LIFAP1 – TP11 et TP12 : Mémory version GrAPiC

Objectifs : Création d'un nouveau projet sous Grapic
Définition d'une structure image

Création d'un nouveau projet dans Grapic

Pouvez créer votre propre projet dans Grapic. Pour cela, suivez les étapes suivantes (http://liris.cnrs.fr/alexandre.meyer/grapic/html/index.html) :

- Dupliquez le dossier `apps/start` dans le dossier `apps/MYPROJ` (en remplaçant `MYPROJ` par le nom que vous souhaitez)
- Renommez `apps/MYPROJ/main_start.cpp` en `main_MYPROJ.cpp`
- Editez, par exemple à l'aide du bloc-notes, le fichier `Grapic/premake4.lua`
- Ajoutez la commande
  `make_project("MYPROJ","apps/MYPROJ/main_MYPROJ.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 `MYPROJ` que vous venez de créer et choisissez `Activate project`. Éditez enfin le fichier `main_MYPROJ.cpp` et commencez le TP.

Dans ce TP nous allons écrire le programme complet permettant de jouer au jeu du Memory (version à 1 seul joueur dans un premier temps).
Le jeu du Memory est un jeu de mémoire qui consiste à retrouver des paires d'images identiques dans une grille d'images. Au départ, toutes les cartes sont cachées puis le joueur sélectionne deux cartes. Si les cartes sont identiques, elles restent visibles, sinon elles sont à nouveau retournées. La partie s'arrête lorsque toutes les paires ont été découvertes.

1. Créez un nouveau projet `MEMORY` dans Grapic et téléchargez l'archive contenant les vignettes utilisées dans le jeu sur la page du TP du jour. Décompressez cette archive dans le dossier `data/MEMORY`.

2. Définissez les structures de données nécessaires au jeu.
   a. Déclarez dans un premier temps toutes les constantes qui vous permettront de paramétrer votre jeu de Memory : `DIMW` (dimension de la fenêtre d'exécution dans Grapic), `MAX_X` et `MAX_Y` (le nombre de lignes et de colonnes **maximum** dans la grille d'images en s'assurant que chacune des deux valeurs est bien paire pour avoir un nombre pair de cases dans la grille), `MAX_IMAGES` (le nombre d'images disponibles dans le dossier `data`), et enfin `MAXCHAR` (taille maximale des chaines de caractères utilisées).

   b. Pour définir la grille de jeu, nous aurons besoin d'une structure case. Chaque case contient 3 informations : une image `ImCase` (celle qui sera affichée lorsque la carte sera choisie par le joueur), le nom de l'image `Nom_Image` (chaine de caractères représentant le chemin d'accès à cette image dans le dossier *data*), et un booléen `visible` qui nous permettra d'afficher uniquement les cases correspondant aux cartes choisies ou bien aux paires remportées.

   c. Définissez la structure du jeu complet du Memory. Cette structure contient les champs suivants : `TGX` et `TGX` (nombre réel de cases dans la grille), un tableau 2D `jeu` de Cases_Grilles, un tableau d'entiers `t_indice_image` (initialisé à 0 et

contenant 1 lorsque l'image d'indice i a déjà été utilisée dans la grille) et l'image `vide` de départ. Le tableau d'entiers `t_indice_image` permettra de mémoriser les indices des images déjà utilisées dans la grille de jeu et éviter ainsi les redondances.

3. Initialisation / création de la grille de jeu (pensez à utiliser le TD 12…).
   a. Les fichiers d'images que vous avez copiés dans le dossier data sont nommés de la manière suivante : `Image_Memory_XX.jpg` avec XX un entier compris entre 01 et 18. Afin de pouvoir positionner aléatoirement les images dans la grille de jeu, il est impératif de pouvoir générer automatiquement le nom du fichier qui sera associé à l'image contenue dans la grille. Pour éviter les doublons (de paires de cartes) il faudra veiller à ce que l'indice choisi de manière aléatoire n'ait pas déjà été sélectionné. On utilisera pour cela le tableau `t_indice_image` de la structure de jeu. Ecrivez une procédure qui prend en paramètre la structure du `memory` et "renvoie" la chaine de caractère construite contenant le nom du fichier.
   b. Nous allons maintenant procéder au remplissage automatique de la grille de memory. Cette procédure devra placer aléatoirement dans la grille `(TGX*TGY)/2` images. Attention de bien vérifier que la grille est vide à l'endroit sélectionné avant d'insérer l'image.
   c. Ecrire le sous-programme permettant d'initialiser tous les champs de la structure `memory` puis de remplir la grille de jeu avec les images.

