TD-TP Allegro 1 – Base, Primitives de dessin, animation basique

OBJECTIFS A ATTEINDRE :

A l'issue du TDTP 1 vous devez être capable :
1) de faire un projet allegro sous CodeBlocks correctement linké soit en mode dynamique (.exe léger), soit en mode static (.exe plus lourd mais pas besoin de DLL)
2) de faire le premier programme basique qui affiche un message via la fonction allegro_message()
3) d'entrer dans le mode graphique de votre choix avec l'utilisation de la fonction set_gfx_mode()
4) de faire une boucle d'événements en utilisant le clavier selon une approche de votre choix (scancode et/ou buffer), lorsque l'utilisateur appuie sur des touches il déclenche des affichages de dessin avec les primitives de dessin.

Exercice 1
Initialisation mode graphique, boucle d'événements

Après initialisation de la librairie (fonction allegro_init()) sélectionner un mode graphique avec les fonctions set_color_depth(...) et set_gfx_mode(...), prévoir un contrôle d'erreur.
Afficher les valeurs contenues dans les variables SCREEN_W et SCREEN_H avec une des fonctions du type textprintf_ex(…) de la librairie.
Faire une boucle dont l'utilisateur sort avec la touche clavier échap. Afficher les coordonnées de la souris.
Tous les affichages se font directement sur l'écran graphique (1er paramètre : "screen")

Exercice 2
Tester les primitives de dessin

Faire au moins un programme qui mette en scène plusieurs fonctions de dessin d'allegro. Par exemple "bombardement" de points, de lignes horizontales, verticales, de rectangles etc. Le tout est contrôlé par l'appui sur différentes touches qui permettent l'appel des différentes fonctions créées (bombarder points, bombarder lignes, etc). L'affichage se fait toujours directement à l'écran sur la bitmap de mémoire vidéo « screen »

Fonctions disponibles (voir doc. en ligne → https://liballeg.org/stabledocs/en/alleg013.html) :

```
void vline(BITMAP *bmp, int x, int y1, int y2, int color);
void hline(BITMAP *bmp, int x1, int y, int x2, int color);
void line(BITMAP *bmp, int x1, int y1, int x2, int y2, int color);

void rect(BITMAP *bmp, int x1, int y1, int x2, int y2, int color);
void rectfill(BITMAP *bmp, int x1, int y1, int x2, int y2, int color);
void triangle(BITMAP *bmp, int x1, y1, x2, y2, x3, y3, int color);
void polygon(BITMAP *bmp, int vertices, const int *points, int color);

void circle(BITMAP *bmp, int x, int y, int radius, int color);
void circlefill(BITMAP *bmp, int x, int y, int radius, int color);
void ellipse(BITMAP *bmp, int x, int y, int rx, int ry, int color);
void ellipsefill(BITMAP *bmp, int x, int y, int rx, int ry, int color);
void arc(BITMAP *bmp, int x, y, fixed ang1, ang2, int r, int color);

void floodfill(BITMAP *bmp, int x, int y, int color);
```

Exercice 3
Faire une animation simple directement à l'écran : économiseur d'écran

Par exemple (simple) un rectangle qui rebondit sur les bords de l'écran. Eventuellement ce rectangle laisse une trace arc-en-ciel ou en dégradés de couleurs. Possibilité d'ajouter un autre graphisme piloté par le clavier...

Autre exemple (plus compliqué), réaliser l'animation d'une courbe de Bézier dont les points de contrôle sont en perpétuel mouvement. L'affichage et le calcul de la courbe sont faits avec la fonction de la librairie :

void spline(BITMAP *bmp, const int points[8], int color);

Cette fonction trace sur la bitmap « bmp » (ici nous passerons « screen ») une courbe de couleur « color » à partir des coordonnées des quatre points du tableau « points » (x1,y1,x2,y2,x3,y3,x4,y4)

Exercice 4 (facultatif, entrainement à la géométrie fractale)
Visualisation de la récursivité : arbre d'appels

```
// sous-programme récursif...
void branches(int x,int y,int largeur,int hauteur,int epaisseur)
{
    int i;
    if (epaisseur>0)
    {
        for (i=0;i<epaisseur;i++){
            line(screen,x+i,y,x+i+largeur,y-hauteur,makecol(255,255,255));
            line(screen,x+i,y,x+i-largeur,y-hauteur,makecol(255,255,255));
        }
        branches(x+largeur,y-hauteur,largeur/2,hauteur/2,epaisseur/2);
        branches(x-largeur,y-hauteur,largeur/2,hauteur/2,epaisseur/2);
    }
}

// exemple d'appel au niveau du main :
branches(400,600,192,192,64);
```

Le code précédent est un exemple de procédure récursive affichant un arbre : chaque appel dessine deux branches et relance 2 appels à l'extrémité de ces deux branches etc... jusqu'à atteindre les branches les plus fines.
En vous inspirant de ce code pouvez-vous obtenir un arbre d'aspect plus naturel ?
Pensez au hasard pour donner de l'irrégularité géométrique ou dans le facteur de branchement, des couleurs, des feuilles, un tronc à la base...

Exercice 5 (facultatif, à titre indicatif)
Possibilité d'afficher des infos

L'appel de la fonction allegro_init() entre autre initialise différentes variables qui donnent des informations. L'ensemble de ces informations est présenté dans la documentation d'Allegro au chapitre « Using allegro ».
En utilisant un projet en mode console correctement linké et après initialisation allegro, afficher quelques informations avec des printf(). Par exemple :

```
extern char allegro_id[];        // contient date et numéro version de la librairie
extern int os_type;              // prend une valeur qui identifie un système (voir documentation pour décodage)
extern int os_version;           // numéro version du système (exemple  Win98 donne 4)
extern int os_revision;          // numéro révision du système (exemple Win98 SE donne 4.10)
extern int os_multitasking;      // si multitâche ou pas
extern char cpu_vendor[];        // marque du processeur si possible
extern int cpu_family;           // type du processeur 3=386, 4=486, 5=Pentium, 6=PPro, etc.
```

Par ailleurs les fonctions :

```
int desktop_color_depth();      // donne le mode couleur (8,16,24,32 bits) du bureau
int get_desktop_resolution(int *width, int *height);       // donne la résolution actuelle  pour le bureau
```