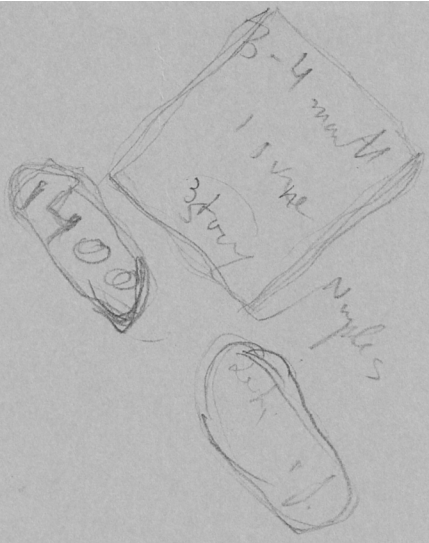


Disp file: of

class: SunRasterCart

type: whale

offset

len

w h

"fu" 5

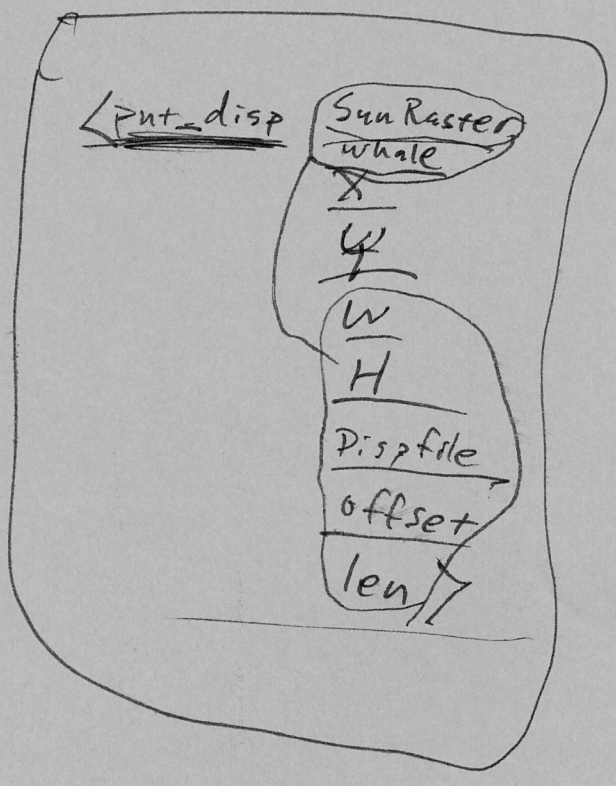
Disp

whale

.target

SunRaster

.picture whale

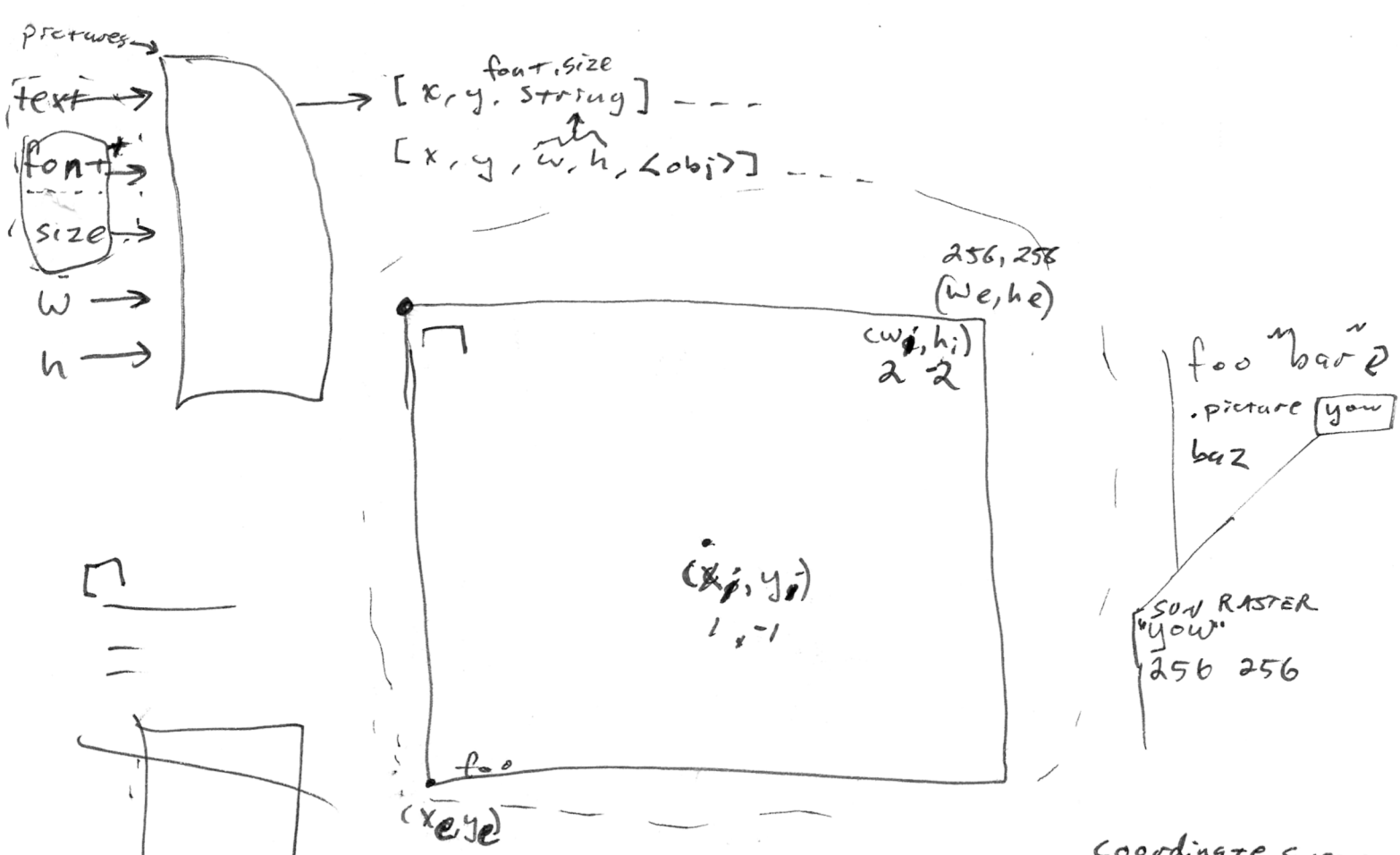


put-target

put_disp(disp, x, y) →

.target target_id ref_id

.picture



coordinate system

% external

% unit square

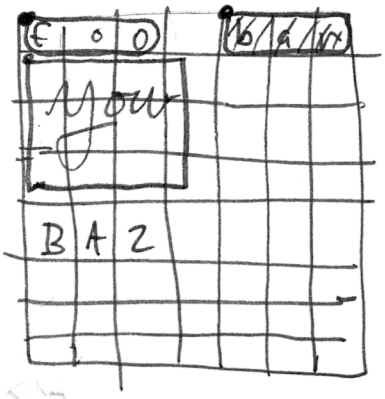
% internal

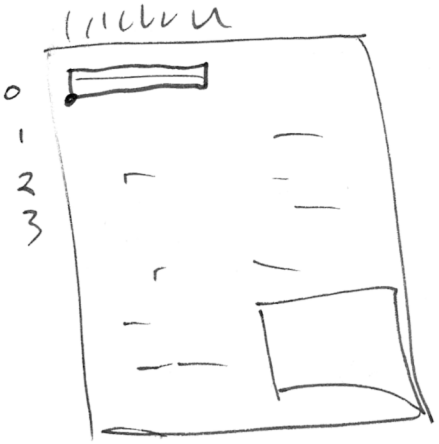
foo ser cannes

clippath pathbbox scale pop pop

1 wi div 1 hi div scale

xi yi translate





.f.o.o.

f.oo

f.o.o.

