Cahier des Charges

Présentation

Ce Cahier des Charges projet conclut l'apprentissage des bases du langage ECMAScript 3/5 à travers un projet dont les objectifs pédagogiques sont :

- Mettre en pratique et développer ses connaissances algorithmiques ;
- Mettre en œuvre et utiliser des structures objet ;
- Mettre en œuvre et utiliser des *design pattern* ;
- Gérer la mémoire et se prémunir des fuites de mémoire ;
- Utiliser et manipuler le BOM et le DOM ;
- Gérer les évènements ;
- Combiner les connaissances traitées en cours pour créer un programme original.

Il s'agit de créer une simulation d'aquarium.

Description

Principe

Vous devez réaliser un programme qui sera une simulation d'aquarium. Cet aquarium contient des poissons. Chaque poisson est créé à l'aide d'un clic de souris. Chaque poisson se déplace dans une direction aléatoire de façon autonome. Un poisson ne peut pas sortir des limites de l'aquarium. Si il arrive à une limite de l'aquarium, il doit choisir une nouvelle direction de déplacement aléatoire. Si un poisson arrive à proximité d'un autre poisson, il doit suivre ce poisson. Si l'utilisateur approche le pointeur de la souris d'un poisson celui-ci doit s'éloigner du pointeur de la souris et, par extension, si l'utilisateur approche le pointeur de la souris d'un groupe de poissons, ceux-ci doivent s'éloigner du pointeur de la souris.

Fonctionnalités

1. **Generate** (Génération) : Créer une zone délimitée de l'affichage qui représentera l'aquarium. Cliquer avec la souris au sein de cette zone délimitée doit entraîner la création d'un élément graphique représentant un poisson aux coordonnées du pointeur de la souris.
2. **Movement** (Mouvement) : Un poisson doit se déplacer de façon autonome dans une direction aléatoire sans jamais dépasser les limites de l'aquarium. Si un poisson atteind une des limites de l'aquarium, il doit opter pour une nouvelle direction aléatoire de déplacement. Cette direction aléatoire de déplacement doit exclure la direction initiale.
3. **Shoal** (Banc) : Si un poisson se trouve à proximité d'un autre poisson, il doit adopter la même direction de déplacement que ce poisson. La distance à partir de laquelle on considère que 2 poissons sont a proximité l'un de l'autre est laissée à votre discrétion. Le premier poisson qui détecte la proximité d'un autre poisson le

suit.

4. **Shark** (Requin) : L'utilisateur peut survoler la zone délimitée représentant l'aquarium avec le pointeur de la souris. Tous les poissons qui se déplacent en direction des coordonnées du pointeur de la souris ou à proximité des coordonnées de la souris doivent changer de direction pour une direction aléatoire. Cette direction aléatoire doit exclure la direction initiale ainsi que les directions correspondant à la position du pointeur de la souris ou à proximité.

Obligations techniques

Gestion du Code Source

Vous devrez utiliser GIT pour gérer votre code source. Vous proposerez un dépôt distant sur un compte privé ou public chez un fournisseur de service tel que GitHUB ou BitBucket ou sur un système auto-hébergé tel que GitLab Community Edition.

En s'inspirant de la méthodologie Git Flow, vous devez créer 5 types de branches :

- Une branche **master** pour les versions définitives. Chaque version devra être taguée avec un numéro de version. C'est sur cette branche que vous fusionnerez vos **releases** ou vos **hotfix**;
- Une branche **release** pour les versions de recette. C'est sur cette branche que vous fusionnerez vos versions de développement stables issues de **develop**.
- Une branche **develop** pour les versions en cours de développement. C'est sur cette branche que vous fusionnerez vos **features** ;
- Plusieurs branches **feature** pour chaque fonctionnalité détaillée plus haut. Par exemple :
  - Une branche **feature/generate** ;
  - Une branche **feature/movement** ;
  - Une branche **feature/shoal** ;
  - Une branche **feature/shark** ;
- Une branche **hotfix** sur laquelle vous corrigerez les bugs des versions de la branche **master**.

Vous veillerez à effectuer des commit régulièrement et à produire des messages de commit suffisamment descriptifs de votre travail.

Ergonomie et graphisme

La représentation graphique de l'aquarium et des poissons est sans importance :

- L'aquarium peut être représenté par un rectangle ou un carré. Les limites de l'aquarium doivent être clairement identifiables.
- Les poissons peuvent être représentés par des carrés ou des ronds. Les poissons doivent être visibles.

Outils de développement

L'ensemble du site sera entièrement développé en HTML5, CSS3 et ECMAScript 3/5. Le document Web sera valide W3C.

Vous pouvez utiliser l'IDE de votre choix.

Vous ne devez utiliser **aucun** framework ou librairies.

