

## Projet — “Quelle tuile !”

L'objectif de ce projet est de mettre en œuvre un jeu de puzzle dans lequel il faut reconstituer une image en déplaçant les tuiles de la zone de jeu.

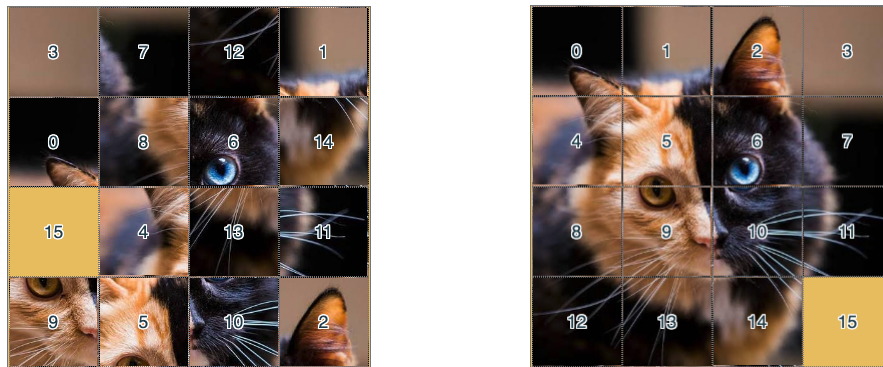


FIGURE 1 – Puzzle non-résolu (à gauche) et puzzle résolu (à droite). Image d'origine provenant de <https://www.instagram.com/p/BLcrm5AgOBC/>

## Instructions

- Le projet est à réaliser en binôme. Indiquez clairement en commentaire dans vos fichiers le nom et prénom de chaque membre du binôme. Un seul rendu est attendu par groupe.
- Vous devez déposer l'ensemble des fichiers (`index.html`, `style.css`, `jeu.js` ainsi que le dossier `./img`) que vous aurez, au préalable, archivé dans un `zip` (aucun autre format ne sera accepté).
- Vous devez remettre votre projet sur Moodle au plus tard le **6 mai 2022** à 20h02. Aucun travail ne sera accepté passé ce délai.
- Un “kit de démarrage” est disponible sur Moodle. Il est obligatoire de l'utiliser, et interdit de le modifier.
- Respectez les instructions et faites uniquement ce qui est demandé dans le sujet.

Le projet comporte deux parties principales :

1. Mise en forme avec CSS ( $\approx 30\%$ )
2. La mise en œuvre du jeu avec jQuery ( $\approx 70\%$ )

# 1 Mise en forme du jeu avec CSS

- Dans cette partie, on ne s’occupe à aucun moment de charger les images du puzzle (cela sera fait avec jQuery ; cf. partie 2).
- Il faut impérativement éviter la redondance de code.
- Tous les agencements doivent être réalisés avec Flexbox.

## 1.1 Mise en forme générale

1. Colorez le fond du corps de la page (le choix de couleur est libre).
2. Spécifiez l’ensemble des fontes comme sans empattement.
3. Le pied de page a une taille de fonte réduite de 10% par rapport à la taille de fonte par défaut et est écrit en italique.
4. Le titre du document et le pied de page doivent être alignés au centre.
5. Le bouton “Mélanger” :
  - n’a pas de bordure ;
  - a les bords arrondis (le degré d’arrondi est libre) ;
  - change de couleur lorsque le curseur de l’utilisateur passe dessus.

## 1.2 Mise en forme de la zone de jeu

Vous avez certainement remarqué que le fichier `index.html` contient un conteneur portant l’identifiant `puzzlearea` : il s’agit de la zone de jeu, dans laquelle vous allez placer les fragments de l’images (les tuiles).

1. Définissez un style pour la zone de jeu : la hauteur et la largeur *maximales* doivent permettre de pouvoir contenir *exactement* 4 tuiles (cf. point 2 pour la dimension d’une tuile).
2. Définissez un style pour la classe `.tuile` : définissez une bordure de 1 pixel grise ; la hauteur et la largeur totales d’une tuile doivent toutes deux faire *exactement* 90 pixel (ni plus ni moins).
3. Les fragments d’images qui seront chargées comme *background* (avec Javascript) dans les tuiles ont une dimension supérieure à la dimension des tuiles. Vous n’avez pas le droit de redimensionner les images, vous devez donc trouver la propriété CSS qui permettra de faire tenir chaque fragment en entier dans leur tuile. Ajoutez cette propriété à votre classe `.tuile`.
4. Pour aider à la résolution du puzzle, on écrira (dans la partie suivante, avec Javascript) le numéro des tuiles dans chacune d’elles. Vous devez définir un style qui s’appliquera *spécifiquement aux paragraphes contenus dans les tuiles* : le texte doit être centré et de couleur gris foncé, la taille de fonte 10% plus grande que la taille de fonte par défaut. Vous ajouterez en outre la propriété donnée suivante :

```
text-shadow: 1px 0 0 #fff,  
            -1px 0 0 #fff,  
            0 1px 0 #fff,  
            0 -1px 0 #fff,  
            1px 1px #fff,  
            -1px -1px 0 #fff,  
            1px -1px 0 #fff,  
            -1px 1px 0 #fff;
```

