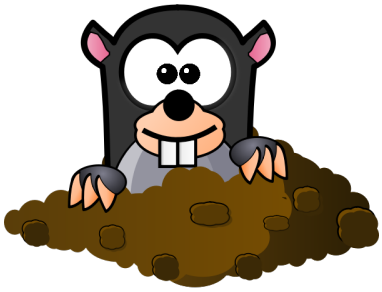
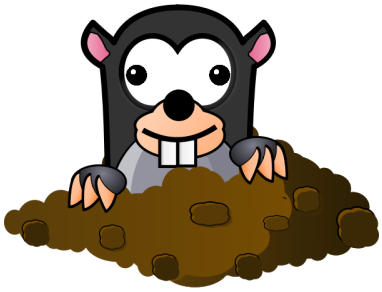
L'objectif de ce TP est d'implémenter le jeu *Guacamole*, une version francisée du jeu *Whack-a-mole*. Le principe est de frapper (ou cliquer, dans notre cas) le plus rapidement possible sur une taupe qui sort de terre. Attention cependant à ne pas se tromper de cible !

Téléchargez les images `bonne_taupe.png` et `mauvaise_taupe.png`, disponibles sur Eprel.



(a) Une bonne taupe.



(b) Une mauvaise taupe.

Question 1 Remplissez un fichier HTML minimal, dont le `body` est de couleur `saddlebrown` et ne contient pour l'instant une seule balise `script`, dont la source est le fichier `tp6.js` (à créer).

Dans la suite, on se contentera de modifier le fichier `tp6.js` : il est donc interdit de modifier le code HTML ou CSS sans passer par JavaScript.

Une première taupe

Question 2 Écrivez une fonction `creerTaupe()` qui prend en argument une chaîne de caractères `image`, et qui

1. crée une nouvelle balise `img` ;
2. ajoute cette balise comme dernier enfant du `body` ;
3. précise que l'image associée à cette balise est `image` ;
4. précise que la `position` (qui est un attribut du `style`) de cette balise est `"absolute"` ;
5. précise que la `width` et la `height` de cette balise sont toutes deux égales à un huitième de la plus petite dimension de la fenêtre (les dimensions de la fenêtre du navigateur sont données par `window.innerWidth` et `window.innerHeight`) ;
6. retourne cette nouvelle balise.

Appelez une première fois `creerTaupe()` sur l'argument `"bonne_taupe.png"` : vous devriez voir apparaître une taupe en haut à gauche de l'écran, comme en Figure 2.

Question 3 On veut maintenant positionner aléatoirement la taupe. Pour cela, implémentez une fonction `placerTaupe()` qui attend en argument une balise `taupe`, et qui

1. génère aléatoirement deux coordonnées à l'intérieur de la fenêtre grâce à `Math.random()` ;

FIGURE 2 – Apparition de la bonne taupe en haut à gauche de la fenêtre

2. place `taupe` à ces coordonnées, en précisant les champs `"left"` et `"top"` de son `style`. Par exemple, pour placer la gauche de `taupe` 10 pixels à droite du bord gauche de la balise mère, on écrira `taupe.style.left="10px"`.

On fera attention à ce que l'image ne puisse pas dépasser de la fenêtre, en prenant de la marge en bas et à droite.

Au chargement de la page, on placera donc aléatoirement la taupe initiale.

Question 4 Pour terminer cette première version du jeu, faites en sorte que chaque clic sur la taupe déplace aléatoirement l'image. On implémentera pour cela un callback `clicBonneTaupe()`.

Introduction des mauvaises taupes

Question 5 Déclarez une liste `mauvaisesTaupes`, initialement vide.

Complétez le code de `clicBonneTaupe()` pour qu'à chaque clic sur la bonne taupe (qui est seule initialement), une nouvelle balise `img` affichant `mauvaise_taupe.png` soit créée et soit ajoutée au `body`. Cette balise devra être ajoutée à la liste `mauvaisesTaupes`.

