Manuel Programmeur

Projet : EF48_taquin_drag_and_drop

Prénom & Nom : Ziad BOUKBIR

Site : https://hugoreveneau2.alwaysdata.net/jeu/

Introduction :

Ce document sert à distinguer et à expliquer la partie du projet codé par Ziad Boukbir vu qu'avec Ilona Baran nous rendons le même site.

Code :

1. **Récupération et affichage des images du jeu sélectionné par le joueur (en collaboration avec Ilona Baran)**

```
fetch("php/jeu.php?nJeu="+String(jeu))
.then(r => r.json())
.then(r => {
  myBD=r;
  let index = 0;
  for (let i = 0; i < 4; i++) {
    for (let k = 0; k < 4; k++) {
      var bulleImage = document.createElement('div');
      var img = document.createElement("img");
      bulleImage.setAttribute("class", "item");
      bulleImage.setAttribute("id", "item_" + i + "_" + k);
      img.setAttribute("draggable", "true");
      if (i != 3 || k != 3) {
        img.src = myBD[0]["img_"+NumToPos.get(liste[index])];
        tabImage.set(bulleImage, liste[index]);
      }
      else {
        img.src = "images/CarreBlanc.jpg";
        tabImage.set(bulleImage, "special");
      }
      bulleImage.appendChild(img);
      fondMap.appendChild(bulleImage);
      index++;
    }
  }
```

tabImage : dictionnaire (clé-valeur) qui associe à chaque case le numéro de tuile qu'elle contient

On crée 16 div avec un id selon sa position dans la grille (item_0_0, item_0_1, …).

On attribue les 15 images aléatoirement sur les 15 premières div à l'aide d'une liste de tirage aléatoire. La dernière div sera la case vide, on lui attribue une image d'un carré blanc pour un meilleur rendu visuelle.

Pour chaque image ajoutée, on sauvegarde son numéro avec sa div associée dans tabImage. La case vide aura comme valeur ''special'' sur tabImage.

Ensuite, on ajoute les div à leur conteneur pour les afficher sur le site.

2. **Ajout des écouteurs de type Dragend sur chaque div de la grille des cases**

```
items = document.querySelectorAll("#conteneur > div.item");
checkScore(tabImage, items);
nombre_de_coup = 0;

items.forEach(item => {
    item.addEventListener('dragend',() => {
        nombre_de_coup += 1;
        nbCoups.innerText = "Vous avez réalisé " + nombre_de_coup + " déplacements";
        possibleMove(item, items, tabImage);
        checkScore(tabImage, items);
    })
});
```

Items : liste des divs contenant les images et appartenant à la classe item

3. **Explication des fonctions utilisées (Toutes les fonctions sont documentées dans le code)**

- **getCoordItem(item) :** fonction qui retourne la position x et position y de l'objet item, donné en paramètre, sur l'écran.

- **getItemByValue(map, searchValue) :** fonction qui cherche dans l'objet map(=dictionnaire) la clé de la valeur donné en paramètre.

- **updateItem(elem1,elem2) :** fonction qui permute deux cases.
  2 cas possibles :
  - Permutation entre la case vide et une case normale
  - Permutation entre deux cases normales

  On récupère le numéro de chacune des deux images grâce à tabImage et on permute le numéro dans la source de l'image. Ensuite, on met à jour le nouveau numéro dans tabImage.

- **possibleMove(item,items,tabImage) :** fonction qui gère les permutations des cases selon tous les cas possibles.
  - Premièrement, on détecte l'éligibilité de la case cliquée à la permutation en vérifiant si elle est dans la même ligne ou la même colonne que la case vide pour respecter les règles du jeu de taquin.
  - Deuxièmement, on distingue les cas en fonction de la position de la case vide sur la grille suivant les 4 lignes ou les 4 colonnes.
  - Dernièrement, on distingue les cas selon la position de la case cliquée par rapport à la case vide (3 cas possibles : elles sont côte à côte, il y a une case entre les deux, il y a deux cases entre les deux) afin d'affecter toutes les pièces entre la pièce activée et la case vide

Explication de la permutation des cases en cas d'une permutation de toute une rangée :

