LIFAP1 – TP7 : Les structures avec GRAPIC

Objectifs : Création d'un nouveau projet sous Grapic
Définition d'une structure image
Opérations de base sur les images / manipulation des structures
Utilisation des menus dans Grapic

1. Création d'un nouveau projet dans Grapic

Pouvez créer votre propre projet dans Grapic. Pour cela, suivez les étapes suivantes :

- Dupliquez le dossier `apps/start` dans le dossier `apps/MYPROJECT` (en remplaçant `MYPROJECT` par le nom que vous souhaitez)
- Renommez `apps/MYPROJECT/main_start.cpp` en `main_MYPROJECT.cpp`
- Editez, par exemple à l'aide du bloc-notes, le fichier `Grapic/premake4.lua` et ajoutez à la fin la commande
  `make_project("MYPROJECT","apps/MYPROJECT/main_MYPROJECT.cpp")`
- Exécutez le script `premake.bat` sous Windows (double-cliquez dessus), ou bien `premake.sh` sous Linux ou MacOS.

En ouvrant à présent le projet `grapic.workspace` vous devriez voir apparaitre votre nouveau projet dans la partie gauche de la fenêtre. Cliquez (bouton droit) sur le projet `MYPROJECT` que vous venez de créer et et choisissez `Activate project`.
Editez enfin le fichier `main_MYPROJECT.cpp` et commencez le TP.

Dans ce TP, nous souhaitons mettre en place certaines fonctionnalités permettant de faire du traitement d'images, comme les images de la page suivante le montre. Il faut donc dans un premier temps définir les différentes structures de données permettant de manipuler et de charger les images à traiter.

2. Définition des différentes structures de données et chargement des images à traiter
   a. Récupérez sur le site LIFAP1 (TP7) l'archive contenant les images à utiliser et décompressez-là dans le dossier `data` de grapic.
   b. Définissez les constantes suivantes : `DIMW` (taille de la fenêtre d'exécution grapic), `MAX_X` et `MAX_Y` (dimensions maximales d'une image donc <= `DIMW`) et `MAXCHAR` (taille maximale des chaines de caractères).
   c. Définissez enfin la structure image qui sera utilisée. Elle contiendra les champs suivants : `taille_x`, `taille_y` de l'image et un tableau 2D de couleurs représentant chacun des pixels de l'image (le type utilisé pour définir chaque couleur sera `unsigned char` pour des besoins de programmation ultérieurs)

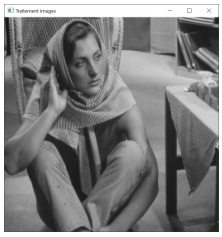
3. Implémentation des opérations de base sur les images. La librairie Grapic comporte déjà un certain nombre de fonctions / procédures permettant de manipuler des images que vous pourrez retrouver en intégralité dans la documentation disponible sur le site. Pour les exercices demandés, vous aurez essentiellement besoin des sous-programmes suivants :

- `void` **`image_draw`** `(Image &im, int x, int y, int w=-1, int h=-1)`
  avec im l'image à afficher, x et y les coordonnées du point inférieur gauche, w et h la taille de l'image.
- `unsigned char` **`image_get`** `(const Image &im, int x, int y, int c=0)`
  retourne la couleur du pixel aux coordonnées (x,y) de l'image im
- `void` **`image_set`** `(Image &im, int x, int y, unsigned char r, unsigned char g, unsigned b, unsigned char a)`
  change la couleur du pixel aux coordonnées (x,y) de l'image im avec la couleur c.
- `void` **`put_pixel`** `(int x, int y, unsigned char r, unsigned char g, unsigned char b, unsigned char a=255)`
  dessine une ligne entre les points de coordonnées (x1,y1) et (x2,y2).
- `Image lena =` **`image`**`("data/barbara.jpg")`
  permet de charger une image du dossier *data* nommée *barbara.jpg* dans une structure image de grapic

a. Ecrivez un sous-programme permettant de remplir la structure image avec les différentes caractéristiques (taille et couleur) d'une image récupérée dans le dossier data. Pour chacun des pixels de l'image, récupérez son intensité lumineuse et écrivez cette valeur dans la structure.
b. Ecrivez un sous-programme permettant d'afficher dans la fenêtre d'exécution de Grapic une image contenue dans la structure.
c. Ecrivez le programme principal permettant de récupérer une image du dossier data, de la charger dans notre structure et de visualiser le résultat obtenu

