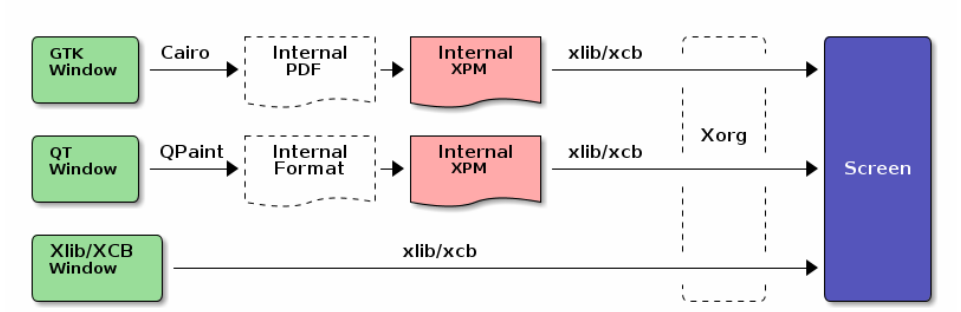
X 中的混成器與 Composite 擴展

目錄

在上篇文章「桌面系統的混成器簡史」中我介紹了其它桌面系統中的混成器的發展史和工作原理， 話題回到我們的正題 Linux 系統上，來說說目前 X 中混成器是如何工作的。 這篇文章將比上一篇深入更多技術細節，不想看太多細節的可以直接跳過看 結論 。

原始的 X 的繪圖模型

首先，沒有混成器的時候 X 是這樣畫圖的：



X 的應用程序沒有統一的繪圖 API。GTK+ 在 3.0 之後統一用 Cairo 繪圖，而 Cairo 則是基於 PDF 1.4 的繪圖模型構建的，GTK 的 2.0 和之前的版本中也有很大一部分的繪圖是用 Cairo 進行，其餘則通過 xlib 或者 xcb 調用 X 核心協議提供的繪圖原語繪圖。QT 的情況也是類似，基本上用 QPaint 子系統繪製成位圖然後交給 X 的顯示服務器。顯示服務器拿到這些繪製請求之後，再在屏幕上的相應位置繪製整個屏幕。當然還有很多老舊的不用 GTK 或者 QT 的程序，他們則直接調用 X 核心協議提供的繪圖原語。

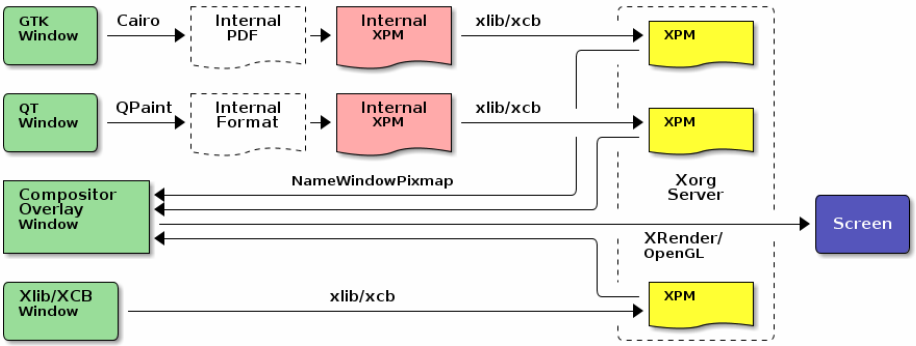
值得注意一點是 X 上除了沒有統一的繪圖模型，也沒有統一的矢量圖格式。X 核心協議的繪圖原語提供的是像素單位的繪圖操作，沒有類似 GDI+ 或者 Quartz 提供的 設備無關 (Device Independence) 的「點」的抽象。所以只用 X 的繪圖原語的話，我們可以把 (1,1) 這個像素點塗黑，但是不能把 (0.5, 0.5) 這個點塗黑，這一設計缺陷在 Unix Hater's Handbook 中已經被吐槽過了。因爲這個缺陷，所以直接用 X 繪圖原語繪製的圖像不能像 矢量圖那樣進行無損縮放。同樣的缺陷導致 X 繪圖原語繪製的字符不能做到 子像素級 (subpixel-level) 抗鋸齒 (anti-aliasing)（這解釋了默認配置下的 xterm 和 urxvt 中的字體渲染爲什麼難看）。相比之下 GDI 有對應的 WMF 矢量圖格式，Quartz 有對應的 PDF 矢量圖格式，而 X 中沒有這樣的格式對應。因爲沒有統一的矢量圖格式，所以無論是 Cairo、QPaint 還是沒有用這些繪圖庫但是同樣在意字體和曲線渲染效果的程序（比如 Firefox 和 Chromium）都需要首先渲染到內部的 XPixMap 位圖格式，做好子像素渲染和矢量縮放，然後再把渲染好的位圖轉交給 X 圖形服務器。