On évitera évidemment de dupliquer le code.

Question 6 Modifiez votre code pour qu'à chaque clic sur la bonne taupe, toutes les taupes (la bonne et les mauvaises) soient repositionnées aléatoirement dans la fenêtre.

Un exemple est donné en Figure 3.

Question 7 Faites en sorte qu'un clic sur une mauvaise taupe fasse perdre le jeu ; dans ce cas, le jeu devra repartir de zéro (en particulier, toutes les mauvaises taupes devront être supprimées).

Question 8 Modifiez votre code pour que deux taupes ne se chevauchent jamais.

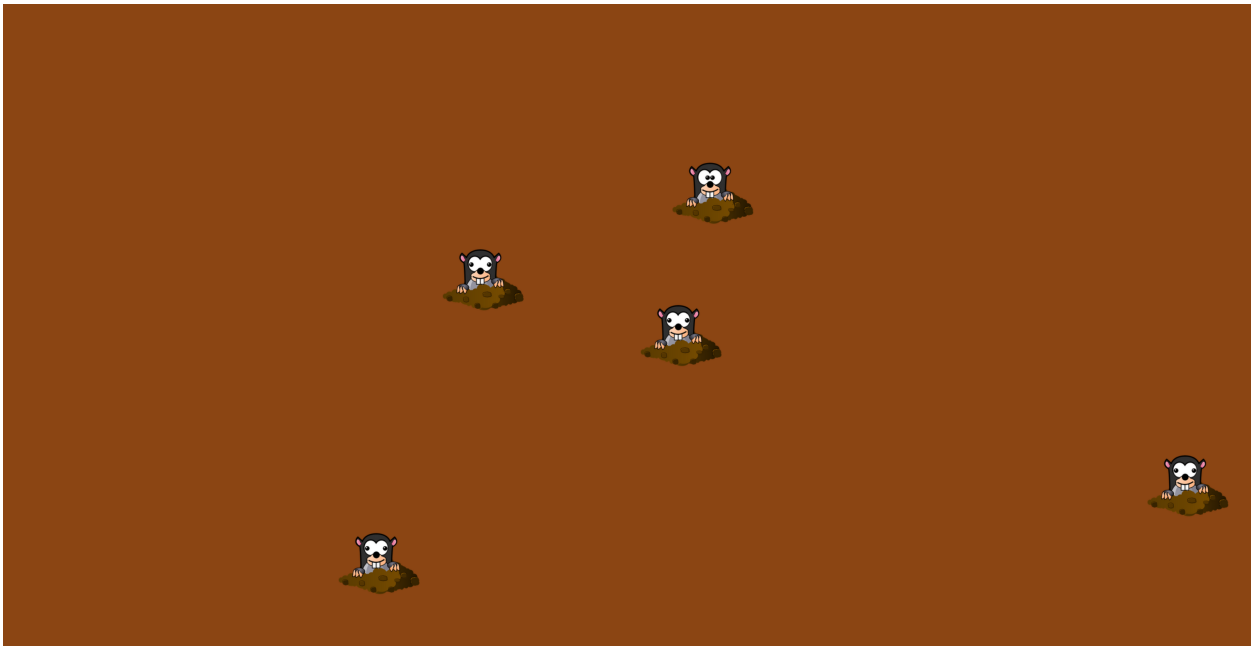


FIGURE 3 – Exemple de configuration après quatre clics sur la bonne taupe : quatre mauvaises taupes ont été introduites.

Ajout d'une limite de temps

Question 9 Pour terminer, modifiez le code pour que le ou la joueuse ne bénéficie que de deux secondes à chaque coup pour cliquer sur la bonne taupe. En cas de non-respect de ce délai, le jeu est perdu et repart de zéro.

On utilisera pour cela les fonctions `setTimeout()` et `clearTimeout()`.