xmonad $\gg=$

Mark Hibberd

Jan 31, 2011

XMonad: Presentation Cannon Fodder

- **Simple**. XMonad defines simple window management.
- **Featureful**. Its small code footprint is packed with great features.
- **Quality**. XMonad is built with high-quality, informative, code.
- **Awesome**. Regardless of language and construction, it is an awesome application.
- Isn't a parser or a compiler.

Overview

- This.

Overview

- This.

XMonad: Form and Function

- History.
- Window Managers.
- Demo.

Overview

- This.

XMonad: Form and Function

- History.
- Window Managers.
- Demo.

XMonad: Code and Construction

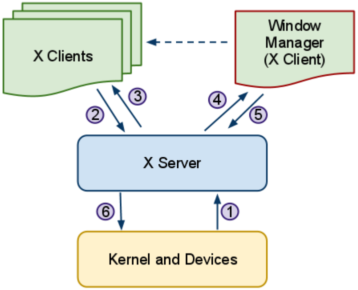
- Architecture and Design.
- Practicality of Pure.
- Data Structures.
- Configuration and Extension.

History

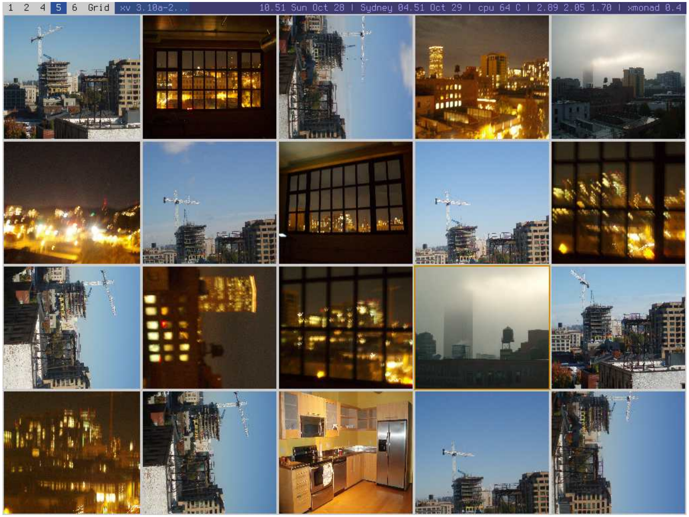
- Created by Spencer Janssen and Don Stewart.
- 1st public commit March 7 2007.
- 0.1 April 2007.
- 0.9 October 2009.
- 0.10 Under development.
- After DWM which set the benchmark for minimal.
- Stated goals:
  - Break down stereo-types of functional programming.
  - Small, quality implementation, big $\implies$ bad.
  - Live the haskell vision - code is more fun when it works.

X11 and Window Managers

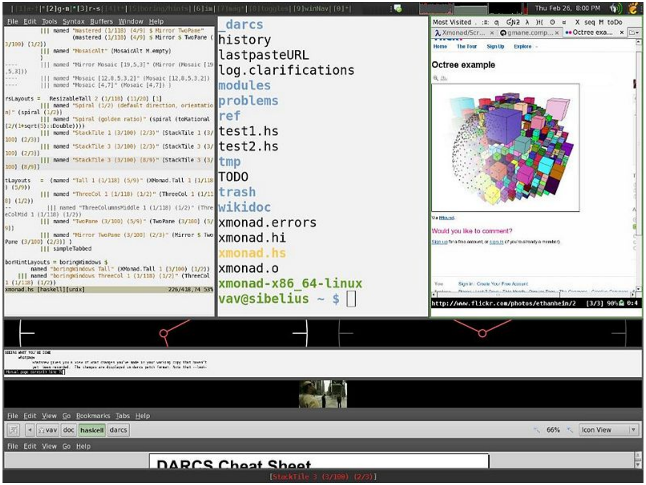
1. Kernel sends events to X server via evdev.
2. X Server passes on events to client to act upon.
3. Clients update and send a rendering event back to X server.
4. X server passes on *damage* event to window manager.
5. Window manager arranges clients and sends an updated rendering event back to X server.
6. X server communicates with kernel and devices to update buffer.



