

open



USE



IMPROVE



EVANGELIZE

Modern Windows Manager on Gnome

Erwann Chenede & Chi Wang
{[erwann.chenede](mailto:erwann.chenede@sun.com),[chris.wang](mailto:chris.wang@sun.com)}@sun.com
OPG, Sun Microsystems

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை



Syllabus

- Background knowledge of Windows Manager
- Case Study: Metacity
- Case Study: Compiz
- Q& A



What is Windows Manager

- A special purpose application that provides the capability of applications to be moved, resized, minimized, and restored dynamically by the user.
- Applies decoration to an application that enable the user to access these features.



Compositing Windows Manager

- Instead of outputting to a common screen, programs each output first to a separate and independent buffer, or temporary location inside the computer, where they can be manipulated before they are shown.



X Windows Manager

- An X window manager is a window manager which runs on top of the X Window System, a windowing system mainly used on Unix-like systems.



Popular X Windows Manager

- Enlightenment
- **Metacity** (the current default for the GNOME desktop environment)
- MWM (Motif Window Manager), Motif Window Manager
- Sawfish (a past default for GNOME, originally called Sawmill)
- Xfwm4 (a window manager for the Xfce desktop environment)
- **Compiz**
- Xfwm (default for Xfce)
- KWin (originally called KWM, default for KDE and has compositing option since 4.0)
- twm (default for the X Window System since version X11R4)



ICCCM: Inter-Client Communication Conventions Manual

- Standard for interoperability between X Window System clients of the same X server
- The ICCCM specifies cut and paste buffers, window manager interaction, session management, how to manipulate shared resources and how to manage device colors.



EWMH: Extended Window Manager Hints

- Builds on the Inter-Client Communication Conventions Manual
- Standardized set of ICCCM additional custom extensions that any desktop environment can adopt.
- Defines interactions between window managers, compositing managers, applications, and the utilities that form part of a desktop environment.

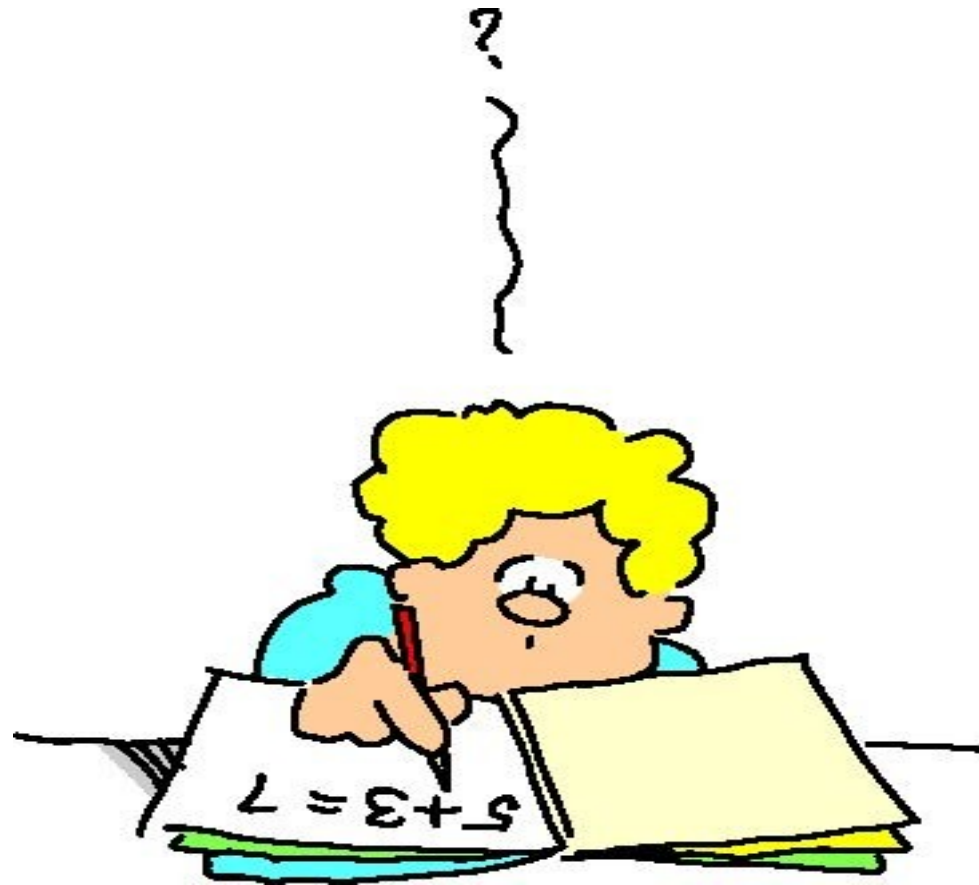


Conclusion

- Generally provide similar functionality with common management features applied to windows
- Difference in Look-and-Feel and decorations



