GNOME Shell

A design for a personal integrated digital work environment

William Jon McCann <william.jon.mccann@gmail.com>
Jeremy Perry <jeremy.perry@redhat.com>

1 July 2009

Introduction

The purpose of this document is to establish goals and intentions, provide high-level guidelines, and create a framework for dialog and action. We hope it will be used to enable cooperation and facilitate innovation. This is not an end product. We expect it to be a stimulus and catalyst for discussion. While it may not keep pace with those discussions and conversations, we expect it to evolve as naturally as they do.

Like many conversations we expect that it may be a little noisy and confusing at first. Eventually we will synthesize these ideas, notes, and chatter into a cohesive story. But we'll need your help to do it.

Problem Definition

The GNOME Project released version 2.0 of the GNOME Desktop in June 2002. It was an important milestone. In the years since then, the developer community has continually and incrementally improved the experience while learning a great deal about what worked and what didn't. The entire personal computing ecosystem has been changing too - partly due to a number of new and disruptive technologies. While we won't dwell on the particulars of those changes it is important to note that there is a growing consensus in the GNOME developer community that we needed to make a leap forward in order to fix many of the flaws in our designs and to generally bring a lot more awesome into the user experience. To do this we need to take a step back and evaluate. What should GNOME feel like? How can it make me more effective? How will it delight me? The GNOME Shell effort may be the keystone for constructing answers to these questions. In this document we'll propose a design for a number of key areas of the core GNOME user experience.

Principles

It may be useful to highlight some of the principles that we wish to guide us as we approach the various challenges in a new design. The intention is not to make an exhaustive or exclusive list but merely to indicate a few standards that may be particularly relevant or interesting for this purpose. For more guidance on principles for good design please refer to some of the sources in the gnome-shell Reference List.

Take responsibility for the user's experience

Principle of non-preemption
"Individual interactive programs operate in a non-intrusive manner with respect to the user's activities. The system does not usurp the attention and prerogatives of the user. A program responds to the user's stimuli, but then quietly retains its context and logical state until the user elects to interact with the program again..."
From Deutsch & Taft 1980 "Requirements for an experimental programming environment"

"...Human attention is the most valuable and scarcest commodity in human computer interaction."
Horvitz, Jacobs, Hovel *Attention-Sensitive Alerting*

Principle of Least Astonishment
- or "uniformity in command interface" - From Deutsch & Taft 1980 *Requirements for an experimental programming environment*

Design a self-teaching interface for beginners, and an efficient interface for advanced users, but optimize for intermediates

Don't unnecessarily rely on (especially mechanical-age) metaphor

Less is More
- Reduce Visual, Memory, Intellectual, and Motor work (and complexity)

Be considerate and forgiving

The technology should act as a mediator
- Be the vehicle, not the destination
- Strive for transparency

Goals

Address problems of Focus, Attention, and Interruption

- Define focal and peripheral regions
- Minimize disruption and facilitate uninterrupted focus
- Make it easy to visualize what I'm doing now
- Make it easy to do something new
- Make it easy to do what I do most often
- Make it easy to recall a previous activity

Address problems of Storing, Finding, and Reminding

- Avoid mental models with categories and hierarchies
- Obviate the need for explicit naming, sorting, filing, or categorizing

Manage Complexity and Encourage Flexibility

- Allow the experience to adapt to the usage
- Work as well for the user that uses only two applications and the one that uses tens on multiple workspaces
- Core concepts should scale to capabilities of devices other than "desktop" computers
- Must be usable with a touch or single button input device
- Must be usable in when rotated or resized

Delight the user

- Better default experience, more consistency, more fun!
- Coherence leads to comfort
    - Incoherence - inconsistency - confusion - discomfort ➢ judgements of incompetence
- Promote a brand identity
- Be aesthetically pleasing
- "Quality isn't job one. Being totally fucking amazing is job one." *gapingvoid.com*
- Provide a consistent experience for all users so that knowledge may be shared and expectations may be stable and deterministic
- Provide a few isolated but highly expressive places for personalization

Non-Goals

Not intended to address task-based or other high-concept, unified computing models at this time

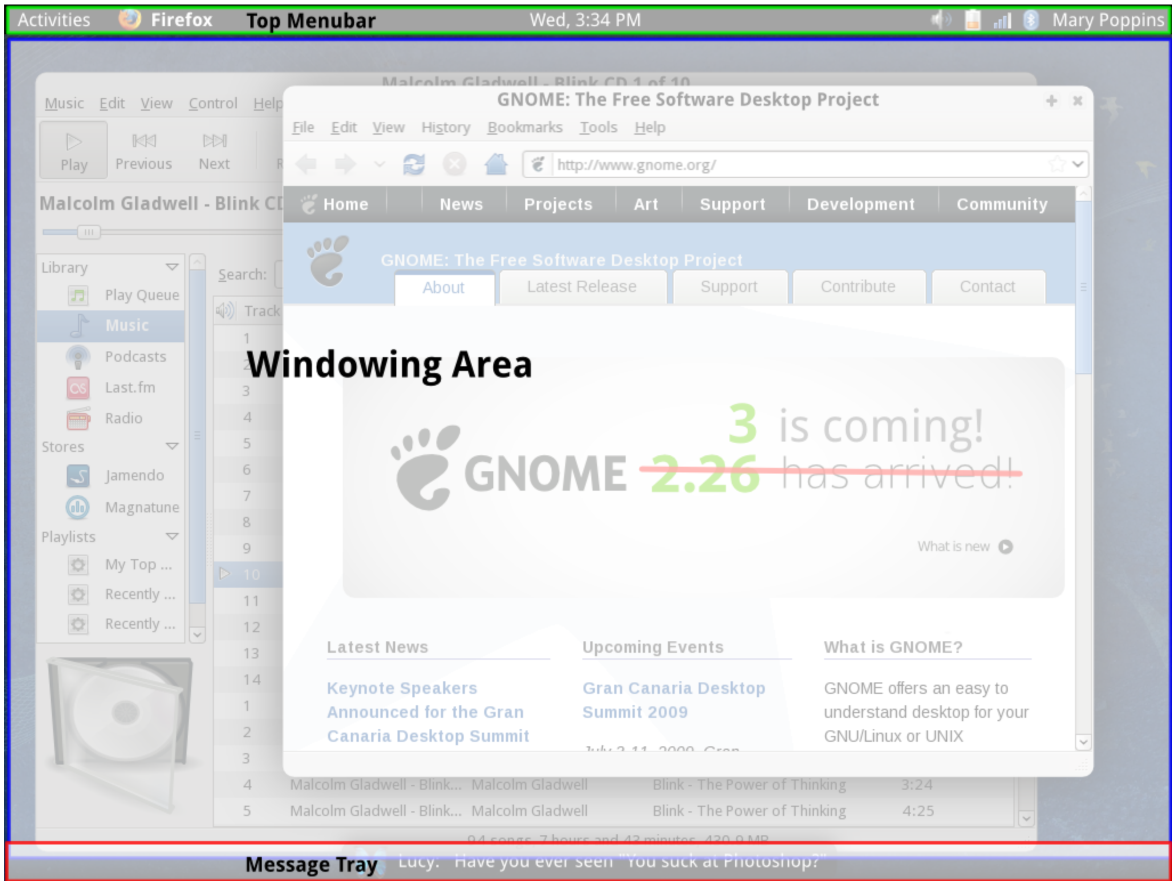
Not optimized for multiple concurrent users

If possible, do not require custom "Shell-aware" applications

Components

For purposes of discussion we'll try to break down the overall design problem into a number of key components. We'll try zoom in on and detail some specific behaviors while we keep an eye on the overall picture to ensure consistency and cohesion.