Tiling Window Managers


1 2 4 5 6 Grid
xv 3.10a-2...
10.51 Sun Oct 28 | Sydney 04.51 Oct 29 | cpu 64 C | 2.89 2.05 1.70 | xmonad 0.4

Tiling Window Managers

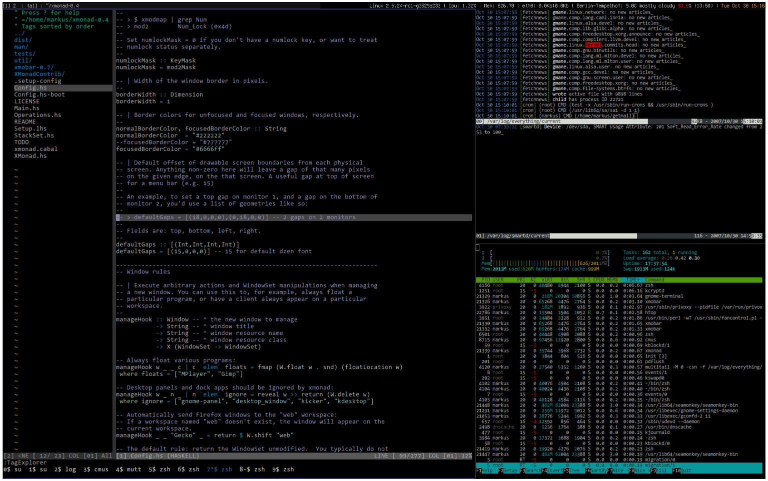


```
            ||| named "mastered (1/118) (4/9) $ Mirror TwoPane"
                    (mastered (1/118) (4/9) $ Mirror $ TwoPane (
3/100) (1/2))
            ||| named "MosaicAlt" (MosaicAlt M.empty)
            )
----        ||| named "Mirror Mosaic [19,5,3]" (Mirror (Mosaic [19
,5,3]))
----        ||| named "Mosaic [12,8,5,3,2]" (Mosaic [12,8,5,3,2])
----        ||| named "Mosaic [4,7]" (Mosaic [4,7]) )

rsLayouts =   ResizableTall 2 (1/118) (11/20) [1]
            ||| named "Spiral (1/2) (default direction, orientatio
n)" (spiral (1/2))
            ||| named "Spiral (golden ratio)" (spiral (toRational
(2/(1+sqrt(5)::Double))))
            ||| named "StackTile 1 (3/100) (2/3)" (StackTile 1 (3/
100) (2/3))
            ||| named "StackTile 3 (3/100) (2/3)" (StackTile 3 (3/
100) (2/3))
            ||| named "StackTile 3 (3/100) (8/9)" (StackTile 3 (3/
100) (8/9))

tLayouts   =  (named "Tall 1 (1/118) (5/9)" (XMonad.Tall 1 (1/118
) (5/9))
            ||| named "ThreeCol 1 (1/118) (1/2)" (ThreeCol 1 (1/11
8) (1/2))
--            ||| named "ThreeColumnsMiddle 1 (1/118) (1/2)" (Thre
eColMid 1 (1/118) (1/2))
            ||| named "TwoPane (3/100) (5/9)" (TwoPane (3/100) (5/
9))
            ||| named "Mirror TwoPane (3/100) (2/3)" (Mirror $ Two
Pane (3/100) (2/3)) )
            ||| simpleTabbed

borHintLayouts = boringWindows $
        named "boringWindows Tall" (XMonad.Tall 1 (3/100) (1/2))
    ||| named "boringWindows ThreeCol 1 (1/118) (1/2)" (ThreeCol
1 (1/118) (1/2))
xmonad.hs [haskell][unix]                         226/418,74 53%
```

```
_darcs
history
lastpasteURL
log.clarifications
modules
problems
ref
test1.hs
test2.hs
tmp
TODO
trash
wikidoc
xmonad.errors
xmonad.hi
xmonad.hs
xmonad.o
xmonad-x86_64-linux
vav@sibelius ~ $
```

