LE Kim Yen

1. Objectif de l'application et comment utiliser

Le but de mon code est de créer une scène d'animation avec différents objets à l'écran et un simple jeu de pêche .

C'est un jeu où le joueur contrôle un filet de pêche pour attraper des poissons qui se déplacent sur l'écran. Le joueur utilise les touches fléchées pour toucher le poisson, lorsque le poisson est touché, il disparaîtra et si les 5 poissons disparaissent, le joueur gagne.

2. EXPLICATION MON CODE

Elle simule un environnement marin dans lequel se trouve un bateau, des poissons, des coraux, des bulles d'air et des étoiles dans le ciel et une lune, à l'intérieur avec les balles se déplaçant selon une simulation de la physique des bulles.

Les objets de la scène sont définis par des structures, notamment "sea" (représentant la mer), "sky" (représentant le ciel avec des étoiles), "ship" (représentant les navires), "fish" (représentant le poisson), " net" (représentant un filet de pêche), "moon" (représentant la lune), "ball" (représentent les balles à l'intérieur de la lune), "bulle" (représentant des bulles d'air) et "coral" (représentant deux coraux).

La mer est simplement dessinée avec un rectangle de coordonnées (0,0) à (500,200).

Le corail : deux images de corail ajoutées pour décorer la fenêtre.

Le ciel avec des étoiles

- *init_Sky ():* cette fonction initialise une structure de ciel en fixant le nombre d'étoiles ainsi que leurs positions et couleurs. La structure du ciel contient un réseau d'étoiles, où chaque étoile a une position et une couleur. La fonction prend deux paramètres : une structure de ciel et un entier représentant le nombre d'étoiles à générer.

Pour chaque étoile, il génère une position aléatoire à l'aide de la fonction *rand()* et générez une nouvelle couleur pour chaque étoile dans le ciel en utilisant la fonction *rand()* qui génère une couleur aléatoire.

Le bateau

Une partie du code qui définit une structure de navire qui représente un bateau.

- *init_Ship() :* initialise la position et du moteur à l'intérieur des deux cercles.
- *draw_Ship() :* dessine ensuite un navire avec un rectangle et deux cercles représentent les deux moteurs du bateau, avec une couleur aléatoire pour chaque moteur. Les deux lignes tournantes continues représentent le bateau en fonctionnement.
- *update_ShipBulle() :* met à jour la position du navire et les angles des deux lignes tournantes dans deux moteurs indiquant que le bateau démarre. La variable "*ecart*" détermine l'amplitude de la variation angulaire, tandis que la variable "*vitesse*" détermine la fréquence de cette variation. La rotation du moteur est modifiée en additionnant le produit de la déviation et du sinus de la vitesse multiplié par le temps écoulé depuis la dernière mise à jour.

La lune avec des bulles mobiles à l'intérieur :

- *ball ball_Init()* : cette fonction initialise l'objet balle avec la position, la vitesse, la masse et la force.
- *void addforce( )* : cette fonction ajoute une force à l'objet balle .
- *void draw_moon ()* : cette fonction dessine un objet lune à l'écran.
- *void update_moon()* : cette fonction met à jour la position et la vitesse et gère les collisions entre elles. Fonction de contrôle de distance, si la balle s'étend au-delà de la zone de la lune, la fonction appliquera les réflexions appropriées et ajustera la position et la vitesse de la balle en conséquence.

Les bulles : une simulation de bulles d'air

- *init_bulle()* : initialise les positions et les couleurs de chaque bulle dans le monde en utilisant des valeurs aléatoires. Le paramètre *nbbulle* spécifie le nombre de bulles à créer.
- *Update_bulle()* : Dans la fonction principale, le code affiche chaque bulle à l'écran en dessinant un cercle rempli à la position de la bulle et avec la

couleur de la bulle. Puis, la fonction met à jour la position de chaque bulle en ajoutant une valeur fixe à la coordonnée y de chaque position. Si une bulle atteint le bord supérieur *(pos.y>200)* , elle va réinitialisée en bas de la fenêtre *(pos.y = 2).*