.pile # ← callbacks do this first

set-pile(n) to set current pile!

do-set_pile(n)

/set-pile % n ⇒ -

These are used to set up new sets of windo

→ .new-pile(x, y, w, h) class

- makes a new pile whose parent is current pile. (or self if no current pile)

.parent-pile

- makes new current pile the parent of current pile.

.definition-pile

- makes a definition-pile link from parent-pile to current pile - (associates /definition-pile with current pileID in PileDict of parent pile.)

.controls-pile

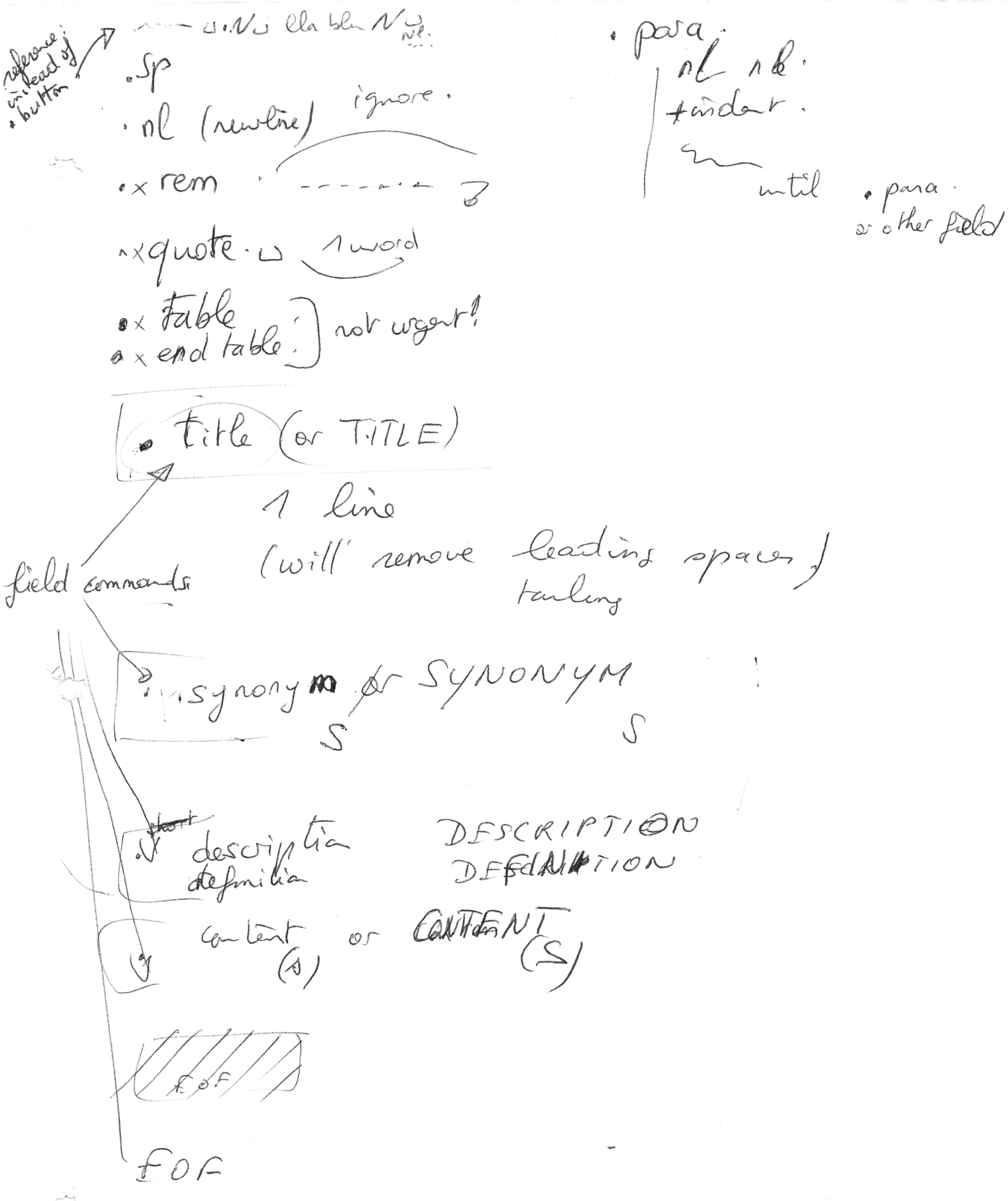
- makes a controls link from parent pile to current pile.