|*[2]g-m|*[3]r-s| [6]im| [9]winNav| [0]*|
Thu Feb 26, 8:00 PM
File Edit Tools Syntax Buffers Window Help
Most Visited
Xmonad/Scr...
gmane.comp...
Octree exa...
Home The Tour Sign Up Explore
Octree example
Would you like to comment?
http://www.flickr.com/photos/ethanhein/2 [3/3] 90%
File Edit View Go Bookmarks Tabs Help
vav doc haskell darcs
66% Icon View
File Edit View Go Help
DARCS Cheat Sheet
[StackTile 3 (3/100) (2/3)]

Tiling Window Managers



```
[1] 2  : tall : ~/xmonad-0.4                 Linux 2.6.24-rc1-g3529a233 | Cpu: 1.32% | Mem: 626.70 | eth0: 0.0Kb|0.0Kb | Berlin-Tempelhof: 9.0C mostly cloudy 93.0% (13:50) | Tue Oct 30 15:16
```

```
" Press ? for help
" =/home/markus/xmonad-0.4
" Tags sorted by order
../
dist/
man/
tests/
util/
xmobar-0.7/
XMonadContrib/
.setup-config
Config.hs
Config.hs-boot
LICENSE
Main.hs
Operations.hs
README
Setup.lhs
StackSet.hs
TODO
xmonad.cabal
XMonad.hs
```

```haskell
--
-- > $ xmodmap | grep Num
-- > mod2        Num_Lock (0x4d)
--
-- Set numlockMask = 0 if you don't have a numlock key, or want to treat
-- numlock status separately.
--
numlockMask :: KeyMask
numlockMask = mod2Mask

-- | Width of the window border in pixels.
--
borderWidth :: Dimension
borderWidth = 1

-- | Border colors for unfocused and focused windows, respectively.
--
normalBorderColor, focusedBorderColor :: String
normalBorderColor  = "#222222"
--focusedBorderColor = "#777777"
focusedBorderColor = "#6666ff"

-- | Default offset of drawable screen boundaries from each physical
-- screen. Anything non-zero here will leave a gap of that many pixels
-- on the given edge, on the that screen. A useful gap at top of screen
-- for a menu bar (e.g. 15)
--
-- An example, to set a top gap on monitor 1, and a gap on the bottom of
-- monitor 2, you'd use a list of geometries like so:
--
-- > defaultGaps = [(18,0,0,0),(0,18,0,0)] -- 2 gaps on 2 monitors
--
-- Fields are: top, bottom, left, right.
--
defaultGaps :: [(Int,Int,Int,Int)]
defaultGaps = [(15,0,0,0)] -- 15 for default dzen font

------------------------------------------------------------------------
-- Window rules

-- | Execute arbitrary actions and WindowSet manipulations when managing
-- a new window. You can use this to, for example, always float a
-- particular program, or have a client always appear on a particular
-- workspace.
--
manageHook :: Window -- ^ the new window to manage
           -> String -- ^ window title
           -> String -- ^ window resource name
           -> String -- ^ window resource class
           -> X (WindowSet -> WindowSet)

-- Always float various programs:
manageHook w _ _ c | c `elem` floats = fmap (W.float w . snd) (floatLocation w)
 where floats = ["MPlayer", "Gimp"]

-- Desktop panels and dock apps should be ignored by xmonad:
manageHook w _ n _ | n `elem` ignore = reveal w >> return (W.delete w)
