Informatique Graphique
Remplir/Peindre

Jean-Baptiste.Yunes@univ-paris-diderot.fr
9/2018

- Comment remplir une forme ?
    - bien entendu de façon efficace…
- Premier problème comment est définie la forme ?
    - Selon les cas les algorithmes peuvent être très différents
        - Forme définie dans le continu ? dans le discret ?

- Une forme définie dans le discret par son contour ? par son intérieur ?

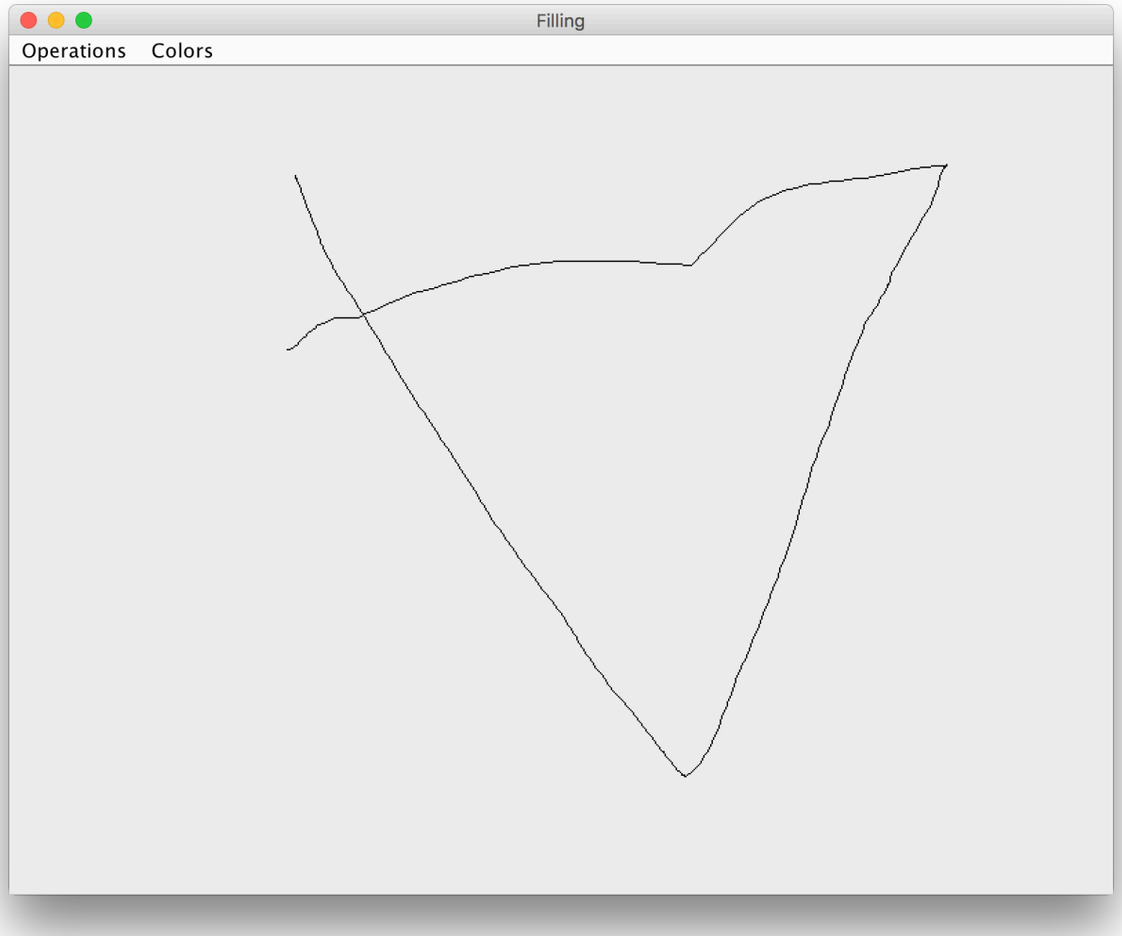
- Quelle est sa connectivité? 4-connexe ? 8-connexe ?

