INFO0030 : Projet de Programmation
Oxo

B. Donnet, E. Marechal

Alerte : Évitez de perdre bêtement des points...

- Nous vous conseillons **vivement** de relire et d'appliquer les guides de style et de langage, mis à votre disposition sur eCampus (INFO0030, Sec. Supports pour le Cours Théorique). Cela vous permettra d'éviter de nombreuses erreurs et de perdre des points inutilement.
- Nous vous conseillons de consulter la grille de cotation utilisée pour la correction des projets afin d'éviter de perdre bêtement des points. Elle est disponible sur eCampus (INFO0030, Sec. Procédures d'Évaluation).
- Votre code sera compilé et testé sur la machine virtuelle qui vous a été fournie et qui sert de référence. Elle est disponible sur eCampus (INFO0030, Sec. Projets). Veillez donc à ce que votre code fonctionne dans cet environnement.

1 Contexte

Oxo est un jeu de stratégie très populaire qui se déroule rapidement. Il permet d'organiser des championnats de courte durée. Oxo est une variante du jeu *tic-tac-toe*. Le jeu se pratique à deux joueurs (au tour par tour) avec pour but de créer le premier un alignement.

Les deux joueurs s'affrontent sur une grille de $n \times n$ cases. Ils doivent remplir, chacune à leur tour, une case de la grille avec un symbole. Le premier joueur inscrit la lettre 'O', le second la lettre 'X'. La partie est gagnée quand l'un des joueurs a réussi à aligner le mot "OXO" sur la grille – à la verticale, à l'horizontale ou en diagonale.

Il vous est demandé, dans ce projet, d'implémenter le jeu Oxo sous la forme d'une interface graphique.

L'objectif de ce projet est de vous permettre d'apprendre à manipuler la programmation événementielle et, en particulier, la librairie Gtk+2 vue au cours.[1] Cette librairie, écrite en C/C++ a été écrite pour réaliser le projet GIMP, qui est une très bonne alternative libre au logiciel de retouche d'image Photoshop. Gtk+2 est aussi à la base de l'environnement graphique GNOME.

2 Interface Graphique

L'objectif de cette section est de vous présenter l'interface graphique générale de votre Oxo.

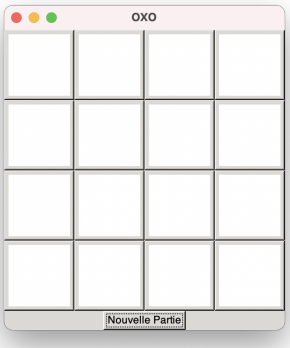
La Fig. 1 présente l'interface graphique générale de votre Oxo ainsi que différentes situations du jeu.

La Fig. 1(a) présente l'interface graphique générale de votre Oxo ainsi que la situation initiale du jeu (i.e., avant de placer la première lettre). Le plateau de jeu est composé de 16 cases (4×4). A l'initialisation, les cases sont "blanches". Le bouton "Nouvelle Partie" permet de créer une nouvelle partie et doit donc créer le plateau tel qu'on le voit sur la Fig. 1(a).

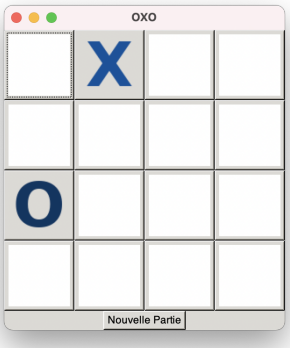
La Fig. 1(b) illustre une partie en cours où deux tours ont déjà été joués.

La Fig. 1(c) illustre une partie se terminant en match nul. Toutes les cases sont remplies mais aucun "OXO" n'a pu être aligné. La seule solution ici, c'est de démarrer une nouvelle partie en cliquant sur le bouton "Nouvelle Partie" ou quitter le jeu (en cliquant sur la croix dans le coin supérieur de la fenêtre – le bouton rouge dans un environnement Mac OS X).

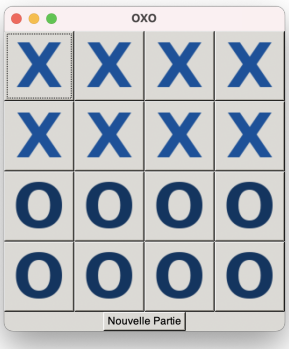
1. cfr. http://www.gtk.org



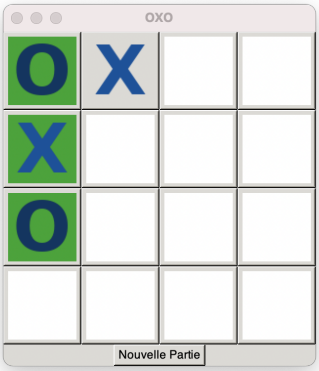
(a) Interface générale. Situation initiale.



(b) Partie en cours.



(c) Partie finie. Match nul.



(d) Partie finie. Victoire.

FIGURE 1 – Interface graphique et situations de jeu.

