Projet de programmation

Structures de données

— Licence 2ème Année—

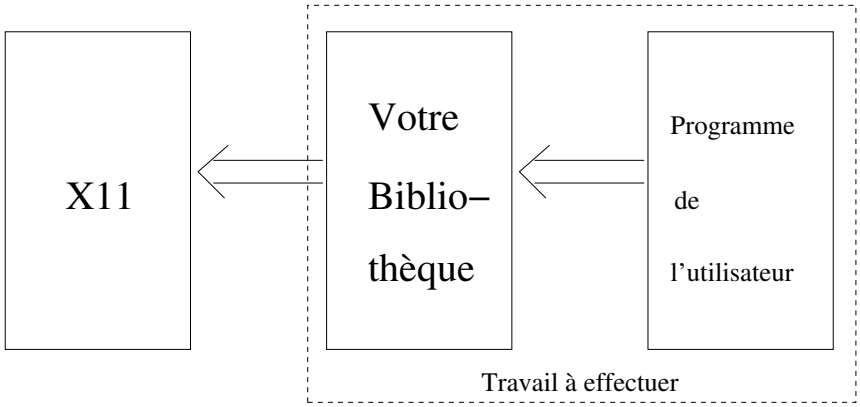
Bibliothèque graphique[1]

Les bibliothèques graphiques comme `GTK`, `allegro`, `qt`, et même `libmlv` sont basées sur `Xlib` (ou `X11`), dans lequel sont développés des concepts de base comme les fenêtres, les événements, ... Un bouton, par exemple, n'est rien de plus qu'une fenêtre qui réagit au clique de souris.

Le but de ce projet est donc de créer une bibliothèque, basée sur `X11`, qui permettra à son utilisateur de programmer facilement une interface graphique dotée d'objets complexes, comme un menu déroulant, un ascenseur...

Commencez par récupérer un fichier d'exemple en `X11` à l'adresse suivante :
`http ://etudiant.univ-mlv.fr/~jdavid01/Enseignement.php`
La ligne de commande nécessaire à la compilation du programme se trouve dans le fichier. Familiarisez vous avec le programme, afin de comprendre comment sont gérés : la création de la fenêtre principale, les évènements, les couleurs...

1 Une bibliothèque ?



Comme l'illustre ce schéma, le programmeur utilisant votre bibliothèque ne devra sous aucun prétexte être obligé de faire appel à une fonction `X11`. Vous devrez donc lui fournir, entre autres, les fonctions suivantes :

[1] Sujet disponible à l'adresse : `http://etudiant.univ-mlv.fr/~jdavid01/Enseignement.php`

- créer une fenêtre et la détruire.
- récupérer évènement souris/clavier
- Changer la couleur d'un élément (Background/Foreground).
- Les fonctions de dessins.
- ...

Ces fonctions feront directement appel a celle de `X11`.
Vous implanterez ensuite des fonctions plus complexes comme :

- créer un bouton.
- créer un menu.
- ...

`ATTENTION : SEUL UNE FENÊTRE PRINCIPALE EST UN OBJET X11. LES OBJETS PLUS COMPLEXES SERONT DÉCRITS A LA MAIN PAR VOS SOINS !`

1.1 Les `Widget`

Un `Widget` est un type englobant tous les objets graphiques.
Il contient les informations suivantes :

- Un lien vers la fenêtre `X11` dont le `Widget` dépend.
- Un pointeur vers le `Widget` parent, de valeur `NULL` s'il s'agit d'une fenêtre `X11`.
- Une structure de donnée qui permet d'accéder aux `Widget` fils. Le choix de cette structure devra être justifié dans le rapport. Pour chaque `Widget` fils, on aura accès ses coordonnées relatives.
- Coordonnées absolues du `Widget`.
- La longueur et la largeur.
- La couleur du background, du foreground.
- Des informations sur la bordure (optionnel).

Un `Widget-Actionable` pourra contenir un pointeur de fonction.
Un `Widget-Zone de texte` contiendra un `char *`.
Une fenêtre `X11` est un `Widget`.

1.2 Exemples

1.2.1 Fenêtre Principale

La fenêtre principale est un objet `X11`. Les fonctions de création, d'affichage et de suppression feront donc appel aux fonctions `X11` que vous trouverez dans l'exemple `x11Exemple.c`

1.2.2 Bouton

Un bouton est forcément contenu dans une fênetre. Lors de sa création vous devrez donc l'apparenter a un `Widget`. A l'affichage, on dessine dans la fenêtre un rectangle plein de la couleur du `background` du bouton. Si le bouton contient du texte, il contient un Widget fils "Zone Texte" que l'on va a son tour afficher. Si l'utilisateur clique sur le

bouton, on doit pouvoir le savoir sans parcourir toute la liste des `Widget`. On récupère donc un pointeur sur le `Widget-Bouton` et on exécute le pointeur de fonction qui y est associée.

