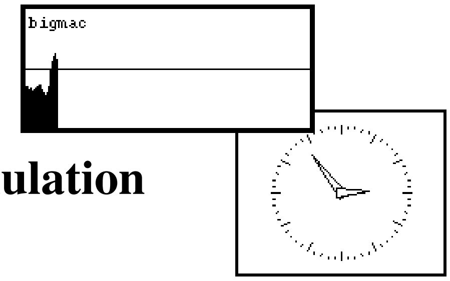
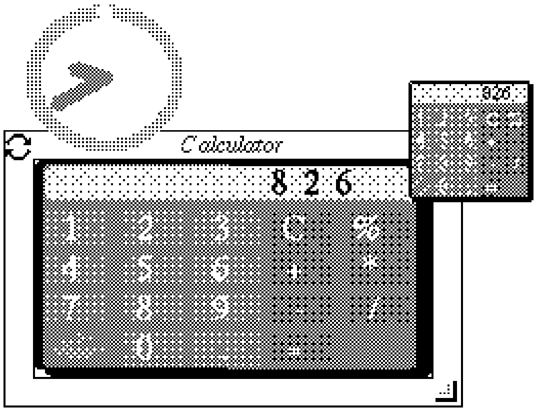
NeWS Styles:

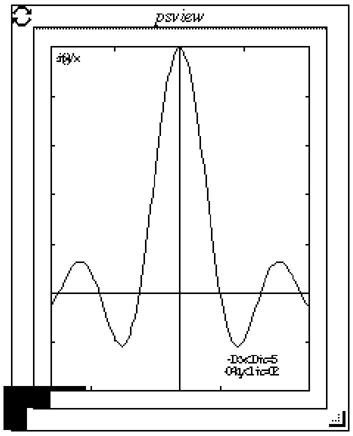
We are limited only by our imagination!



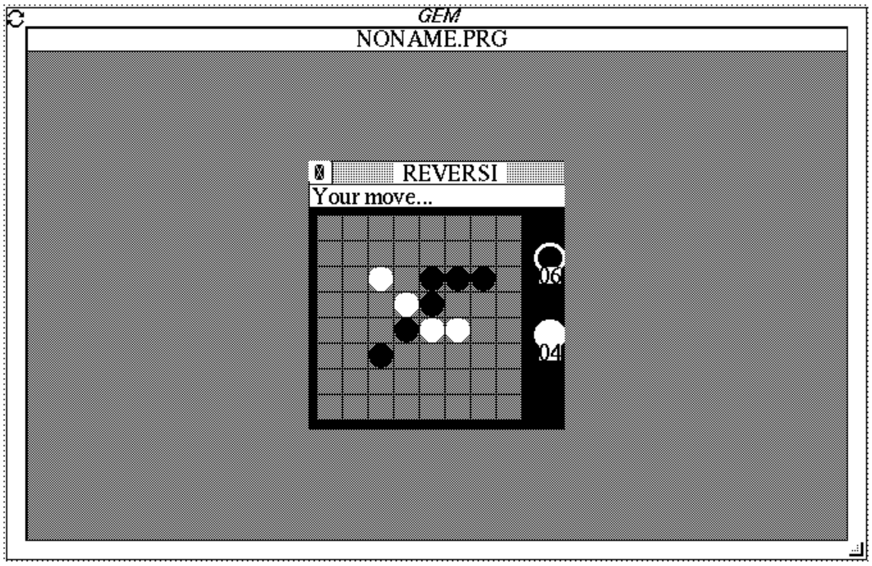
X-10 Emulation



GnuPlot | PSView

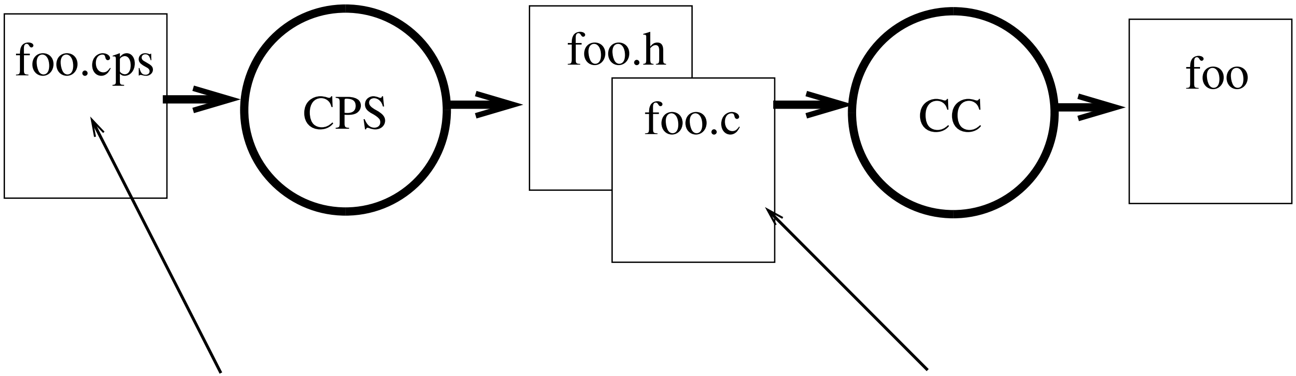


GEM Emulation



NeWS Client Programming:
CPS: The Client - Server Interface

- **CPS: C Language to PostScript Language [and back!]**



```
cdef ps_hand(rot,rad)
   gsave
   rot rotate 0 0 moveto 0 rad rlineto
   stroke
   grestore
```

```
#include "roundclock.h"
main (argc, argv)
   char  **argv;
{
   ....
   ps_hand(-angle*6, radius);
}
```

