man mlx_loop

MiniLibX 3 " 19 septembre 2002 ".
NAME
MiniLibX - Gérer les événements
SYNOPSYS
int mlx_loop(void *mlx_ptr) ;
int mlx_key_hook(void *win_ptr, int (*funct_ptr)(), void *param) ;
int mlx_mouse_hook(void *win_ptr, int (*funct_ptr)(), void *param) ;
int mlx_expose_hook(void *win_ptr, int (*funct_ptr)(), void *param) ;
int mlx_loop_hook(void *mlx_ptr, int (*funct_ptr)(), void *param) ;

EVENEMENTS

Le système graphique est bidirectionnel. D'une part, le programme envoie des ordres à l'écran pour afficher des pixels, des images, etc. D'autre part, il peut obtenir des informations du clavier et de la souris associés à l'écran. Pour ce faire, le programme reçoit des "événements" du clavier ou de la souris.

DESCRIPTION

Pour recevoir des événements, vous devez utiliser mlx_loop(). Cette fonction ne revient jamais. Il s'agit d'une boucle infinie qui attend un événement, puis appelle une fonction définie par l'utilisateur associée à cet événement.
Un seul paramètre est nécessaire, l'identifiant de connexion mlx_ptr (voir le manuel mlx).

Vous pouvez affecter des fonctions différentes aux trois événements suivants :

- Une touche est pressée
- Le bouton de la souris est enfoncé
- Une partie de la fenêtre doit être redessinée
  (ceci est appelé un événement "exposant", et c'est le travail de votre programme de le gérer dans l'environnement X11 d'Unix/Linux, mais à l'inverse cela n'arrive jamais sous MacOS).

Chaque fenêtre peut définir une fonction différente pour le même événement.

Les trois fonctions mlx_key_hook (), mlx_mouse_hook() et mlx_expose_hook() fonctionnent exactement de la même manière.
funct_ptr est un pointeur vers la fonction que vous souhaitez voir appelée lorsqu'un événement se produit. Cette affectation est spécifique à la fenêtre définie par l'identifiant win_ptr. L'adresse param sera transmise à la fonction chaque fois qu'elle sera appelée, et devrait être utilisée pour stocker les paramètres dont elle pourrait avoir besoin.

La syntaxe de la fonction mlx_loop_hook() est identique aux précédentes, mais la fonction donnée sera appelée lorsqu'aucun événement ne se produit.

Lorsqu'il attrape un événement, le MiniLibX appelle la fonction correspondante avec des paramètres fixes :
expose_hook(void *param) ;
key_hook(int keycode, void *param) ;
mouse_hook(int button, int x, int y, void *param) ;
loop_hook(void *param) ;

Ces noms de fonctions sont arbitraires. Ils sont utilisés ici pour distinguer les paramètres en fonction de l'événement. Ces fonctions ne font PAS partie de la MiniLibX.

param est l'adresse spécifiée dans les appels mlx_*_hook. Cette adresse n'est jamais utilisée ni modifiée par le MiniLibX. Lors d'événements de touches et de souris, des informations supplémentaires sont passées : keycode vous indique quelle touche a été pressée (avec X11, cherchez le fichier include "keysymdef.h", avec MacOS, essayez simplement :)),(x, y) sont les coordonnées du clic de souris dans la fenêtre, et button vous indique quel bouton de souris a été pressé.

ALLER PLUS LOIN AVEC LES ÉVÉNEMENTS
La MiniLibX fournit un accès beaucoup plus générique à d'autres événements disponibles. L'include mlx.h définit mlx_hook() de la même manière que les fonctions mlx_*_hook fonctionnent. Les valeurs de l'événement et du masque seront prises dans le fichier X11 include "X.h". Certains événements MacOS sont mappés à ces valeurs, quand cela a un sens, et le masque n'est pas utilisé dans MacOS.

Consultez le code source de la MiniLibX pour savoir comment elle appellera votre propre fonction pour un événement spécifique.

VOIR AUSSI

mlx(3), mlx_new_window(3), mlx_pixel_put(3), mlx_new_image(3)