 where ignore = ["gnome-panel", "desktop_window", "kicker", "kdesktop"]

-- Automatically send Firefox windows to the "web" workspace:
-- If a workspace named "web" doesn't exist, the window will appear on the
-- current workspace.
manageHook _ _ "Gecko" _ = return $ W.shift "web"

-- The default rule: return the WindowSet unmodified.  You typically do not
```

```
[2] <NE [ 12/ 23] COL [01] All [1] Config.hs (HASKELL)                      LINE [ 99/277] COL [01] 32%
:TagExplorer
0$ su  1$ su  2$ log  3$ cmus  4$ mutt  5$ zsh  6$ zsh  7*$ zsh  8-$ zsh  9$ zsh
```

```
Oct 30 15:07:58 [fetchnews] gmane.linux.network: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.lang.caml.inria: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.linux.alsa.devel: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.lib.glibc.alpha: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.freedesktop.xorg.announce: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.compilers.llvm.devel: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.linux.kernel.commits.head: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.gnu.binutils: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.lang.ml.mlton.devel: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.lang.ml.mlton.user: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.linux.alsa.user: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.gcc.devel: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.gnu.screen.user: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.freedesktop.xorg: no new articles_
Oct 30 15:07:59 [fetchnews] gmane.comp.file-systems.btrfs: no new articles_
Oct 30 15:07:59 [fetchnews] wrote active file with 9898 lines
Oct 30 15:07:59 [fetchnews] child has process ID 22721
Oct 30 15:10:01 [cron] (root) CMD (test -x /usr/sbin/run-crons && /usr/sbin/run-crons )
Oct 30 15:10:01 [cron] (root) CMD (/usr/lib64/sa/sa1 -d 1 1)
Oct 30 15:10:01 [cron] (markus) CMD (/home/markus/getmail)
00] /var/log/everything/current                          92KB - 2007/10/30 15:10:01
Oct 30 07:39:11 [smartd] Device: /dev/sda, SMART Usage Attribute: 201 Soft_Read_Error_Rate changed from 2
53 to 100_
```

```
01] /var/log/smartd/current                              116 - 2007/10/30 14:59:35
```

```
  1  [|                          0.7%]     Tasks: 162 total, 1 running
  2  [|                          0.7%]     Load average: 0.20 0.42 0.30
  Mem[|||||||||||||||||||||||||||626/2011MB]  Uptime: 17:37:54
  Mem:2011M used:626M buffers:174M cache:999M   Swp:1913M used:124k

  PID USER     PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 4166 root      20   0 40480  4944  2100 S  0.0  0.2  0:06.67 zsh
 1251 root      15  -5     0     0     0 S  0.0  0.0  0:06.16 kcryptd
21329 markus    20   0  218M 20304 10856 S  0.0  1.0  0:03.64 gnome-terminal