- **cdef**
  - o cdef ps_drawstring (int x, y, cstring s) x y moveto s show

- **tagprint & typedprint**
  - o PS:

```
/CRedraw { % width height  => -
        REDRAW_TAG tagprint typedprint typedprint
} def
```

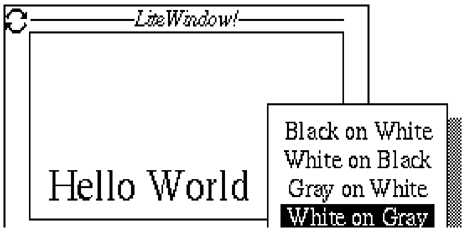
  - o C:

```
foo.cps:    cdef ps_redraw(w, h) => REDRAW_TAG (h, w)
foo.c:      if (ps_redraw(&width, &height)) {..}
```

Hello World!

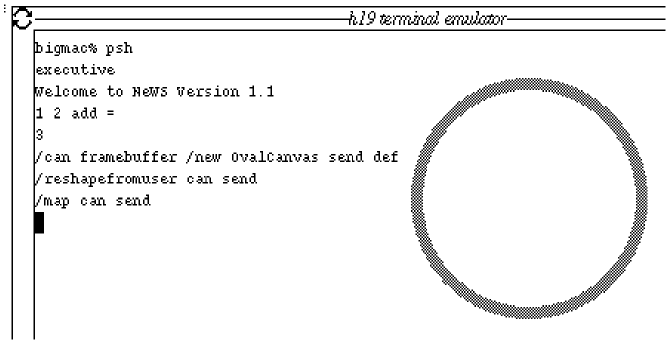
- **Define the PaintClient Code**
- **Define the ClientMenu**
- **Create, Shape, & Map the Window**

```
#! /usr/NeWS/bin/psh
%
% This is a hello_world program with windows & menus.

/TextColor 0 def
/FillColor 1 def
/PaintText&Background {
        FillColor setgray clippath fill
        TextColor setgray 10 10 moveto
        /Times-Bold findfont 24 scalefont setfont
        (Hello World) show
} def
/SetColors { % textcolor fillcolor => -
        /FillColor exch store
        /TextColor exch store
        /paintclient win send
} def

/win framebuffer /new DefaultWindow send def
{       /FrameLabel (LiteWindow!) def
        /PaintClient {PaintText&Background} def
        /ClientMenu [
                (Black on White)        {0 1 SetColors}
                (White on Black)        {1 0 SetColors}
                (Gray on White)         {.5 1 SetColors}
                (White on Gray)         {1 .5 SetColors}
        ] /new DefaultMenu send def
} win send
/reshapefromuser win send
/map win send
```



PSH - The PostScript Program Shell

- **Interactive Usage: Executive**

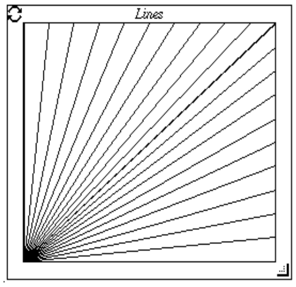


```
bigmac% psh
executive
Welcome to NeWS Version 1.1
1 2 add =
3
/can framebuffer /new OvalCanvas send def
/reshapefromuser can send
/map can send
```

- **Non-Interactive Usage: Scripts**

```
#! /usr/NeWS/bin/psh
% Sun Microsystems, Inc.
% 2550 Garcia Avenue
% Mountain View, California  94043
%lines 6.7 87/03/13
%  Drag a window with a bunch of lines. .
%
/linesperside 20 def
/fillcanvaswithlines { % linesperside => -
    gsave
    1 fillcanvas
    0 setgray
    clippath pathbbox
    scale pop pop
    0 exch
    0 1 3 -1 roll div 1 {
        ColorDisplay? {dup 1 1 sethsbcolor} if
```

/usr/NeWS/demo/lines →



PSH: Using NeWS

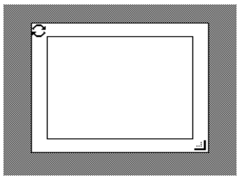
- **Interactive Use: Executive**
- **Non-Interactive Use: Scripts**
- **Hello World**
- **Code Book**

LiteWindow Methods

- **/new: parent => window**

- **/reshape[fromuser]: - => -**

- **/map: - => -**



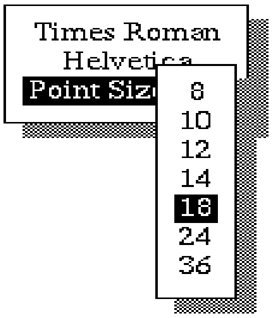
- **Instance Variables:**
  - o /PaintClient, /PaintIcon
  - o /ClientMenu
  - o /FrameLabel, /IconImage

- **Others:**
  - o /paint, /paintframe, /paintframelabel, /paintclient
  - o /reshape, /move
  - o /totop, /tobottom, /flipiconic

LiteMenu Methods

- **/new: key-actionArray => menu**

```
[       (Times Roman)       {/TimesRoman InstallFont}
        (Helvetica)         {/Helvetica InstallFont}
        (Point Sizes =>)    PointSizeMenu
] /new DefaultMenu send
```



