Rapport de projet de RICHAUME Anthéa et LEFEVRE Sebastien

ORush : interface graphique

I. Choix techniques

L'état du port est stocké dans un type etat qui contient l'état du port courant, le bateau courant qu'on veut déplacer et la solution calculée dans solver.
L'état (etat_jeu) est une variable globale pour pouvoir le manipuler dans toutes les fonctions.
Le compteur de déplacement cpt est également global.
Les couleurs sont associées aux lettres des bateaux dans une table de hachage pour pouvoir plus facilement associer une couleur aux bateaux lorsqu'il faudra les dessiner.

Le canvas permet d'afficher la grille et l'etat du port courant, en l'occurrence les bateaux. Une case dans ce canvas fait 100*100px pour simplifier le calcul et les zones sélectionnées dans le programme.

L'ajout des bateaux se fera via un textarea qu'il faudra soumettre.

Les différentes actions seront transmises via différentes commandes. Des commandes directionnelles (représentées par des flèches) pour les mouvements et des commandes de résolution pour terminer une partie.

II. Fonctionnement du programme

Lors de l'ouverture de la page HTML, il faut écrire les bateaux dans la forme 'A2H12' dans la zone de texte en les séparant par un **unique** retour à la ligne et cliquer sur « Add boats » pour les soumettre. Attention de ne pas ajouter de retour à la ligne en plus (notamment après le dernier bateau).

À noter que nous avons essayé d'utiliser des expressions régulières afin de traiter toute erreur éventuelle avec **split** : `regexp -> string -> string list` cependant celle-ci n'arrivait pas à détecter correctement les espacements et retours chariots.

Lorsque le bouton est appuyé cela déclenche un évènement appelant **add_state** qui récupère la chaîne de caractère dans la zone de texte, qui sépare les bateaux avec **split_on_char** et qui ajoute les bateaux dans une liste avec **add_boat**. Cela à pour effet d'initialiser etat_jeu. Par défaut, le cur_boat est le bateau A. On aurait pu utiliser un type optionnel afin d'éviter cet ajout, mais dans cette situation ce n'était pas nécessaire.

Ensuite la fonction **refresh** dessine les bateaux avec leur identifiant via **draw_boat** et encadre le bateau actuellement sélectionné**.** Elle redessine également la grille via **draw_grid** dans le canvas.

draw_boat récupère la couleur correspondant à l'identifiant du bateau à l'aide de la table de hachage colors.

Le jeu peut alors commencer.

Lorsqu'on clique sur le canvas, un évènement est déclenché et **click** est appelée. Cette fonction encadre le bateau sélectionné avec **draw_rect** et met à jour le cur_boat. Pour trouver le bateau à partir des coordonnées du curseur de l'évènement, la fonction utilise **grid_of_state** afin d'avoir la matrice de l'état courant. Si un bateau se trouve à la case cliquée il sera alors sélectionné, sinon l'ancien bateau sélectionné reste conservé.

Pour déplacer le bateau sélectionné, il suffit de cliquer sur les boutons de déplacement, ce qui déclenche un évènement appelant la fonction **click_avance** ou **click_recule** si on a appuyé sur bas/droite ou haut/gauche respectivement. Ces fonctions appellent **apply_move** afin de modifier le state courant, elles incrémentent le compteur de déplacements et mettent à jour cur_boat (car sa position a changé).

A chaque mouvement, **is_win** est appelée pour vérifier si l'état est gagnant. Si c'est le cas, tous les boutons sont désactivés, et le nombre de déplacements effectués est affiché. Pour appliquer visuellement un déplacement (mettre à jour les éléments graphiques sur le canvas), la fonction **refresh** est appelée.

Vous n'arrivez pas à finir la partie ?

→ Solver est là à votre service !

Le bouton Solver permet de résoudre automatiquement le jeu à n'importe quel moment et pourra être appelé plusieurs fois. Lorsque le bouton est appuyé, un évènement est déclenché, ce qui appelle **solver**. Cette fonction calcule la solution en partant de l'état courant à l'aide de la fonction **solve_state** définie dans le module Solver. La solution est alors ajoutée à etat_jeu et affichée sur la page.

Pour appliquer un déplacement de la solution, il faut cliquer sur Next. La fonction **next_move** est alors appelée. Elle récupère les deux premiers caractères de la solution de etat_jeu (le premier déplacement) et les retire. Ensuite le cur_boat est changé via le premier caractère récupéré correspondant à l'identifiant du bateau. Puis on applique **click_avance** ou **click_recule** sur l'état courant suivant le second caractère.

Si plus aucun **next_move** est possible (solution terminée) tous les boutons sont désactivés.

Si on clique sur Finish, **finisher** est appelée. Cette fonction est semblable à **next_move**, mais elle finit totalement le jeu à partir de la solution courante en appliquant tous les mouvements même si des **next_move** ont été effectués au préalable. Si des bateaux sont déplacés après un solve la solution courante ne sera plus correcte il faudra alors l'appeler encore une fois avant de faire un Finish.

Nous aurions voulu ajouter des fonctionnalités sur la désactivation et ré-activation des boutons. Notamment sur ce Finish, on aurait voulu l'activer uniquement si aucun déplacement n'a été fait après un solve. On aurait voulu également désactiver les boutons de déplacement verticaux lorsqu'un bateau horizontal est sélectionné et inversement pour les bateaux verticaux.

Nous n'avons pas pu faire cela car l'attribut disabled attendait un booléen. Ce que set_attribute ne nous permettait pas. Nous avons alors essayé de changer les fonctions dans Dom. La première approche était de modifier le prototype de set_attribut. Cela reviendrait à perdre toutes les utilisations ultérieures qui prenaient en paramètre une string.

L'autre approche était de créer un nouveau module Element dans lequel définir le nouveau set_attribute. Il fallait alors créer un nouveau module Document dans lequel get_element_by_id attend le nouveau Element.t, car get_element_by_id dans le module Document initial attend l'Element.t initial. Cependant cela ne pouvait fonctionner car les nouveaux modules créés ne sont pas présent dans le type nodeType de DOM objects. (recherche : https://ocsigen.org/js_of_ocaml/3.1.0/api/Dom)