21326 markus    20   0 65268  4476  2764 S  0.0  0.2  0:03.10 xmobar
 3922 privoxy   20   0  182M  1892   936 S  0.0  0.1  0:02.97 /usr/sbin/privoxy --pidfile /var/run/privox
22786 markus    20   0 19504  1504  1052 R  0.7  0.1  0:02.58 htop
 3951 root      20   0 24484  3328   912 S  0.0  0.2  0:01.86 /usr/bin/perl -wT /usr/sbin/fancontrol.pl -
21330 markus    20   0 65268  4476  2764 S  0.0  0.2  0:01.65 xmobar
21332 markus    20   0 65268  4476  2764 S  0.0  0.2  0:01.33 xmobar
 6501 root      20   0 40448  4908  2088 S  0.0  0.2  0:00.96 zsh
 8715 markus    20   0 97456 11920  2800 S  0.0  0.6  0:00.92 cmus
   59 root      15  -5     0     0     0 S  0.0  0.0  0:00.69 kblockd/1
21339 markus    20   0 35744  3968  2732 S  0.0  0.2  0:00.67 xmonad
    1 root      20   0  3844   604   516 S  0.0  0.0  0:00.65 init [3]
  201 root      20   0     0     0     0 S  0.0  0.0  0:00.61 pdflush
 4120 markus    20   0 27540  5352  1260 S  0.0  0.3  0:00.57 multitail -M 0 -csn -f /var/log/everything/
    8 root      15  -5     0     0     0 S  0.0  0.0  0:00.56 events/1
  202 root      15  -5     0     0     0 S  0.0  0.0  0:00.46 kswapd0
 4102 markus    20   0 40076  4504  2108 S  0.0  0.2  0:00.41 -/bin/zsh
 4104 markus    20   0 40024  4436  2108 S  0.0  0.2  0:00.40 -/bin/zsh
    7 root      15  -5     0     0     0 S  0.0  0.0  0:00.36 events/0
 4103 markus    20   0 40128  4584  2116 S  0.0  0.2  0:00.35 -/bin/zsh
21448 markus    20   0  462M 62004 23388 S  0.0  3.0  0:00.34 /usr/lib64/seamonkey/seamonkey-bin
21291 markus    20   0  239M 11972  9012 S  0.0  0.6  0:00.34 /usr/libexec/gnome-settings-daemon
21053 markus    20   0 38776  5244  1992 S  0.0  0.3  0:00.33 /usr/libexec/gconfd-2 11
  657 root      16  -4 12592   856   464 S  0.0  0.0  0:00.32 /sbin/udevd --daemon
 2498 dnscache  20   0  5236  1764   388 S  0.0  0.1  0:00.27 /usr/bin/dnscache
  477 root      15  -5     0     0     0 S  0.0  0.0  0:00.25 kjournald
 3984 markus    20   0 37372  3688  1904 S  0.0  0.2  0:00.24 -zsh
   58 root      15  -5     0     0     0 S  0.0  0.0  0:00.23 kblockd/0
21419 markus    20   0 39920  4276  2076 S  0.0  0.2  0:00.23 zsh
21447 markus    20   0  462M 62004 23388 S  0.0  3.0  0:00.19 /usr/lib64/seamonkey/seamonkey-bin
    3 root      RT  -5     0     0     0 S  0.0  0.0  0:00.19 migration/0
    5 root      RT  -5     0     0     0 S  0.0  0.0  0:00.19 migration/1
F1Help  F2Setup F3SearchF4InvertF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Comparison

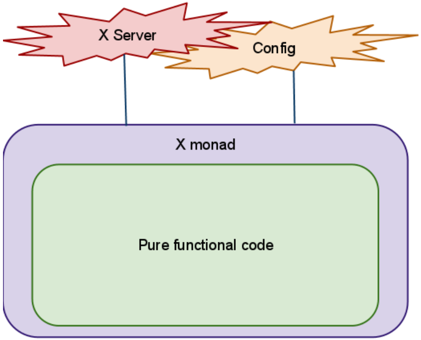
window manager	language	loc	test loc
metacity	C	77683	306
stumpwm	common lisp	17952	226
awesome	C	17130	0
wmii	C	14065	128
dwm	C	2147	0
xmonad	haskell	2222	1215
wm-spec	spec'talk	1712	0

Metrics very roughly gathered Jan 27th 2011.

Demo

Design

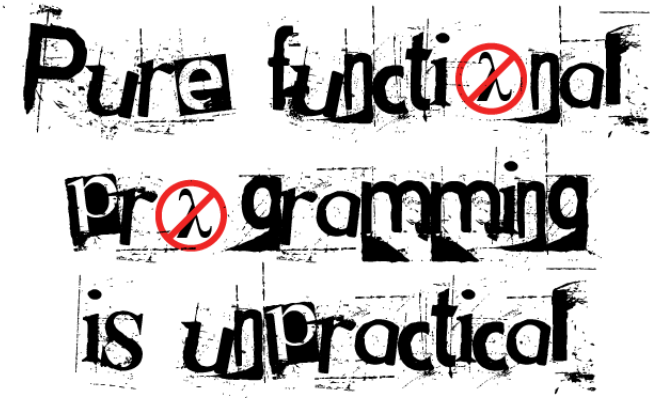
- Purely functional core.