Here is a schematic view of the main scene:



Lucy: Have you ever seen "You suck at Photoshop?"

The main scene has three primary, normally visible, areas for interaction: the **Top Menubar**, **Windowing Area**, and **Message Tray**. In addition to these there is another that is not normally visible, the **Activity Switcher** (also known as the Alt-Tab Switcher). We'll now explore each of these in some depth.

Top Menubar

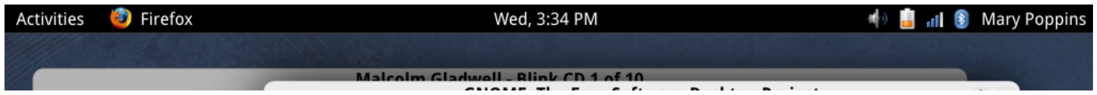


Figure showing **Activities item**, **Active Application item**, **Clock**, **System Status Area**, and **User item**.

The Top Menubar is a horizontal strip across the entire top of the primary screen. It is the primary interface for System and Shell related functionality.

Reasons for using the top of the screen include:

- Menus and menu-like overlays look better when they drop down below the button
- To differentiate the Activities item from the Windows Start button in both behavior and appearance (in part to avoid uncanny-valley type issues and to avoid a sense of unfulfilled expectations)
- At least in the western (RTL-TTB) world the top and left of the interface is the most prominent. This prime real estate must have a considered design.
- The default GNOME configuration already uses a top panel for primary interaction
- Particularly on mobile devices, a top "panel" is becoming standard
- Closing maximized applications is not behavior that is worth optimizing for (by taking advantage of the large upper right target area)

In multi-monitor configurations, it should only be displayed on the primary screen. Secondary monitors are typically used as auxiliary displays unless they are clones of the primary.

The menubar should contain the following items: **Activities**, **Active Application**, **Clock**, **System Status Area**, and **User**. Each of these should look and behave like a menu item in a single menubar. Opening the menu for one item should close another. Each should have a tooltip and prelight on mouse-over.

It is recommended that the visual design of the Top Menubar should reinforce the idea of firmness in order that it act as an anchor and landmark when zooming out to the Activities Summary.

At login time the menubar should slide in from the top of the screen.

We'll now look at each of the elements of the Top Menubar.

Hot corner

The upper left corner of the Activities Item and the screen should behave as a "hot corner." Moving the pointer into this area should automatically activate the Activies Item. This offers a few advantages:

- Allows for quickly accessing functionality
- Eliminates need to aim and click on target button
- May be perceived as more efficient by users

- Obviates the need for "dwell click" for input devices without button click functionality (eg. head/eye tracking pointers)
- May facilitate drag-and-drop into the Overview (if necessary)

Unlike many other forms of hot corners or edges this will likely not be subject to inadvertent triggering for a few reasons:

- The corner in which it resides is already "owned" by a control for identical functionality
- Window caption controls are located on the opposite (right-hand) side

Activities Item

The Activities Item is the primary mechanism for getting things done in GNOME Shell. Clicking this item will bring up the Activities Overview.

It is essential that this item have the affordance of clicking. Button-like hints or a downward pointing arrow may be helpful. When the Summary is open the Activities Item should appear to be "pressed in" to reinforce the relationship between the Item and the Summary. Clicking on the Activities Item while the Summary is shown should cause the user to leave the Summary and return to the most recently used application or workspace.

Active Application Item

The active application must display a menu item positioned directly to the right of the Activities item on the Top Menubar. This item has a few purposes.

It allows the user to unambiguously identify the active (ie. focused) application without having to rely on subtle and idiomatic visual style hints on the application windows that may not be easily seen due to impairments or environmental factors, such as direct sunlight.

It offers functionality that is relevant to the application as a whole. This is in contrast to the menus offered in individual windows which should only show operations that affect the contents of the window itself. For example, a window menu should offer the option to "Close" the instance but the application menu should offer the option to "Quit" (ie. close all related windows).

It also provides a form of application launch feedback (also known as start-up notification). When an application is launched, the Active Application item should appear immediately. Since the menu itself may only be populated with actions after the application has fully loaded a placeholder or option to forcefully quit the application may be presented.

Clock

Tells time and stuff - it is a clock - probably digital.

Open Questions:

- What should we have in the drop down menu?

System Status Area

The System Status Area is a place where System Status Icons represent the status of the system to the user. This is not an area that is variously called the Notification Area or System Tray and should not be used by applications (foreground or background) to indicate their status. This distinction is necessary to ensure the entire top of the screen is designed properly, system owned and coherent, able to be modified or extended, scale well to smaller form-factors, and not become a dumping ground or high-profile branding opportunity. Examples of valid System Status Icons include indicators for: audio volume, network connection, and battery power.

Status icons should use a style that is consistent with the text and menus present on the top panel. In general, these icons should use colors in a considered and measured way. Icons and indicators must not change rapidly or abruptly (not more than once a second). Icon changes should transition smoothly.

Status icons should not be considered primary interaction points. Any action that is available through a status icon must also be accessible from elsewhere in the design. For example, network access must also be able to be configured through the system preferences / control-center. So, status icons must not assume that the user can interact with them. They should only expect that they will be used to indicate status. There are a few reasons for this:

- On smaller form factor devices it may be difficult to interact with small icons
- When a high resolution pointer is not available it may be difficult to interact with small icons
- When the user has a disability the icons may be unusable for interaction
- Functionality must be able to be found by and be accessible to someone using desktop or control center search tools

Like all Top Menubar items, the icons should behave as if they are part of a menu-bar.

User Status Menu

The User Item provides access to personalization, customization, and status change options. The label for this menu item should be the full name of the currently active user as well as an indication of the user's current status. Status is one of:

- Available
- Idle
- Busy
- Invisible

The menu should offer the opportunity to set the status to one of available, busy, or invisible. Idleness should be automatically detected by the system or set automatically when the screen is manually locked.

When the user sets her status to Busy the system should avoid interruptions unless they are of critical severity.

The menu may optionally offer the user the opportunity to set a status message. This may be something like "going to buy beer", "watching a movie", or "missed Lost - no spoilers!"

The menu should also offer the opportunity to access user's "profile" information and the system preferences. It should also allow for locking the screen, and exiting the session.

Open Questions:

- Is this a good place to open my Contacts lists?

Message Tray

Various studies [citations needed. Gonzalez or Horvitz etc] have found that messaging (in the broadest sense) accounts for a significant percentage of task interruptions. As we connect to an increasing number of information sources, managing disruptions has become a real challenge. The primary goal of the Message Tray is to provide the user with enough information to quickly assess an event but limit the severity and duration of the preemption. Another important goal is to allow but not compel the user to respond to the event. The tray also provides an important reminding function for messages that the user has deferred addressing.

More specific goals include:

- permit the user to stay focused on the primary task
- provide awareness of new notifications
- remind for unseen messages
- direct attention to high priority messages and alerts
- unobtrusive display
- provide a uniform interface for messages, notifications, and alerts
- allow the user to control the information display

Notifications are events that are ephemeral and do not require direct action or response. Messages are events that either require or imply a response or should be acknowledged in some way.

Conceptually each Message comes from a Message Source. Sources of messages may include, at the user's discretion: e-mail, calendar appointments, instant message chat, stock market data, weather advisories, and system alerts.

In order to not compel the user to action all Messages (events that request or require action, response, or some form of acknowledgement) should be queued for the user. or Notifications and Messages that do not require acknowledgement or that have alternate indicators should not be queued (eg. A low battery warning does not require a response and the battery level already has an indicator in the System Status Area).

