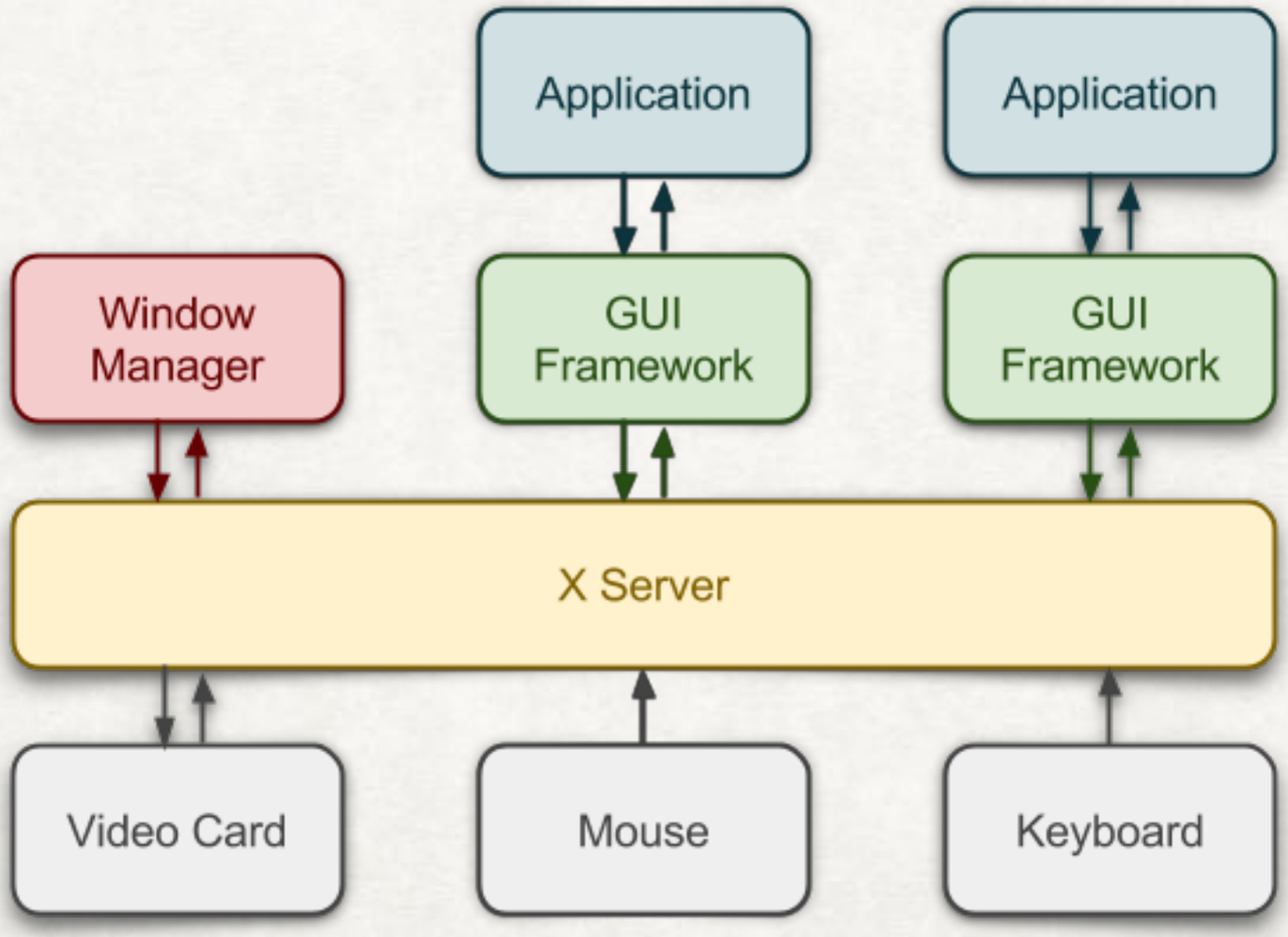
深入浅出窗口管理器

Sian Cao

什么是窗口管理器

是X11/Linux上的一个概念，windows似乎没有这种东西。技术上说它是一个X11的客户端程序，通过标准的X11技术来管理顶层窗口的请求、大小、位置、层次关系等。并且根据口味、窗口类型等信息决定是不是要给顶层窗口提供边框，标题栏、状态栏等界面元素。不同的窗口管理器外表上看可能千差万别。