et indiquerez, en commentaire, ce que produit cette propriété en terme de rendu.

## 2 Mise en œuvre du jeu avec Javascript

Dans cette partie, vous devez utiliser la bibliothèque jQuery.

Le fichier fourni `jeu.js` contient la fonction `alea(min, max)` qui vous permettra de générer des nombres pseudo-aléatoires dans l'intervalle  $[\text{min}; \text{max}[$ . Cette fonction sera utile pour mélanger les pièces du puzzle.

Les images constituant le puzzle sont dans le dossier `./img`. Chaque image a pour dimension  $139 \times 139$  pixels. Vous n'avez pas le droit de les modifier (ni leur nom, ni leur dimension, *etc.*).

### 2.1 Préparation de la zone de jeu

- (a) Modifiez le DOM pour attacher 16 tuiles dans la zone de jeu (`#puzzlearea`) (à faire “intelligemment” : aucun point ne sera accordé si vous répétez l'instruction en dur 16 fois). C'est là que vous chargerez les images du puzzle.
- (b) À la création de chaque tuile, écrivez le numéro de la tuile sur celle-ci (pour rappel, vous avez défini un style pour ce texte dans la question 3 de la partie 1.2).

💡 Pour faciliter la suite de la mise en œuvre, il est très fortement conseillé de créer un identifiant (avec un attribut `html`) pour chaque tuile.

💡 À l'exception de la tuile n°15 —la tuile vide—, chaque tuile  $n$  aura pour image de fond (à spécifier en modifiant la propriété `css` correspondante avec Javascript) le fragment  $n$  présent dans le dossier `./img`.

### 2.2 Mélanger les tuiles

Ajoutez un écouteur d'événement sur le bouton **Mélanger**. Lorsqu'un clic de souris est détecté, la fonction déclenchée est la fonction `shuffle()`, que vous devez définir et mettre en œuvre : la fonction `shuffle()` permet de mélanger aléatoirement les tuiles de la zone de jeu pour démarrer une partie.

💡 Vous aurez besoin de la fonction `alea()` déjà présente dans le fichier `jeu.js`.

### 2.3 Déplacement des tuiles

- (a) Ajouter un écouteur d'événement sur les tuiles.
- (b) Lorsqu'un clic est détecté sur une tuile, la fonction qui s'exécute est la fonction `check_and_swap()`, que vous devez définir et mettre en œuvre.

💡 La fonction `check_and_swap()` permet à l'utilisateur de jouer pour résoudre le puzzle : si la tuile cliquée a pour voisine la tuile vide (numérotée 15), alors il s'agit d'échanger la tuile cliquée avec la tuile vide. On dira que la tuile vide est voisine de la tuile cliquée si cette dernière se situe immédiatement à gauche, en haut, à droite ou en bas de la tuile vide<sup>1</sup>.

---

1. Il s'agit d'un voisinage de von Neumann [https://fr.wikipedia.org/wiki/Voisinage\\_de\\_von\\_Neumann](https://fr.wikipedia.org/wiki/Voisinage_de_von_Neumann)

## 2.4 Puzzle résolu ?

Définissez et mettez en œuvre la fonction `puzzle_solved()` qui permet de vérifier, à chaque fois que l'utilisateur clique sur une tuile, si le mouvement effectué est celui qui a permis la résolution finale du puzzle. Si le puzzle est résolu, il faut :

- (a) Écrire un message dans le paragraphe `#output` indiquant à l'utilisateur que le puzzle est résolu ;
- (b) Détacher l'écouteur d'événement placé sur les tuiles (pour empêcher le joueur de continuer à déplacer les tuiles).

💡 Le puzzle est résolu si les tuiles sont dans l'ordre, c'est-à-dire 0, 1, 2, ..., 15.