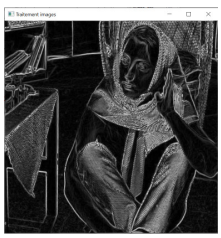
4. Implémentation des opérations de traitement des images

a. Ecrivez une fonction de **seuillage de l'image**. L'opération dite de "seuillage simple" consiste à mettre à zéro tous les pixels ayant un niveau de gris inférieur à une certaine valeur (appelée seuil, en anglais *threshold*) et à mettre à la valeur maximale les pixels ayant une valeur supérieure. Ainsi le résultat du seuillage est une image binaire contenant des pixels noirs et blancs, c'est la raison pour laquelle le terme de binarisation est parfois employé.



b. Ecrivez une fonction permettant d'effectuer une symétrie verticale (ou **miroir**) de l'image d'origine. Afin de conserver l'image d'origine, on construira une nouvelle image que l'on affichera ensuite dans l'interface de Grapic.

c. Dans une image en niveaux de gris, un contour est caractérisé par un changement brutal de la valeur. Le but de l'opération de **détection de contours** (ou filtrage) est de transformer cette image en une autre dans laquelle les contours apparaissent par convention en blanc sur fond noir. Parmi les filtres existants, nous allons implémenter le filtre de Prewitt défini de la manière suivante :



-1	0	+1
-1	0	+1
-1	0	+1

Gx

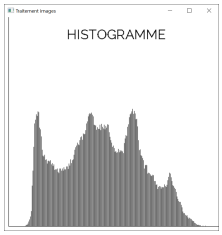
+1	+1	+1
0	0	0
-1	-1	-1

Gy

$|G| = \sqrt{Gx^2 + Gy^2}$ $(|G| = |Gx| + |Gy|)$

Pour chaque pixel de l'image, on calcule la nouvelle intensité lumineuse en fonction de ses 8 voisins (attention aux pixels des bords de l'image) selon les coordonnées x puis y. Puis, à partir des deux valeurs obtenues, on calcule la norme afin d'obtenir la nouvelle intensité lumineuse.

d. Un **histogramme** est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant du plus foncé (à gauche) au plus clair (à droite).



Ainsi, l'histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisses, et le nombre de pixels de l'image en ordonnées. Ecrivez la fonction permettant de dessiner l'histogramme de l'image. Il faudra compter le nombre de pixels de chaque niveau de gris puis dessiner une ligne dont la longueur sera fonction de cette valeur.

