Travaux pratiques d'informatique
CSI3 - Année 2013-2014
TP n°5 : synthèse et graphisme

Première partie

Placez l'archive *tpGfx.tgz* dans le dossier partagé de votre machine virtuelle, puis côté Ubuntu, déplacez cette archive dans votre dossier personnel, à l'endroit désiré. Décompressez cette archive. Placez vous en ligne de commande dans le dossier de cette archive. Faites un *make clean*. Que s'est-il passé ? Faites un *make libisentlib*. Que s'est-il passé ? Faites un *make*. Que s'est-il passé ? Analysez le contenu du fichier *makefile* pour vous aider.

Lancez ensuite le programme qui vient d'être généré. Observez tout ce qu'il fait. Pour le quitter, appuyez sur la touche 'Q' lorsque la fenêtre graphique est au premier plan.

Ouvrez le code source de cet exécutable, qui porte le nom *exemple.c*. Essayez de comprendre ce que fait le programme, à l'aide des indications qui suivent, puis « jouez » avec le code en essayant de le modifier en divers endroits et en analysant ce qui arrive suite à vos modifications.

Le programme commence évidemment par le *main*, qui prend en fait deux paramètres. Ces paramètres sont fournis à la fonction d'initialisation de l'architecture graphique.

```
int main(int argc, char **argv)
{
    initialiseGfx(argc, argv);
```

Cette ligne prépare simplement une fenêtre graphique de nom et de taille données.

```
    prepareFenetreGraphique("OpenGL", LargeurFenetre, HauteurFenetre);
```

Nous avons ensuite la ligne suivante :

```
    lanceBoucleEvenements();
```

Grâce à cette instruction, nous lançons la boucle de gestion des événements. Le système va gérer pour nous tous les clics souris, appuis de touche clavier, demandes de redessin de la fenêtre, et nous en avertir automatiquement à l'aide d'événements.

Il nous en avertit en appelant la fonction *gestionEvenement*, et en passant à cette fonction le numéro d'événement qui convient. Vous pouvez trouver tous les numéros d'événements gérés par *ISENTlib* dans le fichier *GfxLib.h*, au tout début de ce fichier d'en-tête, dans l'énumération *EvenementGfx*.

Passez donc pour l'instant rapidement la fonction *cercle* et intéressons-nous à la fonction *gestionEvenement*.

La fonction *gestionEvenement* est en fait le cœur de notre programme graphique :

```
void gestionEvenement(EvenementGfx evenement)
{
     ...
    switch (evenement)
     {
```

En fonction de l'événement passé en paramètre, cette fonction va aiguiller le déroulement du programme vers tel ou tel cas grâce à une structure de contrôle *switch*.

```
        case Initialisation:
            ...
            break;

        case Affichage:
            effaceFenetre (0, 0, 0);
            ...
            break;

        case Clavier:
            ...
            switch (caractereClavier())
              {
                ...
                case 'Q': /* Pour sortir proprement du programme */
                case 'q':
                    exit(0);
                    break;
                ...
```

Le message *Initialisation* est le premier message reçu par *gestionEvenement*. Il permet entre autres de préparer la suite du programme.

Le message *Affichage* est envoyé automatiquement par le système si la fenêtre doit être redessinée (premier affichage de la fenêtre, fenêtre partiellement ou totalement recouverte puis découverte, demande explicite de redessin, etc).

Il commence par effacer la fenêtre avec la couleur noire (0 de rouge, 0 de vert et 0 de bleu), puis à la place des « … » on met les ordres graphiques à envoyer à la carte. Tous les ordres graphiques doivent être mis dans le case *Affichage*, et jamais ailleurs.

Le message *Clavier* est envoyé par le système dès qu'une touche du clavier est appuyée, et grâce à la fonction *caractereClavier* on peut récupérer le caractère correspondant à la touche. Ici, 'Q' termine notre programme.

```
                case 'R':
                case 'r':
                    rafraichisFenetre();
                    break;
```

Par exemple, appuyer sur la touche 'R' demande au système de forcer un rafraîchissement de la fenêtre, ce qui va faire en sorte que le système d'exploitation nous envoie dès que possible un message *Affichage*. C'est la « bonne manière » de forcer un rafraîchissement de la fenêtre, c'est-à-dire que c'est une manière coopérative à la fois avec le sys-

tème d'exploitation (qui peut envoyer le message *Affichage* au meilleur moment pour optimiser l'affichage) et avec les autres programmes qui tournent sur la machine (et dont l'affichage peut être plus prioritaire que le vôtre).

Nous vous laissons explorer les autres messages. La lecture de *GfxLib.h* vous donnera de précieuses indications sur tout ce qu'il est possible de faire avec la partie graphique d'*ISENTlib*.

Expérimentez un moment avec le code fourni, en modifiant les ordres de tracé de rectangle, de ligne, de texte, etc.

Deuxième partie (volet A)

Dans n'importe quel programme moderne, avoir accès à une interface graphique va nous permettre de contrôler notre programme à la souris ou par pression d'une touche du clavier.

Nous allons essayer de construire une petite interface correspondant à ce que vous avez déjà élaboré :

- élaborez une nouvelle structure de données, nommée *Bouton*, comprenant comme champs au minimum une chaîne de caractères pour le nom du bouton, ainsi que les informations permettant de décrire un rectangle englobant ce texte (astuce : peut-être pourriez-vous réutiliser la structure *Rectangle* créée dans les TP précédents…) ;
- écrivez une fonction permettant d'initialiser un *Bouton* à partir de son texte, ses coordonnées et sa taille (mais peut-être avez-vous déjà écrit du code similaire durant les précédents TP…) ;
- écrivez une fonction permettant de tracer un *Bouton* sur l'écran (texte + cadre : la fonction *tailleChaine* renvoie le nombre de points écran que prend une chaîne arbitraire passée en paramètre pour s'afficher dans la fenêtre, cherchez-la dans *GfxLib.h* afin de connaître ses paramètres et son type de retour) ;
- définissez en *static* un tableau de *Bouton* correctement initialisés (demandez éventuellement la signification de *static* à votre enseignant, et pourquoi il peut être utile ici) ;
- ajoutez, dans la partie de gestion de l'événement initialisation, l'initialisation de ce tableau de *Bouton* ;
- ajoutez, dans la partie de gestion de l'événement affichage, le code qui permet l'affichage du tableau de *Bouton* ;
- ajoutez dans la partie de gestion des événements de bouton souris (*case BoutonSouris*) le code qui, lors d'un clic, vérifie que ce clic a été ou non réalisé sur un bouton, puis qui lance éventuellement une action associée (apparition d'un symbole graphique en fonction du bouton cliqué par exemple).

Améliorez la qualité de votre interface en travaillant le graphisme, en proposant des raccourcis clavier (par exemple appui sur la touche *S* pour sauvegarder à l'identique d'un clic sur le bouton *Sauvegarder*), etc.

Deuxième partie (volet B)

Réfléchissez à une solution permettant à l'utilisateur de rentrer une chaîne de caractères alphanumériques au clavier, mais en mode graphique.

(aide : il faut en fait intercepter les événements clavier, voir si l'on est en mode acquisition d'une chaîne, et le cas échéant ajouter le caractère tapé à une chaîne dite « buffer »)