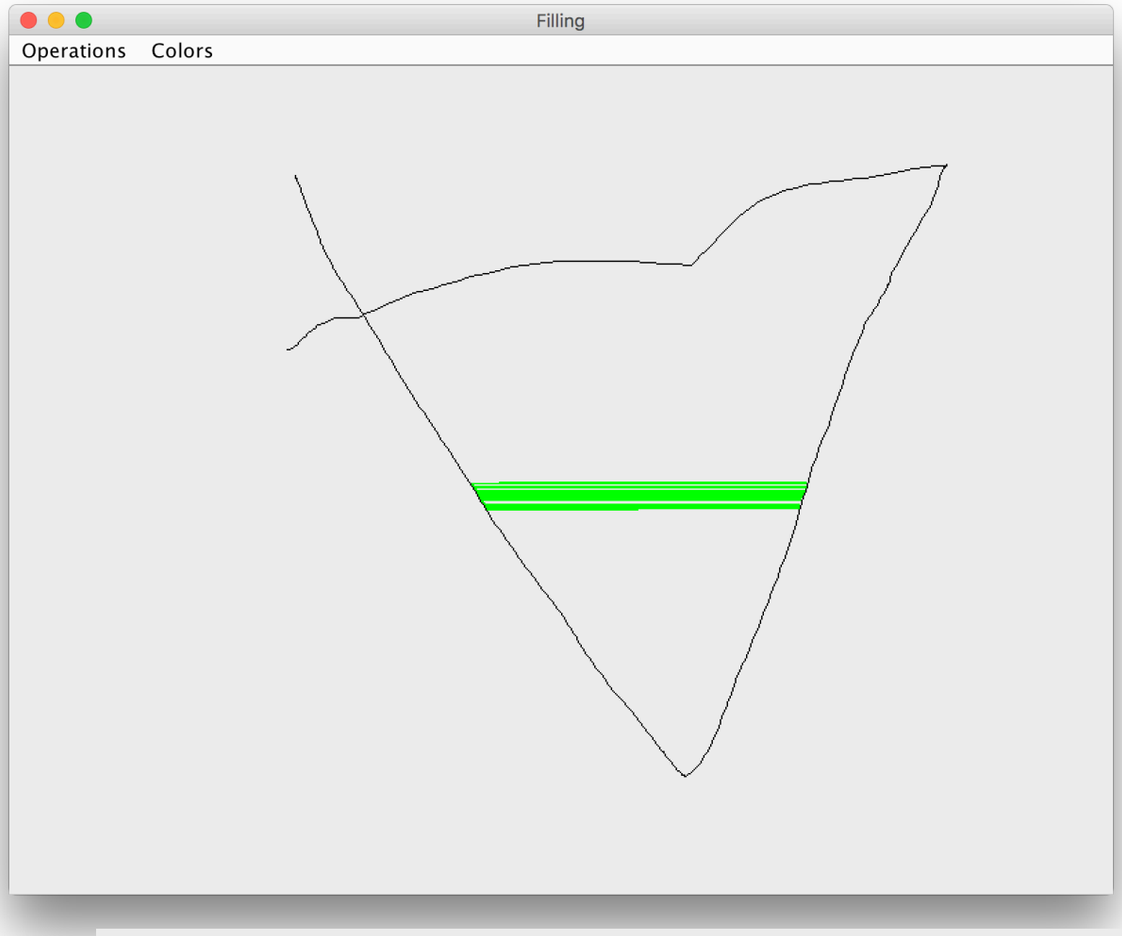
- Proposition : l'algorithme prend en entrée un point de l'intérieur

