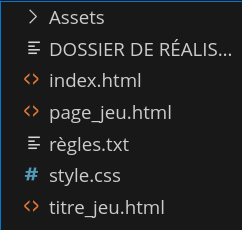
DOSSIER DE RÉALISATION

Le jeu que nous avons créé se nomme **BATAILLON**.

C’est un jeu de cartes qui consiste à affronter des adversaires bien différents et bien plus puissants au fur et à mesure que vous progressez jusqu'à ce que vous perdiez.

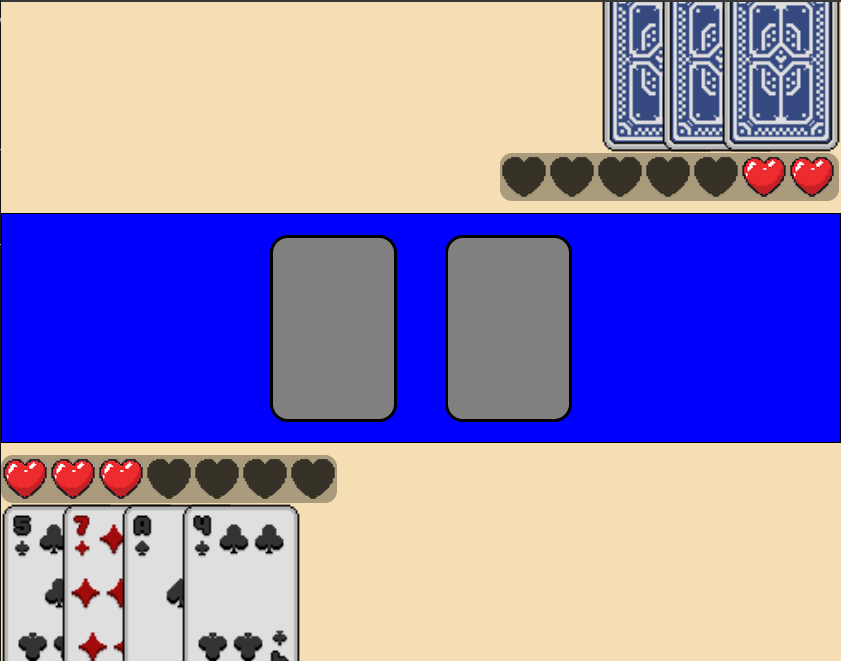
Pour le concevoir, nous avons utilisé les langages HTML/CSS et JavaScript sur Visual Studio Code avec l’extension Live Share pour tous modifier les mêmes fichiers en temps réel. Aucun module externe n’a été utilisé dans le code. Nous avons utilisé ces 3 langages pour 2 raisons principales :

Tout d’abord, ils permettent de faire une interface graphique facilement et rapidement, ce qui est important quand le temps est limité, et ensuite c’est une bonne solution pour créer un jeu accessible sur tous les supports sans nécessité de télécharger quoi que ce soit, car il est accessible par Internet avec l’aide de GitHub Pages.

Le dossier de notre jeu se compose donc de la manière suivante : 

Nous avons donc créé 3 pages html : index, titre\_jeu et page\_jeu, accompagnés d’un fichier css nommé style pour appliquer le css aux pages html. Tout le contenu JavaScript est contenu dans le fichier html page\_jeu car aucune autre page n’utilise de JavaScript et il n’y avait pas une nécessité de faire un fichier séparé pour l’utiliser. Enfin, il y a un dossier Assets dans lequel toutes nos images sont contenues.

Tout le jeu est jouable depuis la page principale index. On y retrouve le titre du jeu, une balise html iframe qui contient les autres pages html où sont contenues le jeu et les règles du jeu juste en dessous.

Sur l’écran principal de jeu, nous avons créé plusieurs blocs de balises html pour contenir, en haut et en bas de l’écran les espaces de jeu de la machine et du joueur, avec chacun leurs cartes et leurs vies qui possèdent un id unique pour le code JavaScript. 

Le code JavaScript nous a permis de rajouter du code interactif dans notre programme dans le but de permettre à l'utilisateur d'interagir avec les éléments qui sont affichés à l'écran. Dans notre cas, l'utilisateur interagit surtout avec les cartes.

Pour gérer les cartes, la vie, et les bonus propres à chaque cartes, nous avons créé plusieurs fonctions dans le code JavaScript qui peuvent être appelées à n’importe quel moment dans la partie. Notre code contient aussi de multiples variables pour des éléments précis du jeu, notamment le nombre de vie et le nombre de cartes de chaque joueur.

