – 1, y _____ 5

X – 1, y - 1 _____ 6

X , y-1 _____ 7

X + 1, y-1 _____ 8

Remplissage par ligne

Rappel :

Algorithme hautement récursif → donc coûteux

Solution

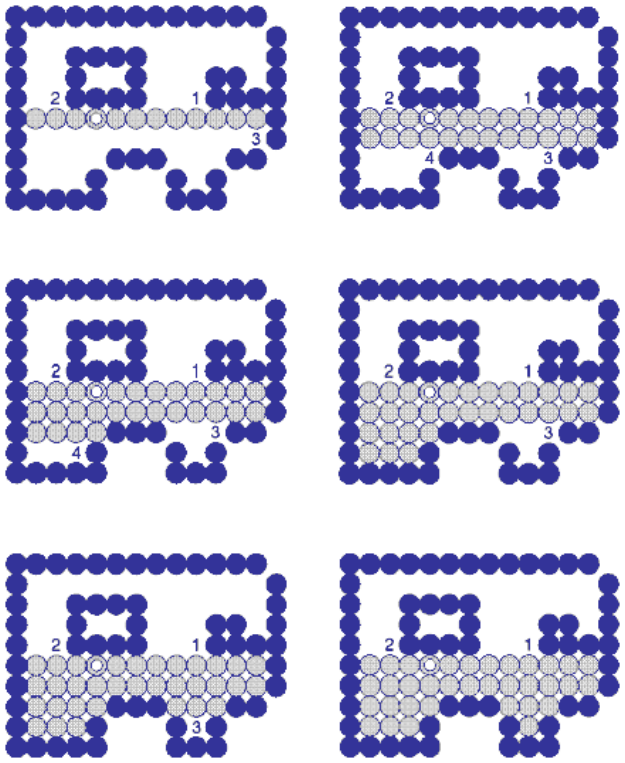
Remplissage par ligne

Principe

Le remplissage ligne par ligne à partir d'un pixel initial germe (x, y)

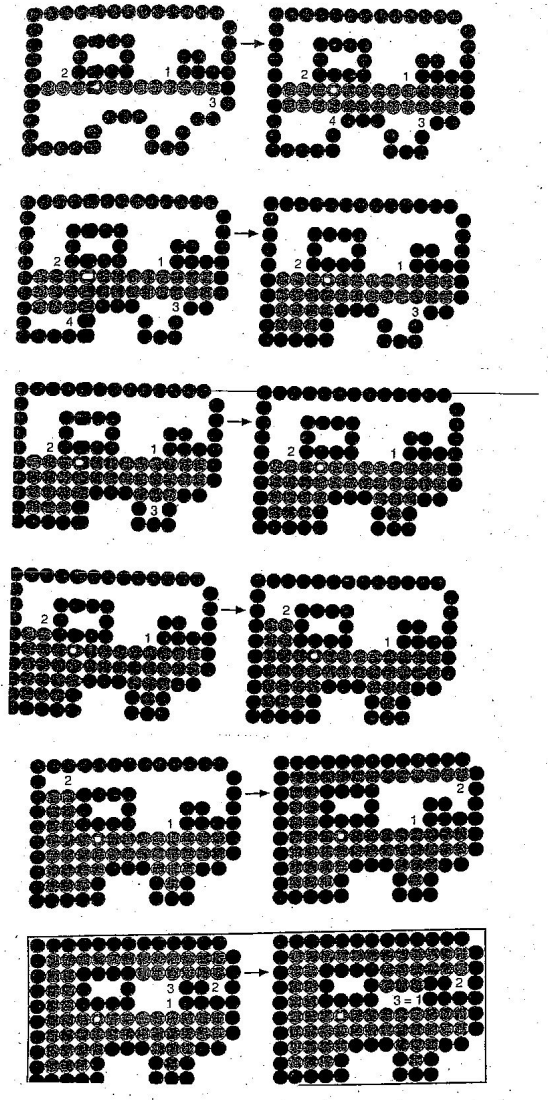
- calculer les pixels frontières pour la ligne d'ordonnée y
- a chaque itération :
  - ➢ on remplit tous les pixels Pi qui se trouvent entre les pixels frontières gauche et droite de la ligne de balayage du germe courant.
  - ➢ On recherche parmi les pixels au dessus et au dessous des Pi ceux qui sont le plus à droite d'une suite horizontale à remplir.
  - ➢ Ces pixels sont empilés comme germes des itérations suivantes.

Exemple



REMPLISSAGE DE RÉGIONS

Algorithme à germes par balayage de ligne



Remplissage_Ligne(x, y, CC, CR)

Paramètres formels

CC : couleur du contour

CR : couleur de remplissage (couleur de la région)

x, y : coordonnée du pixel initial = germe appartient à la région

Début

p ← init_pile()

```
//on empile le germe initial
```

empile((x, y), p)

tant que non vide_pile(p) faire

(x, y) ← sommet_pile(p)

p ← depiler(p)

```
//couleur du pixel avant le remplissage
```

cp ← couleurPixel(x, y)

```
// déterminer les abscisses extrêmes xg et
xd de la ligne de balayage du germe
courant (càd : la ligne y)
```

```
//Recherche de xd = Absci extrême à droite
xd ← x+1
cpd ← CouleurPixel(xd, y)
tantque cpd != CC faire
    xd ← xd +1
    cpd ← couleurPixel(xd, y)
fintq
xd ← xd -1
//Recherche de xg=Absci extrême à gauche
xg ← x-1
cpg ← CouleurPixel(xg, y)
tantque cpg != CC faire
    xg ← xg -1
    cpg ← couleurPixel(xg, y)
fintq
xg ← xg +1
//Remplir la ligne de balayage de xg à xd
avec la couleur de remplissage
afficheLigne(xg, y, xd, y, CR)
```



```
//Recherche de nouveaux germes sur la
ligne de balayage au-dessus : la recherche
s'effectue entre xg et xd
x ← xd
cp ← couleurPixel(x, y+1)
tantque ( x >= xg ) faire
    tantque ( (cp = CC) ou (cp=CR)) et (x>= xg) faire
        x ← x – 1
        cp ← couleurPixel(x, y+1)
    fintq
    si ( (x >= xg) et (cp !=CC) et (cp !=CR) alors
    //Empile le nouveau germe au dessus trouvé
        empile( (x, y+1), p)
    finsi
    tantque (cp !=CC) et (x>=xg) faire
        x ← x – 1
        cp ← couleurPixel(x, y+1)
    fintq
fintq
```

```
//Recherche de nouveaux germes sur la ligne de balayage au-dessous : la recherche s'effectue entre xg et xd
x ← xd
cp ← couleurPixel(c, y-1)
tantque ( x >= xg ) faire
    tantque ( (cp = CC) ou (cp=CR)) et (x>= xg) faire
        x ← x – 1
        cp ← couleurPixel(x, y-1)
    fintq
    si ( (x >= xg) et (cp !=CC) et (cp !=CR) alors
        /*Empiler le nouveau germe au dessous trouvé */
        empile( (x, y-1),  p)
    finsi
    tantque (cp !=CC) et (x>=xg) faire
        x ← x – 1
        cp ← couleurPixel(x, y-1)
    fintq
  fintq
fintq
Fin
```


Remarque

La taille de la pile est faible → ce qui est très économique par rapport à l'algorithme précédent.