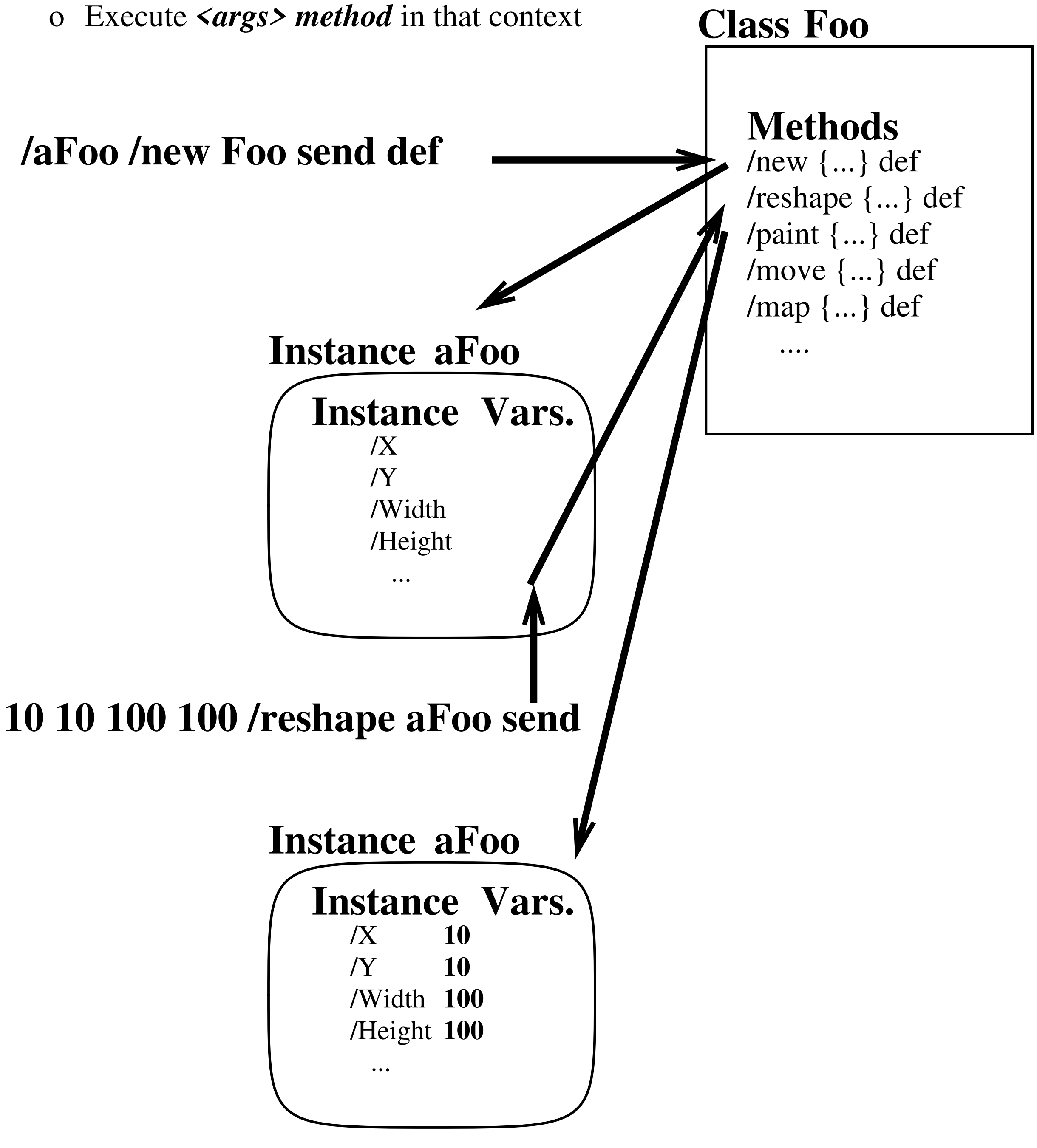
- **/show: menu => -**
  - o Pop-up menu at mouse.
  - o Track & execute action if selected.

{/TimesRoman InstallFont}

- **Others**
  - o /showat
  - o /insertitem, /deleteitem
  - o /currentkey, /currentitem

Classes: Sending Messages

- **Send: <args> /method object send**
  - o Find /method in object's hierarchy
  - o Execute ***<args> method*** in that context



Class Foo

Methods
/new {...} def
/reshape {...} def
/paint {...} def
/move {...} def
/map {...} def
....

/aFoo /new Foo send def

Instance aFoo

Instance Vars.
/X
/Y
/Width
/Height
...

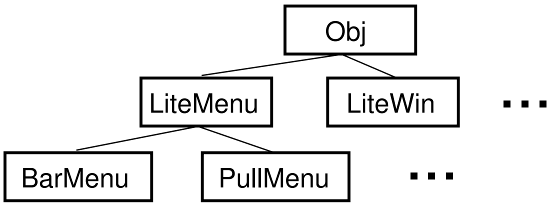
10 10 100 100 /reshape aFoo send

Instance aFoo

Instance Vars.
/X **10**
/Y **10**
/Width **100**
/Height **100**
...