**Présentation de certaines fonctions :**

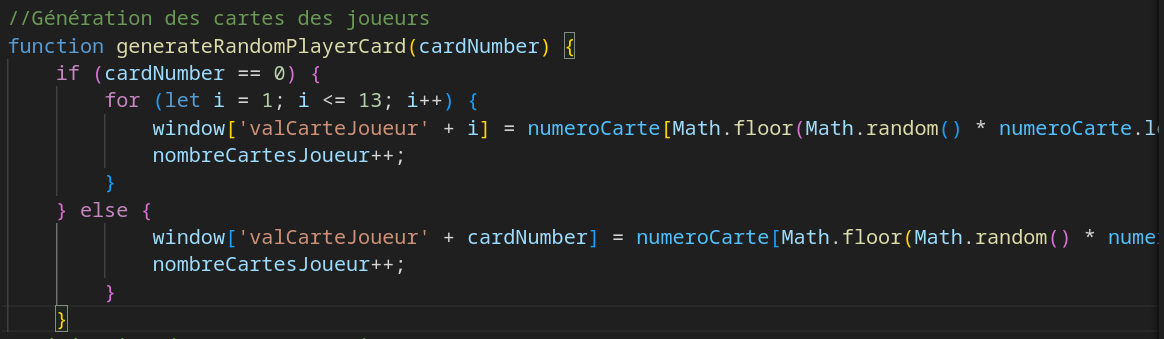
Tout d’abord, la fonction la plus utile est la fonction pour générer des cartes aléatoirement aux joueurs, nommée generateRandomPlayerCard(). La même fonction nommée generateRandomEnemyCard() sert à générer des cartes à la machine.

Pour générer une carte aux joueur, le code se déroule de la manière suivante:

Premièrement, il faut comprendre que les variables de valeur des cartes doivent matcher avec le nom des images des cartes dans nos dossiers. Le nom des cartes se présente comme ceci : il y a d’abord la valeur de la carte, et ensuite son type, séparés par un point. La carte 2 de cœur sera donc 2.2.png, et la variable associée à la carte 2 de cœur devra donc contenir “2.2”.



Avec ceci en tête, nous avons donc créé 2 tableaux : un contenant toutes les valeurs de cartes disponibles et un contenant tous les types de cartes disponibles. Ensuite, un élément de chacun de ces tableaux va être choisi aléatoirement et les 2 vont être assemblés dans une variable de carte pour qu’elle puisse ensuite s’afficher à l’écran.



Après avoir généré des cartes, on fait donc appel à la fonction permettant de les afficher.

Elle va regarder quelles variables de cartes sont égales à 0 (voulant dire que la carte est inutilisée) et va rendre ces dernières invisibles. Pour les cartes dont la valeur de la variable est différente de 0, elle va rafraîchir l’image et la rendre visible.

Pour gérer le jeu et la pose des cartes, une variable de priorité est présente. Si la priorité revient au joueur, le joueur pourra poser une carte en premier, dans le cas contraire, c’est la machine qui posera une carte.

Pour gérer la pose des cartes par le joueur, un eventListener est placé dans tout le document et vérifie que l’id de HTML cliqué comment bien par “carteJoueur”, pour que l’action qui le précède ne s’effectue que lorsqu’une carte est cliquée.

Pour l’ennemi, il posera automatiquement sa dernière carte lorsque c’est son tour.   
  
Une fois que cela est fait, une fonction pour trier les cartes du joueur afin qu’il n’y ait pas de trou dans sa main est appelée. Elle décale d'une carte la valeur de chaque carte à partir de la carte utilisée et fait ensuite appel à la fonction pour rafraîchir l’affichage des cartes.

Lorsque les 2 joueurs ont posé leur carte, leur valeur va être comparée.

Une boucle sera mise en place pour transformer en numéro les cartes au-dessus de 10 qui ont des lettres dans leur nom pour permettre la comparaison. Une fois cela fait, la carte la plus haute l’emporte et la variable de compteur de vies du joueur perdant descend de 1.

Si il y a égalité, les deux joueurs récupèrent leur carte.

Une fois cela terminé, le tour recommence. Une fonction va se charger de rendre invisibles les cartes au milieu du plateau et définir à nouveau les tours des joueurs en fonction de la priorité actuelle.

Il y a d’autres fonctions moins intéressantes à décrire comme la pioche des cartes qui va générer un nombre donné de cartes à un joueur, ou la fonction qui va regarder la variable du nombre de vie de chaque joueur et cacher les vies perdues.