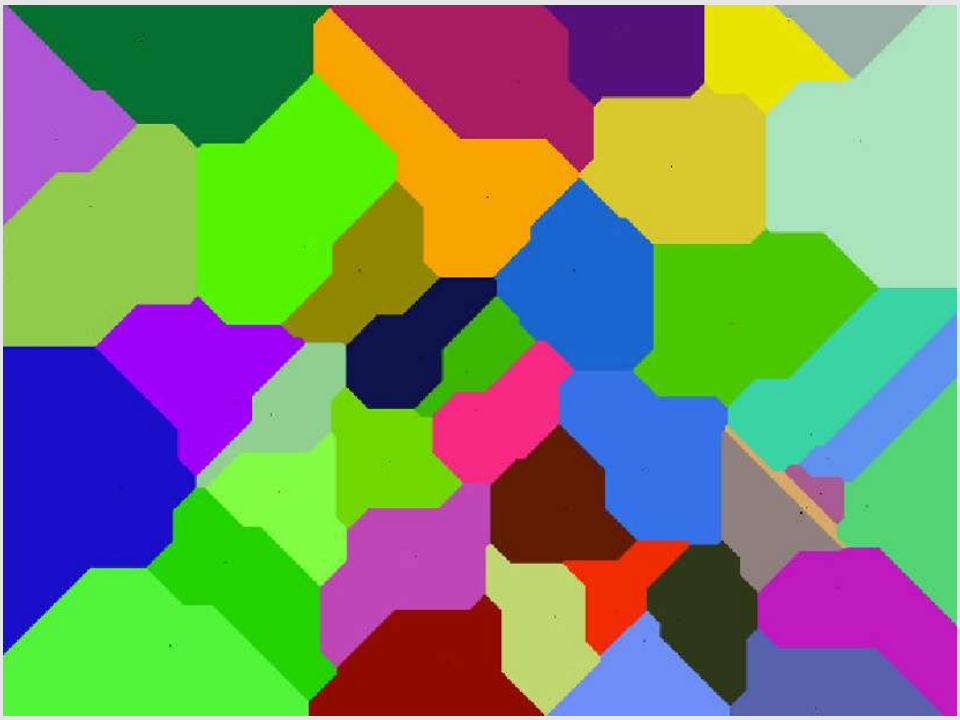
IN 103 - TD 02

Énoncé

ENSTA ParisTech

On souhaite simuler l'extension d'affreuses créatures rampantes s'étalant progressivement autour de leur point d'apparition...

1. On choisit n points initiaux P (par exemple au hasard) auxquels on attribue une couleur.
2. On attribue aux autres points de l'écran la nouvelle couleur telle que chacun soit de la même couleur que la colonie qui l'a conquis.



Q1 Quelle structure de données sous-jacente peut être utile ? Pourquoi (vague idée de comment l'algorithme va fonctionner) ?

Les points initiaux sont décrits par leurs coordonnées et la couleur de leurs colonies respectives.
Cette couleur est définis par trois char, 1 nombre par couleur primaire (r,g,b).
Une autre structure de donnée est utilisée pour décrire une colonie: capacité maximale, nombre de points dedans et leurs positions.

Q2 Quelle est la forme des données à mémoriser dans cette structure ?

Q3 Esquissez l'algorithme pour effectuer cette décomposition.

Dans le fichier `queue.(c|h)` vous sont données les fonctions de manipulation de la structure de donnée discuté précédemment. La suite du TD vise à implémenter l'algorithme de coloriage.

Q4 Quelle est la taille maximale de la structure de mémorisation à créer ?

Q5 Dans quel cas cette structure peut-elle être saturée (par rapport à la taille maximale trouvée dans la question **Q4** ?

Q6 Comment pouvez-vous mémoriser si un point a déjà été colorié ? Comment allez-vous accéder à cette information ?

La couleur d'un point est déterminée par la « quantité » de rouge, de vert et de bleu qu'elle comporte. Ce sont donc 3 entiers qui sont compris dans $[0; 255]$. Pour choisir une couleur au hasard, il faut donc choisir 3 entiers entre 0 et 255.

Vous avez à disposition la fonction :

```
int rand (void)
```

qui renvoie un entier pseudo aléatoire entre 0 et une constante prédéfinie (très grande) inclus.

Cette fonction vous servira également pour choisir aléatoirement les coordonnées des points de départ.

Q7 Comment restreindre la valeur retournée par `rand ()` à $[0; 255]$?

Pour vous aider dans la suite, vous disposez, dans le fichier `gfxprims.(c|h)` d'une fonction d'affichage d'un point selon ses coordonnées x et y et sa couleur via ses composantes rouge / vert / bleu :

void renderPixel (**int** x, **int** y, Uint8 R, Uint8 G, Uint8 B)

Q8 Écrivez une fonction `fill ()` qui implémente l'algorithme de coloriage. Cette fonction devra prendre en arguments la taille de l'écran (largeur, hauteur) et le nombre de points initiaux à partir desquels calculer le diagramme.

Le fichier `test_fill.c` vous fournit le substrat supplémentaire vous permettant de tester votre calcul de coloriage, en considérant que la fonction de calcul du diagramme a le prototype suivant :

void fill (**int** width, **int** height, **int** nb_init)

où `width` est la largeur de l'écran, `height` sa hauteur et `nb_init` est le nombre de points à choisir au hasard pour déterminer les centres de propagation des couleurs.

Pour compiler votre programme, vous aurez besoin de certaines options de compilation qui suivent :

```
gcc -Wall ‘sdl-config --cflags --libs‘ gfxprims.c test_fill.c VOS_FICHIERS
```