为什么要使用窗口管理器

- 没有是什么样的：一般就是没有标题栏，没有边框，背景黑压压的一片，窗口之间没有明显的界限，没有圆角（或者有锯齿的圆角），没有透明等等。但是有些例外，比如使用gtk+的程序有一个趋势，就是绕过wm自己来绘制边框和标题栏。然后常规的操作可能不起作用，比如鼠标点击最大化，最小化，移动，改变大小等，也即无法进行交互，同时也没有输入焦点（跟随鼠标）。

- 有是什么样的：可能有好看的桌面背景，可能有标题栏，可能有边框，大部分时候可以鼠标点击最大化，最小化，移动，改变大小等。可能有透明（取决于是否使用了混成器），反锯齿的窗口圆角等。

- 怎么区分有和没有：看经验，看外表，有时候可能区分不出来，尤其是使用一些所谓的平铺wm。

窗口管理器提供了什么

- 直观上，一般会在client上加入边框和标题栏，并且有丰富的样式，支持漂亮的主题，并且用户至少能够定制某些外观。但是也不一定，比如大多数平铺wm等就不提供标题栏，比如有些窗口类型是splash，tooltip等等也不会有标题栏。所以说不同的wm会有不同的视觉风格。

- 调整大小策略：有很多因素的共同影响：窗口自己的要求，工作区的大小，wm的想法等等。

- 调整位置策略：同类型的窗口，一般wm会有规律的排列。不同类型的窗口，wm一般会尽量分散到workarea各处。

- 虚拟桌面（工作区）：理论上数量不限

- workarea（工作区面积？）：dock的影响，多屏的影响，计算面积（矩形）的算法很复杂

- 焦点切换：鼠标跟随，点击等

窗口管理器分类

按照管理窗口的方式：

- 堆栈式窗口管理器(stacking window manager)
- 平铺式窗口管理器(tiling window manager)
- 动态?

也可以用另外一种方式分类：

- 混成式wm：offscreen渲染（因此可以获取缩略图），alpha混合透明化，
- 非混成式wm：客户端窗口根据服务器报告的expose事件重绘窗口部分区域，X11服务器并没有保存离线的位图。

混成器

混成器：与wm相关的，但是不是同一个东西，视觉上提供了什么

- shadow：窗口底层的一层淡淡的阴影
- 半透明：如果窗口设置了alpha通道，在有混成器的情况下，会透过去
- 窗口背景的动态模糊：基于透明窗口，以及对前后多层窗口的高斯模糊运算
- 实现：可以是OpenGL，也可以是xrender或其他。

技术说明

- X11：CS结构，server就是个事件分发器。通常server接受来自不同client的事件，并根据时间顺序进行处理，然后根据client需要，发送处理通知给client。

- icccm, wm-spec协议：现实是协议只是一个建议，wm并不一定遵守（经常发生），而且各家wm的解释可能不一样。还有许多协议没有明确规定的，比如如何计算工作区的面积，struts有什么限制等。

- wm是一个特殊的client。那么一个client如何成为wm：获取WM_S0（0代表screen number 0）的SelectionOwner。 这是一个友好退让的过程，强制获取WM_S0会使得正在运行的wm收到一个变更通知，使得它能释放资源，清理现场。同一时间只能有一个wm，不能共存。

技术说明（2）

- 窗口管理器只管理顶层窗口：换种说法就是只管理直接子窗口（direct descendants）。

  - 两种方式，一种叫reparent模式，一种是不知道叫什么，独立的管理client，要么没有边框标题栏等装饰，要么用独立的平级窗口同步放在client窗口上面等。

- 窗口管理器如何才能管理窗口：在root窗口上申请Substructure Redirection 。

  - substructure redirection 可以做到以下一些事情：maprequest，configurerequest，CirculateRequest。也既可以截住所有直接子窗口的移动、改变大小、显示、隐藏等请求，有时候对请求进行修改并允许，有时候可以直接拒绝。需要注意的是申请了substructure redirection的窗口只能处理直接子窗口，而不能处理子孙窗口的请求。