Compatibilité

Le jeu sera entièrement fonctionnel sur tous les navigateurs compatibles HTML5, c'est-à-dire : Edge dernière version, Firefox dernière version, Chrome dernière version, Safari dernière version et Opera dernière version.

La compatibilité avec les appareils mobiles n'est pas obligatoire.

Recommandations

Vous veillerez à bien organiser votre code pour améliorer sa présentation (indentation, sauts de ligne, choix de noms explicites). Pour ce faire, vous soignerez particulièrement les commentaires.

Vous n'hésiterez pas à scinder votre code en plusieurs fichiers.

Préalablement à la réalisation du site, vous élaborerez l'organisation de votre code.

En ECMAScript, privilégiez l'usage d'objets plutôt que de nombreuses variables. Vous limiterez l'usage des variables globales en usant préférentiellement de variables locales.

Aides

Pour la partie visuelle :

L'aquarium peut être représenté sous la forme d'un élément HTML canvas ou d'un élément HTML de type div.

Si l'élément choisi pour la représentation de l'aquarium est un élément HTML de type div, un poisson pourra être représenté par un élément HTML de type div également.

Si vous optez pour la stratégie canvas, le code sera plus abstrait mais vous aurez de meilleures performances. Les dessins sur canvas sont accélérés par le GPU.

Si vous optez pour la stratégie div, le code sera moins abstrait mais vous aurez de moins bonnes performances. Pas d'accélération GPU, sauf si vous gérez les animations à l'aide de transitions CSS3.

Pour l'organisation du code :

Vous devez créer une fonction constructeur pour un **poisson** :

- Un poisson contient :
  - Ses coordonnées actuelles ;
  - Une référence à l'aquarium qui le contient ;
  - … et éventuellement d'autres propriétés.
- Le prototype d'un poisson contient :
  - une méthode qui calcule sa position suivante ;
  - une méthode qui détermine si sa position suivante est applicable ;
  - une méthode qui remplace ses coordonnées actuelles par les coordonnées de sa position suivante ;
  - … et éventuellement d'autres méthodes.

Vous devez créer une fonction constructeur pour **l'aquarium** :

- Un aquarium contient :
  - Ses coordonnées actuelles ;
  - Une référence à la simulation qui le contient ;
  - Un tableau de poissons qui seront créés à l'aide de la fonction constructeur de poisson ;
  - …
- Le prototype d'un aquarium contient une méthode qui parcourt à intervalle de temps régulier le tableau de poissons pour :
  - Effectuer les calculs nécessaires à leurs déplacements et aux détections de proximité ;
  - Modifier les propriétés de position des poissons en fonction des résultats des calculs ;
  - Appliquer les propriétés de positionnement de chaque poisson à sa représentation graphique ;
  - …
- Le prototype d'un aquarium peut éventuellement contenir d'autres méthodes…

Vous devez créer une fonction constructeur pour la **simulation** :

- Une simulation contient :
  - Un aquarium ;
  - Les coordonnées du pointeur de la souris ;
  - … et éventuellement d'autres propriétés.
- Le prototype d'une simulation contient :
  - Une méthode qui initialise la simulation en :
    - Créant un aquarium ;
    - Déclarant les gestionnaires d'événements relatifs aux mouvements de la souris
  - … et éventuellement d'autres méthodes.

Pour les calculs :

(Rappels de mathématique de 2nde)

Calculer les coordonnées de la prochaine position d'un poisson en fonction d'une direction :

Théorème 1
Soit D une droite du plan non parallèle à l'axe des ordonnées.
Il existe un unique couple de réels (m, p) tel que la droite D ait pour équation réduite $y = mx + p$.

Définition
Le réel m est appelé **coefficient directeur** de la droite D.

Théorème 2
Soit D et D' deux droites du plan non parallèles à l'axe des ordonnées.
D et D' sont parallèles si et seulement si elles ont le même coefficient directeur.

Théorème 3
Soit D une droite du plan non parallèle à l'axe des ordonnées, et A et B deux points distincts de D de coordonnées respectives (x_A, y_A) et (x_B, y_B).
Le coefficient directeur de la droite D est le réel m tel que :

$$m = \frac{y_B - y_A}{x_B - x_A}.$$

Source : https://euler.ac-versailles.fr/baseeuler/lexique/notion.jsp?id=89

Calculer la distance entre 2 points (pour la proximité avec un autre poisson ou la souris) :

Théorème
Si A et B sont deux points de coordonnées respectives $(x_A ; y_A)$ et $(x_B ; y_B)$, alors la distance AB des deux points A et B est donnée par :

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}.$$

Source : https://euler.ac-versailles.fr/baseeuler/lexique/notion.jsp?id=122