EXAMEN D'INITIATION AUX INTERFACES GRAPHIQUES
Institut Galilée - Université Paris 13 — **Lundi 11 juin 2012 – 14h-16h**
Licence 1 — **Aucun document autorisé**

1 Puissance 4

1.1 Le jeu

Le jeu de puissance 4 se joue entre deux joueurs sur une grille constituée de 6 rangées et 7 colonnes. Un joueur est gagnant s'il parvient le premier à aligner 4 de ses pions successivement sur une ligne, une colonne ou une diagonale de la grille. Les deux joueurs jouent chacun leur tour. Un tour pour un joueur consiste à choisir la colonne où placer son pion : le pion se place alors dans la case la plus basse qui ne contient pas de pion. La partie est ex-æquo si aucun joueur ne gagne et que toutes les cases sont remplies. Exemples de parties gagnées par un joueur :

			1			
		1	2			
	1	2	1			
1	2	1	2			2

Joueur 1 gagne en diagonale

		1				
	2	**2**	**2**	**2**		
2	1	1	1	2		
1	1	2	2	1	1	

Joueur 2 gagne en ligne

			2			
2	1		**2**			
1	2		**2**		1	
1	2		**2**	1	2	
1	2	1	1	2	1	1

Joueur 2 gagne en colonne

1.2 Bibliothèque "puissance 4" fournie

Vous disposez d'une bibliothèque C du jeu de puissance 4, dont le fichier d'entête est joint au sujet. Les structures de données et les méthodes sont documentées dans ce fichier. Précisons quelques points :

• *Gestion du champ 'joueur' de la structure 'jeu_p4'.* Le joueur 1 commence toujours la première partie. Ensuite, c'est le joueur qui n'a pas commencé la partie précédente qui commence une nouvelle partie. Pour gérer le cas d'une première partie, le modèle prévoit la fonction 'p4_new' qu'il faut appeler dés que l'on déclare une variable 'p4_jeu' et qui initialise certains de la structure, afin ensuite de pouvoir proprement décider quel joueur devra entamer la partie. C'est ensuite la fonction 'p4_init' qui met à jour la structure pour entamer une nouvelle partie, et donc en particulier le joueur qui doit jouer (cette fonction s'appuie pour cela sur les champs 'joueur' et 'nbcoup' de la structure 'p4_jeu').

• *Indice des cases.* Les rangées sont indicées de 0 à NB_LIG (de haut en bas), les colonnes sont indicées de 0 à NB_COL (de gauche à droite). Les cases sont stockées dans un tableau unidimensionnel. La case se situant à l'intersection de la ligne i et de la colonne j est donc d'indice $(i*\text{NB\_COL}) + j$ dans ce tableau.

• *Fonction 'p4_jouercoup'.* Cette fonction permet de jouer un coup. Elle utilise une fonction qui n'apparaît pas dans le fichier d'entête car elle est une sous-routine qui n'est déclarée et définie que dans le fichier source. Cette fonction, dont le prototype est `void p4_testercoup(p4_jeu* jeu, int i, int j);` et qui teste si le dernier coup joué est vaiqueur, est documentée comme suit :

```
/* Vérifie et met à jour l'état de la partie à l'issue d'un coup joué.
Pré-conditions : 0<=i<=NB_LIG -1, 0<=j<= NB_COL -1, la case (i,j) est jouée par jeu->joueur
Post-conditions :
- si jeu->joueur == ETAT_CASE_J1 et que le joueur remporte la partie, l'état de la partie est
mis à jour à ETAT_JEU_J1
- si jeu->joueur == ETAT_CASE_J2 et que le joueur remporte la partie, l'état de la partie est
mis à jour à ETAT_JEU_J2
- si la partie est ex-aequo, l'état de la partie est mis à jour à ETAT_JEU_J12 */
static void p4_testercoup(p4_jeu* jeu, int i, int j);
```

2 Prise en mains de la bibliothèque “puissance 4”

1. Suite à une mauvaise manipulation, il faut réécrire certaines fonctions de la bibliothèque “puissance 4”. Il s’agit des fonctions `case_get_tour`, `case_set`, `p4_get_joueur`, `p4_get_case_etat` et `p4_jouercoup`. Pour cette dernière fonction, vous devez faire appel à la sous-routine ‘p4_testercoup’.
2. Il manque certaines fonctionnalités à la bibliothèque “puissance 4”. Donner le prototype et écrire le code de deux nouvelles fonctions :

   (a) La fonction `p4_afficher` qui représente l’état de la partie. Exemple d’affichage :