  - 问题来了：有很多的窗口管理器需要在窗口边缘绘制边框和标题栏，这通常都是通过reparent窗口到由窗口管理器创建的一个frame窗口来实现的。这样一样，原来的窗口就不再是root的直接子窗口，于是root再也无法截住它的map、configure等请求了，怎么办？答案是在frame窗口上申请substructure redirection。

技术说明（3）

- 基本上窗口管理器与窗口通讯是通过两个X11的基本能力：属性和事件（ properties and events ）。

- 基本上与窗口管理器所提供的功能相关的请求都通过这两个方式来实现。窗口管理器会通过selectInput对每个被管理窗口进行跟踪，包括监测请求，属性变化，client message等。

- 首先是一组在root上的属性，定义了当前的一些状态，比如当前工作区，工作区的数量，焦点窗口，工作区的面积。client可以查询和监测这些属性的变化，并作出反应。

- 然后窗口管理器在每个被管理窗口上都维护了一组属性，比如窗口所在的工作区，窗口状态。还有一些窗口属性是有客户程序自己维护的，主要用来与窗口管理器进行通讯的。

- 除了属性外，client可以通过发送client message来请求窗口管理器完成一些任务，比如请求移动到某个工作区，最大化，最小化，移动，改变大小，全屏，置顶等等。

技术说明（4）

混成器（Compositing Manager）

- 混成器依赖一些X11的扩展（extensions）才能工作：Composite，Damage，Shape等。

- XCompositeRedirectSubwindows。它的作用是在X11服务器端为一个窗口的所有直接子窗口创建backing store pixmap，也即离线存储，之后窗口管理器可以通过XCompositeNameWindowPixmap来获取代表该存储的Pixmap。如此一来，混成器就可以获取窗口的内容，并且进行各种运算，创造不同的视觉效果，比如半透明，背景模糊，反锯齿的异形窗口，动画等。

- 每当窗口大小变化或者map state发生变化时，需要更新离线存储。同时可以通过Damage的事件来跟踪窗口哪些区域发生了变化。混成的具体动作不一定需要opengl来完成。举例来说，metacity和xfwm使用xrender（最终会使用硬件来加速）来实现混成，mutter使用clutter（opengl封装）来实现混合和动画，kwin有多个混成后端。

有哪些坑

- 有一些叫做override-redirect（OR）的窗口，窗口管理器基本是三不管（在X11的很多函数的文档里都会说明如果是OR窗口将会特别对待）。这个属性基本上就是请求绕过substructure redirection，意思就是X11不会把ConfigureRequest和MapRequest等发送给窗口管理器去处理。但是一般混成器会监控这些窗口的大小和状态变化，以便更新材质。窗口管理器和混成器一般通过CreateNotify、MapNotify事件来检测是否新建了一个OR窗口。

- 窗口管理器会将窗口分层

```
typedef enum    {
  META_LAYER_DESKTOP          = 0,
  META_LAYER_BOTTOM       = 1,
  META_LAYER_NORMAL       = 2,
  META_LAYER_TOP          = 4, /* Same as DOCK; see EWMH and bug 330717 */
  META_LAYER_DOCK         = 4,
  META_LAYER_FULLSCREEN       = 5,
  META_LAYER_OVERRIDE_REDIRECT = 7,
  META_LAYER_LAST         = 8
} MetaStackLayer;
```

有哪些坑（2）

- time window：理论上每个窗口有唯一一个，而且是一一映射。窗口管理器（mutter）用hash表表示这种关系。

  - NET_WM_USER_TIME_WINDOW属性：指向一个窗口，这个窗口唯一作用是维护一个时间戳属性（NET_WM_USER_TIME），代表窗口最后一次用户活动发生的时间（XServer的时间戳，不是qt或者gtk的）。这个时间戳一般由窗口管理器更新，更新时机有创建时（读取窗口设置，如果有的话（这是个坑，如果设置的时间戳是错的，那就悲剧了）），鼠标事件，键盘事件。（演示？xprop -id 0x7e00004 -spy _NET_WM_USER_TIME）。