Classes: A Tool for Extension

- **Object Oriented Programming: NeWS Classes.**
  - o <args> /method object send
  - o classbegin - classend
  - o self - super
  - o litewin.ps litemenu.ps liteitem.ps



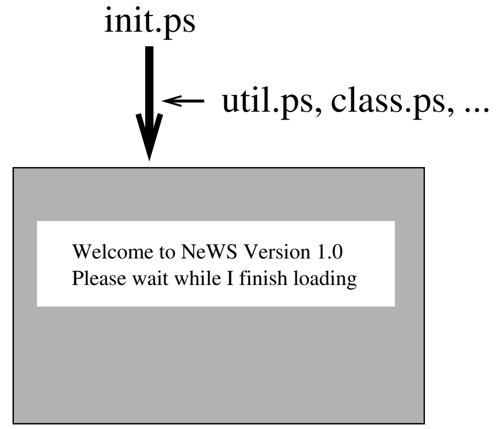
- **Classes & Modules**
  - o Similarities

Modules	Classes
Module Procedures	Methods of Classes
Module Objects	Instances of Classes
Module Globals	Class Variables
Object Variables	Instance Variables

  - o Differences
    Subclassing & Inheritance
    Send
    Self & Super

NeWS++: Extending NeWS

- **Initialization: Read $NEWSHOME/lib/NeWS/*.ps**
  - o init.ps
  - o {util,class,stroke,lite*}.ps
  - o startup.ps user.ps



Welcome to NeWS Version 1.0
Please wait while I finish loading

NeWS++: Extending NeWS

- **Initialization**
- **Classes**
- **Class LiteMenu**
- **Class LiteWindow**

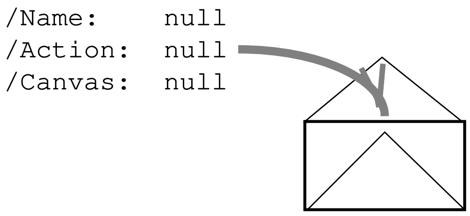
Event Example: "snoop"

```
/snoop {
    /snoopprocess {
        createevent expressinterest {awaitevent ==} loop
    } fork def
} def
/killsnoop {snoopprocess killprocess} def
```

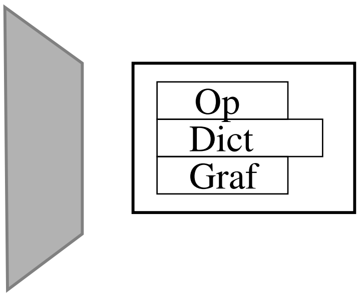
```
snoop
% event([382,223], name(/MouseDragged), action(null))
% event([382,223], name(/LeftMouseButton), action(/DownTransition))
% event([382,223], name(/LeftMouseButton), action(/UpTransition))
% event([382,223], name(28493), action(/DownTransition))
% event([382,223], name(28493), action(/UpTransition))
% event([356,229], name(/MouseDragged), action(null))
killsnoop
```

Event Primitives

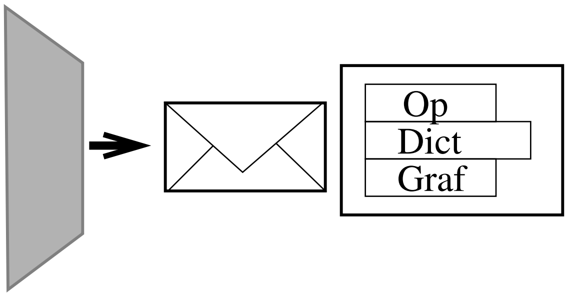
- **createevent: - => event**
  - o Make an empty event.



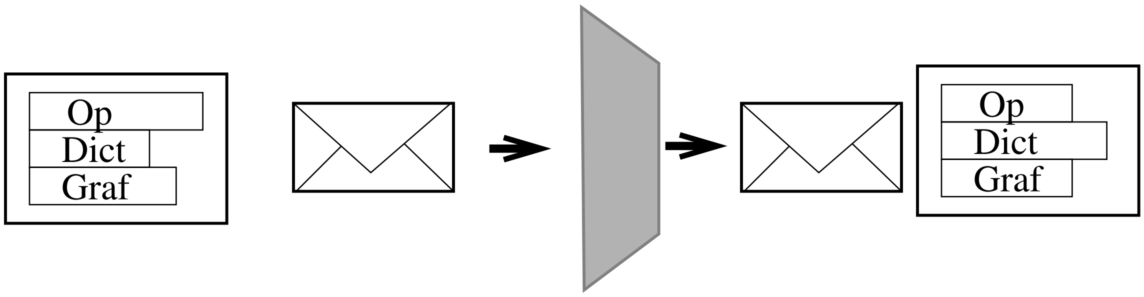
- **expressinterest: event => -**
  - o Add this template to my event filter.



- **awaitevent: - => event**
  - o Block until next quallifying event.



- **sendevent: event => -**
  - o Send an event to others.

