Projet Moteur de jeux

Anton Sokolowsky et Nouh Master

Sommaire :

1. Présentation :

Le jeu est un shoot them up où l'on incarne un vaisseau qui doit essayer de finir les différents niveau sans mourir en tuant les ennemies qui se présentent à lui.

L'univers du jeu se déroule dans l'espace.

Le joueur dispose de 10000 points de vie.
Dans cette version, le code à été fait pour que le vaisseau puisse se dirigé tout seul et tirer tout seul.

Dans cette version du code il y a 7 enemies différents avec 3 niveaux.

Il nous semble important de noté que le jeu fonctionne mieux, (plus fluide) sur chromium que sur firefox.

Pour jouer appuyer sur entrer.

Si vous voulez voir l'écran de victoire mettez 10000 points de vie.
Si vous voulez voir l'écran de défaite mettez 100 points de vie.

ligne 722 :

```
let Joueur = new Player(cnv.width / 2 - 50, cnv.height- 150, 10000);
```

2. Structures de données utilisées :

Dans ce projet on utilise des classes et on gère les différents objects avec des tableaux et des listes chainées.

Pour les décors qu'on voit défilé,
on utilise une classe pour déclarer chaque objet du décors puis on utilise un tableau pour chaque plan dans lequel on push les différents objets.
On utilise plusieur tableau pour pouvoir draw les objets en fonction des plans.
Si on avait mis tout les objets dans un seul tableau on aurait des objets du plan 1 qui pourrait passer derrière des objets du plan 2 par erreur.

Pour les poussières blanches qu'on voit défilé tout au long du jeu, c'est un tableau avec 3 indice, un indice pour pour son x, un pour son y, un pour sa vitesse en y.

Pour les enemies, on utilise une classe enemies, elle est décrite plus précisément dans le code, on gère les ennemis dans une liste chainée.

Pour les projectiles (du joueur ou ennemie), on utilise une classe projectiles, elle est décrite plus précisément dans le code, on gère les projectiles dans une liste chainée.

3. Explications des fonctions les plus importantes :

Fonction draw() :

Cette fonction sert à afficher toutes images qu'on peut voir dans le canvas,
C'est ici qu'on décide de quel plan passe devant un autre , on dessine au fur et à mesure qu'on descend dans la fonction des images les une par dessus les autres, en premier on dessine le fond du niveau. Une grande image, puis on dessine les différents plan du décors, en premier la plan 4 puis le 3 puis le 2 puis le 1 puis les projectiles, puis on affiche les ennemie, puis le joueur, ainsi le joueur est au premier plan.

Fonction update_pos() :

Dans cette fonction on actualise la positions des objets en fonction de leurs vitesse x et y, ici le sens dans lequel on actualise les positions n'importe pas.

Fonction update_pos_enemies(liste) :

Dans cette fonction on actualise la position des ennemies en itérant sur tous les objets de la liste passé en paramètre, en réalité on aurait pu de pas spécifié la liste passé en paramètre, puisqu'on sait qu'on actualise la position des ennemies.

On actualise les positions des enemies en fonction de leurs vitesse et on actualise les coordonnées de leurs surface de collisions en fonction de leurs vitesse.

fonction update_pos_projectiles(liste) :

Dans cette fonction on actualise les coordonnées des objets passé en paramètres, en fonction de leurs vitesse. mais on n'a pas besoin d'actualiser la surface de collisions car les projectiles n'en on pas, ils ne collisionnent les objets que par leurs point coordonnées x,y. Ce choix à été fait pour minimisé les calculs de collisions.

Fonction test_collisions_projectiles(liste_colliders,list_collided) :

List_colliders représente les projectiles, et list_collided représente les enemies,
On va itéré sur chaque élément de la liste List_colliders, et on va prendre comme points de collisions les coordonnées de l'élément de List_colliders (qui est un projectile), puis pour chaque projectile on va itéré sur tous les éléments de la liste List_collided (les ennemies), et on va tester si le points à tester (coordonnées du projectile) est dans la surface de collision d'un des éléments de la liste List_collided, en gros ; si un projectile touche un ennemie.

Fonction test_collisions_joueur(liste) :

Dans cette fonction la liste passé en paramètre est la liste des projectiles ennemies,
On va tester pour chaque projectile, si il est dans la surface de collision du joueur.

Fonction test_collisions_joueur_enemies(liste) :

Dans cette fonction, on teste si les ennemies du type 'Tournant' entrent en collisions avec le joueur.

On va tester pour tous les points de l'ennemie si un seul entre dans la surface du joueur, si oui il y a collisions.

Fonction joueur_cherche_enemie() :

Dans cette fonction on va chercher l'ennemie le plus bas, une fois trouvé le vaisseau du joueur va se dirigé dessus et on va activé le tir. Cependant on test aussi que l'ennemie ne soit pas trop bas, pour que le vaisseau ne se dirige que vers des ennemie qui sont devant lui.

4. Paramétrabilité :

On a essayé de rendre le projet le plus paramétrable possible,
On créé des objets préfabriqué sous la forme de variable globale.

Ainsi on a créé :
5 objets du type projectile pour les ennemies.
2 objets du type projectile pour le joueur.
7 objets du type enemy.

Ainsi il est très simple de fabriqué d'autre type de projectile pour les ennemies et le joueur, et d'autre ennemies.

On a utilisé la même logique pour les Levels et les waves.

Pour créer une wave il suffit de mettre dans le constructeur d'une wave,le nombre d'ennemis et un tableau de différents type d'ennemie.

Pour créer un level il suffit de mettre dans le constructeur un tableau de wave, puis une image, la taille de l'image, puis une image indiquant le numéro du niveau.

puis dans game de mettre un tableau avec tous les levels créé.

5. Animation des explosions et des ennemies tournants :

Les animation des explosions se trouve dans un tableau d'image dans lequel chaque images a été découpée depuis un sprite sheet selon la largeur et la longue et le nombre d'image de celui-ci ,avec la fonction vue en cours img.onload.
On utilise les images du tableau dans la fonction explosion() qui prend en paramètre un points(x,y) ,dans laquelle on utilise un set-interval qui affiche chaque sprite a la position x et y de l'explosion ,elle s'affiche une a une en parcourant le tableau et lorsque celle-ci a atteint la fin du tableau je stop mon set interval. On applique la fonction lorsque qu'il y a une collision.

Pour faire tourner les images on enregistre la canvas actuel , on change l'origine de l'endroit ou l'on va faire tourner la canvas en lui mettant le pts x et y de l'image, je tourne ma canvas selon le degré que j'ai ,je draw l'image tourné , et ensuite je restaure la canvas que j'ai sauvegardé.
Points a améliorer :
Ajouter des animation sur le vaisseau ,
Faire des ennemies moins linéaire ,
Avoir plus de Sprite diffèrent pour faire plus d'animation ,

6. Points à améliorer :

On s'est rendu compte en commentant le code que on aurait pu implémenter le tous de manière encore plus paramétrable et optimisé.
On aurait du faire une class object globale, avec un constructeur contenant les coordonnées, le sprite, la taille de l'image, sa vitesse en x sa vitesse en y, et ensuite dérivé chaque classe d'objet de cette classe.

Avec cette méthode, on aurait plus qu'à faire une seule liste chainée avec tous les objets, une seule fonction update_position, une seule fonction test_collision, une seule fonction draw.