Messages may have one of the following states:

- New
- Seen
- Responded
- Resolved

Message Sources may have one of the following states:

- Empty

The design of the Message Tray should support four modes of operation:

1. Hidden mode - tray is hidden off the screen
2. Notification mode - shows a one line summary of the notification or message along with an icon
3. Summary mode - a peripheral awareness (or low detail) mode showing an icon for each Message Source
4. Detail mode - an interactive mode showing more detail about each Message Source

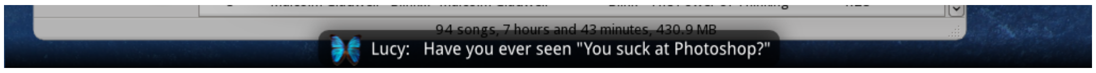
The Message Tray should appear at the bottom of the screen in both landscape and portrait modes (long and short axis respectively).

The Tray should not appear at all for non-critical notifications if the user is marked as Busy.

Open Questions:

- Should a subtle indication appear at the bottom of the screen to indicate a new message when in Busy mode?

Notification Mode


94 songs, 7 hours and 43 minutes, 430.9 MB
Lucy: Have you ever seen "You suck at Photoshop?"

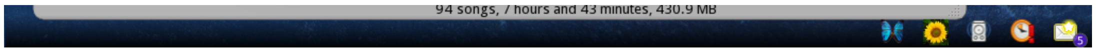
When a new message or notification is available, the Message Tray should slide up from the bottom of the screen in order to display the new Message Item. In this mode, the Tray will display roughly the height of one line of text with a fixed width determined by the size of the screen. The Message Item should display an Icon and a message synopsis. After a brief timeout, if the notification isn't activated or recalled in some way, the Tray should recede off the bottom of the screen. As it moves off the screen the Summary Mode should be displayed momentarily as a reminder function, unless it is empty.

All notifications should pause and remain visible if the user actively moves the mouse over the tray area after it appears. It should not remain visible if the pointer device simply happened to be in the area when the Tray slid out beneath it. At this time (on pointer hover) the message should prelight and have the affordance of clicking if it has a Details Mode available. Conversely, for transient notifications with no Detail Mode there should be no affordance and no prelight.

Effort should be taken to ignore clicks immediately after the Tray slides out when the user doesn't explicitly enter that area of the screen.

It would be nice if the notification or tray offered the user the opportunity to enter Busy mode directly rather than have to find the option in the User menu.

Summary Mode



The Tray should display an icon for each non-empty Message Source. When the Message Source Icon is clicked the Source should expand to display the Detail Mode. The Summary Mode should slide off the screen after a brief time unless the pointer enters the Tray area.

The message source icons should be assembled side-by-side horizontally. Generally, message source icons should queue up from the right to left.

The Summary mode should be displayed when the user returns from idle or busy modes.

A key difference from existing status icon technologies is that Message Source icons can be removed from the Summary simply by dragging them and dropping them somewhere off the tray.

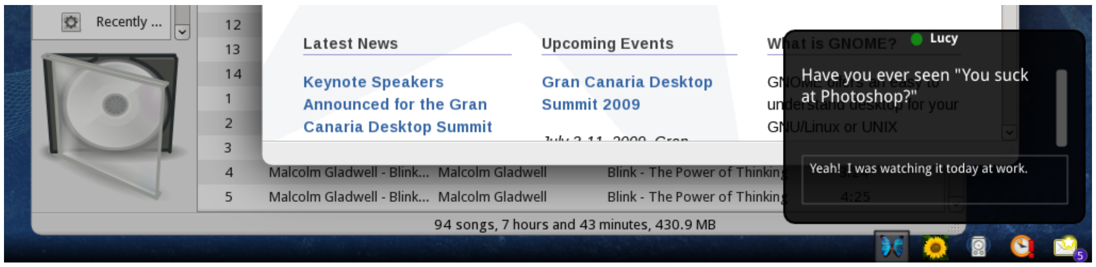
Source Icons may automatically be removed from the Summary for a variety of reasons. Some include:

- Application providing Message Source is again the active (foreground) application
- Application providing Message has exited
- User has acknowledged or otherwise handled the message out of band (perhaps by interacting with the application itself)
- Message Source has timed out (eg. an acknowledged or previously seen chat conversation)

Open Questions:

- Should we have a way to go directly to an application from the Summary mode? Maybe not since some Sources are headless or daemonic processes.

Detail Mode


Recently ...
Latest News
Keynote Speakers Announced for the Gran Canaria Desktop Summit
Upcoming Events
Gran Canaria Desktop Summit 2009
Malcolm Gladwell - Blink... Malcolm Gladwell Blink - The Power of Thinking
94 songs, 7 hours and 43 minutes, 430.9 MB
Lucy

Have you ever seen "You suck at Photoshop?"

Yeah! I was watching it today at work.

The details mode for a Message Source displays information in a way that is appropriate for that Source. This may involve simply showing an informative message or even allow the user to directly respond to an Instant Message.

Instant Message applications are expected to present Message Sources for each currently active conversation. This would make it easier to communicate while doing something else. And eliminate the need for one of the more common window switching tasks.

The conversation message source may also aggregate IM, email, SMS, and Twitter updates for that contact. It should also provide access to a full conversation log - even if the last communication was done on a different system.

Open Questions:

- Should notifications be "chunked" to limit number of disruptions? Likely implies having a way of determining urgency... Probably not worth it if you can find another way to "compress" the notifications.
- Should the tray stop sliding out in collapsed mode on subsequent updates by the same application? Probably best left up to the individual application since you may want to compress notifications for the same user but not for different users.
- Don't show when a fullscreen app is on the primary display?
- If we choose to allow composing messages it should only be via the Contact list. The reason for this is that the method used to contact the person may depend on that person's status. If they are online - IM chat is the way to go. If they are in a meeting email may be better. Sometimes phone is best.
- How do we express message priority?
- Should there be a special system modal type of message? http://mail.gnome.org/archives/gnome-shell-list/2009-April/msg00037.html

Sidebar

The sidebar is an extension point for user-selected content. On displays that are sufficiently large, it should be displayed at the side of the screen by default. It is a container for Gadgets. The sidebar itself may be either: expanded, collapsed, or hidden. Gadgets should have both an expanded and collapsed mode. They may be entertaining, functional, or anywhere in between. However, nothing in the Sidebar should replace functionality that already provided elsewhere in the Shell - it should be auxiliary. Some examples of possible Gadgets include:

- Stock portfolio
- Contextual awareness
- Twitter
- RSS feeds
- Last.fm radio
- Music player
- Weather/Map/Location
- Eyes that follow mouse
- Agenda / To-Do list
- Sticky notes
- System performance monitor
- Dictionary lookup
- A mood lamp that uses the webcam to monitor the user's face for mood changes and reflect them (or soothe them) in a colorful display

The sidebar should not be shown in the Activities Summary workspace overview.

This system should offer compatibility with Google Gadgets.

Open Questions:

- Sides of screen only?
- A fullscreen mode?
- How do we animate the hiding of it when entering the Overview?