Case Study





Metacity

- Metacity is the default Windows manager on Gnome
- Metacity's focus is on simplicity and usability rather than novelties or gimmicks.
- Metacity implements much of the EWMH window manager specification, as well as the older ICCCM.
- Replaceable with other ICCCM-compliant WM
- The "libwnck" library is available for writing Metacity extensions
- Metacity has a simple theme system and a couple of extra themes come with it.



Metacity Theme

- A Metacity theme is based on a defined XML format
- A number of images
- Themes are stored at :
 - `$HOME/.themes/[theme_name]/metacity-1/`
 - `PREFIX/share/themes/[theme_name]/metacity-1/`
- A huge number of such themes can be downloaded from GNOME's art site, art.gnome.org



metacity-theme-1.xml

- metacity-theme-1.xml is the file that contains the XML description for the theme

```
<?xml version="1.0"?>
<metacity_theme>
  <info>
    <name>Nimbus Theme 1.0</name>
    <author>Design Mike Stern - implementation Erwann Chenede</author>
    <copyright>Sun Microsystems Inc. 2006</copyright>
    <date>2003-02-15</date>
    <description>Sun's Nimbus theme for metacity</description>
  </info>
  <!-- This is where we need to start specifying your theme -->
</metacity_theme>
```

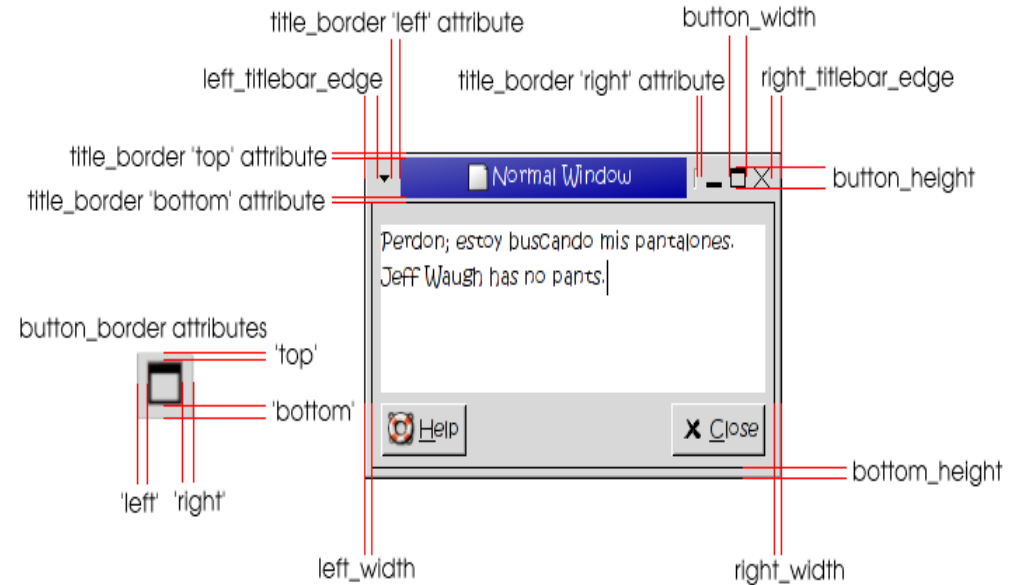
Frame Geometry

- The frame geometry name is referenced later on by a given 'frame style'.
- Can use inheritance to simply overwrites any values inherited from the parent.

```

<frame_geometry name="normal_geometry">
  <distance name="left_width" value="6"/>
  <distance name="right_width" value="6"/>
  <distance name="bottom_height" value="7"/>
  <distance name="left_titlebar_edge" value="6"/>
  <distance name="right_titlebar_edge" value="6"/>
  <distance name="button_width" value="17"/>
  <distance name="button_height" value="17"/>
  <distance name="title_vertical_pad" value="4"/>
  <border name="title_border" left="3" right="12" top="4" bottom="3"/>
  <border name="button_border" left="0" right="0" top="1" bottom="1"/>
</frame_geometry>