Enfin, la Fig. 1(d) illustre une partie victorieuse. La chaîne "OXO" a pu être alignée (ici verticalement – pour rappel, on peut aligner verticalement, horizontalement, et en diagonal). Vous voyez que les caractères sont mis en surbrillance pour illustrer, aux joueur, une situation victorieuse. La seule solution ici, c'est de démarrer une nouvelle partie en cliquant sur le bouton "Nouvelle Partie" ou quitter le jeu (en cliquant sur la croix dans le coin supérieur de la fenêtre – le bouton rouge dans un environnement Mac OS X).

Les images pour une case blanche, les lettres normales ('X' et 'O') et les lettres victorieuses vous sont fournies avec cet énoncé. [2]

3 Compléments GTK

Cette section a pour but de vous apporter des compléments à GTK+2 de façon à pouvoir réaliser ce projet.

3.1 Image et Bouton

Il est possible, en GTK+2, d'afficher une image dans un bouton. Supposons que nous disposions d'une image, `img1.jpg` et qu'on veut l'afficher dans un bouton `pBouton`. Ceci nécessite trois étapes :

1. charger l'image depuis le disque dur. Eventuellement, on peut redimensionner l'image.
2. créer le bouton.
3. placer l'image dans le bouton

Le bout de code ci-dessous illustre ces trois étapes.

```
GtkButton *charge_image_bouton(){
  //1. Charger l'image et la redimensionner (100*100 pixels)
  GdkPixBuf *pb_temp = gdk_pixbuf_new_from_file("img1.jpg", NULL);
  if(pb_temp==NULL){
    printf("Erreur de chargement de l'image img1.jpg!\n");
    return NULL;
  }
  GdkPixBuf *pb = gdk_pixbuf_scale_simple(pb_temp, 100, 100, GDK_INTERP_NEAREST);

  //2. Créer le bouton
  GtkWidget *pBouton = gtk_button_new();

  //3. Placer l'image
  GtkWidget *image = gtk_image_new_from_pixbuf(pb);
  gtk_button_set_image(GTK_BUTTON(pBouton), image);

  return pBouton;
}//fin charge_image_bouton()
```

Si on désire remplacer l'image existante d'un bouton, il suffit d'appeler, sur ce bouton, la procédure `gtk_button_set_image` avec la nouvelle image.

Notez qu'à chaque fois que l'on remplace l'image du bouton, on crée un nouvel objet de type `GtkImage` [3] en appelant `gtk_image_new_from_pixbuf`. On pourrait supposer qu'il serait préférable de libérer l'image avant de créer la nouvelle. En réalité, lorsqu'on appelle `gtk_button_set_image`, GTK+2 libère la mémoire occupée par la `GtkImage` précédemment affichée.

3.1.1 Dimensionnement des Boutons

Une fois un bouton créé (et, plus généralement, n'importe quel élément de type `GtkWidget *`), on peut toujours modifier sa taille par défaut. Il suffit d'utiliser la procédure suivante :

```
void gtk_widget_set_size_request(GtkWidget *wgt, gint largeur, gint hauteur);
```

qui redimensionne le widget `wgt` en un widget de `largeur` × `hauteur`. L'unité de mesure est ici le pixel.

2. voir eCampus, INFO0030, Sec. Projets/Projet 3: OXO
3. A l'instar (par exemple) de `GtkWindow`, un object de type `GtkImage` est aussi de type `GtkWidget`.

4 Enoncé du Projet

Il vous est demandé d'écrire un programme en langage C implémentant le jeu Oxo tel que décrit dans ce document.

En particulier, votre projet devra :

— être soumis dans une archive `tar.gz`, appelée `oxo.tar.gz`, via la Plateforme de Soumission. La décompression de votre archive devra fournir tous les fichiers nécessaires **dans le répertoire courant où se situe l'archive**. Vous penserez à joindre tous les codes sources nécessaires.

— définir les fonctionnalités nécessaires à la mise en place de l'IHM du jeu Oxo (cfr. Sec. 2).

— être modulaire, i.e., nous nous attendons à trouver un (ou plusieurs) header(s) et un (ou plusieurs) module(s).

— s'assurer que les structures de données proposées sont implémentées comme des types opaques (quand cela s'avère pertinent).

— être parfaitement documenté. Vous veillerez à spécifier correctement chaque fonction/procédure/structure de données que vous définirez. Votre documentation suivra les principes de l'outil `doxygen`.

— appliquer les principes de la programmation défensive (vérification des préconditions, vérification des mallocs, ...). Pensez à libérer la mémoire allouée en cours de programmation afin d'éviter les fuites de mémoire.

— comprendre un `Makefile` (voir squelette du `Makefile` à compléter) permettant au moins de

1. compiler votre projet. L'exécution de la commande

```
$>make
```

   permet de générer un fichier binaire, exécutable, appelé `oxo`.

2. Le fichier binaire généré devra pouvoir être exécuté de la façon suivante :

```
$>./oxo
```

3. générer la documentation doxygen. L'exécution de la commande

```
$>make doc
```

   devra produire de la documentation au format HTML dans le sous-répertoire `doc/`.

Il est impératif de respecter **scrupuleusement** les consignes sous peine de se voir attribuer une note de 0/20 pour non respect de l'énoncé.