The Activities Overview

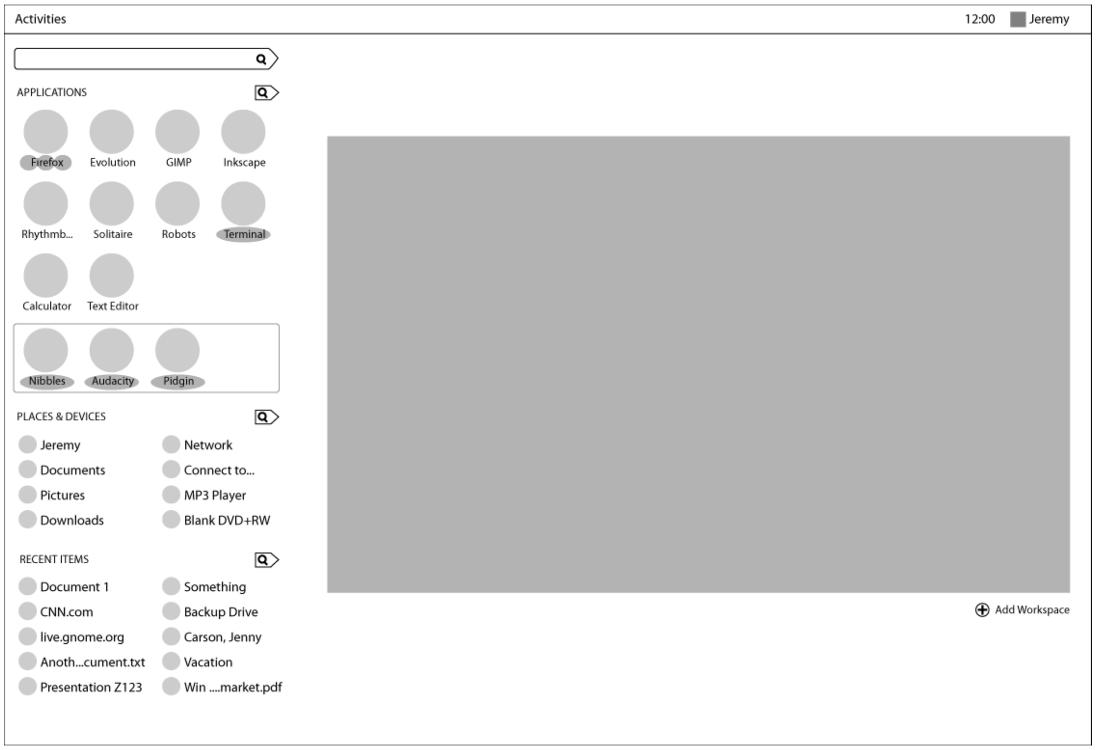


Figure showing a schematic of the Activities Overview

A mediator. A connector. Not a destination.

Designed to facilitate navigation. Optimized for when spatial recognition is insufficient or has not yet been acquired. Or when the mental model does not retain the information of what exactly is sought, what is currently active, what new activities may be launched, etc. With the availability of hibernation and suspend the latter becomes a serious problem. It also allows the switching activity to be mostly mechanical. Also allows one to defer the decision to resume an activity or start a new one until sufficient information is provided.

The summary is a zoomed out overview of all the available activities. It is opened when any of the following occur:

- A hardware menu button is pressed
- The "Menu" or "Window" key is pressed on the keyboard
- The Activities item is clicked in the Top Menubar
- The hot-corner is used

Landmarks and anchors are particularly important in zooming interfaces or 3D environments since they help mitigate disorientation and dislocation. So, the Top Menubar should remain in place when the Activities Summary is shown. Transitions should also be animated to avoid jarring dislocations.

When launching the Summary an animated transition from the currently active Workspace occurs. To the user it should feel as though they take a step back to gain a wider view of the activity-space. The animation should be based on a physical model to reinforce this notion (slow-start-and-stop for example).

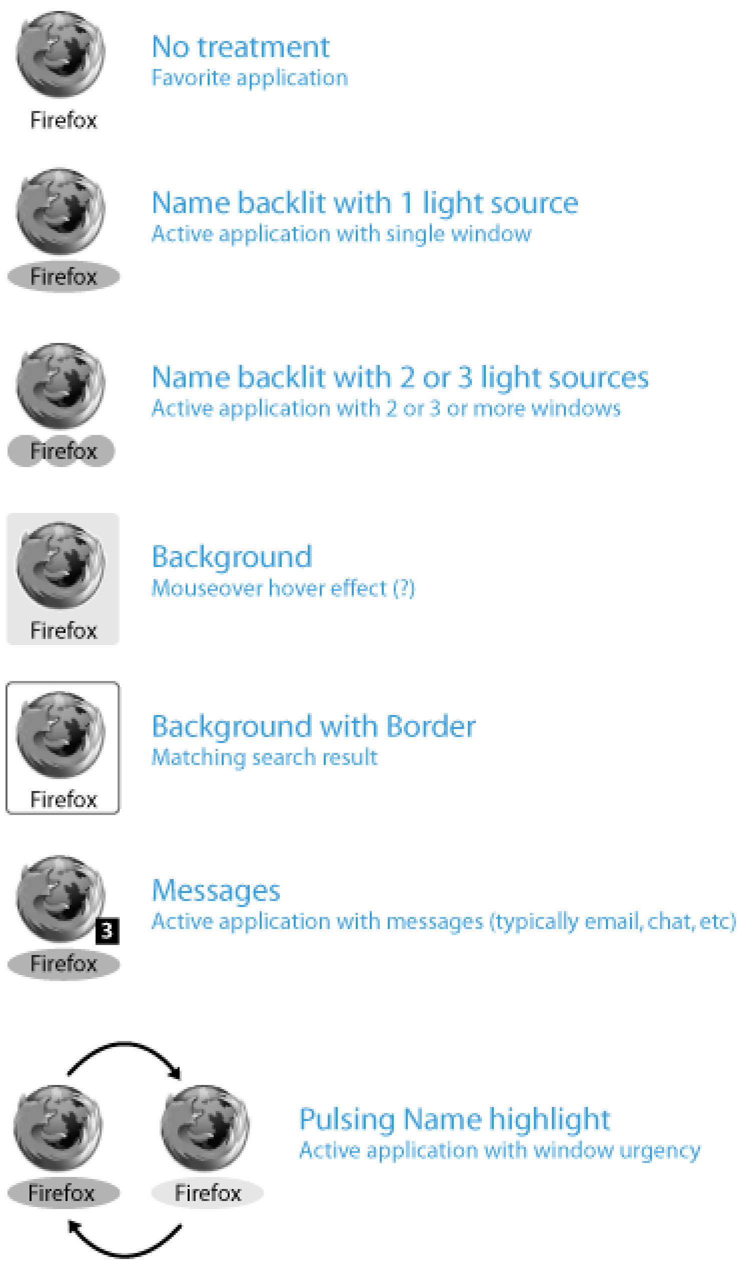
Application Icons

Application Icon shapes may indicate function, brand, or be idiomatic.

They will also be used to indicate the condition or state the application is in. Which may be:

- nominal state
- active application
    - one window
    - two windows
    - multiple windows
- mouse-over/hover focus
- matches search result
- has messages
- requests attention

The following figure shows a possible graphical representation of these states.


Firefox
No treatment
Favorite application
Firefox
Name backlit with 1 light source
Active application with single window
Firefox
Name backlit with 2 or 3 light sources
Active application with 2 or 3 or more windows
Firefox
Background
Mouseover hover effect (?)
Firefox
Background with Border
Matching search result
3
Firefox
Messages
Active application with messages (typically email, chat, etc)
Firefox
Firefox
Pulsing Name highlight
Active application with window urgency

Behaviors for multi-window applications:

- Clicking on an application icon launches the application if it is not running
- When hovering a moment on the icon for a running application, highlight the associated windows in the workspace preview
- When a running application has 1 window, clicking on an application icon goes directly to the application window
- When a running application has 2 or more windows, there are two possible behaviors
    1. Click and release on the icon takes you to the last used window for that application.