💡 **Astuce pour la manipulation de la grille de tuiles** Vous allez très certainement, à un moment donné, vous retrouver avec un tableau 1D qui va contenir les tuiles de votre zone de jeu. Visuellement, votre grille est en 2 dimensions (et, si vous avez fait les choses correctement, cette grille va contenir 16 tuiles disposées en 4×4). Pour une tuile donnée, examiner la tuile à gauche et la tuile à droite est trivial. Mais où sont les tuiles au dessus et au dessous ? La figure 2 ci-dessous illustre ce “problème”. On peut faire les observations suivantes :

- Dans la représentation 1D, il suffit d'ajouter (resp. retrancher) la largeur de la grille 2D (en nombre de tuiles) à l'indice de la tuile examinée pour trouver l'indice du voisin en haut (resp. en bas).
- Quand on regarde dans la description 2D, on se rend compte que certaines tuiles n'ont pas de voisines à gauche ou en haut ou à droite ou en bas (par exemple, la tuile n°11 n'a pas de voisin à droite). Il faut donc prendre en compte ces cas dans votre code !

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

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

FIGURE 2 – Représentation 1D de la zone de jeu. Si on considère la tuile n°11, alors les voisins valides sont les tuiles 7 (en haut), 10 (à gauche) et 15 (en bas). Comme la tuile 11 est au bord droit de la grille, elle n'a pas de voisin à droite.

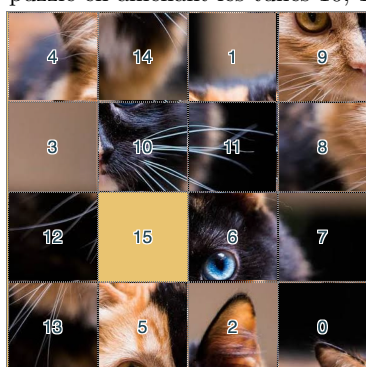


## Annexe : résolution du puzzle

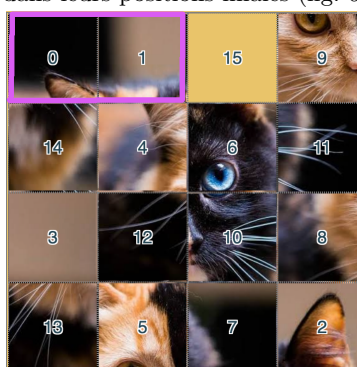
*Il n'y a rien à faire dans cette partie : il s'agit simplement d'indications pour résoudre le puzzle que vous avez mis en œuvre.*

Vous avez tout terminé? Votre jeu fonctionne? Vous avez tenté une partie, et il est apparu que ça n'est pas si simple que ça. Voici l'algorithme pour résoudre ce puzzle à partir de n'importe quel disposition des tuiles au départ.

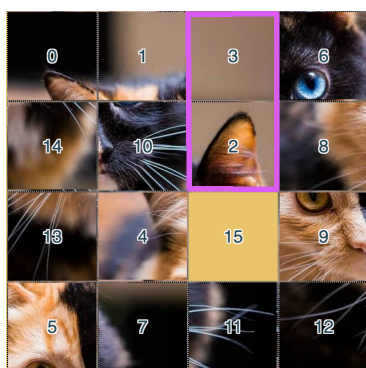
1. Amener les tuiles 0 et 1 dans leurs positions finales (fig. 3b)
2. Amener les tuiles 2 et 3 dans leurs positions préparatoires (fig. 3c)
3. Amener les tuiles 2 et 3 dans leurs positions finales (fig. 3d)
4. Amener les tuiles 4 et 5 dans leurs positions finales (fig. 4a)
5. Amener les tuiles 6 et 7 dans leurs positions préparatoires (fig. 4b)
6. Amener les tuiles 6 et 7 dans leurs positions finales (fig. 4c)
7. Amener les tuiles 8 et 12 dans leurs positions préparatoires (fig. 5a)
8. Amener les tuiles 8 et 12 dans leurs positions finales (fig. 5b)
9. Amener les tuiles 9 et 11 dans leurs positions préparatoires (fig. 5c)
10. Amener les tuiles 9 et 11 dans leurs positions finales (fig. 5d)
11. Terminer le puzzle en amenant les tuiles 10, 11, et 14 dans leurs positions finales (fig. 6a)