5. Complétez le programme principal avec l'affichage d'un menu proposant les différentes opérations.

```
#include <Grapic.h>

using namespace grapic;

const int DIMW = 512;
const int MAX_X = 512;
const int MAX_Y = 512;
const int MASQUE = 9 ;
const int MAXCHAR = 128;

struct Data
{
   int taille_x, taille_y;
   unsigned char pixel[MAX_X][MAX_Y];
};

void draw (Data d)
{
   int i, j ;
      for (i=0; i<d.taille_x; i++)
   {
      for (j=0 ; j<d.taille_y ; j++)
      {
         put_pixel(i,j,d.pixel[i][j],d.pixel[i][j],d.pixel[i][j]);
      }
   }
}
void init(Data& d, Image lena)
{
   int i,j;
   d.taille_x = DIMW;
   d.taille_y = DIMW;
   for (i=0; i<d.taille_x; i++)
   {
      for (j=0 ; j<d.taille_y ; j++)
      {
```

```
            d.pixel[i][j]= int(image_get(lena,i,j,0) );
        }
    }
}

void draw_init(Data &d)
{
    int i,j;
    for (i=0; i<d.taille_x; i++)
    {
        for (j=0 ; j<d.taille_y ; j++)
        {
            put_pixel(i,j,d.pixel[i][j],d.pixel[i][j],d.pixel[i][j]);
        }
    }
}


void seuillage(Data &d)
{
    int i,j;
    for (i=0; i<d.taille_x; i++)
    {
        for (j=0 ; j<d.taille_y ; j++)
        {
            if (d.pixel[i][j]>=128) d.pixel[i][j]=255;
            else d.pixel[i][j]=0;

        }
    }
}

void miroir(Data &d)
{
    int i,j,temp;
    for (i=0; i<d.taille_x/2; i++)
    {
        for (j=0 ; j<d.taille_y; j++)
        {
            temp = d.pixel[i][j] ;
            d.pixel[i][j] = d.pixel[d.taille_x-i-1][j] ;
            d.pixel[d.taille_x-i-1][j] = temp ;
            //put_pixel(i,j,d.pixel[i][j],d.pixel[i][j],d.pixel[i][j]);
            //put_pixel(d.taille_x-i-1,j,d.pixel[d.taille_x-i-1][j],d.pixel[d.taille_x-i-
1][j],d.pixel[d.taille_x-i-1][j]);

        }
    }
}

void contours(Data &d)
{
    int i,j,k,l;
    // unsigned char res[512][512] ;
    Data d2=d ;
    for (i=1; i<d.taille_x-1; i++)
    {
        for (j=1 ; j<d.taille_y - 1 ; j++)
        {
```

```
            k = -d.pixel[i-1][j-1]- d.pixel[i][j-1]-d.pixel[i+1][j-1]+ d.pixel[i-1][j+1]+
d.pixel[i][j+1]+ d.pixel[i+1][j+1];
            l =  d.pixel[i-1][j-1] + d.pixel[i-1][j]+d.pixel[i-1][j+1]- d.pixel[i+1][j-1]- d.pixel[i+1][j]-
d.pixel[i+1][j+1];
            d2.pixel[i][j] = sqrt(k*k + l*l);
            if (d2.pixel[i][j]>255) d2.pixel[i][j]=255;
            //put_pixel(i,j,d2.pixel[i][j],d2.pixel[i][j],d2.pixel[i][j]);
        }
    }
     d = d2  ;

}

void floutage(Data &d)
{
    int i,j,k,l;
    float ng;
    Data d2=d ;
    for (i=MASQUE/2; i<d.taille_x-MASQUE/2; i++)
    {
        for (j=MASQUE/2 ; j<d.taille_y - MASQUE / 2 ; j++)
        {

            ng = 0 ;
            for (k=i-MASQUE/2 ; k<= i +MASQUE/2 ; k++)
            {
                for (l=j-MASQUE/2 ; l<=j+MASQUE/2 ; l++)
                {
                    ng += d.pixel[k][l];

                }

            }
            d2.pixel[i][j] = ng / (MASQUE*MASQUE);
            //put_pixel(i,j,d2.pixel[i][j],d2.pixel[i][j],d2.pixel[i][j]);
        }
    }
     d = d2  ;

}

void warhol (Data &d)
{
    int i,j,k,l;
    float ng;
    Data d2=d ;
    for (i=1; i<d.taille_x-1; i++)
    {
        for (j=1 ; j<d.taille_y - 1 ; j++)
        {

           /* ng = 0 ;
            for (k=i-1 ; k<= i + 1 ; k++)
            {
                for (l=j-1 ; l<=j+1 ; l++)
                {
                    ng += d.pixel[k][l];

                }
```

```
            }
            ng /= 9;*/
            if (d.pixel[i][j] <=150)
                put_pixel(i,j,255,0,255);
            else
                put_pixel(i,j,255,255,0);
        }
    }
     d = d2  ;
}

void draw_histogramme (int xmin, int ymin, int xmax, int ymax, int nb[256], char
titre[MAXCHAR])
{
    int i,j;
    line(xmin,ymin,xmax,ymin);
    line(xmin,ymin,xmin,ymax);
    fontSize(32);
    print(150,450,titre);
    for (i=0; i<256; i++)
    {
        line(2*i + xmin,ymin,2*i + xmin, ymin+((ymax-ymin)*nb[i]*64)/(DIMW*DIMW));
    }

}

void histogramme (Data &d, int nb[256])
{
    int i,j,k,l;
   for (i=0; i<256; i++) nb[i]=0;

    for (i=0; i<d.taille_x; i++)
    {
        for (j=0 ; j<d.taille_y ; j++)
        {
            nb[d.pixel[i][j]]++;
        }
    }

}

int main(int argc, char** argv)
{
    Data d;
    Menu m;
    int gris[256]={0};
    bool stop=false;
    winInit("Traitement images", DIMW, DIMW);

   // Image lena = image("data/lena512.gif");
   // Image lena = image("data/hongkong.jpg");
    Image lena = image("data/barbara.jpg");
    init(d,lena);

    menu_add( m, "Initiale");
    menu_add( m, "Seuillage");
    menu_add( m, "Miroir");
    menu_add( m, "Contours");
    menu_add( m, "Floutage");
    menu_add( m, "Warhol");
```

```
    menu_add( m, "Histogramme");


    while(!stop)
    {
        winClear();

        switch(menu_select(m))
        {
        case 0 :
            init(d,lena);
            draw_init(d);
            break;
        case 1 :
            init(d,lena);
            seuillage(d);
            draw(d);
            break;
        case 2 :
            init(d,lena);
            miroir(d);
            draw(d);
            break;
        case 3 :
            init(d,lena);
            contours(d);
            draw(d);
            break;
        case 4 :
            init(d,lena);
            floutage(d);
            draw(d);
            break;
        case 5 :
            init(d,lena);
            warhol(d);
            break;
        case 6 :
            init(d,lena);
            histogramme(d, gris);
            draw_histogramme(10,10,512,512,gris,"HISTOGRAMME");
            break;
        default:
            draw_init(d);
            break;
        }
        menu_draw(m, 5,5, 100, 102);
        stop = winDisplay();
    }
    // pressSpace();
    winQuit();
    return 0;
}
```