2. Click and hold down displays a list of current windows to choose from. New windows should be created in the same workspace as the most recent window. Applications may extend this list to include an option to open a new window.

Favorites Well

The Favorites Well contains a set of Application Icons that represent favorite or frequently used applications.

The behavior of the Application Icons is largely the same as described above. With the addition of the following drag and drop behaviors.

Dragging Application Icons within the Favorites Well will spatially re-arrange applications. An animation should be used to indicate the shifting icons.

Dragging Application Icons from the Running Applications Well or dragging a window from the Workspace View into the Favorites Well will add the application as a favorite.

Dragging Application Icons from the Favorites Well or the Running Applications Well onto a workspace will launch the application or a new instance. While dragging, a cursor indicator should cue the user.

Dragging Application Icons from the Favorites Well to any other unused area will remove it from favorites. While dragging, a cursor indicator should cue the user. A small animation may be shown to indicate completion.

To remove a favorite application the user can right click and choose Remove from Favorites.

Open Questions:

- What happens when it fills up?
- Do we have a limit?
- Should also be able to launch documents in a specific application by dragging the document icon onto an application icon?

Running Applications Well

The Running Applications Well contains a set of Application Icons that are currently running but do not already have a position in the Favorites Well. Applications Icons behave the same as in the Favorites Well.

Workspace View / Window Montage

The goal of the Workspace View is to identify and select windows in their running context. In order to facilitate this, each Workspace will present the application windows associated with it in the form of a Window Montage. The Window Montage has the following properties:

- All non-utility windows running on the workspace should be appear in a non-overlapping, tiled array

- Utility and dock windows should not appear in the window array
- Windows should appear as a scaled down versions of the running application window (complete with live updates)
- It is extremely important that as much of the fidelity of the original window be preserved in the scaled version in order to facilitate identification
- An application icon should appear over the center of each scaled window
- A text label for the window should appear over the window on pointer hover
- One of the following layout strategies may be used (must be tested):
    - Taskpose' WindowRank style PDF
    - predictable and stable positioning algorithm in order to engage spatial cognition
    - motion is minimized during the zoom-out transition to the Activities Summary from the active workspace

Open Questions:

- Allow control+click to select multiple windows to be non-overlapping (layout assistance)
- Use metrics such as time focused, number of focus switches to determine size of window image
- Zoom in on window when hovering for a certain time? to full size? and interactive if holding a key down

Recent Items and History

[NOTE: This section is only a set of notes so far. Needs much more research and design]

Filing, finding, reminding, archiving, and summarizing

Open Questions:

- Episodic rather than semantic memory
- Limit or restrict exposure to filesystem and organizational hierarchies?
- Easily identifiable retrieval cues?
- How to handle Ephemeral, Working, Archived sets?
- (recent, ?, cabinet?) or (today, this week, the past?) Add Tomorrow, Next Week, Someday? for reminding.
- All other sets are only a "view" of the archive? So using a deletion metaphor may be wrong because you really just want it to vanish from the present view. So labelling or tagging? "Move to archive?"
- Add a working set area/target? Or pin to Recent Items... Or need more space... Not physically present - moves to archive?
- Is spatial memory useful for long term use? Automatic layout appropriate for short term use?
- Should link the "encoding conditions [of event] and the retrieval environment" Tulving, Thomson 1973
- Show preview of document in the application it was used in last? workspace?
- Previews for all types of files (audio video too)?
- Sense of neighborhood / proximity
- Engage sense memory? visual, auditory cues?
- Engage associative memory?
    - relationships / referrer

    - time / date
    - place (home/work/etc)
- See what was used at the same time? Bundle?
- Maybe provide new "file save" dialog that shows a large preview of the file (maybe even allows changing the preview - flipping to another page or a later frame in a movie) and allows the application to provide context cues to be associated with the file. The context fields will be prefilled by the application but allow the user to override or clear them. Examples include: title, from-web-site, from-referrer-page, from-email-address, summary, author, file-type, etc. Some of these fields are filled by the application that is saving the file and some are detected from the file-type handler. It may also allow the file to be tagged with a user-defined set of tags. Recently used tags and especially tags that were recently used in the current application or workspace should be suggested. It should not expose the file-system hierarchy to the user. It should be able to detect duplicates before saving. In the case of a duplicate it should allow the existing information to be supplemented or revised. Should allow the file to be placed into agenda slots or just stored in the archive.
- Should the "file-open" dialog should allow for searching and scanning based on the attributes stored at filing time?
- Maybe have the ability to add notes to documents at any time. Maybe even have an item on the window titlebar to turn "notes" on and off. Probably shouldn't be stored in the file or the application itself because they aren't really properties or attributes of the object - they belong to the user, the user's thoughts. Maybe visualized as a layer over the window. Would it also be useful to pin notes to specific parts of the document? Even allow freeform shapes - I'll call these "scribbles"? This may even allow a form of collaboration. Send file+notes in an email or an IM chat? Default to using the person the file was received from.
- Important that notes not be part of of the object itself because the "document" may not even be writable by the user or even exist in any concrete form? Also for privacy reasons... The shell should make a snapshot of the window at the time of the note.
- Should provide a mechanism for applications to automatically save files and perform version control (saved undo/redo)?
- Instead of delete... use forget? To remove from archive/memory and remove all linkage.

Search

[NOTE: This section is only a set of notes so far. Needs much more research and design]
Should be able to search Applications, Documents, Contacts, Music, Videos, Images, Messages, and Web History.

Journal Summary

[NOTE: This section is only a set of notes so far. Needs much more research and design]

Open Questions:

- Temporal, historical record?

- Include a Today folder that has a spatial layout of activities and documents?

Application Switcher

Application window switching has to be as simple as possible. Keyboard activated application window switching is a powerful tool for intermediate and expert users on Linux, Windows, and Mac platforms. The Shell should implement similar functionality. It may be expected that novice users will use the overview until they have been trained to use the Application Switcher keyboard shortcut.

Some of the differences from the existing GNOME switcher should include:

- Align this switcher with the layout norms and visual style of the shell - Ex: bigger icons
- Fast switching to any running application regardless of workspace.
- Show one icon per application, but provide a way to switch to any specific window.
- Add a way to preview or improve the usefulness of selected window highlighting
- Allow mouse selection
    - Two handed use is more efficient
    - Avoids the tab cycling problem

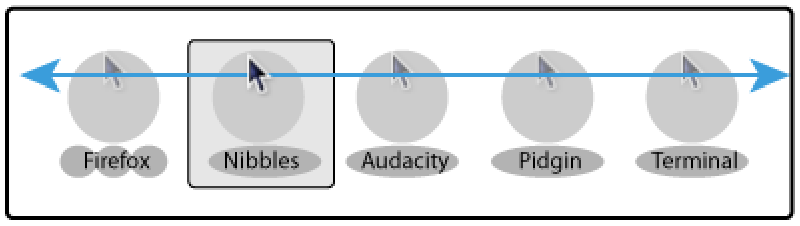
Basic control

- The Alt-Tab keystroke calls up the switcher
- All Active applications in all workspaces shown
- Applications are ordered by time of last use.
- Single icon per application, regardless of number of windows. Users can optionally select specific windows for multi-window apps (see more below)
- One application is always selected
- Per current behavior, releasing Alt key is how the user selects the application to switch to. Clicking is not required.