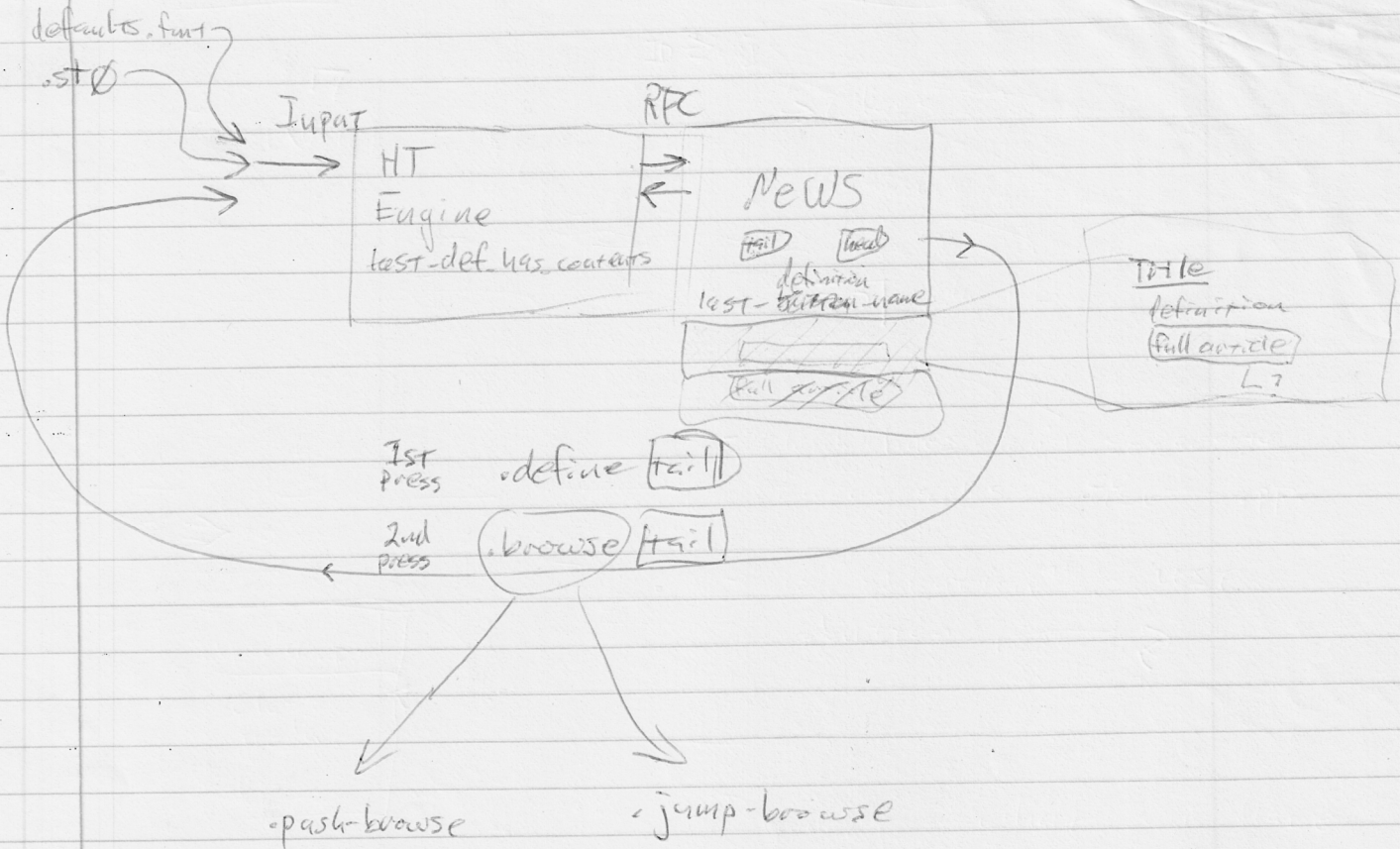
.local

.global

.system

.background


reference: instead of button
.sp
.nl (newline)
ignore.
.x rem
.x quote
1 word
.x table
.x end table
nor urgent!
.para
.nl .nb
.under
until .para
or other field
.title (or TITLE)
1 line
(will remove leading spaces/trailing
field commands
.synonym or SYNONYM
s
s
.description or DESCRIPTION
content (s) or CONTENT (s)
.eof
EOF



- remember if the last thing we defined has a .contents

- double click.