```



```

<frame_geometry name="borderless_geometry" rounded_top_left="true" rounded_top_right="true" parent="normal_geometry">
  <distance name="left_width" value="0"/>
  <distance name="right_width" value="0"/>
</frame_geometry>

```

Drawing operations

- In order to successfully draw a part of the frame, you will need to specify a drawing operation for that 'frame piece'.

```

<draw_ops name="close_button_pressed">
  <image filename="button-close-icon-pressed.png" x="0" y="0" width="width" height="height" />
</draw_ops>

<draw_ops name="menu_button_pressed">
  <image filename="button-menu-icon-pressed.png" x="0" y="0" width="width" height="height" />
</draw_ops>

<draw_ops name="title_text_focused">

  <icon x="(3 `max` (width- (title_width + mini_icon_width + 5))) / 2"
        y="(((height - title_height) / 2) `max` 0)"
        width="mini_icon_width"
        height="mini_icon_height"/>

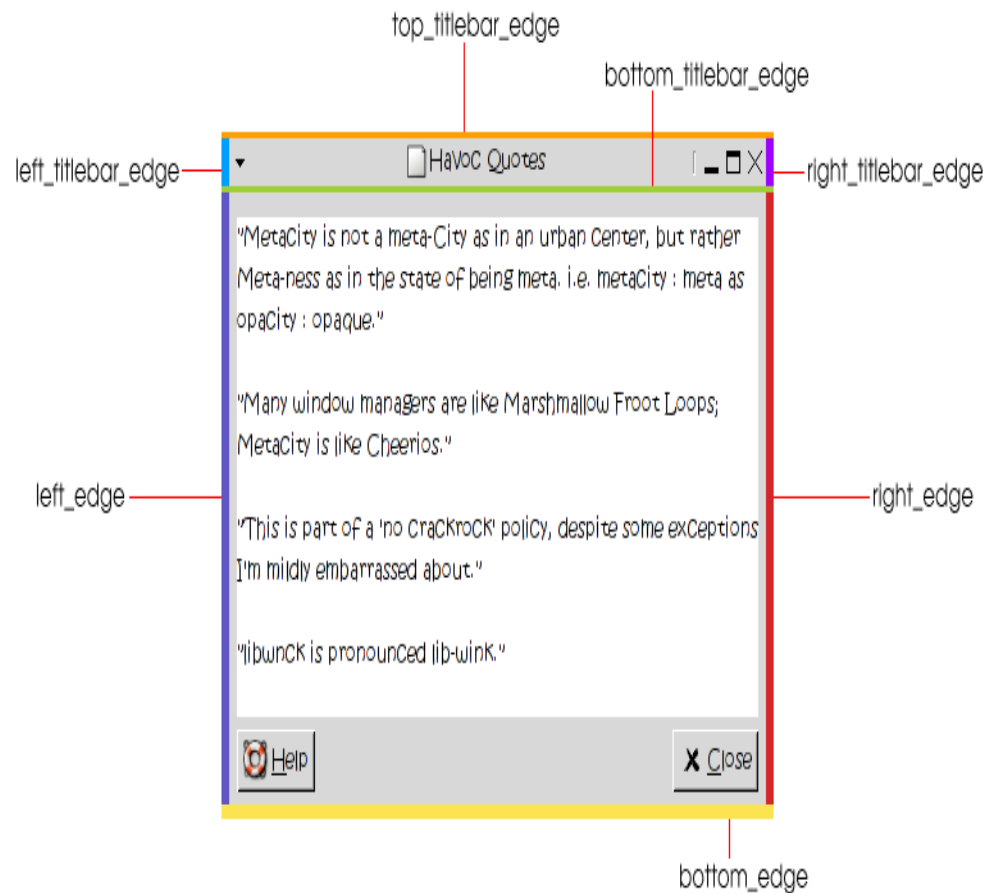
  <title color="#c4c7ce"
        x="(((3 `max` (width- (title_width + mini_icon_width + 5))) / 2) + (mini_icon_width + 5))"
        y="(((height - title_height) / 2) `max` 0) + 1"/>

  <title color="black"
        x="(((3 `max` (width- (title_width + mini_icon_width + 5))) / 2) + (mini_icon_width + 5))"
        y="(((height - title_height) / 2) `max` 0)"/>

</draw_ops>