通過 Composite 擴展重定向窗口輸出

2004年發佈的 X11R6.8 版本的 Xorg 引入了 Composite 擴展 。這個擴展背後的動機以及前因後果在一篇文章 The (Re)Architecture of the X Window System 中有詳細的表述。Composite 擴展允許某個 X 程序做這幾件事情：

1. 通過 RedirectSubwindows 調用將一個窗口樹中的所有窗口渲染重定向到 內部存儲(off-screen storage) 。重定向的時候可以指定讓 X 自動更新窗口的內容到屏幕上或者由混成器手動更新。
2. 通過 NameWindowPixmap 取得某個窗口的內部存儲。
3. 通過 GetOverlayWindow 獲得一個特殊的用於繪圖的窗口， 在這個窗口上繪製的圖像將覆蓋在屏幕的最上面。
4. 通過 CreateRegionFromBorderClip 取得某個窗口的邊界剪裁區域（不一定是矩形）。

有了 Composite 擴展，一個 X 程序就可以調用這些 API 實現混成器。 這裏有篇 教學解釋如何使用 Composite 擴展 。開啓了混成的 X 是這樣繪圖的：



整個 X 的混成器模型與 Mac OS X 的混成器模型相比，有如下幾點顯著的區別：

1. 混成的部分是交由外部的程序完成的，對混成的繪製方式和繪製普通窗口一樣。 出於效率考慮，絕大多數 X 上的混成器額外使用了

XRender 擴展或者 OpenGL/EGL 來加速繪製貼圖。不過即使如此，還是不能避免同樣的位圖（內容不一定完全一致，比如 X 可以在窗口交給它的位圖上加上邊框然後再返還給混成器）**在不同的三個程序之間來回傳遞**。

2. RedirectSubwindows 調用針對的是一個窗口樹，換句話說是一個窗口及其全部子窗口，不同於 Mac OS X 中混成器會拿到全部窗口的輸出。這個特點其實並不算是限制，因爲 X 中每個虛擬桌面都有一個根窗口，只要指定這個根窗口就可以拿到整個虛擬桌面上的全部可見窗口輸出了。反而這個設計提供了一定的自由度，比如我們可以用這個調用實現一個截圖程序，拿到某個特定窗口的輸出，而不用在意別的窗口。
3. 爲了讓窗口有輸出，窗口必須顯示在當前桌面上，不能處於最小化狀態或者顯示在別的虛擬桌面，用 X 的術語說就是窗口必須處於被映射(mapped)的狀態。因此直接用上述方法**不能得到沒有顯示的窗口的輸出**，比如不能對最小化的窗口直接實現 Windows 7 中的 Aero Peak 之類的效果。這個限制可以想辦法繞開，比如在需要窗口輸出的時候臨時把窗口隱射到桌面上，拿到輸出之後再隱藏起來，不過要實現這一點需要混成器和窗口管理器相互配合。
4. 不像 Mac OS X 的基於 OpenGL Surface 的繪圖模型是設備無關(device independent)的，這裏 X 的繪圖模型是設備相關(device dependent)的。這既是優點也是缺點。從缺點方面而言，顯示到 X 的位圖輸出因爲設備相關性，所以嚴格對應顯示器的點陣，並不適合作爲文檔格式打印出來。當然無論是 Cairo 還是 QPaint 都提供了到 PostScript 或者 PDF 後端的輸出，所以實用層面這個並不構成問題。設備相關這一點的優點在於，繪製到 XPM 位圖的時候，程序和繪圖庫是能拿到輸出設備（顯示器）的特殊屬性的，從而繪圖庫能考慮不同的色彩、分辨率、DPI 或者子像素(subpixel)佈局(layout)這些屬性以提供最好的渲染效果。Mac OS X 10.4 在設計的時候也曾考慮過提供無極縮放的支持，而這種支持到了 Mac OS X 10.5 中就縮水變成了 Retina 的固定 2 倍縮放。這種局面在 X 上沒有發生正是因爲 X 的繪圖模型的這種設備相關性，而 Mac OS X 的混成器纔用的 OpenGL Surface 則無視了這些設備相關的屬性。