			2			
2	1		2			
1	2		2		1	
1	2		2	1	2	
1	2	1	1	2	1	1

   (b) La fonction `p4_derouler` qui permet de retracer le déroulement d’une partie en affichant, pour chaque coup joué de 1 au nombre de coups joués dans la partie en cours : l’indice de colonne de la case jouée lors de ce coup, le joueur qui a joué ce coup. Exemple d’affichage :

```
coup 1:   colonne 0  joueur 1
coup 2:   colonne 1  joueur 2
coup 3:   colonne 1  joueur 1
coup 4:   colonne 6  joueur 2
coup 5:   colonne 2  joueur 1
coup 6:   colonne 3  joueur 2
coup 7:   colonne 3  joueur 1
coup 8:   colonne 3  joueur 2
coup 9:   colonne 3  joueur 1
coup 10:  colonne 2  joueur 2
coup 11:  colonne 2  joueur 1
```

3 Interface graphique du jeu de Puissance 4

Vous devez développer l'interface graphique du jeu de puissance 4, en gtk. L'interface doit afficher la grille de jeu et indiquer dans son titre "Tour du joueur 1", "Tour du joueur 2", "Partie remportée par le joueur 1", "Partie remportée par le joueur 2" ou "Partie ex-æquo", selon l'état courant du jeu. Au cours d'une partie, elle doit permettre aux joueurs de spécifier la colonne dans laquelle ils souhaitent jouer. Lorsqu'un partie est finie, elle doit permettre aux joueurs de demander une nouvelle partie.

1. Composants graphiques (en s'appuyant sur un schéma) :
   (a) Quels sont les objets graphiques impliqués (*ex.*, 3 boutons, 2 libellés, etc.) ? *Considérez les composants dans leur intégralité ; ne tenez pas compte du libellé d'un bouton, du titre d'une fenêtre, etc.*
   (b) Comment ces objets sont-ils imbriqués entre-eux (*ex.*, le bouton `toto` dans la fenêtre `tata`, *etc.*) ?
   (c) Pour chaque imbrication d'un objet dans un autre, indiquer et justifier la fonction gtk à utiliser. Vous pouvez pour cela donner le prototype de la fonction si vous le connaissez, ou simplement décrire la fonction, l'objet gtk sur lequel elle est définie, sa vocation, ses paramètres essentiels.
2. Leur comportement :
   (a) Indiquer pour chaque composant pouvant interagir avec l'utilisateur son état d'activité en fonction de l'état du jeu.
   (b) Indiquer pour chaque libellé (dont le titre de la fenêtre, le contenu d'une zone de saisie, *etc.*) l'évolution de sa valeur en fonction de l'état du jeu.
3. Sollicitations faites au modèle :
   (a) Quelle(s) fonction(s) de la bibliothèque "puissance 4" faut-il appeler préalablement au lancement de l'interface graphique ?
   (b) Quelle(s) fonction(s) de la bibliothèque "puissance 4" faut-il appeler lorsque sur l'interface, le joueur a choisi de jouer une colonne ?
   (c) Quelle(s) fonction(s) de la bibliothèque "puissance 4" faut-il appeler lorsque sur l'interface, l'utilisateur demande une nouvelle partie ?
4. Comment faire en sorte que votre interface ait le comportement décrit dans les réponses précédentes ? Indiquer :
   (a) le prototype et la vocation des fonctions que vous envisagez devoir écrire (mais sans écrire les fonctions) ;
   (b) l'ensemble des appels que vous envisagez de faire à la fonction `gtk_signal_connect` ;
   (c) les éventuelles structures de données que vous envisagez de définir pour faire appel à ces fonctions.

4 Langage C et gtk

1. Le programme suivant produit une erreur à la compilation ; quelles sont les instructions fautives ?

```
1   #include <gtk/gtk.h>
2   int main (int argc, char *argv[])
3   {
4       GtkWindow* pFenetre = NULL;
5       gtk_init(NULL, NULL);
6       pFenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
7       gtk_window_set_title(pFenetre, "toto");
8       gtk_widget_show_all(GTK_WIDGET(pFenetre));
9       gtk_main();
10      return 0;
11  }
```

2. Le programme suivant compile, mais produit des erreurs à l'exécution. Lesquelles, et pourquoi ?

```
1   #include <gtk/gtk.h>
2   int main (int argc, char *argv[])
3   {
4       GtkWidget* pFenetre = NULL;
5       GtkWidget* pBoutonQuitter = NULL;
6       gtk_init(NULL, NULL);
7       pFenetre = gtk_window_new(GTK_WINDOW_TOPLEVEL);
8       pBoutonQuitter = gtk_label_new("_Quitter");
9       gtk_container_add(GTK_CONTAINER(pFenetre), pBoutonQuitter);
10      gtk_window_set_title(GTK_WINDOW(pFenetre), "tata (pour changer)");
11      gtk_button_set_label(GTK_BUTTON(pBoutonQuitter), "titi (soyons fous !)");
12      gtk_widget_show_all(GTK_WIDGET(pFenetre));
13      gtk_main();
14      return 0;
15  }
```