

TP n° 1

Préliminaires

Afin de faciliter le débogage des programmes, il est conseillé d'utiliser le même navigateur Web que celui utilisé en cours, à savoir Google Chrome. Une fois que vos programmes fonctionnent, il est intéressant de les tester sur d'autres navigateurs afin d'en assurer la robustesse.

Pong

Le but de ce TP est d'implémenter en Javascript le jeu Pong pour deux joueurs. Il permettra de se familiariser avec la console Javascript du navigateur, la manipulation du document HTML depuis Javascript et les constructions de base du langage.

Questions

1. Récupérer les fichiers `index.html`, `pong.js` et `style.css` sur la page du cours. Placer ces fichiers dans un répertoire `ld/tp01`. Si ce dernier est dans votre répertoire `public_html`, alors vous pourrez y accéder depuis le serveur Web du département (depuis une machine du PUIO) à l'adresse

`http://tp-ssh1.dep-informatique.u-psud.fr/~prenom.nom/ld/tp01/index.html`

Même si cela n'est pas nécessaire pour l'instant, certaines fonctionnalités de Javascript ne sont autorisées par les navigateurs que si la page est servie par un serveur Web et non pas en local. Se familiariser avec le contenu des fichiers.

2. Ajouter dans l'élément `head` de la page HTML une balise :

```
<script type="text/javascript" src="pong.js"></script>
```

Recharger la page (Ctrl-R), afficher la console Javascript (Ctrl-Shift-J), et vérifier qu'aucune erreur n'y apparaît.

3. Dans le fichier `pong.js` compléter la fonction javascript `init_ball` qui initialise l'objet `ball` en définissant ses coordonnées à (120,290), et qui initialise la propriété `display` avec l'objet HTML correspondant à l'élément d'id `ball` (on pensera à recharger la page une fois la fonction écrite pour vérifier la syntaxe du code Javascript). Une fois la page rechargée, afficher le contenu de l'objet `ball` dans la console (en évaluant l'expression `ball`) puis appeler la fonction `init_ball()`. Regarder de nouveau le contenu de `ball` et constater que les champs sont bien initialisés.
4. Ajouter maintenant après la définition de la fonction `init_ball()` un appel à cette fonction. Recharger la page. Que se passe-t-il? Déplacer maintenant l'élément `<script>` se trouvant dans la balise `<head>` juste avant la balise fermante `</body>` (après la fin du `</div>` d'id `canvas`). Recharger la page. Proposer une explication pour cette différence de comportement (laisser la balise `<script>` en fin de document pour la suite mais retirer l'appel à `init_ball`).
5. Écrire sur le même modèle qu'`init_ball` deux fonctions `init_players` et `init_walls` qui initialisent les coordonnées des deux joueurs et des deux murs respectivement (les murs sont en haut et en bas, les deux « raquettes » à gauche et à droite. La raquette de gauche est au coordonnées (80,230) et la raquette de droite est symétrique).
6. Écrire une fonction `draw(o)` qui prend en argument un objet possédant des propriétés `x`, `y`, `width` et `height` et une propriété `display` contenant un objet DOM et qui mets à jour le style CSS `left`, `top`, `width` et `height` de l'objet DOM (attention, les propriétés CSS sont des chaînes de caractères avec des unités, donc si `o.x` vaut 42, on placera `"42px"` dans `o.display.style.left`). Appeler dans la console `draw(ball)`, `draw(wall1)`, ... après avoir appelé les fonction d'initialisation.

7. Ajouter sur l'objet document (représentant toute la page) un gestionnaire d'évènement pour l'évènement "keydown". Ce dernier doit tester le code de la touche pressée et réagir comme suit :
 - touche 'e' : remonter la raquette du joueur 1 de 10 pixel (sauf si on touche le mur du haut)
 - touche 'd' : baisser la raquette du joueur 1 de 10 pixel (sauf si on touche le mur du base)
 - touche 'o' : remonter la raquette du joueur 2 de 10 pixel (sauf si on touche le mur du haut)
 - touche 'l' : baisser la raquette du joueur 2 de 10 pixel (sauf si on touche le mur du base)
 Le code de la touche pressée se trouve dans la propriété keyCode de l'évènement. Les codes des touches e, d, o et l sont 69, 79, 68 et 76. Vérifier que le gestionnaire fonctionne correctement en affichant l'objet player1 dans la console après avoir pressé d plusieurs fois. Ajouter le code pour appeler les trois fonctions d'initialisation puis afficher les deux murs.
8. Écrire une fonction update() qui redessine les objets ball, player1 et player2. Utiliser ensuite la fonction setInterval(update, xxx) où xxx est un intervalle de répétition en milliseconde, afin que l'affichage soit redessiné 60 fois par secondes.
9. Écrire une fonction update_ball() qui mets à jour la position de la balle en fonction de sa vitesse (on suppose que l' unité de speed_x et speed_y est en pixel/frame). Appeler cette fonction au début de update. Recharger la page et modifier dans la console les valeurs de ball.speed_x ou ball.speed_y. Constater que la balle avance.
10. Modifier la fonction update_ball() pour détecter les collisions. Après la mise à jour des coordonnées, calculer les coordonnées x_c et y_c du centre de la balle.
 - si y_c est telle que la balle touche l'un des murs (haut ou bas), changer ball.speed_y en -ball.speed_y
 - si x_c est telle que la balle est au delà des raquettes alors tester si y_c est compris entre le haut et le bas de la raquette
 - si c'est le cas, changer ball.speed_x en -ball.speed_x
 - sinon renvoyer 1 si la raquette du joueur 2 est dépassée (le joueur 1 marque un point) ou renvoyer 2 si la raquette du joueur 1 est dépassée.
 - renvoyer 0
 Recharger la page et « lancer » la balle manuellement (en faisant modifier ball.speed_x et ball.speed_y dans la console) pour tester les rebonds.
11. Modifier la fonction update pour récupérer le résultat de update_ball. Si ce dernier est différent de 0, mettre la vitesse de la balle à 0, réinitialiser la position des raquettes et incrémenter le score du joueur correspondant et placer le numéro de l'autre joueur dans une variable globale to_play. Positionner la balle devant la raquette qui a perdu le point.
12. Écrire une fonction launch qui lance la balle de manière aléatoire. Pour cela on tire un angle aléatoire θ compris entre $-\frac{\pi}{3}$ et $\frac{\pi}{3}$ et on calcule les coordonnées du vecteur vitesse de la balle avec :

$$\begin{aligned} \text{speed}_x &= \pm r \times \cos(\theta) \\ \text{speed}_y &= r \times \sin(\theta) \end{aligned}$$

Le signe du déplacement horizontal dépend du joueur (si to_play vaut 1 positif, sinon négatif). La constante r constitue la norme du vecteur vitesse (la balle avancera de r pixels par *frame*). Associer cette fonction à la touche h du clavier (code 72). Les fonctions mathématiques nécessaires sont dans l'objet Math (.random(), .PI, .sin(), .cos()).

13. Quels sont les problèmes de ce code Javascript?
14. (facultatif) Afficher le score dans un nouveau div, améliorer le style, ...