輸入事件的重定向，這可能做到麼？

通過上述 Composite 擴展提供的 API，混成器可以把窗口的 **輸出** 重定向到自己的窗口上。但是僅僅重定向輸出，整個 X 還不處於可用狀態，因爲 **沒有重定向輸入**。考慮一下用戶試圖用鼠標點擊某個按鈕或者文本框，這時鼠標處於的位置是在 OverlayWindow 上繪製的位置，這個鼠標事件會交給 OverlayWindow，而用戶期待這個事件被發送給他看到的按鈕上。

需要重定向的事件主要有鍵盤和鼠標事件兩大類（暫時先不考慮觸摸屏之類的額外輸入）。由於 Composite 擴展並沒有直接提供這方面的重定向 API，這使得輸入事件處理起來都比較麻煩，

假設要重定向鍵盤事件，混成器需要效仿輸入法框架（fcitx, ibus, scim）那樣處理一部分按鍵事件並把其餘事件轉給具有輸入焦點的程序。看看現有的輸入法框架和諸多程序間的問題，我們就能知道這裏的坑有多深。於是 **大部分 X 的混成器都不處理鍵盤事件重定向**。再來看重定向鼠標事件，這邊的坑比重定向鍵盤事件的坑更多，因爲不像重定向窗口輸出那樣只需要考慮 top-level 頂層 窗口，重定向鼠標輸入的時候要考慮所有子窗口（它們有獨立的事件隊列），以及要準確記錄輸入事件事件發生時的鍵盤組合鍵狀態，還要正確實現 ICCCM/EWMH 中描述的轉交窗口焦點的複雜規則，所有這些都已經在 X 中實現過的事情需要重新實現一遍。

由於坑太多難以實現，所以所有 X 下的混成器的實現方式都是直接忽略這個繁重的任務，**不重定向輸入事件** 而把它交給 X 處理。具體的實現方式就是通過 XFixes 擴展提供的 SetWindowShapeRegion API 將 OverlayWindow 的 **輸入區域** ShapeInput 設爲空區域，從而忽略對這個 OverlayWindow 的一切鼠標鍵盤事件。這樣一來對 OverlayWindow 的點擊會透過 OverlayWindow 直接作用到底下的窗口上。

因爲選擇了不重定向輸入事件，X 下的混成器通常會處於以下兩種狀態：

1. 選擇狀態下可以縮放窗口的大小，扭曲窗口的形狀，並且可以把窗

口繪製在任意想要繪製的位置上（並不是移動窗口的位置），**但是不能讓用戶與窗口的內容交互**。

2. 正常狀態下可以讓用戶與窗口的內容交互，但是**繪製的窗口位置、大小和形狀必須嚴格地和 X 記錄的窗口的位置、大小和形狀保持一致**。持續時間短暫的動畫效果可以允許位置和形狀稍有偏差，但是在動畫的過程中如果用戶點擊了 變形縮放過的窗口，那麼鼠標事件將發往錯誤的（X 記錄中的而非顯示出的）窗口元素上。