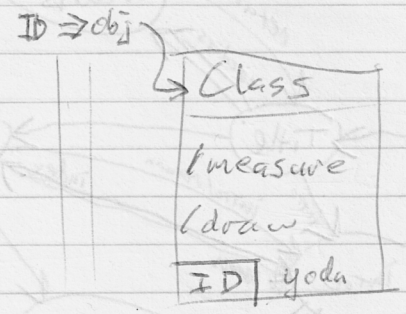
---

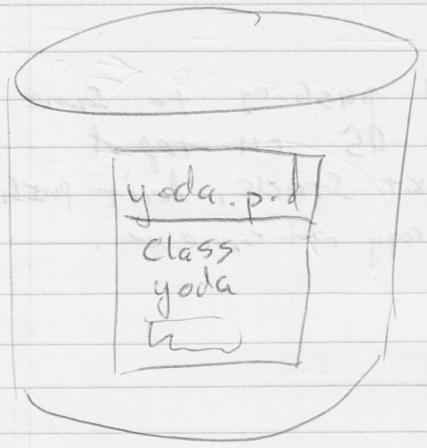
union
HWnd * instance_header

#define sb_instance_header(x) (x).u.

TIES Net NeWS

.yoda 17
(w,h)





- blank lines don't separate paragraphs. use .pp.
- no ~~vbutton~~. use .button

- Document what we have

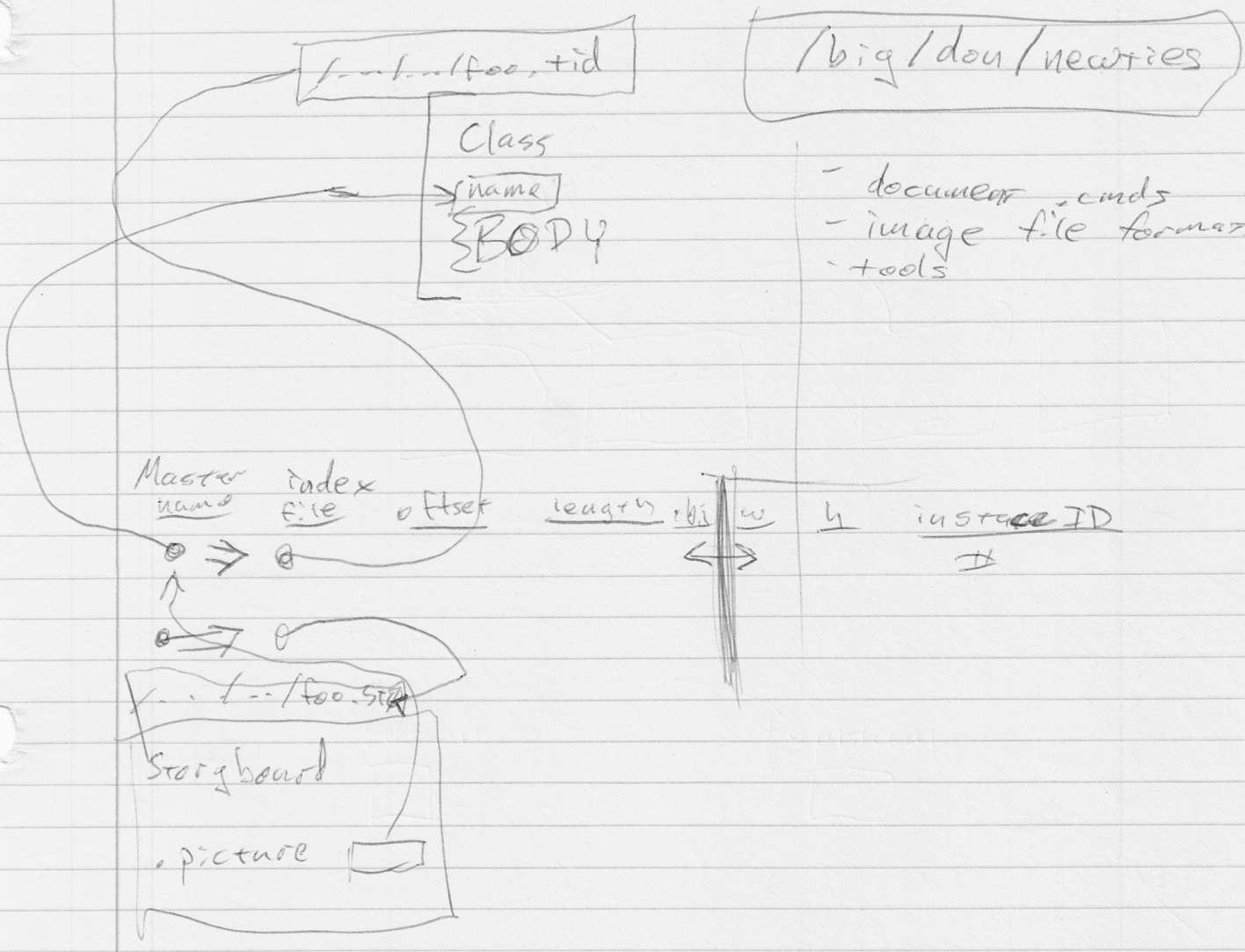
Run browser with root directory as argument.

