Jeu type Pacman

Dans ce cours nous allons voir comment écrire les bases d’un jeu dans un labyrinthe.

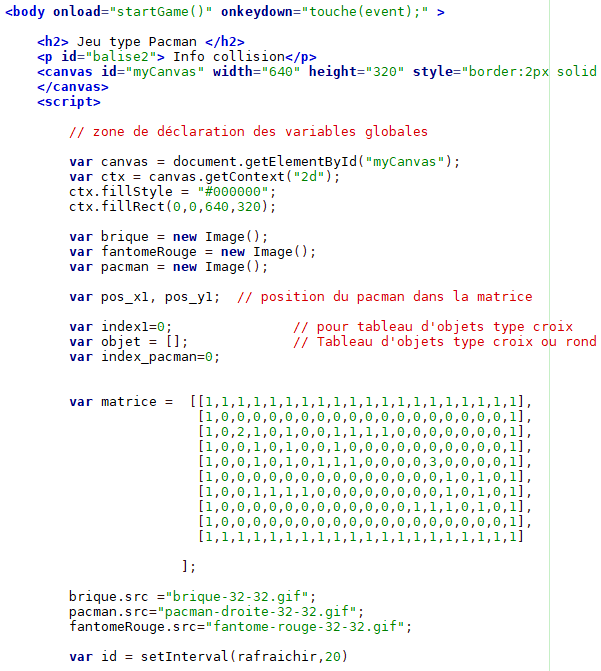
On va partir d’une trame de base pour le code qui est fortement inspiré de ce qui a été fait dans les cours précédents

Editer le fichier **pacman1.html** et faire une copie **pacman2.html** sur laquelle vous allez travailler. Vous devez aussi récupérer les fichiers images associés.

Ressources graphiques gratuites téléchargeables directement :

<https://www.gameart2d.com>

**Etape 1 : création du labyrinthe et des objets graphiques (pacman, fantôme, brique …)**



Taille du canevas 640\*320

On remplit le canevas avec un rectangle noir pour le fond.

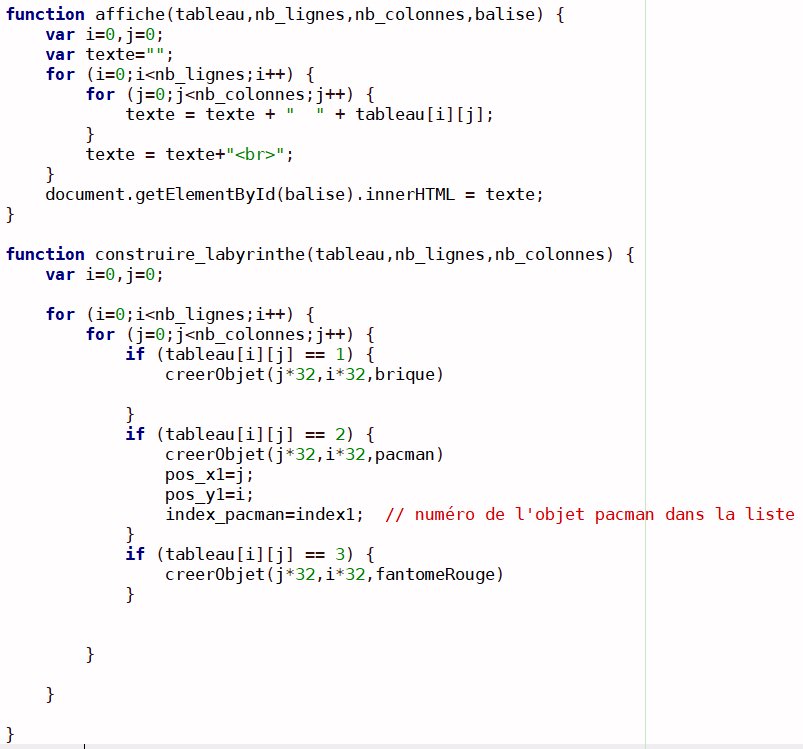
Déclaration des objets images pour les murs, le fantôme et pacman

Liste qui contient tous les objets du jeu

Matrice du jeu . Chaque nombre correspond à un élément graphique (mur, pacman, fantôme …)