  - 之前wine的某些程序（比如QQ）的几个窗口映射到同一个time window，导致窗口管理器内部管理的混乱。

- 时间戳：这个对于窗口管理器而言很重要，因为外部发生的事件都是随机和异步的，窗口管理器经常要根据时间戳来正确的处理某些竞争的问题。比如点击dock上的一个图标时，从而发送一个activate窗口的client essage消息，但是在消息里不包含时间戳或包含了一个错误的（或者说过去的一个时间戳）时间戳，可能会出现请求的窗口并没有置顶（因为已经有更新的时间戳存在，表示用户有新的活动发生）。deepin-metacity和deepin-mutter为了照顾大家，在有些情况禁止了时间戳检查。

有哪些坑（3）

- _NET_WM_BYPASS_COMPOSITOR：这基本上是一个优化提示，要求混成器不对该窗口进行混合，这样可以减少资源开销。类似的，一般不透明的全屏窗口置顶时会被自动uncomposite，因为这种情况完全没必要和后面的窗口进行混合运算。

- NET_WM_STRUT/NET_WM_STRUT_PARTIAL：struts对于窗口管理器有很多影响，主要是工作区面积的计算方式，窗口能够最大化的范围，窗口呈现的位置等。一般OR窗口比较屌，不会受这些的影响。

- WORKAREA的计算：这是个坑，而且百家争鸣。因为窗口管理器的协议里并没有规定算法。deepin-metacity和deepin-mutter的做法是（考虑多屏环境，并且适当简化一下）计算不包含struts的最大面积的矩形。这个区域会对窗口管理器如何放置窗口有很多深远的影响。

- 窗口的位置和大小是个复杂的过程：窗口管理器在收到移动、改变窗口大小的请求时，并不是完全听从程序的要求，而是会根据很多限制去修改请求，规则特别多，是各家窗口管理器区别比较大也是很复杂的一个功能。窗口管理器要综合考虑当前屏幕大小，struts设置，窗口的最小大小，建议大小等因素，进行一组constraints计算，并最后得到一个结果。

有哪些坑（4）

- **窗口类型关系混乱：窗口类型的使用是有一些限制和要求的，不正确的逻辑关系可能会导致窗口管理器混乱。**
  - 比如dialog类型父窗口，弹出一个normal类型的子窗口作为菜单。
  - 对话框窗口之间是有关联关系（transient_for）的，这可以帮助窗口管理器合理布置对话框。不正确的设置或者不设置transient_for会出现许多问题。
  - MAXIMIZED不等于FULLSCREEN，不要拿maximized当fullscreen用，比如上面提到窗口在窗口管理器里是分层排列的，maximized窗口在normal这个层次，而fullscreen是在顶层。
  - 窗口的大小最大跟窗口状态是MAXIMIZED对于窗口管理器来说，是有本质区别的，不要等同，这个问题经常出现在使用wine运行的程序里，导致窗口管理器无法正常管理。

- 一个非混成的窗口管理器，在最小化窗口时会对其进行unmap操作，释放窗口资源。但是在使用混成窗口管理器的时候，最小化的窗口并不是withdraw的，而是记录了一个特殊的NET_WM_STATE=HIDDEN。从上面提到的层次来看，HIDDEN的窗口其实是放置在DESKTOP类型的窗口之后的。

- 窗口管理器创建了很多私有窗口用于管理和实现一些效果。比如用来分隔HIDDEN窗口与非HIDDEN窗口的所谓guard window。比如用来实现边角热区功能的不可见小窗口。

参考

- wm-spec：https://specifications.freedesktop.org/wm-spec/wm-spec-latest.html
- icccm：https://www.x.org/releases/X11R7.6/doc/xorg-docs/specs/ICCCM/icccm.html
- xlib programming manual
- deepin-metacity：https://github.com/linuxdeepin/deepin-metacity
- deepin-mutter：https://github.com/linuxdeepin/deepin-mutter
- https://seasonofcode.com/posts/how-x-window-managers-work-and-how-to-write-one-part-i.html