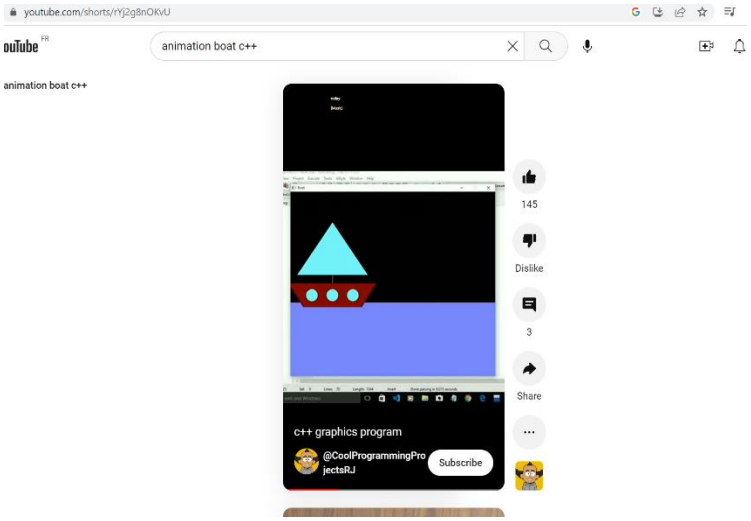
Les poissons avec le jeu de pêche :

Une structure de données pour les poissons avec leur position, une image, un rayon et le nombre total de poissons.

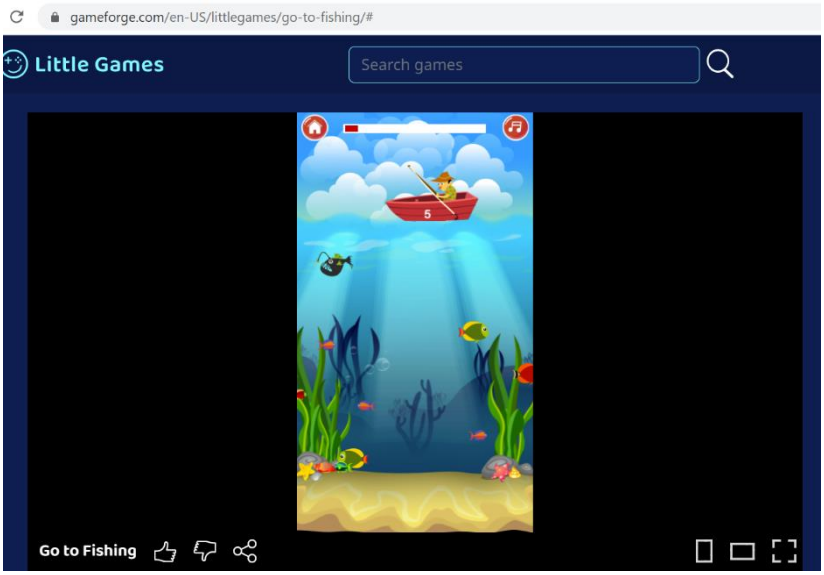
- *Init_fishNet () :* initialise les données des poissons et d'un filet des positions et des images spécifiques.
- *Update_FishNet () :* prend en entrée deux références d'objets : *fish, net* et une référence de booléen *iswin*. Elle met à jour la position des poissons et du filet dans le jeu. La fonction commence par déplacer le filet en fonction des les touches fléchées du clavier pressé par le joueur. Ensuite, elle parcourt tous les poissons et les déplace en fonction de leur vitesse et de leur direction. Si un poissons atteint le bord de la fenêtre, il réapparaît de l'autre côté. Enfin, elle calcule la distance entre chaque poisson et le filet, si cette distance est inférieure à la somme de leurs rayons, le poisson disparaîtra (son rayon est mis à 0). Si tous les poissons disparaissent, le booléen *iswin* est mis à vrai. Et il va changer la couleur de la fenêtre et afficher You Win !

3. Des références vers des documents

Mon idée est née après avoir regardé une courte vidéo sur YouTube sur le mouvement d'un navire comme indiqué ci-dessous :



Jouez et voyez comment fonctionnent certains jeux de pêche en ligne:



4. L'historique des évolutions des 3 dernières semaines

   Je réfléchis à l'idée du projet depuis la mi-mars, j'ai donc commencé à travailler sur mon idée avant celle qui était de dessiner des objets sur la fenêtre et de faire quelques mouvements simples pour les objets. La semaine dernière avant de déposer la version finale, j'ai eu l'idée que je voulais que mon projet soit un simple jeu de pêche où les utilisateurs pourraient interagir avec le jeu. Donc en une semaine je le perfectionne et cette semaine, j'ai juste besoin de revoir et modifier le code.