```

Frame piece and Window Button



Frame Style

- A 'frame style' tie in the various different 'frame pieces' and 'window buttons' to a specific 'frame geometry'.
- Generally need to create a style for window states normal, maximized, shaded, maximized_and_shaded and depending on whether the window is in focus or not.
- If you omit any of the pieces, then nothing will be drawn for that piece

```

<frame_style name="normal_unfocused" geometry="normal">
  <piece position="entire_background" draw_ops="background_unfocused"/>
  <piece position="title" draw_ops="title_text_unfocused"/>
  <piece position="left_edge" draw_ops="left_edge_unfocused"/>
  <piece position="right_edge" draw_ops="right_edge_unfocused"/>
  <piece position="bottom_edge" draw_ops="bottom_edge_unfocused"/>
  <piece position="top_titlebar_edge" draw_ops="top_titlebar_unfocused"/>
  <piece position="bottom_titlebar_edge" draw_ops="bottom_titlebar_unfocused"/>

  <button function="close" state="normal" draw_ops="close_button_unfocused"/>
  <button function="close" state="prelight" draw_ops="close_button_prelight"/>
  <button function="close" state="pressed" draw_ops="close_button_pressed"/>
  <button function="minimize" state="normal" draw_ops="minimize_button_unfocused"/>
  <button function="minimize" state="prelight" draw_ops="minimize_button_prelight"/>
  <button function="minimize" state="pressed" draw_ops="minimize_button_pressed"/>

  <button function="maximize" state="normal" draw_ops="maximize_button_unfocused"/>
  <button function="maximize" state="prelight" draw_ops="maximize_button_prelight"/>
  <button function="maximize" state="pressed" draw_ops="maximize_button_pressed"/>

  <button function="menu" state="normal" draw_ops="menu_button_unfocused"/>
  <button function="menu" state="pressed" draw_ops="menu_button_unfocused"/>
</frame_style>

```



Frame style set

- A 'frame style set' maps frame styles onto the various window state
- The name attribute of a 'frame style set' is referenced later on by a a given 'window type'

```
<frame_style_set name="normal">  
  <frame focus="yes" state="normal" resize="both" style="normal_focused"/>  
  <frame focus="no" state="normal" resize="both" style="normal_unfocused"/>  
  <frame focus="yes" state="maximized" style="maximized_focused"/>  
  <frame focus="no" state="maximized" style="maximized_unfocused"/>  
  <frame focus="yes" state="shaded" style="normal_focused_shaded"/>  
  <frame focus="no" state="shaded" style="normal_unfocused_shaded"/>  
  <frame focus="yes" state="maximized_and_shaded" style="maximized_focused_shaded"/>  
  <frame focus="no" state="maximized_and_shaded" style="maximized_unfocused_shaded"/>  
</frame_style_set>
```



Windows

- Window type:normal, dialog, modal_dialog, menu, utility and border
- Each window type needs a style set

```
<window type="normal" style_set="my_normal_style_set"/>
```

```
<window type="dialog" style_set="my_dialog_style_set"/>
```

```
<window type="modal_dialog" style_set="my_modal_dialog_style_set"/>
```

```
<window type="menu" style_set="my_menu_style_set"/>
```

```
<window type="utility" style_set="my_utility_style_set"/>
```

```
<window type="border" style_set="my_border_style_set"/>
```



Menu Icons

- Specify icons for the menu entries Close, Maximize, UnMaximize and Minimize.
- It is enough to specify drawing operations for normal state only

```
<menu_icon function="close" state="normal" draw_ops="close_button_not_over"/>  
<menu_icon function="maximize" state="normal" draw_ops="maximize_button_not_over"/>  
<menu_icon function="unmaximize" state="normal" draw_ops="maximize_button_not_over"/>  
<menu_icon function="minimize" state="normal" draw_ops="minimize_button_not_over"/>
```



Conclusion

- Creating a Metacity theme will take a somewhat large amount of time.
- It is advisable to take existing themes and modify, instead of starting afresh.
- Theme comprised of images might look at times very tempting, consider that it takes an appreciable amount of time to render the theme.