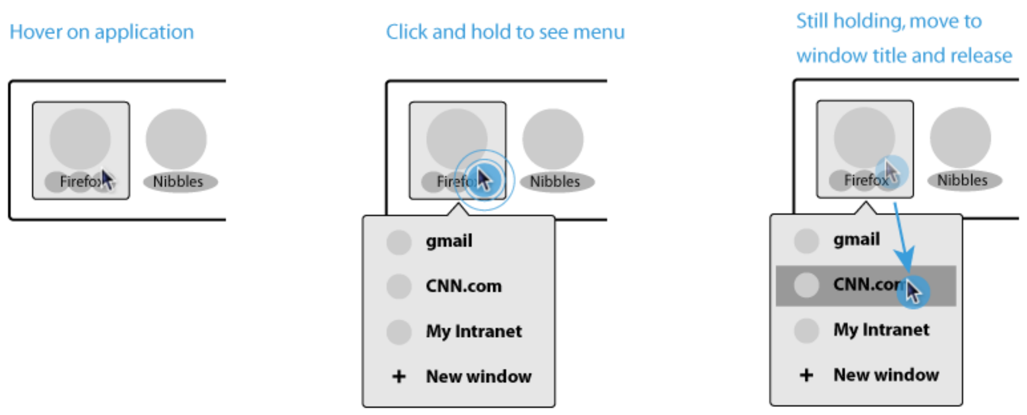
Using the mouse to set focus

Using the mouse is a second, more direct way to set the focus in addition to repeatedly pressing the Tab key. Users are frequently annoyed with tabbing when they tab too far. Employing the mouse, which is likely to already be in their hand, lets the user quickly select arbitrarily from the application list.

Use mouse hover in conjuntion with tabbing to set switcher focus



Multiple Windows



- Click and hold to reveal the window(s) for the selected application (note: need a keyboard equivalent like space or down arrow)
- We could use the same window list as in the overview design. And, like the overview, show a preview for each window as you hover over it.

Open Questions:

- If a window to reveal is in a different workspace than the current one, quickly switch to that workspace first.
- If an application has windows in multiple workspaces, switch to the multiple workspaces view as needed. Same effect as in the shell.
- Perhaps we can even show a hint to use alt-tab when we detect that the user is going in and out of the shell to switch windows in rapid succession?

Windowing Area

Window Decorations

Window decorations are the frame around the application content area of a Window. It is sometimes referred to as the chrome. Historically, these frames were used to:

- indicate input focus by changing the frame color
- provide access to a window management menu
- provide access to window management shortcuts such as close, minimize, and maximize

This frame was historically not drawn by the application and as a result appeared to be different from the rest of the window, both in visual appearance and behavior. There are a few reasons why we may wish to reconsider this approach.

- Windows may request that they not be provided decorations
- Allow decorations to be drawn by the application in order to achieve a consistent look across the application
- Minimize use of decorations so that they don't distract from the focal area
- Allow the rendering system to draw the entire window at once

Therefore, we should carefully consider how we use decorations to indicate status or provide core functionality.

Since window decorations, especially the title bar, may appear indistinct from the rest of the application, the user should be able to click-and-hold on any inactive/unused area of the window in order to move it. For the same reason all windows should have a handle in the lower right hand corner to afford the ability to resize the window.

Open Questions:

- Would it be useful to add some pointer friction to the edges of windows so that controls at the edge of window have a larger effective target area. This may be a big improvement for usability of scrollbars in particular.

Window Captions / Title-bars / Banners

The title line of a window is variously called the window caption, title-bar, or window banner. This is typically one part of the window decoration. Historically, it was used to label and identify a window, facilitate moving a window, indicate input focus, and host window manager controls.

Should not be used as the primary mechanism for indicating input focus. Rather, it should appear integral with the rest of window.

The window-manager menu should not be accessed from a highly prominent button in the title area. It may be accessed from a right click if necessary.

Window States

Maximized

On displays with less than 768 pixels of vertical space windows should default to open maximized.

Open Questions:

- Is there a good way to collapse caption/title-bar into top menubar?
    - Probably not since the top menu-bar should only appear on the primary monitor
- Should the window consume all of screen or use an application defined optimal size (a la OS X)?
    - Obviously this is would be moot on small screens.

Minimized

Open Questions:

- Could animate and appear to be consumed by the Activities button
- Could "iconify" to a live snapshot that lives on the wallpaper
- Could simply push the window back in the window stack
- Could behave like the "hide window" option in OS X

Full-screen

Open Questions:

- Should we allow the hotcorner to be used? If not we should use an audio cue to indicate why it isn't working
    - Maybe not since we don't have the Activities button to "guard" the corner. And also the fullscreen window may be on a monitor that doesn't have the top menu-bar.
- A consistent way to leave fullscreen - or is that the hotcorner?

Moving Windows

Since window decorations may appear indistinct from the rest of the application it is required that the user be able to click-and-hold on any inactive/unused area of the window in order to move it.

When windows are moved more than half their size between drawing updates (as a result of dropped redraws or discontinuous pointer movement) a motion blur animation effect should be used. This may appear as a partially translucent streak between the positions. In some cases, this blur effect may be used even for small discontinuities such as when a password dialog shakes side to side (oscillates) to indicate failure. Perhaps both the relationship of distance travelled to size and the magnitude of the acceleration of the object can be metrics for determining if the effect should be used.

Resizing Windows

Windows should have a resize handle at the lower right corner of the window with an affordance of dragging. Windows may be resized at the other corner control-points. Windows should not be resized on any of the horizontal or vertical edges of the windows. This obviates the need to have window decorations on those sides. It also allows for the possibility of using Fitt's Law advantages for reaching application controls when not using maximized windows.

When the frame rate is insufficient to smoothly transition between window sizes the change should use an animation effect similar to those used when moving windows. In the case where even this is not possible, a wireframe should be used instead.

Open Questions:

- Support a modifier key (for experts) to snap resizing to vertical-only or horizontal-only?

New Windows

Newly created windows should be placed in an unused region of the currently active workspace, if possible.

Requiring the user to manually resize the window should be avoided. On small screens new windows should default to open maximized.

Open Questions:

- Animation to use?
- Document windows should remember last size?
- Avoid manual resizing.

Closed Windows

Open Questions:

- Animation to use?

Window Urgency

The following technical description is from Section 4.1.2.4. WM_HINTS Property of the Inter-Client Conventions Manual (ICCM):

> *The UrgencyHint flag, if set in the flags field, indicates that the client deems the window contents to be urgent, requiring the timely response of the user. The window manager must make some effort to draw the user's attention to this window while this flag is set. The window manager must also monitor the state of this flag for the entire time the window is in the Normal or Iconic state and must take appropriate action when the state of the flag changes. The flag is otherwise independent of the window's state; in particular, the window manager is not required to deiconify the window if the client sets the flag on an Iconic window. Clients must provide some means by which the user can cause the UrgencyHint flag to be set to zero or the window to be withdrawn. The*

user's action can either mitigate the actual condition that made the window urgent, or it can merely shut off the alarm.

Rationale

This mechanism is useful for alarm dialog boxes or reminder windows, in cases where mapping the window is not enough (e.g. in the presence of multi-workspace or virtual desktop window managers), and where using an override-redirect window is too intrusive. For example, the window manager may attract attention to an urgent window by adding an indicator to its title bar or its icon. Window managers may also take additional action for a window that is newly urgent, such as by flashing its icon (if the window is iconic) or by raising it to the top of the stack.

