LMI 1 - Semestre 2 Mars-Avril-Mai 2018

Université d'Artois
Faculté des Sciences Jean Perrin

ALGO2 – Algorithmique et Programmation 2

Projet : Same

Étape 2 - On supprime un groupe de billes

L'objectif ici est, en cliquant sur une bille, de supprimer toutes les billes de même couleur qui forment un groupe avec elle. Dans la suite, on appellera *composante* toutes les billes d'une même couleur qui se touchent horizontalement ou verticalement.

Les composantes

L'idée principale est qu'on peut numéroter tous les groupes de billes. Ainsi, pour la situation suivante (l'illustration se fait sur un "petit format" pour plus de clarté) :

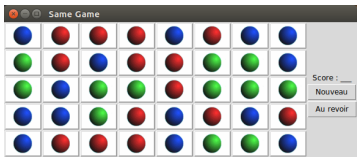

FIGURE 1 – Le jeu Same

On pourrait numéroter les composantes ainsi (en gras les numéros de lignes et de colonnes) :

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	1	2	2	2	3	4	5	5
1	6	2	7	2	2	8	8	5
2	6	9	10	10	2	8	8	8
3	9	9	10	11	12	13	14	15
4	9	11	11	11	12	16	16	17

L'objectif de cette étape est donc de calculer la numérotation des composantes.

La classe `Case`

La première chose à faire est de modifier de la classe `Case` pour prendre en compte dans chaque case du numéro de composante. Ce numéro va être un attribut supplémentaire de la classe `Case`. Certains numéros de composantes ont une signification particulière : -1 signifie que pour cette case, le numéro de la composante à laquelle elle appartient n'a pas encore été calculé. C'est le cas par exemple lorsque la case est créée, ou lorsqu'elle change de couleur ; 0 signifie que la case est vide (c'est le cas lorsqu'on la supprime).

1. Modifiez le constructeur de la classe `Case` pour ajouter un attribut `self.__compo` qui sera initialisé comme indiqué ci-dessus ;
2. Mettez à jour les méthodes `change_couleur` et `supprime` selon les indications ci-dessus ;

3. Ajoutez les méthodes suivantes :
   (a) une méthode `composante` qui retourne le numéro de la composante ;
   (b) une méthode `pose_composante` qui prend en paramètre un entier et l'affecte comme numéro de composante pour le case ;
   (c) une méthode `supprime_compo` qui désaffecte un numéro de composante à la case (en le remettant à -1). Toutefois, une case vide aura son numéro de composante affecté à 0 :
   (d) une méthode `parcourue` qui teste si la case a été affectée à un numéro de composante.

La classe `ModeleSame`

L'idée est la suivante : à l'initialisation du jeu, on calcule les composantes (on numérote chaque composante). Lorsque l'utilisateur clique sur une case, si le nombre d'éléments de la composante est au moins égal à 2, alors les billes de la composantes sont supprimées. Puis les composantes sont recalculées, et on peut recommencer.

Un nouvel attribut

On va conserver dans une liste le nombre d'éléments de chaque composante. L'indice de la composante représentera le numéro de la composante et la valeur de la liste à cet indice donnera le nombre d'éléments de cette composante. La composante numéro 0 étant associée aux cases vides, on "trichera" en notant systématiquement que la composante 0 contient 0 éléments, pour ne pas essayer de supprimer une bille déjà supprimée.

Dans l'exemple ci-dessus, la valeur de la liste devrait être (en haut les indices de la liste) :

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	1	7	1	1	3	2	1	5	4	3	4	2	1	1	1	2	1

1. Ajoutez un attribut `self.__nb_elts_compo` qui est une liste initialisée à `[]` dans le constructeur de `ModeleSame` ;
2. Ajoutez une méthode `composante()` qui prend deux paramètres `i` et `j` et qui renvoie la composante de la bille en `(i,j)`.

Le calcul des composantes

Ce calcul nécessite deux méthodes. La méthode `calcule_composante_numero(self, i, j, num_compo, couleur)` attribue le numéro `num_compo` aux cases qu'elle peut toucher depuis la case en `(i,j)` et qui sont de couleur `couleur`. Cette méthode retourne le nombre de cases de la composante `num_compo`. La méthode `calcule_composantes(self)` lance le calcul des composantes sur toutes les cases de la matrice.