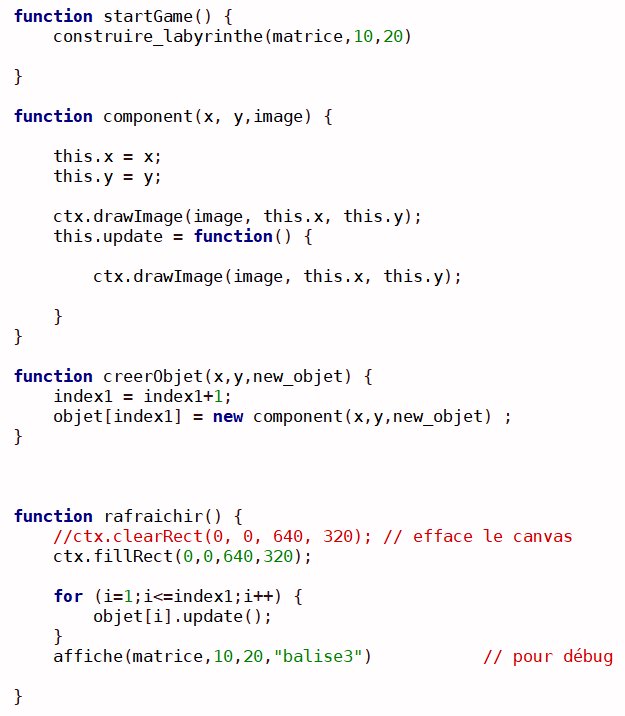
Chargement des fichiers images dans les objets images.

« Rafraichir » est une fonction qui est appelée toutes les 20ms



Affiche le tableau représentant le labyrinthe en mémoire (pour debug)

Affiche les éléments graphiques en fonction des données du tableau en mémoire.  
Suivant la valeur de la case (1,2 3 ..) on va afficher une brique, pacman ou un fantôme. Chaque élément graphique correspond à un nombre en mémoire



Initialisation au démarrage

Fonction qui permet de créer un objet ou d’actualiser sa position

Chaque fois que l’on crée un objet, index1 s’incrémente (c’est le numéro de l’objet dans la table « objet »)

Fonction qui met à jour la position de tous les objets de la table objet.

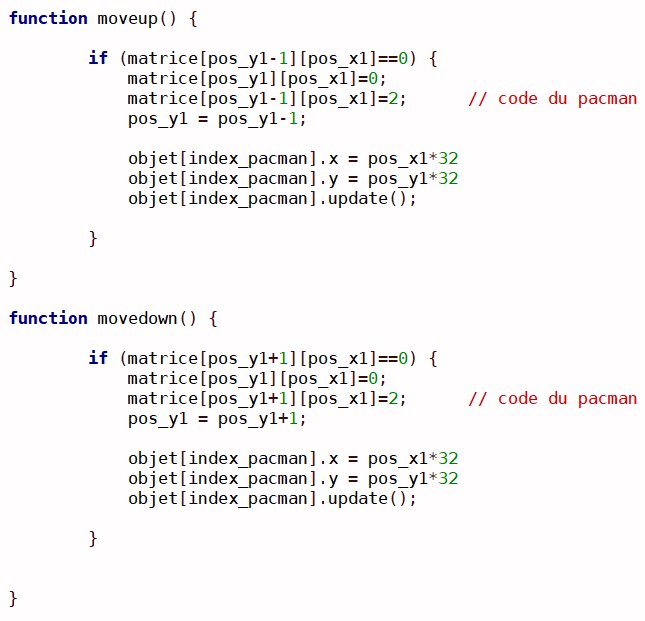
On teste si position au dessus est libre

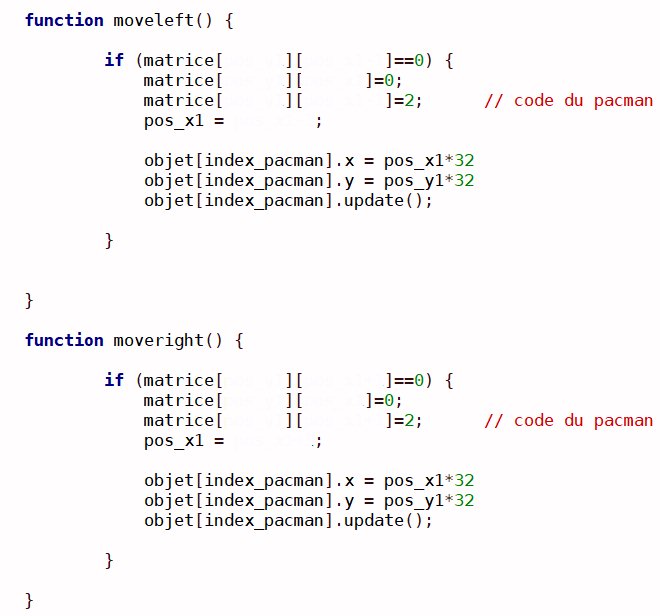
On efface la position actuelle

On place le 2 dans la nouvelle case

Mise à jour de la coordonnée

Mise à jour des éléments graphique





En vous inspirant du code pour monter/descendre, **compléter** le code pour le déplacement à gauche et à droite du pacman.