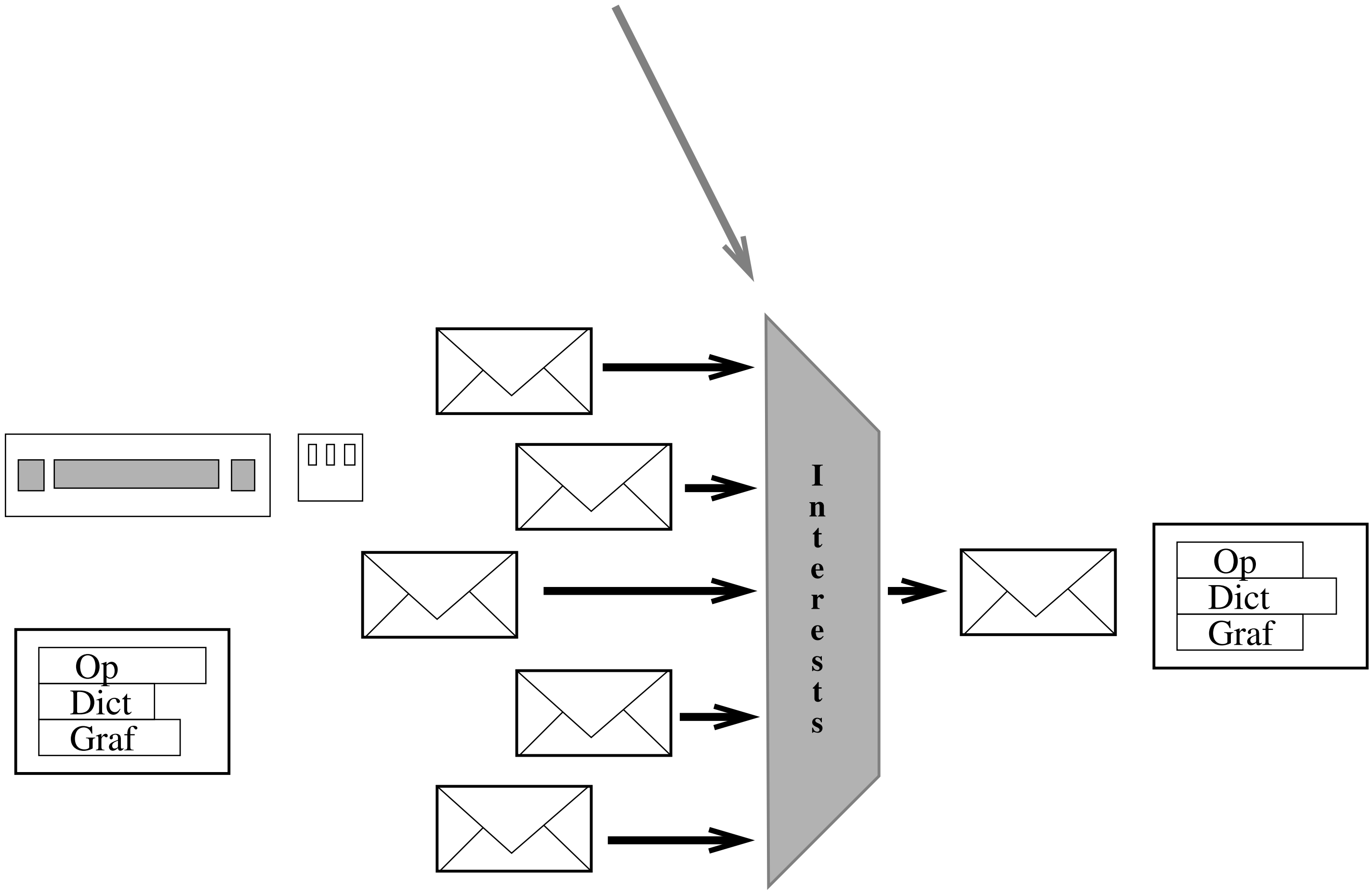


Event Distribution: Interests

- **Interests are Events**
  - o "Templates" for Event Delivery

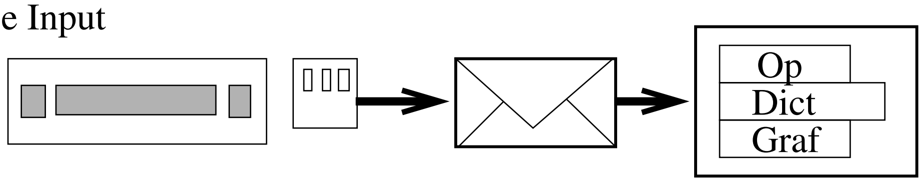
Interest:

```
/Name:     /RightMouseButton
/Action:   /DownTransition
/Canvas:   MyCanvas
```

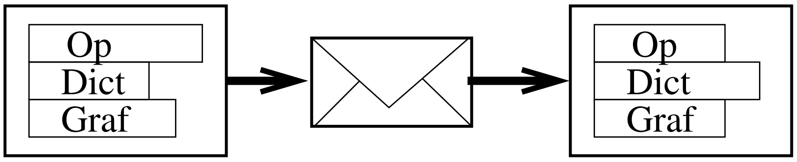


PostScript Language++: Events

- **Communication:**
  - o Device Input



  - o Interprocess "messages"



- **The Event Dictionary:**

o /Name	***/RightMouseButton***
o /Action	***/DownTransition***
o /Canvas	***MyCanvas***
o /TimeStamp	***currenttime 1 60 div add***
o /XLocation /YLocation	***100 100***

Process Examples:

"fork" and "waitprocess"

```
/p {2 2 add} def
p =
     % 4
/pp {p} fork def
pp =
     % process(7663140, runnable)
pp waitprocess =
     % 4
```

"pause"

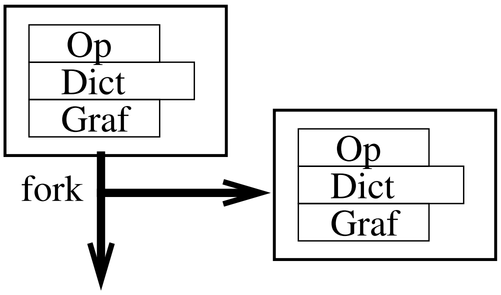
```
/p {2 2 add} fork def
p =
     % process(7762460, runnable)
pause
p =
     % process(7762460, zombie)
p waitprocess =
     % 4
```