Case Study: Compiz

- Conforms to the Inter-Client Communication Conventions Manual standard
- Compiz is the compositing window managers for the X Window System that is able to take advantage of OpenGL-acceleration.
- The integration allows it to perform compositing effects in window management
- Compiz is built on the Composite extension to X and the `GLX_EXT_texture_from_pixmap` extension to OpenGL.
- Compiz is designed with a highly extensible plugin system, these plugins can extend the basic functionality of Compiz



Window Decorator

- The Decoration plugin provides a window border for your windows
- Different decorators that you may use with the decoration plugin.
- GTK Window Decorator for use with Metacity themes
- Emerald - a custom Compiz window themer



Plugin design – Load plugin

- Plugin is loaded by compiz
- Calls a function to get a list of function pointers

```
CompPluginVTable *  
getCompPluginInfo20070830  
(void)  
{  
    return &shotVTable;  
}
```

```
static CompPluginVTable shotVTable = {  
    "screenshot",  
    shotGetMetadata,  
    shotInit,  
    shotFini,  
    shotInitObject,  
    shotFiniObject,  
    shotGetObjectOptions,  
    shotSetObjectOption  
};
```

Init and Fini

- shotInit is called when the plugin is loaded
- shotFini is called when the plugin is unloaded
- Allocate an index into an array that exists for each display which allows us to store a pointer to a custom structure

```

static Bool
shotInit (CompPlugin *p)
{
    if (!compInitPluginMetadataFromInfo (&shotMetadata,
                                         p->vTable->name,
                                         shotDisplayOptionInfo,
                                         SHOT_DISPLAY_OPTION_NUM,
                                         0, 0))

        return FALSE;

    displayPrivateIndex = allocateDisplayPrivateIndex ();
    if (displayPrivateIndex < 0)
    {
        compFiniMetadata (&shotMetadata);
        return FALSE;
    }
    compAddMetadataFromFile (&shotMetadata, p->vTable->name);
    return TRUE;
}

static void
shotFini (CompPlugin *p)
{
    freeDisplayPrivateIndex (displayPrivateIndex);
    compFiniMetadata (&shotMetadata);
}
    
```



Display Init and Fini

- Stores another custom array index, this time into an array of custom data structs for each screen
- Use Wrap() to make our function being called instead of normal function

```
#define WRAP(priv, real, func, wrapFunc) \  
    (priv)->func = (real)->func;    \  
    (real)->func = (wrapFunc) \  
#define UNWRAP(priv, real, func) \  
    (real)->func = (priv)->func
```



```

static Bool
shotInitDisplay (CompPlugin *p, CompDisplay *d)
{
    ShotDisplay *sd;

    if (!checkPluginABI ("core", CORE_ABIVERSION))
        return FALSE;
    sd = malloc (sizeof (ShotDisplay));
    if (!sd)
        return FALSE;

    if (!compInitDisplayOptionsFromMetadata (d, &shotMetadata, shotDisplayOptionInfo, sd->opt, SHOT_DISPLAY_OPTION_NUM))
    {
        free (sd);
        return FALSE;
    }

    sd->screenPrivateIndex = allocateScreenPrivateIndex (d);
    if (sd->screenPrivateIndex < 0)
    {
        compFiniDisplayOptions (d, sd->opt, SHOT_DISPLAY_OPTION_NUM);
        free (sd);
        return FALSE;
    }
    WRAP (sd, d, handleEvent, shotHandleEvent);
    d->base.privates[displayPrivateIndex].ptr = sd;
    return TRUE;
}

static void
shotFiniDisplay (CompPlugin *p, CompDisplay *d)
{
    SHOT_DISPLAY (d);
    freeScreenPrivateIndex (d, sd->screenPrivateIndex);
    UNWRAP (sd, d, handleEvent);
    compFiniDisplayOptions (d, sd->opt, SHOT_DISPLAY_OPTION_NUM);
    free (sd);
}

```



Plugin Options

- Plugin options specified in the gconf.
- Plugin can store options for each screen and each display

```
static const CompMetadataOptionInfo shotDisplayOptionInfo[] = {  
    { "initiate_button", "button", 0, shotInitiate, shotTerminate },  
    { "directory", "string", 0, 0, 0 },  
    { "launch_app", "string", 0, 0, 0 }  
};
```