**Tester** votre code et faire **valider** par le professeur

**VALIDATION :**

**A quoi** correspond la valeur 32 ?

**Réponse :**

**OMJS**

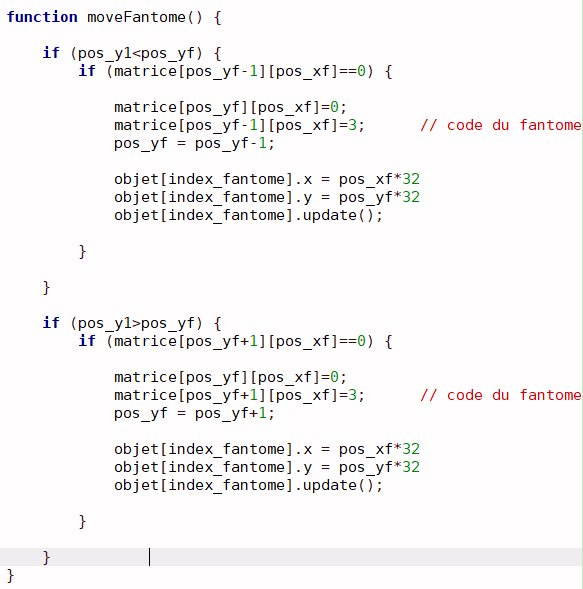
**Etape 2 : déplacement automatique du fantôme.**

A partir de cette étape, le codage relève davantage de l’algorithmique que de la connaissance du langage javascript. Nous allons explorer quelques pistes :

1- Pour que le fantôme puisse se déplacer seul, il faut qu’il y ait une fonction deplacement qui soit appelée automatiquement toutes les X millisecondes

2- Le fantome doit avoir une stratégie. La plus basique est qu’il tente de se diriger vers le pacman . Pour cela, il faudra comparer les coordonnées (x,y) du fantome avec ceux du Pacman pour déterminer le sens du déplacement.

3- Enfin, il faut autoriser le déplacement du fantôme uniquement si pas de mur.



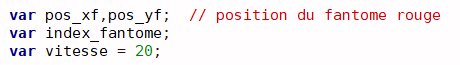
Test si pacman est au dessus du fantome

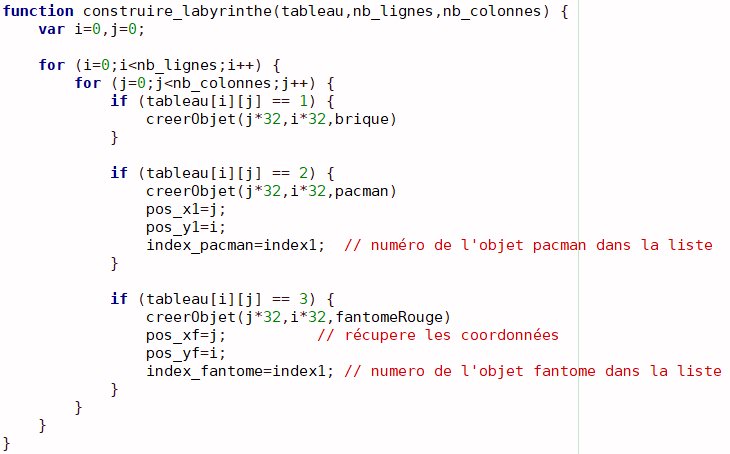
Test si déplacement possible

Test si pacman est en dessous du fantome

Test déplacement possible

Compléter la déclaration des variables





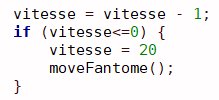
Compléter le code associé au fantome

**²**

Rajouter ce code dans la fonction « rafraichir » afin d’appeler la fonction moveFantome .  
Pouvez vous dire toutes les combien de millisecondes moveFantome() sera appelé ?

(la réponse n’est pas 20 …)

**Réponse :**



**Faire valider** le fonctionnement par le professeur (le fantome doit se déplacer vers le haut ou le bas en fonction de la position du pacman) :

**Validation :**

En vous inspirant de ce que vous venez de faire, **compléter la fonction moveFantome()** afin que le fantôme se déplace vers le pacman sur l’axe X

Attention : le déplacement en diagonale n’est pas autorisé ….

Faire valider par le professeur :

**Validation :**

**Rajouter un deuxième fantôme.**

Proposer une autre stratégie de déplacement ou bien une amélioration de la stratégie actuelle (car le fantôme peut rester bloquer dans un angle du labyrinthe )

OMJS