One of the problems is that while the event may seem urgent from the point of view of the application or the window manager, it is very rarely urgent from the point of view of the user. Most environments today follow the recommendations of the ICCM and attempt to draw the attention of the user by blinking, flashing, or bouncing something on the screen in front of the user. This behavior is acutely distracting and preempts the user's focus and attention. [Citation needed] What makes matters even worse is that the motion does not give the user any context or reason for the disruption. So, the distracting motion when coupled with the user's curiosity and fear of "missing" something important results in what should be a modeless indication becoming, in practice, modal and preempting.

Some uses of the urgency hint today include:

1. Launched, non-focused application has finished starting up
2. Not explicitly started application pops open a window
3. A new window opened on another workspace
4. An unfocused tab finished loading in browser
5. Requested window is open on another workspace (eg. Documents folder is open on workspace 1 and it is requested from workspace 2)

[todo: make a list of examples]

Of which, types 3, 4, and 5 should probably not happen at all.

We present the following as an alternative indication for 1 and 2.

When an application window pops open behind an active window due to focus-stealing prevention the active window should momentarily become partially translucent and the new window should appear underneath. A subtle sound effect should be played. We should provide sufficient information to applications so that they may avoid even attempting to open a window while the user is busy and thereby avoid the focus-stealing scenario in the first place.

When an application window pops open on an inactive workspace it should appear as a ghosted image of the actual window and point in the direction of the workspace it opened on.

In the case where the initial indication was missed or forgotten we should also offer a reminding function. This suggests that using the Message Tray. In the Activities Summary the Application Icon should gently pulse (see Application Icon States) and the application window may as well.

Open Questions:

- Show a quick light trace around the new window outline?
- Should we extend the urgency hint to provide a "reason"?
- Should we also show the application icon in the Message Center?
- Should the message tray be used instead of the application item - could provide more context?
- Perhaps slightly delay pop up if the user is in the act of typing or using the pointer?

Startup Notification

When starting a new application, an Active Application Item will be placed in the Top Menubar. This will serve as an indication that the application is starting and hasn't yet created any windows.

Application Tabs

Tabbed windows are a type of interface where the application assumes part of the role that the window manager typically plays. Tabs are essentially docked or grouped windows. These should be integrated into the window and application switching functionality of the Shell.

Finding open windows / Resuming Activities

There are two basic interactions for finding and resuming things that are in progress: the Activities Overview, Application Switcher.

Open Questions:

- One of the problems with resuming activities is reminding yourself where you left off. One thing may be to provide a visual cue to at least where you last interacted with the window. Maybe show a circular pulse of light in that area?

Workspaces

Workspaces may be useful for spatial or logical arrangement of windows or to define a locality for a conceptual task. The usage is highly personal. The shell must not use a pre-determined number but start with one and allow the user to add as needed.

[Explain how this differs from "rooms" or "tasks". Cite "When One Isn't Enough" and "Rooms"]

Open Questions:

- How does the workspace view in the overview appear when using wallpaper slideshows?

Multiple Monitors

Additional monitors should be treated as secondary or auxillary when not cloned from the primary display. The Top Menubar should only appear on the primary display.

Open Questions:

- What about http://en.wikipedia.org/wiki/ASUS_Eee_Keyboard ?
    - I suppose having the overview (perhaps without window selector?) and the Message Tray would be pretty nice.
    - Or simply having the Application Switcher (alt-tab) always there would be cool.

Window Postures

Sovereign windows

- Side by side and maximized states are important
- Should support mouse gestures to move into side-by-side or maximized state

Modal, transient windows

- Must not be moved independently from the parent window
- Must not be closed independently from the controls offered
- Should not have a title
- Must not appear in any window lists independently of the parent
- Therefore, should not have a caption/title-bar
- Since the transient window will obscure the parent window and in order to support the case where information from the parent window is needed in order to complete interacting with the transient window we should temporarily make the transient window translucent when the mouse button is held down over the parent window area.

Input focus identification

Windows must be able to be clicked to bring them into the foreground and in order for them to receive keyboard input focus (ie. make active). This is particularly important for windows that are already partially visible. However, it can be difficult to find or identify an area to click in an unfocused window (particularly when overlapped or partly obscured) that won't be responsive to the click itself. This is even more difficult for windows without window decorations or when the decorations are obscured.

So, most windows should not be responsive to pointer clicks unless they are already in the foreground. Scrolling events are not subject to the same issues and since they may be desirable even for unfocused windows they should always be allowed.

There are, however, some cases where sending click events to windows not on top of the window order may be desirable or necessary. For example, any application with a multiple window workflow such as an Image Editor or any application with floating tool palettes. So, for this to work, the visual appearance of the window should indicate a) if it has keyboard input focus b) whether the window is responsive to mouse clicks.

Unfocused windows that will not have mouse click events forwarded to them should be dimmed. It is thought that desaturating (or removing color) from windows should be reserved for windows that are no longer responding and that may be "dead".

Applications must have a way to tell the window manager whether pointer events should be forwarded to an unfocused window. The unfocused window may allow one of the following:

1. should never receive clicks
2. should receive clicks when any window from the same application is in focus (eg. _NET_WM_WINDOW_TYPE_UTILITY)
3. should receive clicks when a _NET_WM_WINDOW_TYPE_UTILITY window of the same application is active
4. should always receive clicks

So the follow window states are possible:

1. active window : normal appearance
2. inactive and click forwarding is allowed
3. inactive and click forwarding is not allowed

The currently active application should display a single menu item on the Top Menubar using the name of the application as the item label. This will augment the window shading indicators.

When windows are in side-by-side comparison mode mouse clicks should be sent to each and therefore neither window should be dimmed.

Open Questions:

- Should we only dim when there is an overlap?
- What about secondary displays?
- Does window group play a role here?
- Should we just say that utility windows always appear on top of the windows that they act as controllers for - and just never forward clicks?

Window Control Gestures

It is very common for users to want to see two windows simultaneously or side-by-side for transcription and comparison purposes (see the Windows 7 blog for usage data). Also, with the increasing prevalence of widescreen format displays it may be increasingly difficult to size a window to a comfortable width for reading. Historically, this was easily achieved by "maximizing" the window. So it would be advantageous to offer ways to efficiently view two windows at once, with a minimal amount of set up and to resize to an optimal size.

A side-by-side mode should be offered so that two windows can split the viewing area of a single display. This should be accomplished by dragging the window against the left or right side of the display. The window should then snap into the half-screen mode.

To facilitate using an optimal reading width the "maximize" control should instead use a "maximal" or "optimal" size that is determined by the application. There may also be a keyboard shortcut for this mode.

Keyboard shortcuts

The Shell should offer a number of standard keyboard shortcuts in order to enhance the efficiency of expert users.