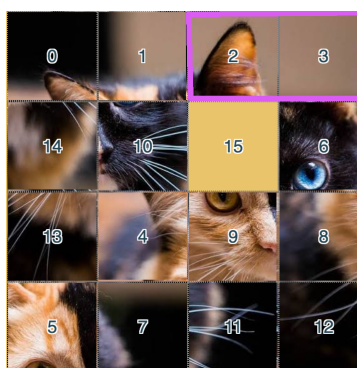
(a) Puzzle mélangé



(b) Tuiles 0 et 1 placées

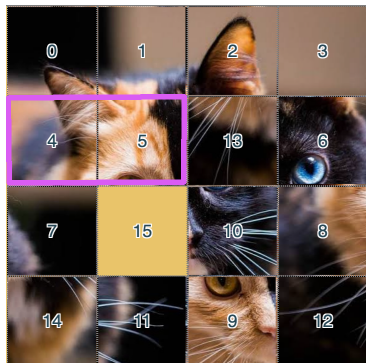


(c) Tuiles 2 et 3 préparées

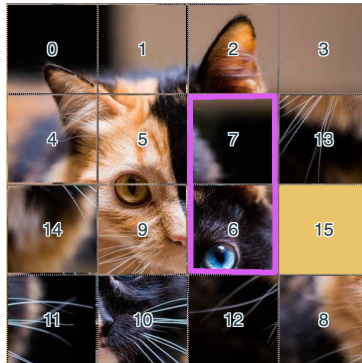


(d) Tuiles 2 et 3 placées

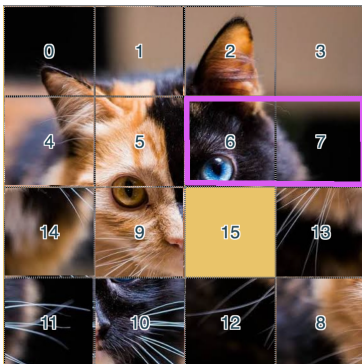
FIGURE 3



(a) Tuiles 4 et 5 placées

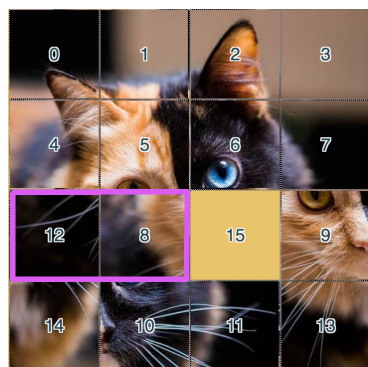


(b) Tuiles 6 et 7 préparées

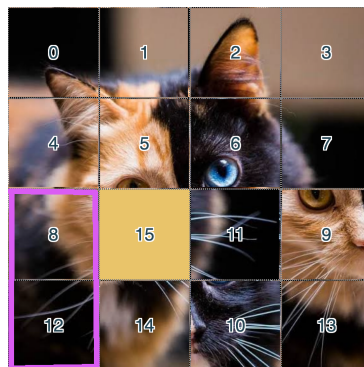


(c) Tuiles 6 et 7 placées

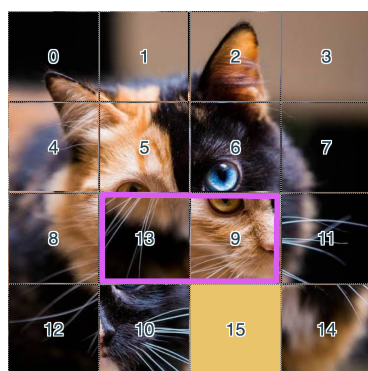
FIGURE 4



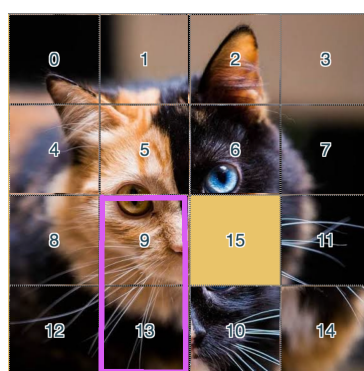
(a) Tuiles 8 et 12 préparées



(b) Tuiles 8 et 12 placées

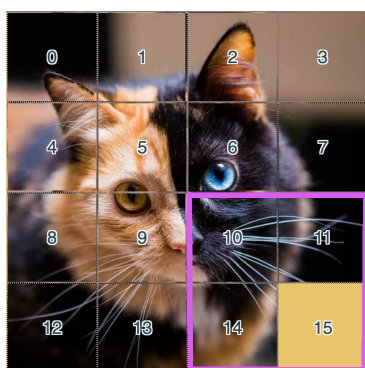


(c) Tuiles 9 et 13 préparées



(d) Tuiles 9 et 13 placées

FIGURE 5



(a) Puzzle résolu

FIGURE 6