  - Sinon il faut en trouver un, c'est un autre problème…

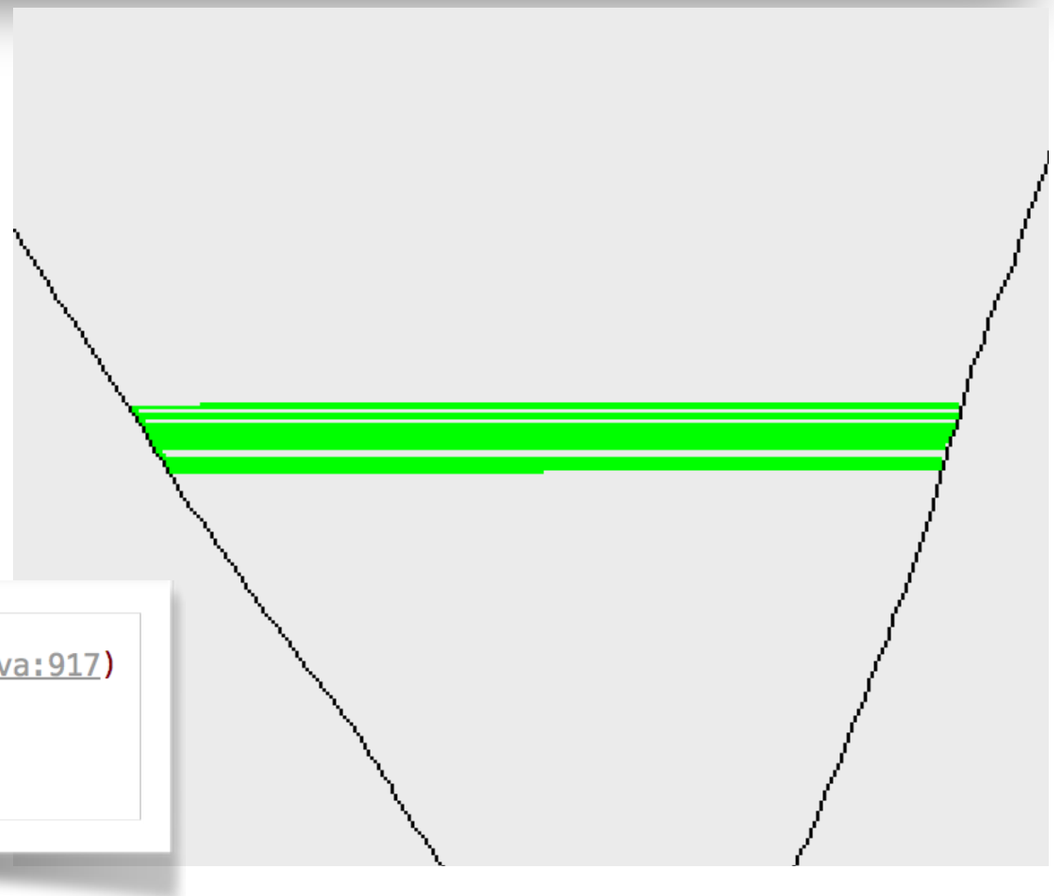
- Algorithme naïf (version «intérieur»):
  remplirNaivement(x,y,couleur) {
    if point(x,y) n'est pas à l'intérieur ou déjà peint return;
    setPixel(x,y,couleur);
    remplirNaivement(x+1,y,couleur);
    remplirNaivement(x-1,y,couleur);
    remplirNaivement(x,y+1,couleur);
    remplirNaivement(x,y-1,couleur);
  }

- Assez rapidement : explosion de la pile d'exécution! Pourquoi ?





- Pourquoi cet effet ?