1.2.3 Zone texte

De la même manière que le bouton, la zone texte à forcément un `Widget` parent. A l'affichage, on dessine le rectangle du `background`, qui sera généralement de la même couleur que le `Widget` parent, mais on laisse le choix a l'utilisateur. On affiche ensuite le texte aux coordonnées indiquées à l'aide de la commande `XDrawString`.

1.2.4 Check-box

Un check-box un bouton associé a une variable booléenne. Si la variable vaut 1, on dessine un croix dans le bouton.

1.2.5 Ascenseur

Un ascenseur est un `Widget` complexe. Il contient 2 boutons qui vont incrémenter et décrémenter la variable a laquelle l'ascenseur est associé et un bouton plus large contenant un rectangle dont la position et la taille dépend de la valeur et du maxima de la variable. Lorsque l'on clique dans ce dernier, la valeur de la variable sera modifiée en fonction des coordonnées de la souris.

2 Niveaux

2.1 Niveau 1

- Implantez toutes les structures de données. La méthode de hiérarchisation des `Widget` devra être réfléchie et justifiée dans le rapport.
  Les algorithmes de manipulation de ces structures devront être optimisé le mieux possible.
  `CE POINT EST LE PLUS IMPORTANT DU PROJET. FAITES DONC ATTENTION ET SOYEZ SÛR DE VOS STRUCTURES`
- L'affichage est en mode `ASCII` : l'appel aux fonctions de votre bibliothèque ne fait que des `printf`, mettant en évidence le bon fonctionnement des structures.
- Deux `Widget` frères ne peuvent pas se superposer.
- Définissez des `Widgets` conteneurs NON-ACTIONNABLE : fenêtres, zones textes...

2.2 Niveau 2

- La bibliothèque fait appel aux fonctions `X11` : on obtient un rendu graphique simple : les couleurs du background/foreground ne sont pas pris en compte.
- Deux `Widget` frères peuvent se superposer : l'utilisateur précise si c'est le cas. A l'affichage, si l'on clique sur un `Widget`, il passe au dessus de tous les autres.

– Créer des `Widget` actionnables simples : bouton, check-box...

2.3 Niveau 3

– Les couleurs du background/foreground sont pris en compte.
– Créer des `Widget` actionnables complexes : menus, ...
– On doit pouvoir sélectionner un ensemble de `Widget` à la souris. Chaque `Widget` possède donc deux structures pour stocker ses fils : les sélectionnés et les non-sélectionnés. On doit pouvoir déplacer facilement les éléments d'une structure à l'autre. À l'affichage, on doit pouvoir voir que les objets ont été sélectionnés.

2.4 Niveau 4

– Rendre possible le déplacement d'un `Widget` ou d'un ensemble de `Widget` dans le même parent ou dans un parent différent.
– Créer des `Widget` complexes : ascenseur, zone image...

2.5 Bonus

– Des `Widget` non rectangulaires.
– Créer le .so (voir votre cours de Système)
– Des `Widget` originaux...

3 Modalités

Le projet devra être envoyé par mail à votre chargé de TP avant la date limite, sous la forme d'un fichier `tar.gz` (compilez avec la commande `tar czvf NOM1_NOM2_L2.2.tgz REPERTOIRE/`)) Les sources seront organisés de la manière suivante :
– Chaque fichier contiendra des commentaires, aidant à la compréhension du code source.
– Le répertoire `lib/` : les fichiers de votre bibliothèque.
– Le répertoire `src/` : les sources d'un petit programme utilisant la bibliothèque : `Ce programme sera exécuté lors de la soutenance`
– Le répertoire `doc/` : le rapport.
– Un fichier `Makefile`.
– Un fichier `README` contenant les instructions utiles a l'examinateur : option du makefile, nom de d'exécutable...

Concentrez vous au maximum sur les structures et ne passez au graphique que lorsque vous serez sur de vos choix.
`Chaque niveau doit être implanté correctement avant que vous ne passiez au niveau suivant. Faire le niveau 1 correctement équivaut déjà à une note supérieure où égale a la moyenne.`