→ master-index

defaults ⇒ .init
controls .do controls
.do intro
return

.include
.browse info

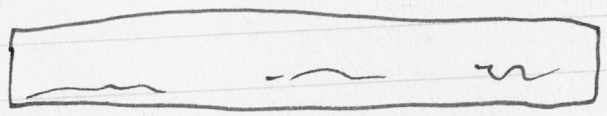
Rename defaults storyboard "defaults


/.../.../foo.tid
Class
name
BODY
/big/dou/newties
- document cmds
- image file formats
- tools
Master name
index file
offset
length
big w
h
instance ID
#
/.../.../foo.sta
Storyboard
picture
BK
Mac PaintWorks
Mac PaintWorks Raster
Serial
Converter Sun
Raster File
MacDraw
Mac PostScript
Serial cable
HyperTIES pid
PostScript

~foo bar~. ← this should work.

of pic. ID

put-picture(x, y, w, h, ~~picture ID~~ offset, len)

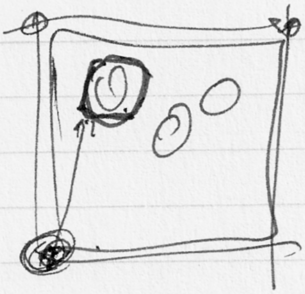
.string "  "

read opcode
~~switch~~ (opcode) {

case put_picture:
read(x, y, w, h, ~~pic ID~~ pid_offset, pid_len);
read_pid(pid_offset, pid_len, pic_ID)
index(pic_ID, file, offset), class)
send(class, PUT_PIC, x, y, w, h, file, offset

~~send~~

foobar [~this~] is a button.



text (x, foobar

text (x, y , [this]

target (" " , [ ] .

[foobar]