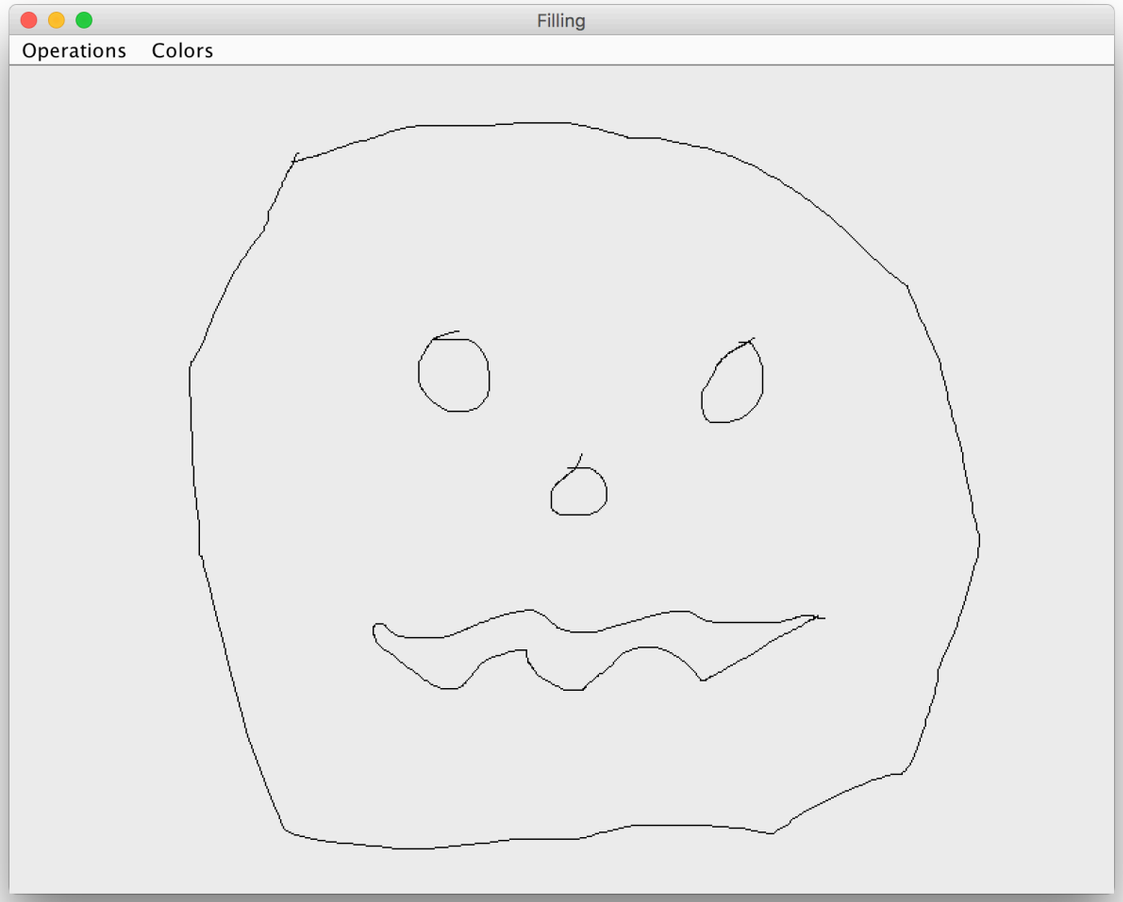


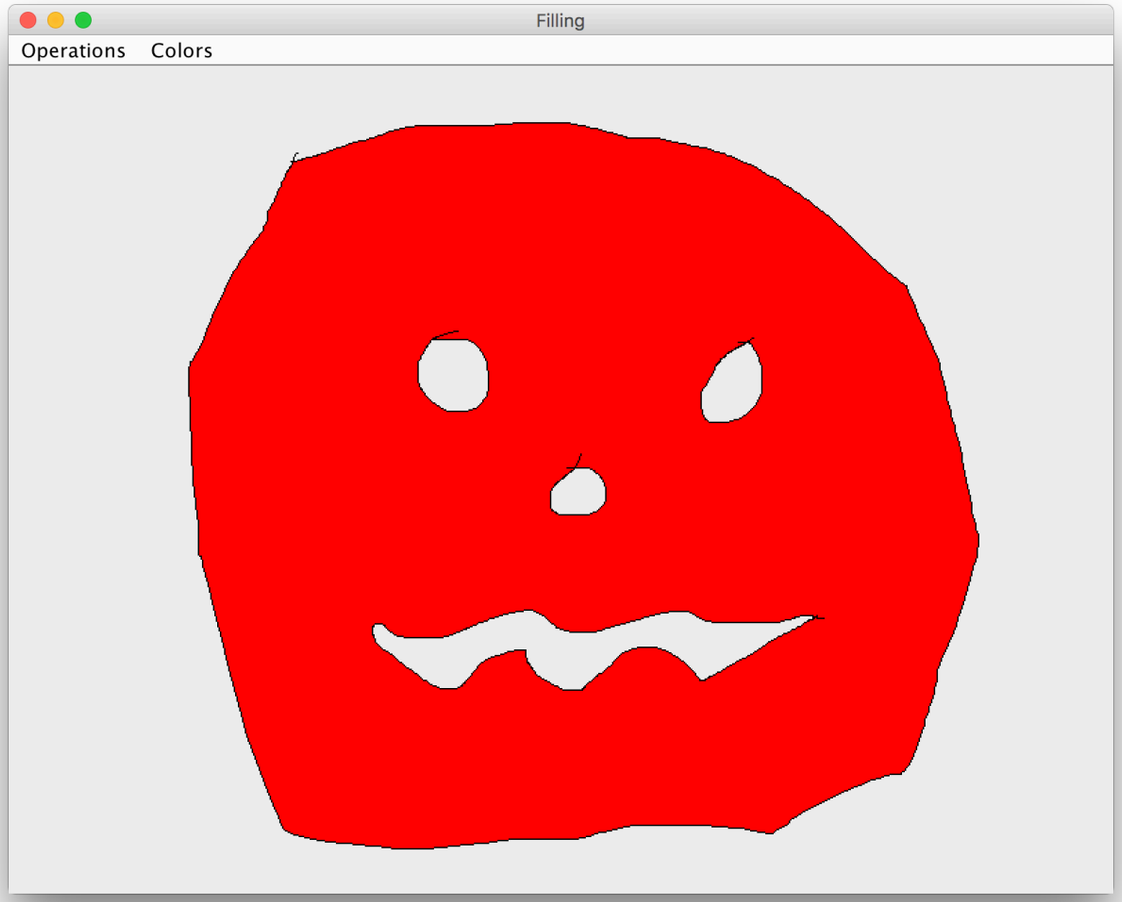
```
Exception in thread "AWT-EventQueue-0" java.lang.StackOverflowError
    at java.desktop/java.awt.image.BufferedImage.getRGB(BufferedImage.java:917)
    at DrawingPanel.getPixel(DrawingPanel.java:101)
    at Algorithms.naiveRecursiveFilling(Algorithms.java:51)
    at Algorithms.naiveRecursiveFilling(Algorithms.java:53)
    at Algorithms.naiveRecursiveFilling(Algorithms.java:53)
```

