TP n°3 : une librairie js : jQuery

1. **Petit jeu de balle.**
   Télécharger la dernière version de **jQuery** sur https://code.jquery.com (jQuery est une bibliothèque JavaScript qui porte sur l'interaction entre JavaScript et HTML et a pour but de simplifier des commandes communes de JavaScript) et prendre le fichier `template_exo5.html` à l'adresse habituelle : http://isis.truck.free.fr/Site/ens Le but est de programmer le fichier `scriptJeuBalle.js`. Créez ledit fichier JavaScript dans un sous-répertoire `js`. Le but du programme est de **simuler le mouvement d'une balle sur un plan**. C'est-à-dire qu'il faut dessiner une balle qui aura une vitesse donnée puis simuler son déplacement en utilisant une approximation des lois de la physique. Pour simuler le déplacement, il faudra mettre à jour les différentes coordonnées, vitesse et accélération.

   1.1 **Commencer par déclarer les variables globales (sans type) :** la balle a un diamètre `diamBalle` (20), une **position** en 2D notée `x`, `y` (300,300), une **vitesse** notée `vx`, `vy` (aléatoire) ainsi qu'une **accélération** notée `ax`, `ay` (0,0). Le paramètre `d` (1.0) représente le pas de temps alors que le paramètre `e` (0.004) représente le coefficient de frottement de la balle. Donner à ces variables les valeurs par défaut indiquées entre parenthèses.

   1.2 Faire une fonction de dessin pour représenter la balle. Pour dessiner en JavaScript, il suffit de récupérer le contexte de dessin
   (`var context = document.getElementById("myCanvas").getContext("2d");`)
   sur lequel on fera appel à des fonctions comme dessiner des cercles, des rectangles, remplis ou non... (http://www.w3schools.com/tags/ref_canvas.asp). Ce contexte est associé à un élément html de type `canvas`. Utiliser la fonction ready() de jQuery : `$(document).ready(function() { /* … */ })` : tout ce qui se trouve dans les accolades et les parenthèses de ready sera chargé dès que le DOM sera chargé et avant le contenu-même de la page. **Dessiner la balle en une couleur de son choix**, en position x,y en utilisant arc().

   1.3 « **Effacer** » la zone de jeu avant d'afficher la balle. Comment peut-on faire, sachant que le fond est blanc ? (Faire un test en dessinant 2 fois la balle à deux endroits différents et en effaçant entre les 2 affichages).

   1.4 Écrivez maintenant une fonction maj() réalisant les opérations suivantes: incrémenter x de la valeur de la vitesse à chaque pas de temps d. Idem pour y. Incrémenter également la vitesse et l'accélération :

   ```
   vx = vx + d * ax;  vy = vy + d * ay;  ax = - e * vx; ay = - e* vy;
   ```

   Appeler successivement dessine() puis maj() plusieurs fois pour voir le déplacement de la balle.

   1.5 Animer « proprement » l'image en utilisant une fonction anime() qui appelle dessine() puis maj() et qui est appelée grâce à la méthode setInterval(). Que se passe-t-il au bout de quelques instants ?

   1.6 Faire en sorte de résoudre le problème en testant les coordonnées x,y de la balle. Que se passe-t-il, malgré tout, au bout d'un certain temps (à cause de e) ?

   1.7 Faire une fonction qui relance le jeu en repositionnant de façon aléatoire les vitesses vx et vy. Utiliser $("#idElement") pour sélectionner un élément et effectuer une action (ici, événement `click`) sur l'élément : `$("#myCanvas").click(function() { relance(); });`

   1.8 Lors de la relance, on souhaite changer aléatoirement la couleur de la balle. Dans la fonction relance(), ajouter ce changement de couleur.

2. Lire les exemples proposés ici (système terrestre animé ou horloge animée) : https://developer.mozilla.org/fr/docs/Tutoriel_canvas/Animations_basiques et regarder également l'exemple complet assez ressemblant à l'exercice 5, mais avec de la programmation objet : https://developer.mozilla.org/fr/docs/Tutoriel_canvas/Advanced_animations