可以發現這兩種狀態就直接對應了 Gnome 3 的普通狀態和縮略圖狀態（點擊 Activity 活動 或者戳畫面左上角之後顯示的狀態），這也解釋了爲什麼儘管 Gnome 3 的窗口有碩大的關閉按鈕，但是在縮略圖狀態下 Gnome 3 仍然需要給窗口加上額外的關閉按鈕：**因爲處於縮略狀態下的窗口只是一張畫而不能點**。

Composite 擴展的這些限制使得 X 下的混成器目前只能實現 Mac OS X 那樣的 Exposé 效果，而不能實現 LG3D 那樣直接在 3D 空間中操縱窗口內容。

解決重定向問題曾經的一縷曙光是 Sun Microsystems 昇陽公司 在開發 LG3D 的過程中同時提議過另一個 X 擴展叫做 Event Interception 或者簡稱 XEvIE，這個擴展的設計目的就是提供 API 讓某個程序接收並操縱全部的鍵盤和鼠標事件。可惜這個擴展隨着昇陽公司本身的隕落而 處於無人維護的狀態，這一點也在它的官方網頁上說明了：

> It has been suggested that this extension should not be used because it is broken and maintainerless.

Composite 擴展的不足

通過上面的介紹，我們就已經可以看到 Composite 擴展的不足之處

了。 總結起來說，主要有兩大不足：

1. 繪圖效率低。因爲同樣的位圖從應用程序傳到 Xorg ，再從 Xorg 傳到混成器， 最後從混成器再繪製到屏幕上，繞了一個大彎。這就是爲什麼 Wayland 的開發者在他的slide the real story behind Wayland and X 裏這麼說：

   > and what's the X server? really bad IPC
   >
   > 那麼 X 服務器到底做了什麼呢？ 非常糟糕的進程間通訊

2. 沒有重定向輸入事件。如果我們要在 X 的混成器裏做這個事情，基本上我們要全部重寫一遍 X 已經寫好的窗口事件分發邏輯。

既然同樣要重寫，爲什麼不直接重寫一遍 X 呢，扔掉那些歷史負擔，扔掉那些無用的 API ，重新設計可擴展的 API ，做好快速安全的 IPC ——嗯，重寫 X 就是 Wayland 的目的。

不過這麼重寫了的 Wayland 還是我們熟悉可愛的 X 麼？它有哪些地方變樣了？ 這將是我下一篇文章的內容。

附錄：擴展閱讀

我自己沒有寫過窗口管理器，沒有寫過混成器，沒有寫過 Wayland 程序，以上說的都是我從互聯網上看到的整理出來的內容。寫下本文的過程中我參考了這些文章：

The (Re)Architecture of the X Window System 這篇2004年寫的文章描述了 Composite 擴展出現的動機和歷史，介紹了繪圖庫的實現情況，涉及了上面所說的那些 X 擴展被用到的情況和可能。 同時這篇文章還展望了很多現在的 X 已然實現了的功能，比如 OpenGL 和 X 的結合方面我們有了 GLX 和 AIGLX ，比如內核的顯卡支持方面我們有了 DRI 和 KMS 。總之這是一篇描述 Linux 桌面未來的發展軌跡的非常有閱讀價值

的歷史文獻。

so you want to build a compositor 這是一篇 2008 年寫的博文，介紹如何用 Clutter 實現一個最簡單的混成器。

Composite tutorial 這是另一篇介紹如何實現一個簡單的混成器的博文，用 Qt 實現，但是同樣很底層。

unagi 這是一個可用的（但是已經長期沒有開發的）類似 xcompmgr 的混成器。這個項目貌似 是一位研究生的碩士畢業設計，同時他公開了碩士學位的畢業論文 Master thesis: Writing an X compositing manager 其中也對實現一個簡單的混成器做了詳盡描述，包括介紹了相關的 X 擴展和調用。