- Utilisation d'une pile explicite :

```
remplirAvecPile(x,y,couleur) {
  stack s;
  s.push(p(x,y))
  while (s.notEmpty()) {
    p = s.pop();
    if point p sur contour ou peint continue;
    setPixel(p,couleur);
    s.push(point(x+1,y));
    s.push(point(x-1,y));
    s.push(point(x,y+1));
    s.push(point(x,y-1));
  }
}
```

- Pas la même mémoire pour la pile…






```
/Library/Java/JavaVirtualMachines/jdk
Naive recursive with explicit stack
Max stack size is 327340
```

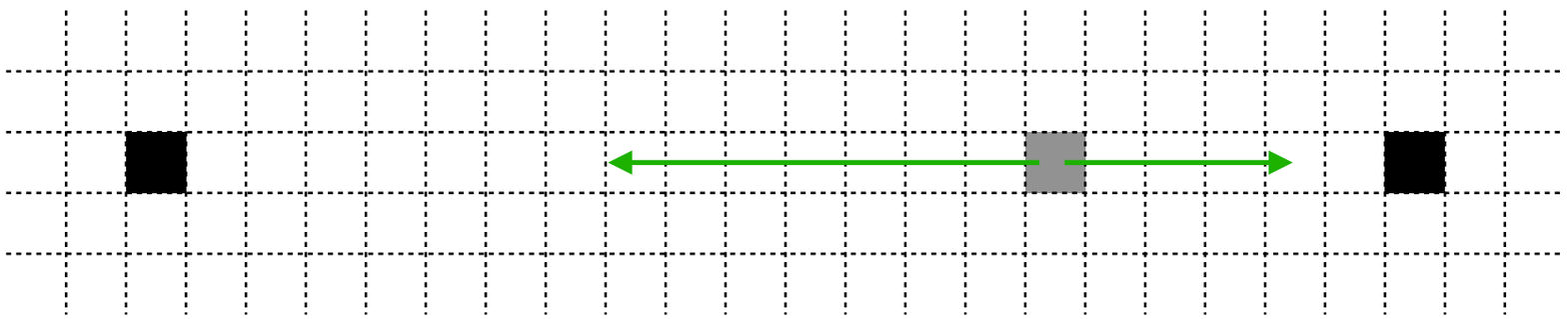
- Grosse pile!

