LIFAMI – TD / TP : Evolution de la couleur des insectes cherchant à se camoufler

Objectifs : Révision sur les structures, boucles, etc.
Savoir écrire un enchainement de plusieurs fonctions dans un but applicatif
Un exemple d'application inspirée de la biologique

Notre objectif est de programmer un mini jeu où l'ordinateur « apprend » un camouflage pour une population d'insectes se faisant dévorer par votre souris. La phase d'apprentissage est une simulation de l'évolution génétique des gènes de couleur que pourrait faire une espèce d'insectes voulant maximiser ses chances de survie.

Initialement (gauche) les insectes ont toutes les couleurs possibles (il y a un cercle de couleur pour chaque insecte sur l'image). Après plusieurs itérations (droite) les meilleurs insectes qui ne se sont pas fait mangés rapidement sont gardés pour construire la population suivante. La population a des couleurs vertes/jaunes qui offre un meilleur camouflage.

1. Un insecte est représenté par une position (x,y), une couleur (r,g,b) et une durée de vie. Une population d'insectes est représentée par un tableau de NB_INSECTS insectes et une image de fond représentant le paysage dans lequel les insectes vivent.

   Déclarez ces deux structures.

2. Ecrivez les deux procédures suivantes d'initialisation du monde des insectes.

   ```
   void initInsect(SomeInsects& si, Color& good, int range)
   ```
   →Initialise les insectes. Leur position est choisie au hasard. Leur couleur sera choisi au hasard dans un rayon *range* autour de la couleur *good*. Le champ de la durée de vie est initialisé à -1 : un chiffre négatif signifie que l'insecte est toujours vivant, un positif indique combien de temps il a vécu.

   ```
   void init(SomeInsects& si)
   ```
   →Initialise l'image du paysage et appelle la procédure qui initialise les insectes

3. Ecrivez la procédure *draw* qui prend en paramètre la population d'insectes
    a. Affichez les insectes et l'image de paysage. Chaque insecte est un cercle plein de rayon 3 avec la couleur stockée dans la structure.
    b. Ajoutez les instructions pour que les insectes dans un rayon de 20 pixels de la souris deviennent morts. La souris est le prédateur. Un insecte mort n'est plus visible et aura son champ de durée de vie qui contiendra la durée qu'il a vécu avant de se faire manger par la souris. Utilisez :
    `ElapsedTime()` qui renvoie la durée depuis le lancement du programme

4. Une fois tous les insectes morts nous allons garder les insectes les mieux adaptés à leur environnement, et sélectionner leur couleur pour régénérer une population.
    a. Ecrivez la procédure *minMaxLifeTime* qui trouve la durée de vie minimale et la durée de vie maximale dans une population d'insectes.
    b. Ecrivez la fonction *averageColorOfGoodInsects* qui calcule la couleur moyenne des insectes dont la durée de vie a été supérieure à une certaine durée.
    c. Dans la procédure *draw* ajoutez du code qui régénère une nouvelle population avec une couleur mieux adaptée calculée par les questions a. et b.

De nombreuses améliorations sont possibles …

Annexe

L'algorithme principal pourra ressembler à ceci :

```
int main(int , char** )
{
    bool stop=false;

    winInit("Interpolation !", DIMW, DIMW);
    backgroundColor( 240, 230, 255 );

    Menu menu;
    menu_add( menu, "Init");
    menu_add( menu, "Run");
    menu_setSelect( menu, 1);

    SomeInsects li;
    init(li);

    while( !stop )
    {
        winClear();
        switch(menu_select(menu))
        {
            case 0 : init(li); menu_setSelect(menu, 2); break;
            default: break;
        }
        draw(li);
        menu_draw( menu );
        stop = winDisplay();
    }
    winQuit();
    return 0;
}
```