Returning multiple results

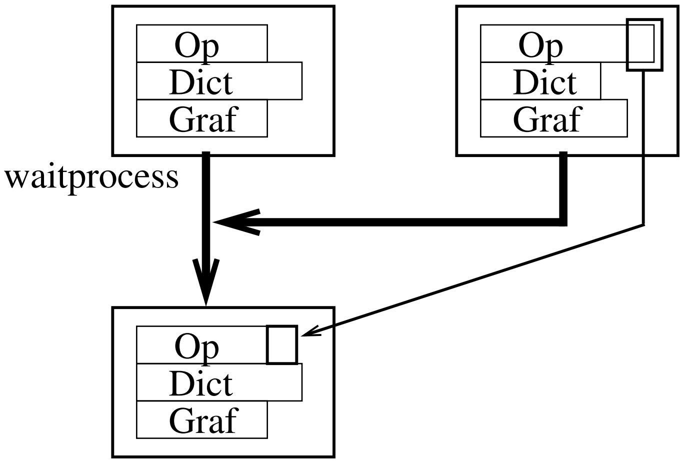
```
/p {1 2 3} fork def
p waitprocess =
     % 3
/p {[1 2 3]} fork def
p waitprocess dup ==
     % [1 2 3]
aload pop pstack
     % 1 2 3
```

Light Weight Process Primitives

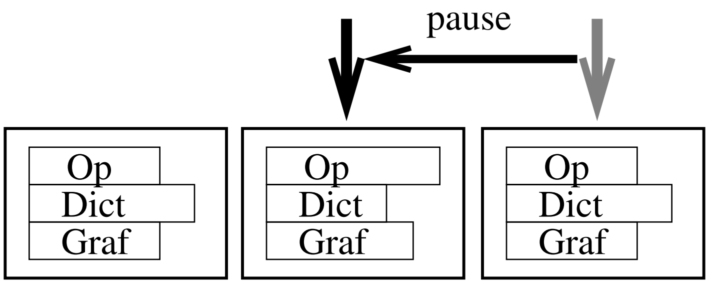
- **fork: procedure => process**
  - o Inherits parent's environment



- **waitprocess: process => value**
  - o "Returns" top of stack, or null.



- **pause: - => -**
  - o Lets other processes run

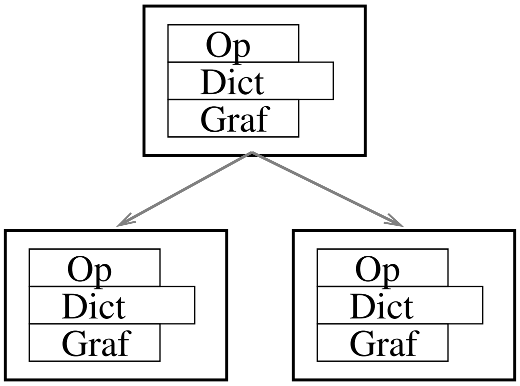


- **Others:**
  - o newprocessgroup, {suspend,continue}process, kill{process,processgroup}, monitors

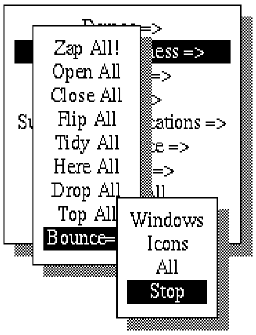
PostScript Language++: Processes

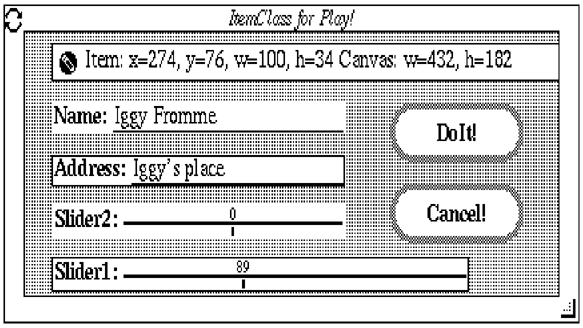
- **Independent thread of execution**
  - o Non-preemptive

- **Inherit parent’s environment:**
  - o Operand Stack
  - o Dict Stack
  - o Gaphics State



- **Light Weight!**
  - o A process is forked each menu invocation
  - o A process is forked each button push





ItemClass for Play!

Item: x=274, y=76, w=100, h=34 Canvas: w=432, h=182

Name: Iggy Fromme

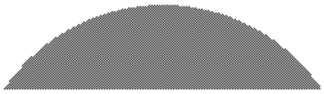
Address: Iggy’s place

Slider2: 0

Slider1: 89

Canvas Example:
Creating a Simple Canvas

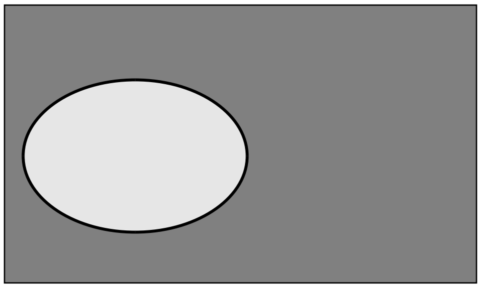
```
gsave
framebuffer setcanvas
100 100 translate 0 0 300 300 ovalpath

/can framebuffer newcanvas def
can reshapecanvas
can /Mapped true put

can setcanvas .5 fillcanvas
grestore
```



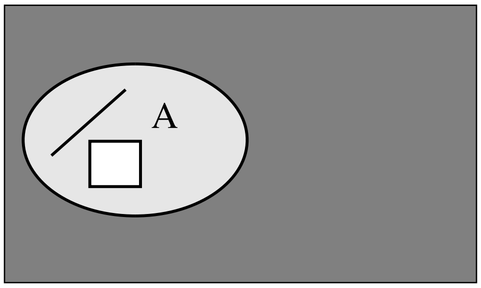
Canvas Primitives

- **newcanvas: parentcanvas => canvas**
  - o Create a new "child" canvas.
  - o Canvases nest to form a "Tree".