4. L'affichage de la grille de jeu va nécessiter de convertir les informations de coordonnées de cases dans le tableau 2D en coordonnées "réelles" dans la fenêtre d'exécution de Grapic. Seules les images découvertes devront être visibles, l'image vide sera affichée par défaut. Il faudra également redimensionner les vignettes pour que leur taille soit adaptée à la taille de la cellule.

   Nous utiliserons la fonction suivante :
   `void` **`image_draw`** `(Image &im, int x, int y, int w=-1, int h=-1)`
   qui affiche l'image `im` à partir du point de coordonnées `(x,y)` avec une taille `w` et `h` (ou -1 si la taille d'origine est conservée).

5. Pour rendre cette partie plus interactive, le joueur utilisera la souris pour sélectionner les deux cases qu'il souhaite retourner. Afin de mieux comprendre la gestion de la souris sous Grapic, reportez-vous au tutoriel qui lui est consacré (http://liris.cnrs.fr/alexandre.meyer/grapic/html/index.html#tuto5).
   a. Ecrivez un sous-programme permettant de récupérer et retourner les coordonnées (tableau) de la case qui a été cliquée par le joueur. Cette case sera rendue visible et un nouvel affichage de la grille effectué.
   b. Ecrivez une fonction qui compare deux cases choisies par le joueur et retourne 1 si elles sont identiques, 0 sinon. Si les deux vignettes sont identiques elles resteront affichées jusqu'à la fin de la partie, sinon elles seront à nouveau retournées.

Ecrivez le programme principal qui permet de jouer au Memory. On pourra en fin de partie afficher le nombre d'échecs (images différentes retournées) pour parvenir à découvrir toutes les paires.

