TP2 : Et maintenant … le damier !!!

Tristan Colombo 2002

1) Affichage des coordonnées

Récupérer l'ossature de programme du TP1 en ne gardant que l'affichage du tableau (gtk_table). La *gtk_table* est une matrice de 11 x 11 cases indicées pour le programmeur par :

(0, 0)	(1, 0)	(2, 0)	…	(10, 0)
(0, 1)	(1, 1)	(2, 1)	…	(10, 1)
…	…	…	…	…
(0, 10)	…	…	…	(10, 10)

Or nous avons vu que pour désigner une case dans le *gtk_table_attach* il fallait donner les coordonnées du point supérieur gauche et du point inférieur droit de la case. La case (i, j) du tableau ci-dessus doit donc être appelée par i, i+1, j, j+1.
Pour l'affichage des coordonnées proprement dites, on doit stocker la valeur à afficher dans une variable de type tableau de caractère (par ex. char s[5]) car la fonction *gtk_label_new* n'accepte pas le formatage de type *printf* (nous utiliserons la fonction *sprintf* qui permet de copier dans une variable de type tableau de caractères un affichage formaté. Ex. : `sprintf(a, "i=%d", i)` copiera dans a la chaîne « i=5 » si la variable i vaut 5.
Les coordonnées verticales sont de type entier : dans la case (0, 0) de notre tableau il faudra afficher 10, dans la case (0, 1) ce sera 9, etc. (seules les ordonnées varient). Il s'agit d'une boucle décroissante de 10 à 1 :

```
char num_case[3];
int i;
GtkWidget *Label;
…
for (i=10; i>=1; i--) /* i-- équivaut à i=i-1 */
{
  sprintf(num_case, "%d", i);
  Label=gtk_widget_new(num_case);
  gtk_table_attach(GTK_TABLE(Table), Label, 0, 1, 10-i, 10-i+1, GTK_FILL,
GTK_FILL, 0, 0);
}
```

Pour les coordonnées horizontales qui sont des lettres nous allons utiliser une propriété du C : lorsque l'on donne un entier et que l'on demande au C d'afficher un caractère … il va le faire car tout caractère possède un codage dit ASCII : ici il suffit de savoir que 65 correspond au A, 66 au B, etc. La portion de code pour l'affichage de ces coordonnées sera alors (pour la *gtk_table*, seules les abscisses varient) :

```
for (i=1; i<=10; i++) /* i++ équivaut à i=i+1 */
{
  sprintf(num_case, "%c", i+64);
  Label=gtk_widget_new(num_case);
  gtk_table_attach(GTK_TABLE(Table), Label, i, i+1, 10, 11, GTK_FILL,
GTK_FILL, 0, 0);
}
```

2) Affichage des cases (boutons contenant un pixmap) :

Nous avons vu la dernière fois comment déclarer des pixmaps GDK. Il faudra ici déclarer 4 types de pixmaps GDK correspondant à la case blanche, la case noire, le pion blanc et le pion noir (nous appelerons ces pixmaps GDK Pix_blanc, Pix_noir, Pix_pion_noir, Pix_pion_noir).
La phase de création correspond à :

```
#include <case_blanche.xpm>
…
GdkPixmap *Pix_blanc;
GdkBitmap *Masque_blanc;
…
Pix_blanc = gdk_pixmap_create_from_xpm_d(Fenetre->window,
                                         &Masque_blanc,
                                         &Transparent,
                                         case_blanche_xpm);
```

Pour pouvoir attacher un pixmap à un bouton il faut transformer le pixmap GDK en pixmap GTK. De plus nous voulons placer les boutons dans une case (i, j) de la *gtk_table*. Nous allons créer une fonction qui prendra en paramètre un pixmap GTK, un bouton, une table et les coordonnées d'une case et placera le pixmap dans le bouton et le bouton dans le tableau. Il est à noter qu'à chaque bouton nous voulons affecter une opération différente (afficher le nom de la case) : il nous faudra donc autant de boutons que de cases soit un `GtkWidget *Damier[10][10]`. Voici la fonction qui permet d'effectuer ces opérations :

```
void Affiche_case(GtkWidget *Pix, GtkWidget *Bouton, GtkWidget *Tab, int i,
int j)
  {
    Bouton = gtk_button_new();
    gtk_button_set_relief(GTK_BUTTON(Bouton), GTK_RELIEF_NONE);
    gtk_container_add(GTK_CONTAINER(Bouton), Pix);
    gtk_table_attach(GTK_TABLE(Tab), Bouton, i+1, i+2, j, j+1, GTK_FILL,
GTK_FILL, 0, 0);
    sprintf(c[i][j], "%c-%d", i+65, 10-j);
    gtk_signal_connect(GTK_OBJECT(Bouton), "clicked", (GtkSignalFunc)
Selection_de_case, (gpointer) c[i][j]);
  } /* Fin de Affiche_case */
```

Note : Chaque bouton est relié à la fonction *Selection_de_case* à laquelle on passe en paramètre la chaîne c[i][j] qui contient le nom de la case. Cette fonction très simple ne fait qu'afficher cette chaîne sur le terminal :

```
void Selection_de_case(GtkObject *objet, gpointer num_case)
  {
    int x, y;
    fprintf(stderr, "Case %s\n", (char *) num_case);
  } /* Fin de la fonction Selection_de_case */
```

Il ne nous reste plus maintenant qu'à écrire les boucles pour afficher dans un premier temps toutes les cases noires puis les cases blanches avec ou sans pions. Pour les cases noires, nous savons que la première case sera (1, 0), puis (3, 0), … donc une progression de +2 sur les abscisses. Sur la seconde ligne il y aura une inversion : (0, 1), puis, (2, 1), … En posant i l'abscisse de la case et y l'ordonnée, pour y pair i sera initialisé à 1 et pour y impair i sera initialisé à 0 (penser à créer le pixmap GTK avant d'appeler la fonction *Affiche_case*) :

```
/* Dessin des cases noires */
  i=1;
  for (j=0; j<10; j++)
  {
    while (i<10)
    {
      Pixmap = gtk_pixmap_new(Pix_noir, Masque_noir);
      Affiche_case(Pixmap, Damier[i][j], Table, i, j);
      i=i+2;
    }
```

```
    if (i==11)
      i=0;
    else
      i=1;
  }
```

Pour les cases blanches le système est identique mais il faudra afficher des pions noirs sur les lignes 0 à 3 et des pions blancs pour les lignes 6 à 9 :

```
/* Dessin des cases blanches (avec ou sans pions) */
  i=0;
  for (j=0; j<10; j++)
  {
    while (i<10)
    {
      if (j<4)
      Pixmap = gtk_pixmap_new(Pix_pion_noir, Masque_pion_noir);
      else if (j>=6)
      Pixmap = gtk_pixmap_new(Pix_pion_blanc, Masque_pion_blanc);
      else
      Pixmap = gtk_pixmap_new(Pix_blanc, Masque_blanc);
      Affiche_case(Pixmap, Damier[i][j], Table, i, j);
      i=i+2;
    }
    if (i==11)
      i=0;
    else
      i=1;
  }
```

Et voilà …