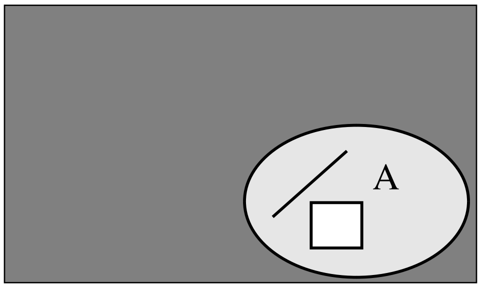
- **reshapecanvas: canvas => -**
  - o [Re]shape the canvas to the current path.
  - o Inherits Parent’s Transform.



- **setcanvas: canvas => -**
  - o Graphics done in the "current" canvas.

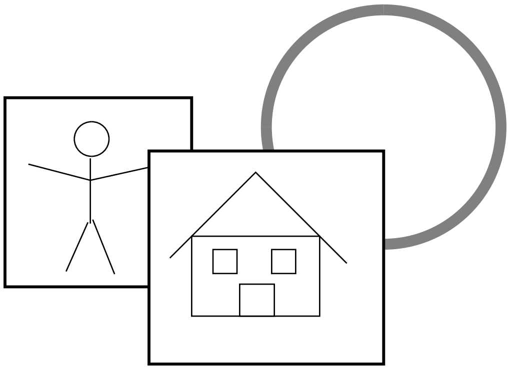


- **movecanvas: x y => -**
  - o Move the canvas

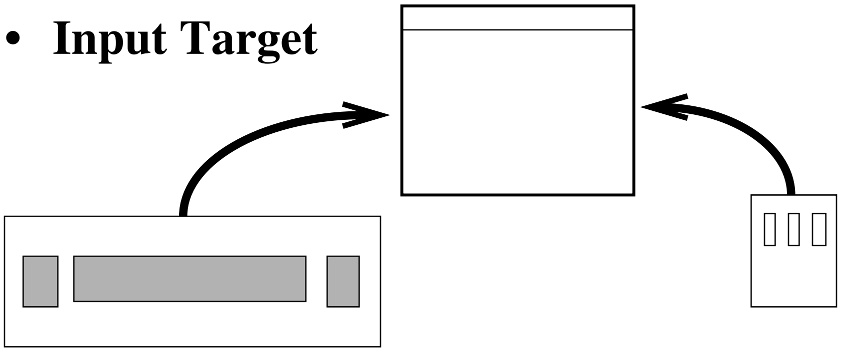


PostScript Language++: Canvases

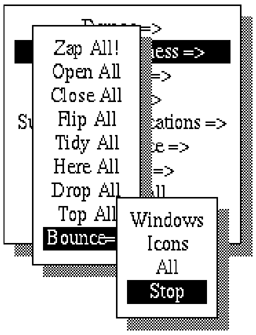
- **Multiple Drawing Surfaces**

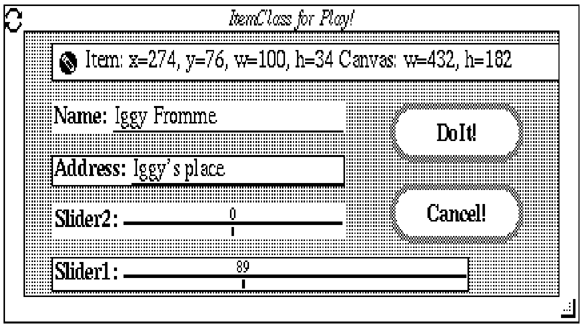


- **Input Target**



- **Light Weight!**
  - o Each Menu is a Canvas
  - o Each Button, Slider, .. is a Canvas





ItemClass for Play!

Item: x=274, y=76, w=100, h=34 Canvas: w=432, h=182

Name: Iggy Fromme

Address: Iggy's place

Slider2: 0

Slider1: 89

NeWS Objects as Dictionaries

- **Key - Value access:**

```
currentcanvas /Mapped get =
false

currentcanvas /Mapped true put
```

- **begin .. end dict stack usage:**

```
createevent dup begin
    /Name   /DoItEvent def
    /Action /Window def
end
==
event(0xf993c, [0,0], name(/DoItEvent), action(/Window))
```