1. On commence par la méthode `calcule_composantes(self)`. Elle initialise `self.__nb_elts_compo` avec un 0 pour traiter le cas particulier des cases vides (cf. explication ci-dessus). Elle initialise le numéro de composante courant `num_compo` à 1. Le principe général est le suivant : on parcourt toutes les cases de la matrice. Pour chaque case aux coordonnées `(i,j)` dont la composante est encore -1, on récupère sa couleur, on lance le calcul de la composante `num_compo` à partir de `(i,j)`. Le nombre d'éléments retournés est ajouté au bout de la liste `self.__nb_elts_compo`. Puis on peut passer au numéro de composante suivant.
2. La méthode `calcule_composante_numero(self,i,j,num_compo, couleur)` est une méthode récursive. Elle s'arrête lorsqu'en `(i,j)` se trouve une case qui a déjà été parcourue ou dont la couleur est différente du paramètre `couleur`. Dans ce cas, le nombre d'éléments

retourné pour la composante est `0`. Sinon, elle pose la valeur `num_compo` comme composante sur la case en `(i, j)` et lance un appel récursif vers les cases adjacentes, dans la mesure où elle ne sort pas des limites de la matrice. Le nombre d'éléments de la composante est alors la somme des valeurs retournées + 1 pour la case `(i, j)` elle-même.

Lancer le calcul des composantes

Le premier appel au calcul des composantes doit se faire dans le constructeur de `ModeleSame`.

Les appels suivants correspondront à un nouveau calcul : il sera alors nécessaire dans une premier temps de supprimer la composante attribuée à chaque case. Écrivez la méthode `recalc_composantes(self)` qui fait ce travail puis appelle la méthode `calcul_composantes()`.

La suppression d'une composante

Dans la classe `ModeleSame`, vous devez maintenant écrire une méthode `supprime_composante(self, num_compo)` qui supprime toutes les billes de la matrice qui sont dans la composante numéro `num_compo`. Elle ne fera cette action que si le nombre de billes de la composante est au moins égal à 2, ce qui permettra de ne pas prendre en compte un clic sur une case vide ou sur une bille isolée. Cette méthode devra mettre à jour le score : pour chaque composante supprimée, si n est le nombre de billes de la composante, on augmente le score de $(n-2)^2$. Si des billes ont été supprimées, c'est à ce moment-là qu'il faudra relancer un calcul de composantes.

La méthode retourne un booléen qui indique si oui ou non une suppression a eu lieu.

La classe `VueSame`

Il ne reste plus qu'à changer l'action du contrôleur : plutôt que de demander au modèle de supprimer la bille sur laquelle le joueur a cliqué, il lui demandera de supprimer la composante de cette bille.

La méthode `redessine` pourra aussi mettre à jour le score.

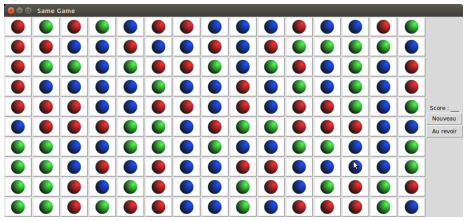

FIGURE 2 – Le joueur cible une composante

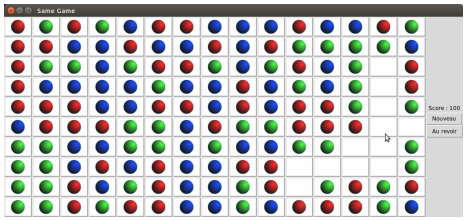

FIGURE 3 – La composante est supprimée

Fin de l'étape 2

Testez votre application. Bravo, c'est fini pour cette étape !

Il vous ne reste plus qu'à préparer l'étape suivante : copiez vos fichiers `modele.py` et `vue.py` dans le répertoire `etape3`. Déposez dans un fichier au format `.zip` (ou `.tgz`, mais attention, les `.rar` sont exclus et ne seront pas corrigés) votre étape 2. Maintenant, vous êtes prêt.e.s pour la suite !