Screen Init and Fini

- Exactly the same type of thing happens for the screen, too

```
typedef struct _ShotScreen {
    PaintOutputProc paintOutput;
    PaintScreenProc paintScreen;
    int grabIndex;

    int x1, y1, x2, y2;
    Bool grab;
} ShotScreen;
```

```
static Bool
shotInitScreen (CompPlugin *p, CompScreen *s)
{
    ShotScreen *ss;
    SHOT_DISPLAY (s->display);
    ss = malloc (sizeof (ShotScreen));
    if (!ss)
        return FALSE;
    ss->grabIndex = 0;
    ss->grab = FALSE;

    WRAP (ss, s, paintScreen, shotPaintScreen);
    WRAP (ss, s, paintOutput, shotPaintOutput);

    s->base.privates[sd->screenPrivateIndex].ptr = ss;

    return TRUE;
}

static void
shotFiniScreen (CompPlugin *p, CompScreen *s)
{
    SHOT_SCREEN (s);

    UNWRAP (ss, s, paintScreen);
    UNWRAP (ss, s, paintOutput);

    free (ss);
}
```

Plugin Operations

- Our own function is called instead normal function for the display

```
static void
shotHandleEvent (CompDisplay *d,
                 XEvent *event)
{
    CompScreen *s;

    SHOT_DISPLAY (d);

    switch (event->type) {
    case MotionNotify:
        s = findScreenAtDisplay (d, event->xmotion.root);
        if (s)
            shotHandleMotionEvent (s, pointerX, pointerY);
        break;
    case EnterNotify:
    case LeaveNotify:
        s = findScreenAtDisplay (d, event->xcrossing.root);
        if (s)
            shotHandleMotionEvent (s, pointerX, pointerY);
    default:
        break;
    }

    UNWRAP (sd, d, handleEvent);
    (*d->handleEvent) (d, event);
    WRAP (sd, d, handleEvent, shotHandleEvent);
}
```

```
static Bool
shotPaintOutput (CompScreen *s, const ScreenPaintAttrib *sAttrib,
                 const CompTransform *transform, Region region,
                 CompOutput *output, unsigned int mask)
{
    Bool status;
    SHOT_SCREEN (s);
    UNWRAP (ss, s, paintOutput);
    status = (*s->paintOutput) (s, sAttrib, transform, region, output, mask);
    WRAP (ss, s, paintOutput, shotPaintOutput);
    if (status && ss->grab)
    {
        int x1, x2, y1, y2;
        x1 = MIN (ss->x1, ss->x2);
        y1 = MIN (ss->y1, ss->y2);
        x2 = MAX (ss->x1, ss->x2);
        y2 = MAX (ss->y1, ss->y2);

        if (ss->grabIndex)
        {
            glPushMatrix ();
            prepareXCoords (s, output, -DEFAULT_Z_CAMERA);

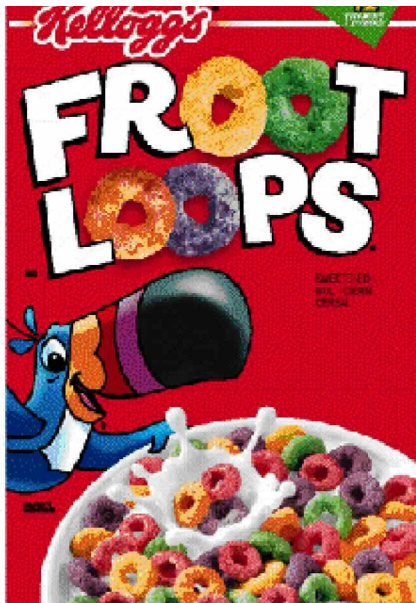
            glPopMatrix ();
        }
    }
    return status;
}
```



Summary

- What is a windows manager and what it does
- Standards the windows manager should follow
- Metacity windows manager and its theme design
- Compiz windows manager and its plugin structure

Choose your favour



OR



open



USE



IMPROVE



EVANGELIZE

Thank you!

Erwann Chenede

<http://blogs.sun.com/erwann>

Chi Wang

<http://blogs.sun.com/chriswang>

開
放
的
열린
مفتوح
libre
मुक्त
ಮುಕ್ತ
livre
libero
ముక్త
开放的
açık
open
nyílt
•••••
πικρ
オープン
livre
ανοικτό
offen
otevřený
öppen
открытый
வெளிப்படை