42 Docs

Events

TABLE OF CONTENTS

Introduction

Events are the foundation of writing interactive applications in MiniLibX. It is therefore of essence that you fully comprehend this chapter as this will be of use in your future graphical projects.

All hooks in MiniLibX are nothing more than a function that gets called whenever a event is triggered. Mastering all these events won't be neccessary, however, we will quickly go over each X11 event accordingly.

MacOS version

Note: On MacOS - Cocoa (AppKit) and OpenGL - version, minilibx has partial support of X11 events and doesn't support X11 mask (x_mask argument of mlx_hook is useless, keep it at 0).

Supported events:

```
enum {
	ON_KEYDOWN = 2,
	ON_KEYUP = 3,
	ON_MOUSEDOWN = 4,
	ON_MOUSEUP = 5,
	ON_MOUSEMOVE = 6,
	ON_EXPOSE = 12,
	ON_DESTROY = 17
};

// usage:
mlx_hook(vars.win, ON_DESTROY, 0, close, &vars);
```

X11 interface

X11 is the library that is used alongside of MiniLibX. It therefore is no secret that this header is very useful for finding all the according events of MiniLibX.

X11 events

There are a number of events to which you may describe.

Key	Event		Key	Event	
02	KeyPress		14	NoExpose	
03	KeyRelease		15	VisibilityNotify	
04	ButtonPress		16	CreateNotify	
05	ButtonRelease		17	DestroyNotify	
06	MotionNotify		18	UnmapNotify	
07	EnterNotify		19	MapNotify	
08	LeaveNotify		20	MapRequest	
09	FocusIn		21	ReparentNotify	

Key	Event		Key	Event	
`10`	FocusOut		`22`	ConfigureNotify	
`11`	KeymapNotify		`23`	ConfigureRequest	
`12`	Expose		`24`	GravityNotify	
`13`	GraphicsExpose		`25`	ResizeRequest	

If you can't figure out what some events are, don't worry, because you probably won't need them. If you do, go read the documentation of each X11 events.

X11 masks

Each X11 event, also has an according mask. This way you can register to only one key when it triggers, or to all keys if you leave your mask to the default. Key masks therefore allow you to whitelist / blacklist events from your event subscriptions. The following masks are allowed:

Mask	Description		Mask	Description
`0L`	NoEventMask		`(1L<<12)`	Button5MotionMask
`(1L<<0)`	KeyPressMask		`(1L<<13)`	ButtonMotionMask
`(1L<<1)`	KeyReleaseMask		`(1L<<14)`	KeymapStateMask
`(1L<<2)`	ButtonPressMask		`(1L<<15)`	ExposureMask
`(1L<<3)`	ButtonReleaseMask		`(1L<<16)`	VisibilityChangeMas
`(1L<<4)`	EnterWindowMask		`(1L<<17)`	StructureNotifyMask
`(1L<<5)`	LeaveWindowMask		`(1L<<18)`	ResizeRedirectMask
`(1L<<6)`	PointerMotionMask		`(1L<<19)`	SubstructureNotifyM
`(1L<<7)`	PointerMotionHintMask		`(1L<<20)`	SubstructureRedirec
`(1L<<8)`	Button1MotionMask		`(1L<<21)`	FocusChangeMask
`(1L<<9)`	Button2MotionMask		`(1L<<22)`	PropertyChangeMas

Mask	Description		Mask	Description
`(1L<<10)`	Button3MotionMask		`(1L<<23)`	ColormapChangeMa
`(1L<<11)`	Button4MotionMask		`(1L<<24)`	OwnerGrabButtonM

Hooking into events

mlx_hook

Hooking into events is one of the most powerful tools that MiniLibX provides. It allows you to register to any of the aforementioned events with the call of a simple hook registration function.

To achieve this, we call the function `mlx_hook`.

```
void mlx_hook(mlx_win_list_t *win_ptr, int x_event, int x_mask, int (*f)(), void *param)
```

Some version of mlbx doesn't implement `x_mask` and whatever the value there will be no mask.

Prototype of event functions

Event functions have a different prototype depending of the hooking event.

Hooking event	code	Prototype
ON_KEYDOWN	2	`int (*f)(int keycode, void *param)`
ON_KEYUP*	3	`int (*f)(int keycode, void *param)`
ON_MOUSEDOWN*	4	`int (*f)(int button, int x, int y, void *param)`
ON_MOUSEUP	5	`int (*f)(int button, int x, int y, void *param)`
ON_MOUSEMOVE	6	`int (*f)(int x, int y, void *param)`
ON_EXPOSE*	12	`int (*f)(void *param)`
ON_DESTROY	17	`int (*f)(void *param)`

**Has mlx_hook alias.*

Hooking alias

Minilibx api has some alias hooking function:

`mlx_expose_hook` function is an alias of mlx_hook on expose event (`12`).
`mlx_key_hook` function is an alias of mlx_hook on key up event (`3`).
`mlx_mouse_hook` function is an alias of mlx_hook on mouse down event (`4`).

Example

For example instead of calling `mlx_key_hook`, we can also register to the `KeyPress` and `KeyRelease` events. Lets take a look:

```
#include <mlx.h>

typedef struct  s_vars {
      void *mlx;
      void *win;
}                   t_vars;

int   close(int keycode, t_vars *vars)
{
      mlx_destroy_window(vars->mlx, vars->win);
      return (0);
}

int   main(void)
{
      t_vars    vars;

      vars.mlx = mlx_init();
      vars.win = mlx_new_window(vars.mlx, 1920, 1080, "Hello world!");
      mlx_hook(vars.win, 2, 1L<<0, close, &vars);
      mlx_loop(vars.mlx);
}
```

Here we register to the `KeyPress` event with the according `KeyPressMask`. Now whenever we press a key, the window will be closed.

Test your skills!

Now that you have a faint idea of how all of this works, we encourage you to make the hook handlers. Whenever the:

- ESC key is pressed, your window should close.
- window is resized, you should print something in your terminal.
- the red cross is clicked, your window should close.
- you press a key longer than x seconds, you should print something in your terminal.
- mouse enters the window, you should print `Hello!` in your terminal, when it leaves, you should print `Bye!`.