.target target_id ref_id

~ . . . . ~

.target tid rid <[foo bar]>

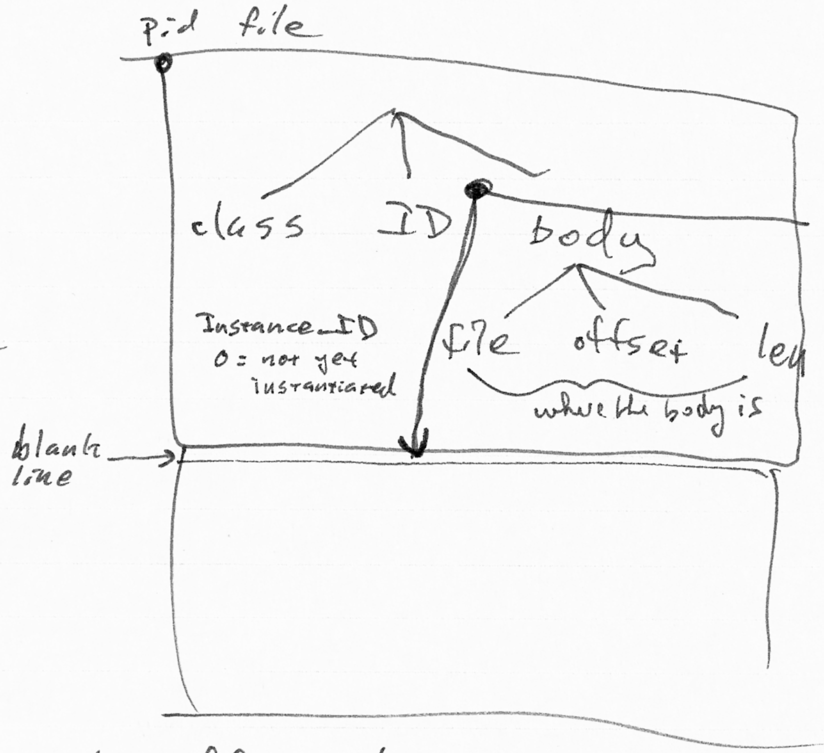
.picture moose / .itallics foo

.target tid rid

.target tid rid

Tid / pid

Class
ID
body < offset, len



Rectangle
"foo"
| 50 50 |

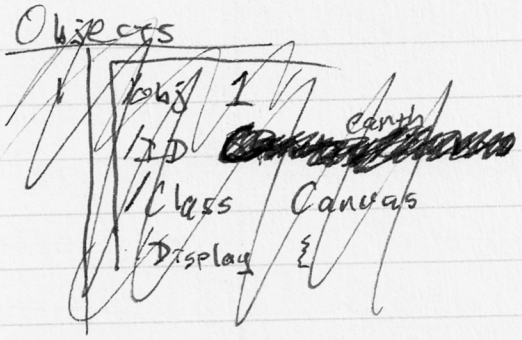
file, offset, len
string

send (Class, ~~ID~~, BODY)
Rect, ~~foo~~

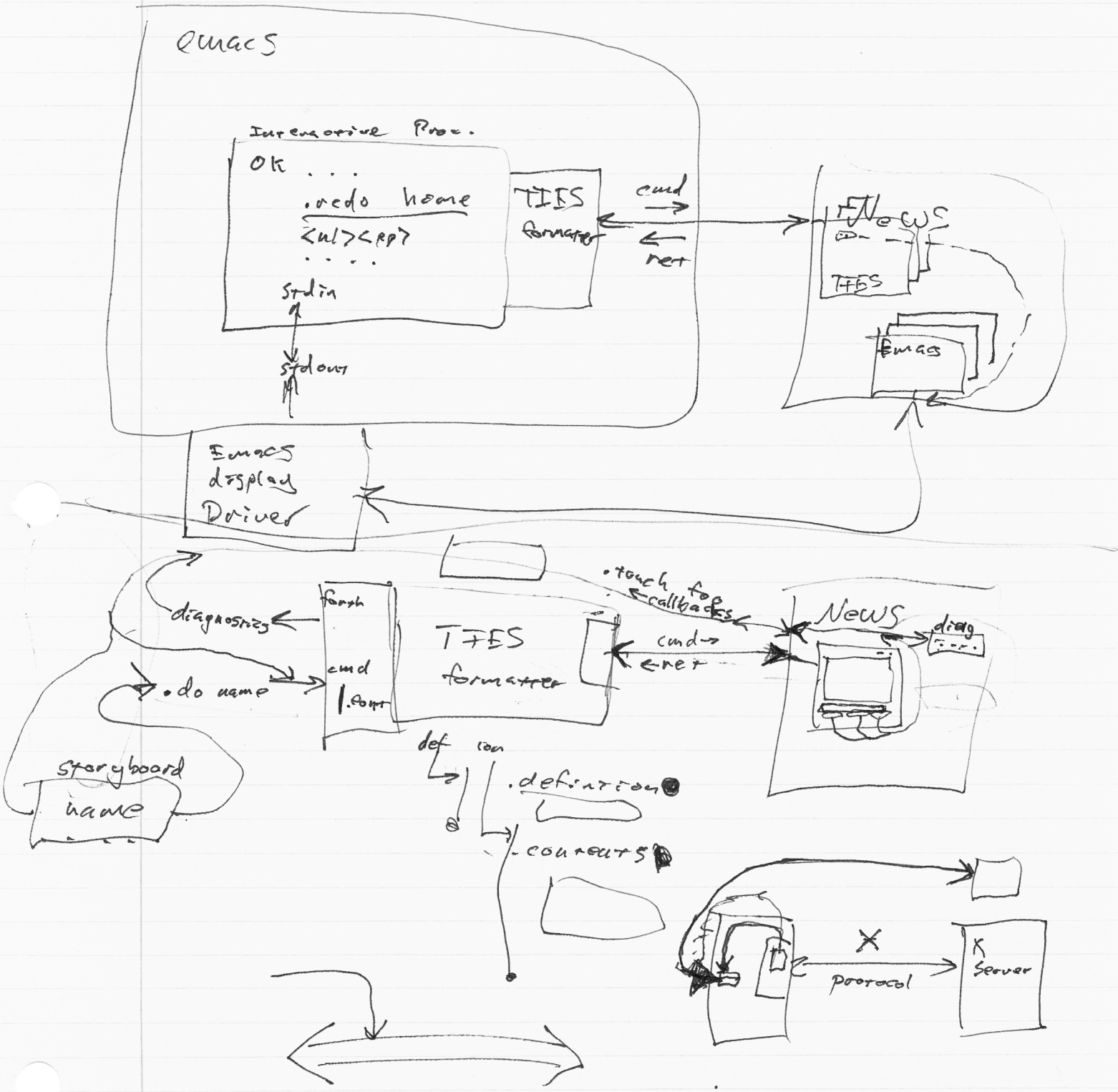
index (→Id, class→, body→)
send (→class, →body, →msg, →Id, →args - - - -

send (→class, →Id, [→file, →offset], →len)

- Namespace of ID's
  ID's have a type:
  - reference_ID
  - target_ID
  - picture_ID
  ? - command_ID



/ Char
/ Name


Qmacs
Interactive Proc.
Ok . . .
.vedo home
<rp>
. . . .
stdin
stdout
TIES
formatter
cmd
ret
FNews
TIES
Emacs
Emacs
display
Driver
diagnostics
forth
cmd
.do name
.forth
TIES
formatter
touch for
callbacks
cmd
ret
News
diag
storyboard
name
def
con
.definition
.constants
X
protocol
X
Server

- ~~Find~~ InIndex (~~index~~, key) ⇒ ENTRY*

- how do we control windows?
- how do we display index? (i.e., anything w/o a file)

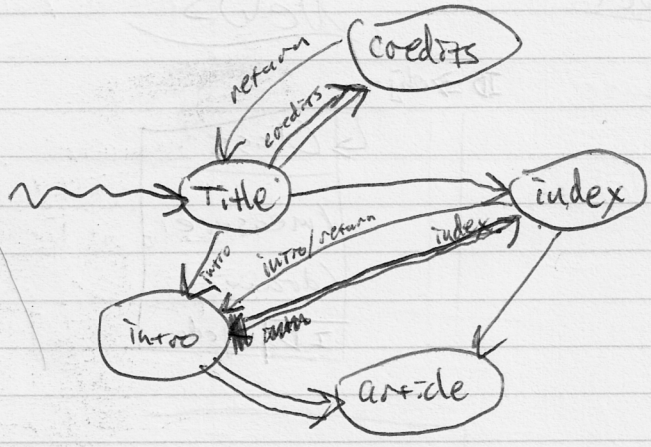
ENTRY struct

type (nearby main or synonym)
- ident.
- filename
- offset
  length

add class → string
  instance-id

OpenFile (entry) ⇒ FILE*

entry* file*



pushing context: when pushing to same place as on top of context stack, don't push extra copy of context.

~~about page~~

jump to index.

jump to same article on top of stack.

push to different article.