- Thin monadic skin provides a solid, managed edge to the system.
- Interacts with X server and config via X monad (ReaderT, StateT, IO).
- Leverage haskell and its tools for maximum profit.



XMonadContrib

- Core is kept small.
- Users were doing amazing things with their configs.
- XMonadContrib evolved out of user demand for a mechanism share these custom window management hacks.
- Configs are really easy to re-use, and so XMonadContrib exploded.
- http://xmonad.org/xmonad-docs/xmonad-contrib/index.html

Practical Programming



Practical Programming

Pure functional
programming
is unpractical

Practical Programming

Definition

Abstraction: Highlighting essential concepts by omitting specific and needless characteristics.

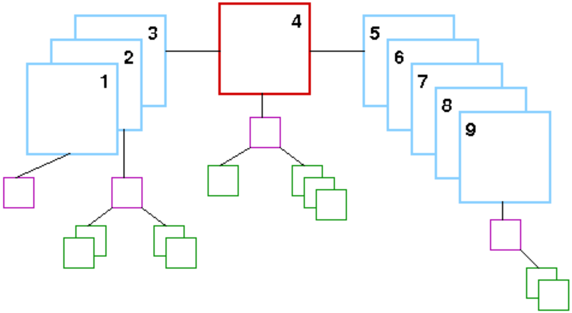
- The pure model taken by xmonad allows for real abstraction.
- The X apis are bad - really bad - but xmonad makes them easy.
- A novice at dealing with X or haskell can still be productive.
- More importantly developers can have a higher level of confidence in the correctness of their program.

Practical Programming

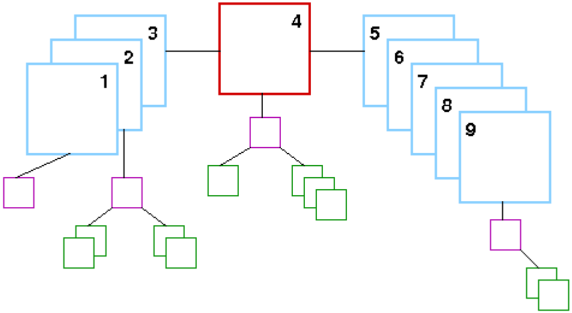
- XMonad is one of the only window managers with robust testing.
- API was driven from QuickCheck. Anytime it was difficult define properties, it triggered a revisit of the data structure.
- 100% coverage on core data structures, verified with HPC.
- Use the type system to prevent bugs.
- Static analysis using Neil Mitchel's *Catch* library.
- Referential transparency and the story of bug 177.

How would you model workspaces and windows?

How would you model workspaces and windows?



How would you model workspaces and windows?



- Purely functional data structure.
- A pointer into a set of workspaces, each with a view into a list of windows.
- Perfect fit for zippers (more a one-hole context than a traditional zipper).

Zippers and One-hole contexts

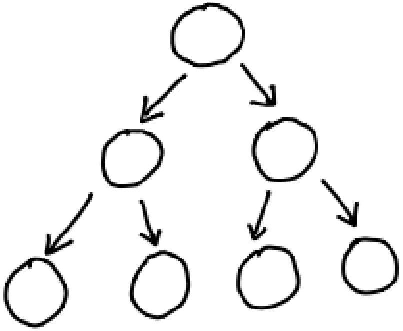
- Efficient navigation for immutable data structures.
- The techniques may be familiar, origins in the 1960s.
- The term *zipper* and its application to purely functional data structures was introduced by Gérard Huet.
- Generalisation of one-hole contexts is presented in Conor McBride's aptly named paper: *Clowns to the Left of me, Jokers to the Right*.

Zippers

Thanks

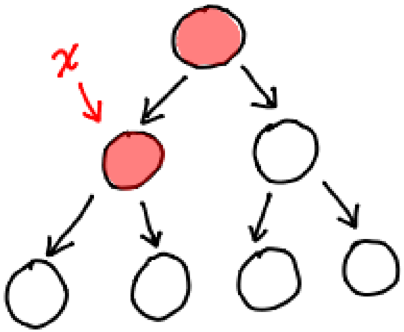
A note of thanks for Edward Yang who gave permission to reproduce the following diagrams to explain zippers.

Zippers



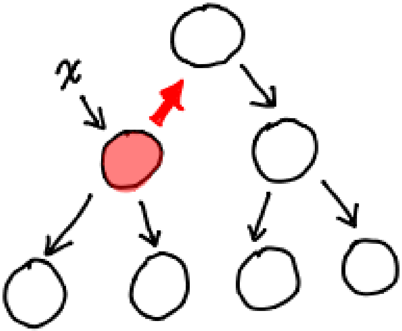
```
data Tree a = Nil | Node a (Tree a) (Tree a)
```

Zippers



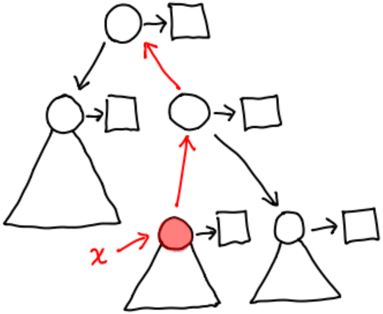
Can traverse with path copying, but lose accessibility to some segments.

Zippers



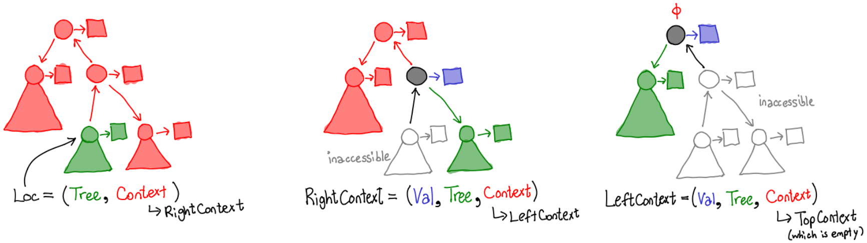
Flip a pointer and you have a zipper.

Zippers



A more comprehensive example.

Zippers