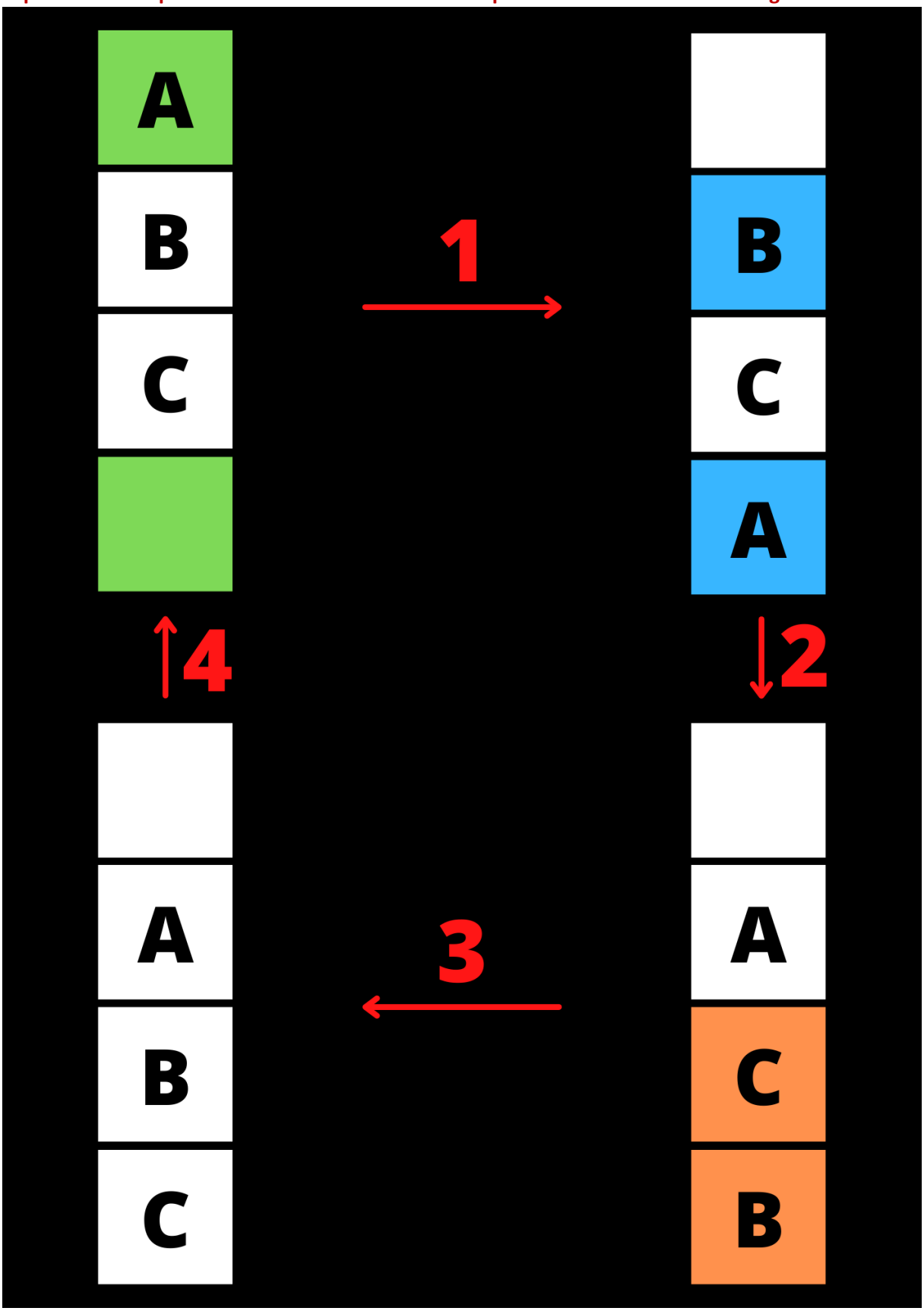

Figure 1 : Exemple si on fait un drag sur la case A

D’abord on permute la case A et la case vide. Ensuite, on permute la case A et la case B. Enfin, on permute la case B et la case C.

- **checkScore(tabImage, items) :** fonction qui gère la fin du jeu. Elle vérifie si toutes les tuiles sont bien placées.
  La tuile de la div X est bien placée si cette div a comme valeur (numéro de l’image) dans tabImage X+1.
  On parcourt notre objet map tabImage et on vérifie si :
  tabImage.get(items[index]) == index + 1
  Sachant que items[0]=la div de la case 1 et items[1]=la div de la case 2 **…**
  La partie est finie si le nombre de tuiles bien placées = 15
  Une fois la partie finie, on informe le joueur par un Pop-Up félicitant le joueur et on remplace le carré blanc de la div spéciale (la case vide) par img_3_3 de la base de données pour reconstruire l'image entière.