- Des optimisations possibles ?
    - Oui si on peint vers la gauche il est inutile de revenir à droite…
        - Sur une ligne on balaye à gauche et à droite en empilant uniquement les points nécessaires au dessus et en dessous
- Gain pas très flagrant

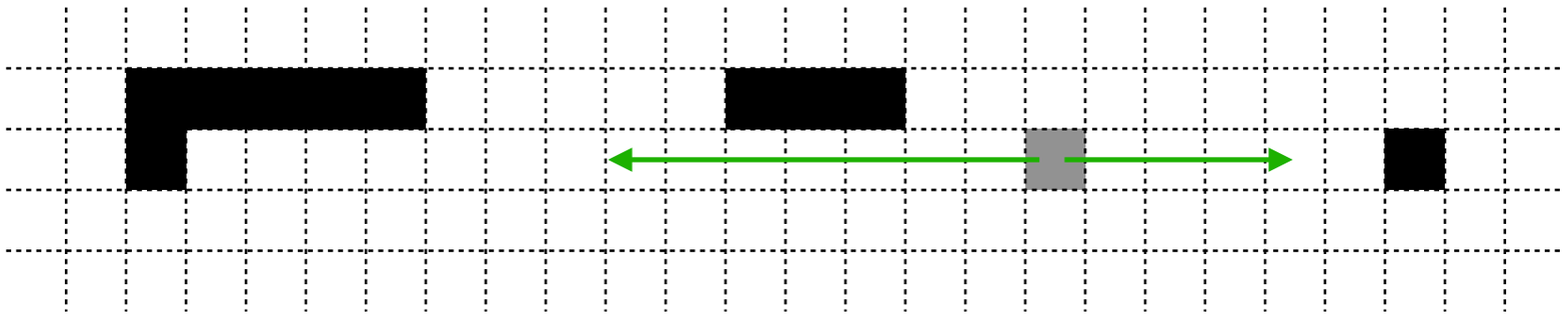
```
/Library/Java/JavaVirtualMachines/jdk-9.0.1.jdk
Naive recursive with explicit stack
Max stack size is 301611
Naive recursive with explicit stack optimized
Max stack size is 161633
```

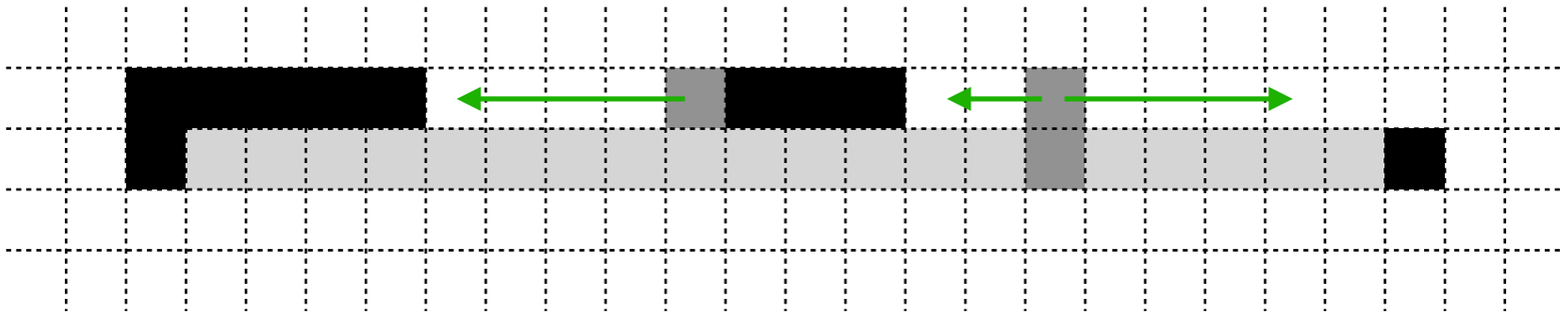
- Pousser l’idée plus loin…
    - Ne pas systématiquement empiler tous les points en dessus et en dessous
        - On balaye déjà de façon continue, on peut donc être intelligent dans ce qu’il y a à empiler
- Algorithme de Smith

- Idée de base :
  - À partir d’un point appartenant à un segment à colorier
    - Colorier à gauche et à droite par itération



- Repérer sur la ligne du dessus (resp. dessous) les segments…





```
Smith optimized
Max stack size is 178
```

- Références

  - Tint Fill
    Alvy Ray Smith
    1979, ACM SIGGRAPH Conference proceedings