```
data Loc a = Loc (Tree a) (Context a)
data Context a = Top
               | Left a (Tree a) (Context a)
               | Right a (Tree a) (Context a)
```

Stacked Set

```
-- i = tag
-- l = layout
-- a = window
-- sid = screen id
-- sd = screen detail

data StackSet i l a sid sd =
    StackSet { current  :: !(Screen i l a sid sd)
             , visible  :: [Screen i l a sid sd]
             , hidden   :: [Workspace i l a]
             , floating :: M.Map a RationalRect
             } deriving (Show, Read, Eq)
```

Tracking the current workspace, visible - but not focused - workspaces for multi-head support, the hidden workspaces and floating layers.

Supporting Characters

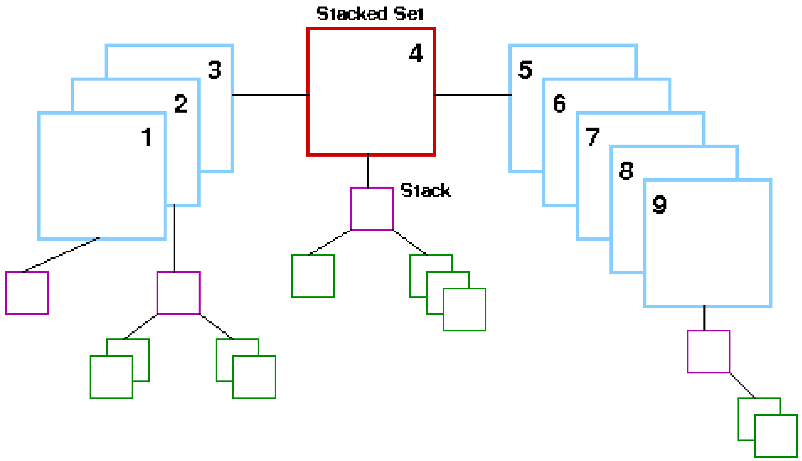
```
data Screen i l a sid sd =
        Screen { workspace :: !(Workspace i l a)
               , screen :: !sid
               , screenDetail :: !sd }
    deriving (Show, Read, Eq)

data Workspace i l a =
        Workspace  { tag :: !i
                   , layout :: l
                   , stack :: Maybe (Stack a) }
    deriving (Show, Read, Eq)
```

Stack

```
data Stack a =
       Stack { focus  :: !a         -- focus
             , up     :: [a]        -- clowns to the left
             , down   :: [a] }      -- jokers to the right
    deriving (Show, Read, Eq)
```

This is a pointed list where the cursor represents the focused windows. And the left most element represent the master window.

Windows revisited.



Window Management API (Simplified)

A simplified api for window management. StackSet is parametrized over a number of variables in reality.

