Live Wallpapers for the X Window System

John Tsiombikas

nuclear@member.fsf.org
Live Wallpapers?

- Animated desktop wallpapers as opposed to static pictures
- Inspired from android
Window Managers are regular X clients
no special privileges

WM conventions
- ICCCM
- Motif WM Hints
- NetWM protocols

Inter-Client Communications Conventions Manual
Animated desktop?

- X client drawing continuously on the desktop
  - Needs to be fast & efficient
  - Must keep redraw rate to a minimum

- OpenGL context bound to the desktop

What is the desktop though?
Finding the desktop...

Attempt #1: Root window

```
win = RootWindow(dpy);
ctx = glXCreateContext(dpy, ...);
glxMakeCurrent(dpy, win, ctx);
```
Finding the desktop......

Attempt #2: Virtual root window

Window manager frame

WM virtual root window

NetWM property: _NET_VIRTUAL_ROOTS

X server root window
Finding the desktop ........

NetWM property:
_NET_VIRTUAL_ROOTS

Dennis Ritchie
Window manager
Finding the desktop ........

Attempt #3: give up...
xpenguins already did this

win = ToonGetRootWindow(dpy, ...);
Finding the desktop ........

Attempt #3: give up...
xpenguins already did this

win = ToonGetRootWindow(dpy, ...);

Sadly still not enough...
Finding the desktop

KDE

I'll make my own desktop window with blackjack, and hookers

$ xlivebg -n

- Create new window
- Set the property: _NET_WM_WINDOW_TYPE
  _NET_WM_WINDOW_TYPE_DESKTOP
Xlivebg project

Goal: framework for 3rd party live wallpapers

- Must make it really simple to implement live wallpapers, by handling:
  - All interactions with the X window system
  - OpenGL context creation and setup
  - Configuration and options management
  - Providing helpers for common tasks
  - Background images loading & management
Xlivebg parts

Xlivebg program

- live wallpaper
- live wallpaper
- live wallpaper

plugins (so)

Xlivebg-cmd
Xlivebg-gui
Configuration files

~/.xlivebg/config
~/.config/xlivebg.conf
/etc/xlivebg.conf

xlivebg {
  active = "ripple"

  # background image
  image = "bgimage.jpg"
  fit = "stretch"

  # --- plugin-specific configuration ---
  distort {
    amplitude = 0.025
    frequency = 8.0
  }

  ripple {
    raindrops = 0
  }
}

} } search paths
global settings

settings for the "distort" wallpaper
settings for the "ripple" wallpaper
Interactive configuration

- AF_UNIX socket: /tmp/xlivebg.sock
- Command-line client: xlivebg-cmd
- plugin property description

```plaintext
proplist {
  prop {
    id = "count"
    desc = "number of foo"
    type = "integer"
    range = [500, 5000]
  }
  prop {
    id = "size"
    desc = "foo size"
    type = "number"
    range = [0.25, 4.0]
  }
  prop {
    id = "video"
    desc = "background video file"
    type = "filename"
  }
}

wallpaper plugins provide a list of tweakable parameters they support
```
Interactive configuration GUI

- Communicates through the socket
- Left pane: general settings
- Right pane: auto-generated active plugin prop. UI

Global settings

Plugin properties

Save to config file
Bundled live wallpapers (v1.0)

- Color cycling
- Ripple effect
- Starfield
- Distortion
- Video playback
**Plugin Example (1/2)**

```c
#define PROPLIST
"proplist \n" prop \n" id = "speed\n" desc = "animation speed\n" type = "number\n" range = [0, 10]\n" }
"
"
static struct xlivebg_plugin plugin = {
  "minimal",
  "Minimal live wallpaper example",
  PROPLIST,
  XLIVEBG_20FPS,
  init, 0,
  start, 0,
  draw, prop,
  0, 0
};
static float speed;

int register_plugin(void)
{    return xlivebg_register_plugin(&plugin);
}
```

```c
struct xlivebg_plugin {
  char *name, *desc;
  char *props;
  long upd_interval;
  xlivebg_init_func init;
  xlivebg_cleanup_func cleanup;
  xlivebg_start_func start;
  xlivebg_stop_func stop;
  xlivebg_draw_func draw;
  xlivebg_prop_func prop;
  void *data, *so;
};
```

```
in xlivebg.h

plugin init

so = dlopen("foo.so", ...);
reg = dl_sym(so, "register_plugin");
reg();
```

**Requested Redraw Rate**

**Property List**

- id = "speed"
- desc = "animation speed"
- type = "number"
- range = [0, 10]

**Callbacks**
plugin example (2/2)

```c
static int init(void *cls)
{
    xlivebg_defcfg_num("xlivebg.minimal.speed", 1.0f);
    return 0;
}

static void start(long tmsec, void *cls)
{
    prop("speed", 0);
}

static void prop(const char *prop, void *cls)
{
    if(strcmp(prop, "speed") == 0) {
        speed = xlivebg_getcfg_num("xlivebg.minimal.speed", 1.0f);
    }
}

static void draw(long tmsec, void *cls)
{
    int i, num_scr = xlivebg_screen_count();
    xlivebg_clear(GL_COLOR_BUFFER_BIT);

    /* for every screen ... */
    for(i=0; i<num_scr; i++) {
        xlivebg_gl_viewport(i);
        /* ... draw using OpenGL ... */
    }
}
```

Callbacks
- default value if not configured
- called on plugin activation
- called when a property is modified
- called continuously to redraw
Links

- web site:
  http://nuclear.mutantstargatoat.com/sw/xlivebg

- github repo:
  https://github.com/jtsiomb/xlivebg

- Setup & demo video:
  https://www.youtube.com/watch?v=JZ_RXO6BWPD8
Links

- web site:
  http://nuclear.mutantstargoat.com/sw/xlivebg

- github repo:
  https://github.com/jtsiomb/xlivebg

- setup & demo video:
  https://www.youtube.com/watch?v=JZ_RXnBWPD8

Thanks for watching!