- throw away definition when moving ~~back~~
- try making definition title a button
- confirmation on quit

- PostScript
  - Why? - Standard - High level, device independant imaging model.
    - Flexible, High level language. Weird, but powerful. Interpreting
    - Integrates text + graphics

- X11/NeWS - Distributed by AT&T -
  - Two parallel protocol interpreters
  - on top of the same graphics library
  - X11 and NeWS interpreters both in C
  - same event queue — NeWS processes comm. w/ X clients over IPC using special primitives and decode X events
  - same window "forest"
  - NeWS canvas = X11 window ← NeWS can draw on X windows; X can draw on NeWS canvases

- NeWS
  - Extensions to the PostScript language
    - Light Weight Processes
      - process input on behalf of the client, locally. (where I/O does not
        - without consuming comm bandwidth
        - fast, responsive graphical feedback
        - Share data
      - autonomous processes - "desk accessories", system services
    - Input Queue
      - Virtual event mechanism - clean
      - used by NeWS processes to communicate, schedule events, etc.
      - Virtual Devices easy to implement.
    - Receive low level input (left button down at (X,Y) in can C at time T)
    - Give graphical feedback (erase, draw, highlight, rubberband)
    - Translate input from the user into high level app. specific events.
    - High level events are sent to client, over net.

Window managers run in same addr. space as windows they manage!

- Extensible window server ⇒ less client/server traffic
  - fine grained, application specific extensions. Dynamic Postscript CMD
- Object Oriented toolkit / Interactive prog. env.
  - toolkit runs in server, More responsive, less traffic
  - code + data shared by clients.
    - smaller clients, more independent of UI
    - modular look + feel
  - customizable via subclassing.
  - Hack the UI w/out changing, recompiling, relinking app.
  - Show Class Browser
  - Dynamic Debugging, Interactive. Examine & change state of env.
- Build custom UI gadgets by subclassing!
- Building on top of what's already there.
- Multitasking is invaluable for UI (people multitask)

local input processing and echoing.

local menus & control panels just send high level results of interaction.

local repainting of windows when damaged or resized.

CD load Display List (RE exec when needed)

- Arbitrarily Shaped Windows
  - Canvas
    - Shape defined by PS path. Holes, disconnected regions
    - Shape influences: clipping of graphical output, distribution of input events
    - Has its own coordinate system
    - can be shaped to gets, mouse enters, leaves, moves in, clicks
  - Sprites - LW PS proc that blinks, waves, paints. Cursor, space invader

- Emacs -
- Multiple frames
- Dired
- Menu Compile -
- Control Panel Compiler
- Edit/execute PS code
- Psh, shell windows
- tab windows
- text selection (rubber banding, newlines, multiple clicks.)
- Menus - bind, describe-key
- UI styles: change window & menu classes
- font & color menus

- News - PS IO
- Tab / Dired
- menu

- HyperTIES - Hypermedia Browser.
- Embeded Menus - buttons, highlights.
- Space Telescope Database
- Popups, Items, etc. - Demos
- page & button menus.
- Multiple page documents.

- Companies supporting NeWS
- Unipress, SGI, Parallax, Grasshopper, ImageSoft

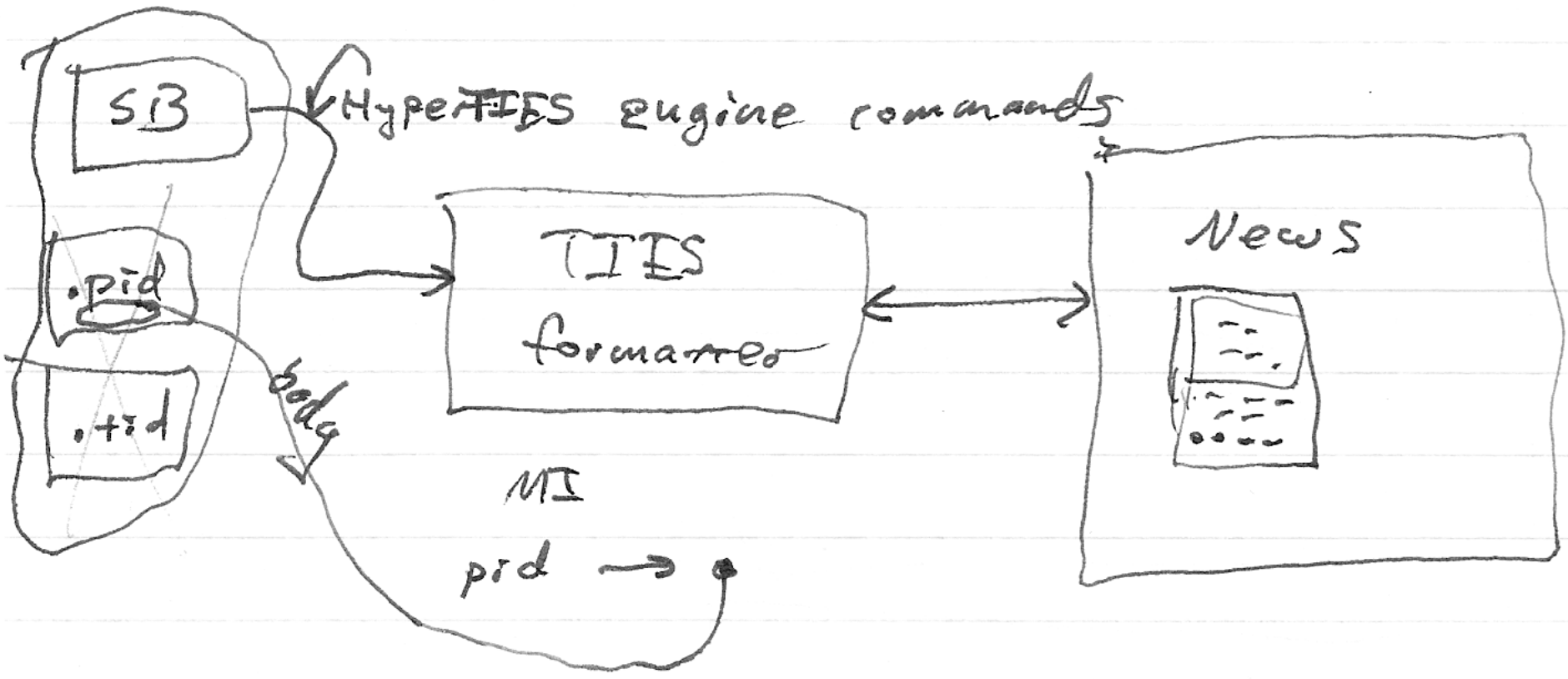
- Pie Menus
- Window manager menus
- Font Menus
- Color Menus
- Pullout Menus
- Precision Pie

