info224 : algorithmique et programmation
TD / TP à la maison

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Chaque étudiant doit envoyer son travail à son enseignant de TD :
- Pierre Hyvernat `Pierre.Hyvernat@univ-savoie.fr`,
- Thomas Seiller `Thomas.Seiller@univ-savoie.fr`,
- Tom Hirschowitz `Tom.Hirschowitz@univ-savoie.fr`.

Vous pouvez récupérer un fichier `rectangles.py` avec un fonction d'affichage graphique sur ma page web. Vous devrez fournir un fichier Python valide. N'oubliez pas de préciser vos nom, prénom et filière en haut de votre fichier.

Attention : si votre fichier n'est pas du Python valide, votre correcteur se réserve le droit d'enlever 5 points sur votre note !

Vous pouvez travailler en petits groupes. Si c'est le cas, n'oubliez pas de le préciser dans vos fichiers.

Partie 1 : fonctions simples

Nous allons manipuler des rectangles dans le plan. Ces rectangles ont leurs côtés verticaux et horizontaux.

Chaque rectangle est représenté par un dictionnaire contenant les coordonnées de ces coins "en haut à gauche" et "en bas à droite" :

```
r1 = { 'x1' : 10 , 'y1' : 10  ,  'x2' : 25 , 'y2' : 33 }
```

Question 1. On considère que l'origine se trouve en haut à gauche, avec les coordonnées qui augmentent vers la droite et en bas. Écrivez une fonction `valide(rect)` qui renvoie vrai ou faux suivant que le rectangle passé en argument est correct ou pas.

Question 2. Écrivez une fonction `surface(rect)` qui renvoie la surface (en pixels) du rectangle passé en argument.

Question 3. Écrivez une fonction `dans(rect,x,y)` qui renvoie vrai ou faux suivant que le point de coordonnées `x` et `y` est dans le rectangle `rect` ou pas.

Question 4. (Bonus) Écrivez la fonction `intersection(rect1, rect2)` qui calcule l'intersection de deux rectangles. Le résultat de cette fonction sera :
- la valeur `None` si les rectangles ne s'intersectent pas,
- le rectangle intersection si les rectangles s'intersectent.

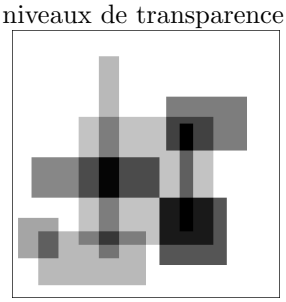
Attention, il y a plusieurs cas. Je vous conseille de commencer par regarder ce qui se passe si on calcule l'intersection de deux segments.

Partie 2 : transparence

On ajoute maintenant une *transparence* à chaque rectangle : il s'agit d'un entier entre 255 (opaque) et 0 (complètement transparent).

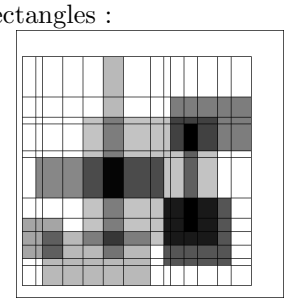
```
r1 = { 'x1':10 , 'y1':10  ,  'x2':25 , 'y2':33  , 'transparence':127}
```

Le but est de trouver les différents niveaux de transparence pour une liste de rectangles :



Ceci est important car certaines interfaces graphiques ne peuvent pas gérer la transparence.

Pour découvrir les niveaux de transparence sur une superposition de rectangles, nous allons découper toute l'image en petits rectangles :



Ensuite, pour chacun des petits rectangles, il suffit d'additionner les transparences des rectangles initiaux qui le contiennent.

Question 1. Écrivez la fonction `gere_transparence(Rs)`. Cette fonction prend en argument une liste de rectangles, avec des transparences et renvoie une liste de petits rectangles qui pavent l'image.

Par exemple, sur la liste

```
[ {'x1':100  , 'x2':200  , 'y1':100  , 'y2':200  , 'transparence':70 },
  {'x1':150  , 'x2':300  , 'y1':140  , 'y2':170  , 'transparence':120 } ]
```

votre fonction devra renvoyer

```
[ {'x1': 100, 'x2': 150, 'y1': 100, 'y2': 140', transparence': 70 },
  {'x1': 100, 'x2': 150, 'y1': 140, 'y2': 170', transparence': 70 },
  {'x1': 100, 'x2': 150, 'y1': 170, 'y2': 200', transparence': 70 },
  {'x1': 150, 'x2': 200, 'y1': 100, 'y2': 140', transparence': 70 },
  {'x1': 150, 'x2': 200, 'y1': 140, 'y2': 170', transparence': 190 },
  {'x1': 150, 'x2': 200, 'y1': 170, 'y2': 200', transparence': 70 },
  {'x1': 200, 'x2': 300, 'y1': 140, 'y2': 170', transparence': 120 } ]
```

(Faites un dessin pour voir d'où viennent ces rectangles...)

Pour cela, vous aurez besoin de :

- récupérer toutes les ordonnées des rectangles de `Rs`,
- les trier par ordre croissant,
- récupérer toutes les abscisses des rectangles de `Rs`,
- les trier par ordre croissant,
- parcourir ces deux listes pour parcourir les petits rectangles,
- pour chacun de ces petits rectangles, parcourir la liste `Rs` pour connaitre la transparence globale du petit rectangle, (la fonction `dedans` vous sera bien utile),
- renvoyer la liste de tous ces petits rectangles.

Votre fonction devra donc comporter trois boucles imbriquées : pour les abscisses, les ordonnées et les rectangles initiaux !