Correction version Texte

```
#include <iostream>
#include <string.h>
#include<stdlib.h>
#include<time.h>

using namespace std;
const int MAX = 32 ;
const int TG = 2 ;

void init_grille (char grille[TG][TG][MAX])
{
    int i,j;
    for (i=0;i<TG;i++)
    {
        for (j=0;j<TG;j++)
        {
            strcpy(grille[i][j],"*");
        }
    }
}

void affiche_grille (char grille[TG][TG][MAX])
{
    int i,j;
    for (i=0;i<TG;i++)
    {
        for (j=0;j<TG;j++)
        {
            cout<<grille[i][j]<<" ";
        }
        cout<<endl;
    }
}

void affiche_solution (char grille[TG][TG][MAX],char sol[TG][TG][MAX],int l1, int c1, int l2, int c2 )
{
    int i,j;
    for (i=0;i<TG;i++)
    {
        for (j=0;j<TG;j++)
        {
            if ((i==l1 && j==c1)||(i==l2 && j==c2))
                cout<<grille[i][j]<<" ";
            else cout<<sol[i][j];
        }
        cout<<endl;
    }
}
void remplir_grille (char grille[TG][TG][MAX])
{
    int i,j,x,y;
    char mot[MAX];
    for (i=0;i<TG*TG/2;i++)
    {
        cout<<"Donnez un mot"<<endl;
        cin>>mot;
        for (j=0;j<2;j++)
        {
            do
            {
                x=rand()%TG;
```

```
            y=rand()%TG;
        }
        while (strcmp (grille[x][y],"*")!=0);
        strcpy(grille[x][y],mot);
      }
   }
}

void selectionne_case (int &l1, int &c1, int &l2, int&c2)
{
   cout<<"donnez les coordonnées des deux cases à sonder"<<endl;
   do
   {

   cout<<"premiere case : "<<endl;
   cin>>l1>>c1;
   cout<<"deuxième case : "<<endl;
   cin>>l2>>c2;
   }
   while (l1<0 || l1>=TG || c1<0 || c1>=TG || l2<0 || l2>=TG || c2<0 || c2>=TG || (l1==l2) && (c1==c2));
}

bool verifie_cases (char grille[TG][TG][MAX], char solution[TG][TG][MAX])
{
   int l1,c1,l2,c2;
   selectionne_case(l1,c1,l2,c2);
   affiche_solution(grille,solution,l1,c1,l2,c2);
   if (strcmp (grille[l1][c1],grille[l2][c2])==0)
   {
      strcpy(solution[l1][c1], grille[l1][c1]);
      strcpy(solution[l2][c2], grille[l2][c2]);
      return true;
   }
return false;
}
int main (void)
{
   char jeu[TG][TG][MAX],sol[TG][TG][MAX];
   int scoreJ1=0,scoreJ2 = 0, nb_paires = TG*TG / 2;
   int joueur=1;
   srand(time(NULL));
   init_grille(jeu);
   init_grille(sol);
   remplir_grille(jeu);
   //affiche_grille(jeu);
   do
   {
      if (verifie_cases(jeu,sol))
      {
         cout<<"cases identiques, rejoue"<<endl;
         nb_paires--;
         if (joueur==1) scoreJ1++;
         else scoreJ2++;
      }
      else
      {
         cout<<"cases differentes on change de joueur"<<endl;
         if (joueur==1) joueur = 2;
         else           joueur = 1;
      }
   }
   while (nb_paires!=0);
   if(scoreJ1>scoreJ2) cout<<"Joueur1 a gagné !!"<<scoreJ1<<" contre "<<scoreJ2<<endl;
```

```
    else if(scoreJ1<scoreJ2) cout<<"Joueur2 a gagné !!"<<scoreJ2<<" contre "<<scoreJ1<<endl;
    else cout<<"egalité !"<<endl;

 return 0;
}
```