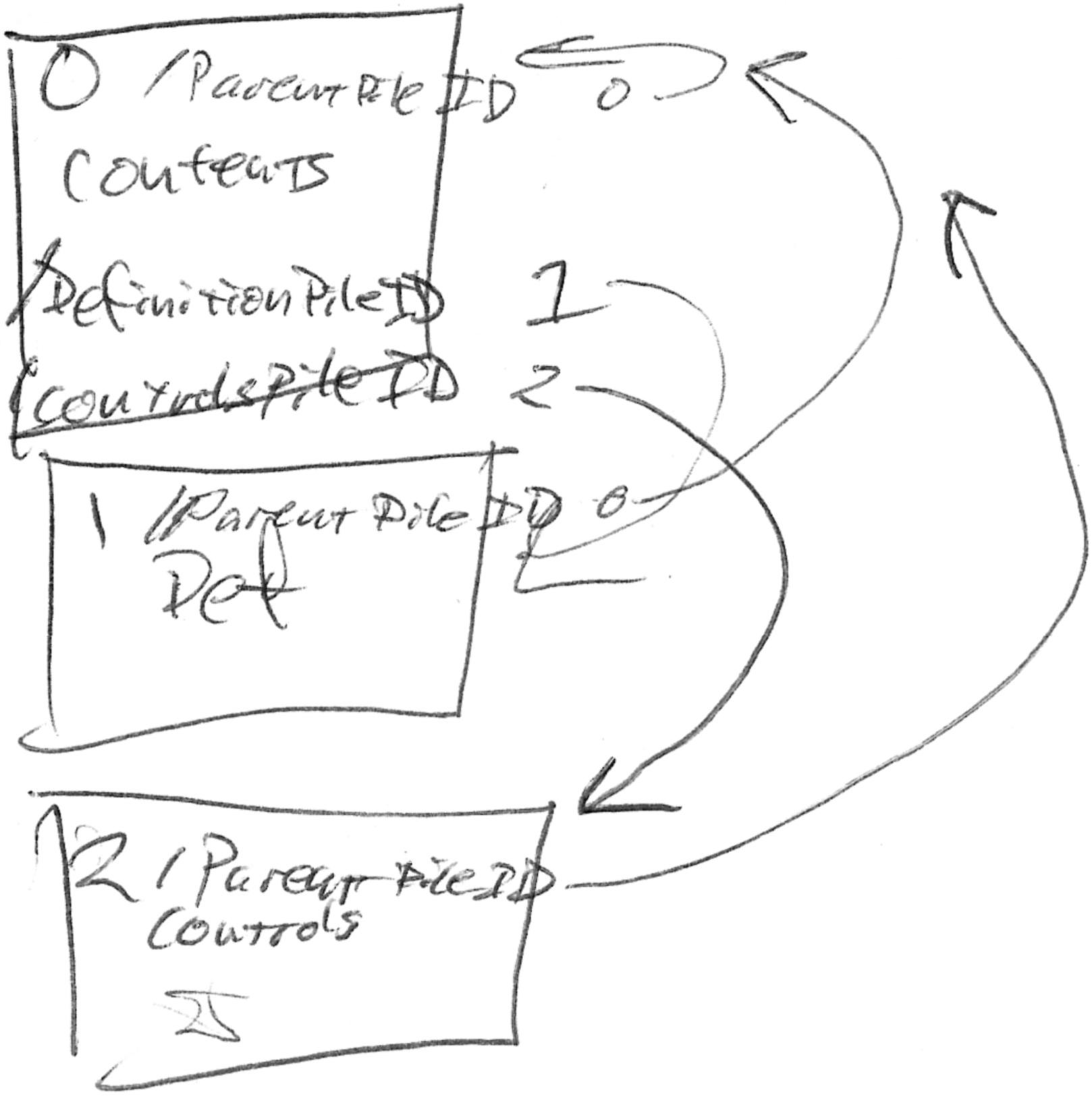
- Object Browser / Pseudoserving if + Visualizer

- Utilities
- ~~Mousee~~ warp
- Keysee
- PSterm - CAM Reagan Bytes
- animator ← Yin Yang

- Hacks - eyes - NewWin Reshape
- like NLP: ● ⇒ reshape

+ Title page: no initial cap!
smaller name font
+ Outline ...


SB
HyperTIES engine commands
.pid
.tid
body
TIES
formatter
News
MI
prd


0 ParentFileID 0
Contents
DefinitionFileID 1
ControlsFileID 2
1 ParentFileID 0
Det
2 ParentFileID
Controls