Some of these are:

- Alt+Tab: Move forward to the next most recently used application in a list of open applications
- Shirt+Alt+Tab: Move backward through a list of open applications (sorted by recent use)
- CTRL+Tab: Switch to the next child window of a Multiple Document Interface (MDI) or tabbed program
- Alt+F1 or Logo key: Go to Overview
- Alt+F2: Quick launcher
- Alt+F4: Close the current window
- Alt+F7: Move the current window
- Alt+F8: Resize the current window
- Alt+F9: Hide window
- Shift+Alt+F9: Hide all but the active window
- Alt+F10: Maximize the active window
- ALT+Enter: Open the properties for the selected object
- CTRL+Alt+left/right/up/down: Move to workspace left/right/up/down
- Shift+Alt+left/right/up/down: Move window to another workspace
- F1: Help
- Logo+(+/-): Zoom in/out
- PrtSc: Print screen
- Power Button: Suspend system
- Power system off
- CTRL+A: Select all the items in the current view
- CTRL+B: Bold
- CTRL+C: Copy
- CTRL+F: Find
- CTRL+G: Find again
- CTRL+I: Italic
- Control+J: Jump to selection or cursor
- CTRL+N: New
- CTRL+O: Open
- CTRL+P: Print
- CTRL+Q: Quit
- CTRL+S: Save
- CTRL+U: Underline
- CTRL+V: Paste
- CTRL+W: Close
- CTRL+X: Cut
- CTRL+Y: Re-do
- CTRL+Z: Undo
- Press Shift five times: Toggles StickyKeys on and off
- Esc: Cancel
- ?: Lock screen
- ?: Tile or untile all open windows on active workspace
- ?: Tile or untile all open windows on active workspace for active application
- ?: Tile most recent N windows
- ?: Peek through active window (ie. make transparent)

- ?: Move focus to the Top Menubar
- ?: Move focus to the Message Tray

Persistence

After a reboot the Shell should return to a nominal state with a single workspace and no currently running applications unless they have been set up to automatically start. In cases where persistence is desired the user should use the system sleep and hibernate functionality as appropriate. To encourage this the default power button action should be to put the system to sleep.

Applications should be expected to maintain their own state and to recover from power loss or application failure gracefully and with no loss of data. Document-based applications should retain the state of the document if it is closed and opened. Not-document based applications should retain application state as much as possible.

Personalization Opportunities

The shell aims to be a vehicle and not a destination and as transparent as possible so the number of personalization opportunities should be carefully selected and limited.

We should support three primary types of extensions to the core functionality and experience:

1. At the platform level to shape a different user experience (usually for devices with different form-factors and goals). I would expect this to probably be only at build/integration time.
2. Something like a status area where extensions behave in a very consistent and well-defined manner.
3. A "place" where the rules are more relaxed and fun things can happen - maybe a sidebar - maybe Vegas...

Notes

Notifications

Notification bubbles are an attempt to balance peripheral alerting with the non-preemption principle. They have been widely deployed and are currently used in one form or another on almost every available platform. As an alternative to modal interactions they have been largely successful. However, close inspection reveals a number of problems with the current designs:

- Notifications implemented as "bubble" overlays may unintentionally intercept the user's pointer clicks. At minimum, this causes an extra click to be required by the user and accidental and irretrievable dismissal of a notification. And at worst, this causes the accidental triggering of an action associated with the notification.
- Notifications on OS X and Linux typically appear near the upper right corner of the windowing area of the screen.&nbsp; On Linux, this was done so that they may point to an active status icon. This is problematic because the upper right hand corner is used frequently for window caption actions (eg. window close). In addition to this, there are a number of items in the status status area or top "panel" that may act like menus and drop down below into the target area for the notification bubbles. So, bubbles appearing in the upper right corner dramatically increases the likelihood of collisions. (This problem is less severe on OS X because the caption controls are on the left hand side.)
- Coupling notifications to status icons leads to an unnatural association between the two. It implies that notifications may become "resident" by adding a status icon. This is partially responsible for the dilution of the status area and the perception that it is a "notification area".

One attempt to address this problem calls for the removal of actions associated with the notifications. For a design where notifications appear as bubbles overlaid on active areas of the screen, removal of actions makes a good deal of sense. But doing so implies providing a different way of handling things that are messages and not simple non-critical, transient notifications. Messages are much less interesting when considered in isolation. It is more natural to be able to respond to a message through direct manipulation. However, providing different interfaces for notifications and messages may not be ideal as the distinction may be too subtle or pedantic for most to appreciate.

We may solve the problem by providing a distinct place for notifications and messages.

Social Dimension / Collaboration

Should be addressed in version 2.0.

Applications and Activities

Whenever possible applications and activities should adapt to the capabilities of the device.

Generally, most applications should be single-instance so that clicking on the application launcher behaves consistently with opening a new file or window from within the application. For example, clicking on the Inkscape launcher twice should not launch two separate instances of the application that are unaware of each other.

Desktop

Used for both ephemeral, and working set data finding and reminding. Given time, the constant stream of things to do, the constant remainder that does not get done, and the unwillingness to categorize and archive manually, and the fact that the solution doesn't scale (due to being spatially bound) results in the system breaking down. On top of this - so to speak - is the problem that this data lives underneath all of the current activities on the computer and is therefore very difficult to reach. Which also tends to reduce its effectiveness for finding and reminding. It also doesn't provide any form of prioritization.

In the Shell design, the "desktop" folder should no longer be presented as if it resides behind all open windows. We should have another way of representing ephemeral and working set objects.

The reminding function of the desktop is really only available immediately after login. Once any activities are started its effectiveness is dramatically diminished. Starting the Journal automatically at login will have a equivalent effect and have the advantage of being easier to access later.

Trashcan Target

The need for a trashcan target is greatly diminished by removing the desktop folder from the computer background. However, to remove the need for it one needs to go a step farther and drop the use of the spatial file management (ie. the desktop metaphor) altogether. It will be necessary to build a Trash capability into the Journal Summary and history archive that are described in a later section. If we need a metaphor at all then perhaps a better one is to "forget" the object.

Taskbars

Some of the problems with taskbars include:

- useless if only windows open for the application are docks etc that skip the taskbar
- group windows by application or class and not spatially or contextually
- are useless for finding applications or windows located on other workspaces
- provide no visual cues for identifying the window
- do not integrate at all with application launchers
- number of open window may outstrip the space available

- in effect, the deficiencies motivated applications to maintain their own taskbar (in tabs) resulting in a form of nested tabs interface

Multi-Pointer X

- Can we make the shell "chrome" per-user
- Use different mouse pointers per user / device
- Use separate "active-application" menus that indicate the mouse pointer
- How would you handle the Activities Summary

Activities and Tasks

At this time we will not be addressing the concept of tasks as stored sets of activities as described in:
[citations needed]

Any solution would need to address the difficulties of separating activities into tasks, distinguishing tasks, sharing activities between tasks, naming, saving task state, identifying and selecting tasks to restore, etc. Overall it seems a bit too analytical and reflective for the way many people work - or just "do".

There is also the problem of scale. Many people are involved in hundreds of different tasks per day. How would you represent this in an interface cleanly without overly taxing the user with process.

This may be an area for further research.

References

- http://blogs.msdn.com/e7/archive/2008/09/23/user-interface-starting-launching-and-switching.aspx
- http://blogs.msdn.com/e7/archive/2008/09/29/follow-up-starting-launching-and-switching.aspx
- http://blogs.msdn.com/e7/archive/2008/10/01/user-interface-managing-windows-windows.aspx
- http://blogs.msdn.com/e7/archive/2008/10/04/follow-up-managing-windows-windows.aspx
- http://blogs.msdn.com/e7/archive/2008/11/20/happy-anniversary-windows-on-the-evolution-of-the-taskbar.aspx
- https://wiki.ubuntu.com/NotifyOSD
- https://wiki.ubuntu.com/NotificationDevelopmentGuidelines#Avoiding%20actions
- http://live.gnome.org/Boston2008/GUIHackfest/WindowManagementAndMore
- http://www.engadget.com/2009/06/03/palm-pre-review-part-1-hardware-webos-user-interface/
- http://tronche.com/gui/x/icccm/
- http://mail.gnome.org/archives/desktop-devel-list/2009-April/msg00314.html
- http://en.wikipedia.org/wiki/Table_of_keyboard_shortcuts
- http://support.apple.com/kb/HT1343
- http://support.microsoft.com/kb/126449
- http://lifehacker.com/5132073/the-best-new-windows-7-keyboard-shortcuts
- http://blog.fishsoup.net/2004/09/06/building-a-tool/