Correction version graphique

```
#include <Grapic.h>
#include<string.h>
#include <iostream>
using namespace grapic;
using namespace std;

const int DIMW = 500;
const int MAXCHAR = 100;
const int MAX_X = 10;
const int MAX_Y = 10;
const int MAX_IMAGES = 18;

struct CaseGrille
{
    Image ImCase;
    char nom_image[MAXCHAR];
    bool visible;
};

struct Memory
{
    int TGX=4;
    int TGY=4;
    struct CaseGrille jeu[MAX_X][MAX_Y];
    int t_indice_image[MAX_IMAGES];
    Image vide;

};

void Creer_Nom_Fichier (Memory &m,char nom_fichier[MAXCHAR])
{
    int num_image;
    char numero[3];
    do
    {
        num_image=rand() % MAX_IMAGES + 1;
    }
    while (m.t_indice_image[num_image-1]!=0);
    m.t_indice_image[num_image-1]=1;

    strcpy(nom_fichier,"data/memory/Image_Memory_");
    if (num_image<10)
    {
        strcat(nom_fichier,"0");
        numero[0]='0'+num_image;
        numero[1]='\0';
    }
    else
    {
        numero[0]='1';
        numero[1]=num_image % 10 + '0';
        numero[2]='\0';
    }
    strcat(nom_fichier,numero);
```

```
    strcat(nom_fichier,".jpg");

}


void init_images(Memory& m)
{
    int i,j;
    int ind_ligne,ind_colonne;
    char nom_image[MAXCHAR];

    for (i=0; i<(m.TGX*m.TGY)/2; i++)
    {
        Creer_Nom_Fichier(m,nom_image);
        for (j=0; j<2; j++)
        {
            do
            {
                ind_ligne=rand()%m.TGX ;
                ind_colonne=rand()%m.TGY ;
            }
            while (strcmp(m.jeu[ind_ligne][ind_colonne].nom_image,"data/memory/vide.jpg")!=0);
            strcpy(m.jeu[ind_ligne][ind_colonne].nom_image,nom_image);
            m.jeu[ind_ligne][ind_colonne].ImCase=image(nom_image);
            //
image_draw(m.jeu[ind_ligne][ind_colonne].ImCase,ind_ligne*DIMW/m.TGX,ind_colonne*DIMW/m.TG
Y,DIMW/m.TGX,DIMW/m.TGY);
        }
    }
    m.vide = image("data/memory/vide.jpg");
}


void init(Memory& m)
{
    int i,j;
    for (i=0; i<MAX_IMAGES; i++)
    {
        m.t_indice_image[i]=0;
    }
    for (i=0; i<m.TGX; i++)
    {
        for (j=0; j<m.TGY; j++)
        {
            m.jeu[i][j].ImCase=image("data/memory/vide.jpg");
            strcpy(m.jeu[i][j].nom_image,"data/memory/vide.jpg");
            //image_draw(m.jeu[i][j].ImCase,i*DIMW/m.TGX,j*DIMW/m.TGY,DIMW/m.TGX,DIMW/m.TGY);
            m.jeu[i][j].visible=false;
        }
    }
    init_images(m);
}


void choix_case (Memory m, int &l1, int &c1)
{
    int l,c;
    bool b;
    l1 = -1;
    while  ( l1==-1 )
    {
        if (isMousePressed(SDL_BUTTON_LEFT))
        {
```

```
            int x,y;
            mousePos(x, y);
            l=x/(DIMW/m.TGX);
            c=y/(DIMW/m.TGY);
            if (!m.jeu[l][c].visible)
            {
                l1=l;
                c1=c;
            }
        }
        winDisplay();
    }
}


void Affiche_Grille(Memory &m)
{
    int i,j;
    Image Im;
    for (i=0; i<m.TGX; i++)
    {
        for (j=0; j<m.TGY; j++)
        {
            if (m.jeu[i][j].visible==false)
            {
                image_draw( m.vide,i*DIMW/m.TGX,j*DIMW/m.TGY,DIMW/m.TGX,DIMW/m.TGY);
            }
            else

image_draw(m.jeu[i][j].ImCase,i*DIMW/m.TGX,j*DIMW/m.TGY,DIMW/m.TGX,DIMW/m.TGY);

        }
    }

}

int VerifiePaire(Memory &m)
{
    int l1,c1,l2,c2;
    choix_case(m,l1,c1);
    m.jeu[l1][c1].visible=true;
    Affiche_Grille(m);
    winDisplay();

    choix_case(m,l2,c2);
    m.jeu[l2][c2].visible=true;
    Affiche_Grille(m);
    winDisplay();
    //pressSpace();
    if ( strcmp(m.jeu[l1][c1].nom_image,m.jeu[l2][c2].nom_image)==0)
    {
        return 1;
    }
    else
    {
        delay(2000);
        m.jeu[l2][c2].visible=false;
        m.jeu[l1][c1].visible=false;
        return 0;
    }

}
```

```
int main(int , char** )
{
    Memory m;
    int score = 0 , echec=0;
    int l1,c1,l2,c2;
    bool stop=false;
    srand(time(NULL));
    winInit("MEMORY", DIMW, DIMW);
    winClear();
    init(m);
    Affiche_Grille(m);
    stop=winDisplay();

    while ((!stop)&&(score!=(m.TGX*m.TGY)/2))
    {
        if (VerifiePaire(m))
        {
            score+=1;
        }
        else
        {
            echec++;
        }
        Affiche_Grille(m);
        stop = winDisplay();
    }
    color(255,0,0);
    fontSize(48);
    print(150,250,"GAGNE!!!");
    fontSize(28);
    print(200,150,echec);
    print(250,150,"erreurs");
    color(0,0,0);
    fontSize(12);
    pressSpace();
    winQuit();
    return 0;
}
```