```
-- Constructing a new window manager with 'n' workspaces.
new    :: Int -> StackSet a

-- Extract the currently visible window.
peek   :: StackSet a -> Maybe a

-- Extract the windows on the current workspace.
index  :: StackSet a -> [a]

-- Move the currently focused window to workspace 'n'
shift  :: Int -> StackSet a -> StackSet a
```

Window Management API (Simplified)

```
-- Move focus to the left or right window
focusLeft, focusRight :: StackSet a -> StackSet a

-- Bring a new window under management
insert   :: a -> StackSet a -> StackSet a

-- Delete the currently focused window
delete :: StackSet a -> StackSet a

-- View the virtual workspace to the left or right.
viewLeft, viewRight   :: StackSet a -> StackSet a
```

Notice the symmetry, favour idempotent and reversible operations. Easier to assert properties.

X monad

The X monad is a typical transform stack for an effectful application. It is used to store the configuration environment, track application state and interact with the outside world. In this case to the X Server via FFI.

```
newtype X a = X (ReaderT XConf
                  (StateT XState
                    IO) a)
    deriving (Functor, Monad, MonadIO,
              MonadState XState,
              MonadReader XConf, Typeable)
```

Monadic Armour

```
doLayout     :: layout a -> Rectangle -> Stack a
             -> X ([(a, Rectangle)], Maybe (layout a))
pureLayout   :: layout a -> Rectangle -> Stack a
             -> [(a, Rectangle)]

handleMessage :: layout a -> SomeMessage
              -> X (Maybe (layout a))
pureMessage   :: layout a -> SomeMessage
              -> Maybe (layout a)
```

Configuration

- Pure haskell library, import and run.
- defaultConfig based upon core.
- Use XMonadContrib to pimp your config.

Extension

- Extension and Configuration are equivalent.
- Configurations are highly composable and can be packaged up like any haskell library.

A custom layout

The Reliability Toolkit

XMonad Development Philosophy.

Taken from Don Stewart's presentation: Design and Implementation of XMonad.

- Cabal
- -Wall
- QuickCheck
- HPC
- Type system
- Catch

Kicking Butt with Haskell

XMonad Development Philosophy.

Taken from Don Stewart's presentation: Design and Implementation of XMonad.

- Model effectful systems in purely functional data structures.
- Use QuickCheck as your design assistant.
- Use HPC to keep QuickCheck honest.
- Enforce code quality with serious testing on every commit.
- Don't be tempted by partial functions.
- Don't be tempted by side effects.
- Be responsive to bug reports.
- Look at your competition's bugs, audit and prevent them.

More information

1. XMonad - http://xmonad.org
2. XMonadContrib - http://xmonad.org/xmonad-docs/xmonad-contrib/index.html
3. IRC - #xmonad - irc.freenode.org

References

1. Roll your own window manager - http://cgi.cse.unsw.edu.au/ dons/blog/2007/05/01, http://cgi.cse.unsw.edu.au/ dons/blog/2007/05/17.
2. Design and Implementation of XMonad - http://www.cse.unsw.edu.au/ dons/talks/xmonad-hw07.pdf
3. Functional Pearl, The Zipper - http://www.st.cs.uni-saarland.de/edu/seminare/2005/advanced-fp/docs/huet-zipper.pdf
4. Clowns to the Left, Jokers to the Right - http://strictlypositive.org/CJ.pdf.
5. You could have invented zippers - http://blog.ezyang.com/2010/04/you-could-have-invented-zippers/.
6. Lighter introduction to zippers - http://learnyouahaskell.com/zippers

References

1. Monad transformers - http://book.realworldhaskell.org/read/monad-transformers.html
2. More Monad transformers - http://en.wikibooks.org/wiki/Haskell/Monad_transformers
3. Quick Check - http://haskell.org/haskellwiki/Introduction_to_QuickCheck
4. Coverage - http://projects.unsafeperformio.com/hpc/
5. XMonad and Catch - http://neilmitchell.blogspot.com/2007/05/does-xmonad-crash.html
6. Bug 177 - http://code.google.com/p/xmonad/issues/detail?id=177