POLYTECH MARSEILLE Aix*Marseille Université **PEIP2**

Ingénierie Informatique

TP n°6

Étape 30 – Fonction supprimeBonbons

Cette fonction prend en paramètre les tableaux `tab_jeu` et `tab_match` vus dans les étapes précédentes. Pour chaque cellule `tab_match[col,lig]` ayant une valeur 0, elle supprime le bonbon correspondant dans `tab_jeu` en lui affectant la valeur -1 dans `tab_jeu`.

Réécrivez la fonction `supprimeBonbons`. Pour la tester, il suffit de voir si les match3 suppriment réellement des bonbons.

Descente des bonbons

Idées pour faire descendre les bonbons :

Afin de réaliser l'animation des bonbons qui descendent par gravité parce que certains bonbons situés en dessous ont été détruits, l'idée consiste à déterminer, pour chaque colonne, la ligne `lig` du bonbon le plus bas qui a été supprimé. C'est l'objectif de la fonction `calculeLigneDescente`, qui va stocker cette information pour l'ensemble des colonnes dans un tableau 1D `tab_ligne_descente`. Ensuite, pour chaque colonne, on va afficher tous les bonbons toujours existant dans l'espace de jeu et qui sont situés au dessus de la ligne `lig` en les déplaçant d'un nombre δ de pixels vers le bas, et on va afficher normalement, c'est-à-dire sans déplacement de δ pixels, les bonbons existants situés en dessous de la ligne `lig`. C'est l'objectif de la fonction `afficheDescente`. En faisant varier δ de 0 à `TAILLE_BONBON`, on aura l'impression que les bonbons situés au dessus `lig` descendent progressivement d'une case vers le bas. Quand ce déplacement a eu lieu, on met à jour les deux tableaux `tab_jeu` `tab_match` afin de prendre en compte cette descente d'une case vers le bas des bonbons. C'est l'objectif de la fonction `descendTableau`. Enfin, en haut des colonnes qui ont été descendues d'une case, il faut rajouter de nouveaux bonbons. C'est l'objectif de la fonction `ajouteNouveauxBonbons` du prochain TP.

Le programme principal réitère ces opérations tant qu'il reste des bonbons supprimés dans l'espace de jeu. Toutes ces opérations donnent à l'utilisateur l'illusion que toutes les cases vides de l'espace de jeu sont progressivement comblées par les bonbons situés au dessus.

Étape 31 – Fonction calculeLigneDescente

La fonction `calculeLigneDescente` prend en paramètre le tableau `tab_match` créé dans les étapes précédentes ainsi qu'un tableau 1D appelé `tab_ligne_descente` et dont la taille est égale au nombre de colonnes de bonbons. Le but de cette fonction est, pour toutes les colonnes `col`, de placer dans `tab_ligne_descente[col]` l'indice `lig` le plus grand tel que `tab_match[col,lig] = 0` (autrement dit, il s'agit du bonbon le plus bas qui a été supprimé). Si aucun bonbon n'a été supprimé dans la colonne `col`, alors on affecte la valeur `-1` à `tab_ligne_descente[col]`. Enfin, la fonction renvoie un booléen indiquant si au moins une des colonnes a un bonbon supprimé (autrement dit, si ce booléen est `True`, cela signifie qu'on aura des descentes de bonbons, et s'il vaut `False`, aucune descente n'aura lieu dans l'espace de jeu).

Réécrivez la fonction `calculeLigneDescente` et testez la en réalisant des match3 **verticaux**. Vous devez voir les bonbons au dessus de ceux impliqués dans le match3 descendre pour combler les vides résultant de la suppression des bonbons du match3.

Étape 32 – Fonction afficheDescente

Cette fonction prend en paramètre le `painter` qui va réaliser les affichages, le tableau « modèle » `tab_jeu`, un tableau contenant toutes les images des bonbons, le tableau `tab_ligne_descente` calculé dans l'étape précédente et, enfin, un entier `deplacement`. Elle réalise l'affichage de la bordure du jeu, puis celui des bonbons de la manière suivante : pour chaque colonne `col`,

- les bonbons situés au dessus de la ligne `tab_ligne_descente[col]`, c'est-à-dire ceux dont l'ordonnée `lig` est inférieure à `tab_ligne_descente[col]`, sont affichés avec un décalage de `deplacement` pixels vers le bas ;
- les bonbons situés en dessous de `tab_ligne_descente[col]` sont affichés comme d'habitude (c'est-à-dire sans déplacement).

Attention : on rappelle que, dans `tab_jeu`, une cellule de valeur `-1` représente un bonbon supprimé, il ne faut donc pas l'afficher. Réécrivez la fonction `afficheDescente` et testez la en réalisant des match3 verticaux. Vous devez voir les bonbons au dessus du match3 descendre pour combler les vides.

Aide sur l'écriture de la fonction :

Afin d'écrire rapidement la fonction `afficheDescente`, je vous suggère de copier-coller le code de la fonction `afficheTableauJeu`, du TP n°3, qui réalise les mêmes affichages mais sans déplacement et sans case vide (bonbon supprimé), puis d'adapter ce code.

Étape 33 – Fonction descendTableau

La fonction `descendTableau` prend en paramètres un tableau 2D `tab` de `NB_COLONNES_JEU` colonnes et `NB_LIGNES_JEU` lignes ainsi que le tableau `tab_ligne_descente` des étapes précédentes. Pour chaque colonne `col`, elle fait descendre le contenu de toutes les cellules de la colonne situées « au dessus » de `tab_ligne_descente[col]` d'une case vers le bas. De plus, s'il y a eu descente dans la colonne `col` (autrement dit, si `tab_ligne_descente[col]` > 0), elle place -1 dans `tab[col,0]` afin de signifier que cette case est maintenant vide.

Utilité de la fonction :

Après avoir calculé le tableau `tab_ligne_descente` et exécuté la fonction `afficheDescente`, le joueur a vu des bonbons tomber progressivement d'une case vers le bas. Il faut maintenant rendre le tableau « modèle » `tab_jeu` cohérent avec ce qu'a vu le joueur. C'est précisément ce que l'on fait en exécutant la fonction `descendTableau` en lui passant `tab_jeu` en premier paramètre. De la même manière, il faut rendre le tableau `tab_match` cohérent avec `tab_jeu` et, pour cela, on appelle la fonction `descendTableau` en lui passant `tab_match` en premier paramètre.