- **forall:**

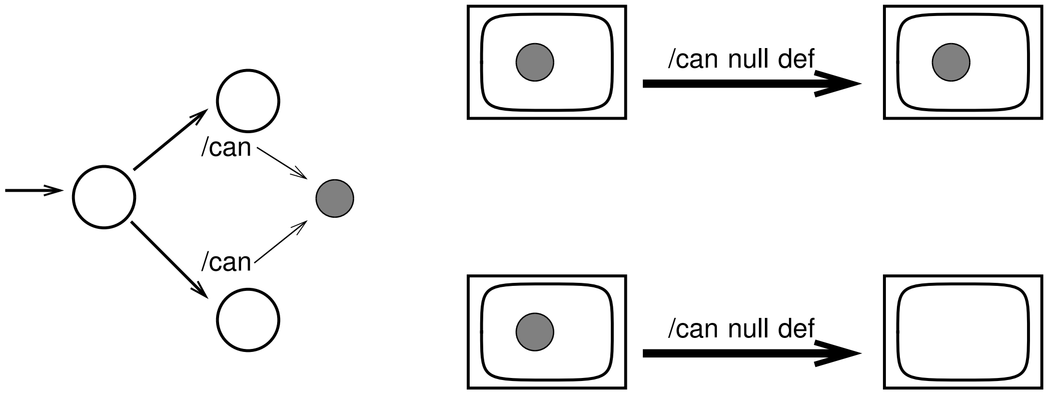
```
currentprocess {pop =} forall
/Interests
/OperandStack
/State
/ErrorCode
/ExecutionStack
/DictionaryStack
/Execee
```

- **..but "type" returns a unique name**

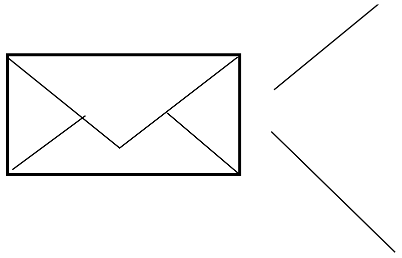
```
currentcanvas type =
canvastype
```

PostScript Language++: Overview (Cont.)

- **Garbage Collection: Shared Heap Memory.**



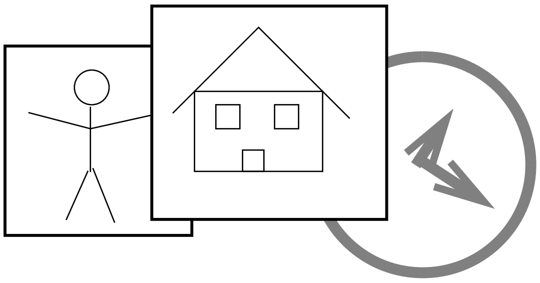
- **Other NeWS Facilities:**
  - o Cursors, Colors, Overlays, Getenv, ...
- **NeWS Objects as Dictionaries:**
  - o Canvases, Processes, Events:



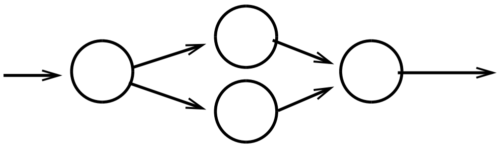
/Name	/RightMouseButton
/Action	/DownTransition
/Canvas	/MyCanvas
.....	

PostScript Language++: Overview

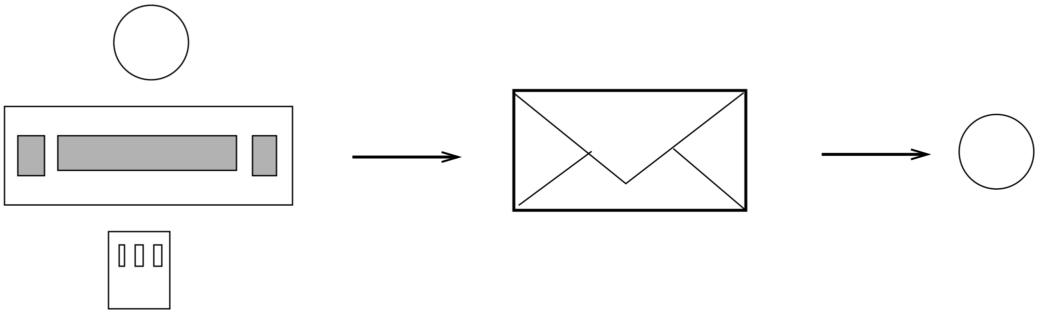
- **Additional NeWS Objects:**
  - o **Canvases**: Multiple Overlapping Drawing Surfaces.



  - o **Light Weight Processes:** Independent threads of control..



  - o **Events:** Input & Inter-process communication.



PostScript Language++:
Additional NeWS Features

- **Overview**
- **Details:**
  - o Canvases
  - o Processes
  - o Events

Our Agenda

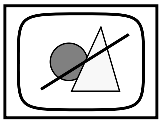
- **PostScript Language++: Additional NeWS Features**
- **NeWS++: Extending NeWS**
- **PSH: Using the NeWS Server**

NeWS Programming: *Programmable* Window Tools

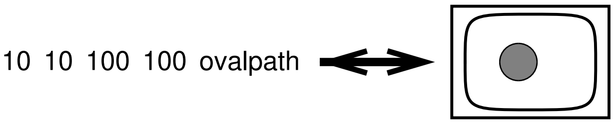
- **The PostScript Language, a new Language in your life!**
  - o PostScript "Red Book"
  - o NeWS Technical Overview
  - o NeWS Manual

```
/setshade { % GrayOrColor => - (set gray or color)
    dup type /colortype eq {setcolor} {setgray} ifelse
} def
% ---------------------------------------------------------
/fillcanvas { % GrayOrColor => - (Fills current canvas w/ GrayOrColor)
    setshade clippath fill
} def
```

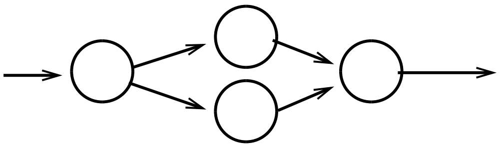
- **..the good NeWS:**
  - o PostScript language Graphics



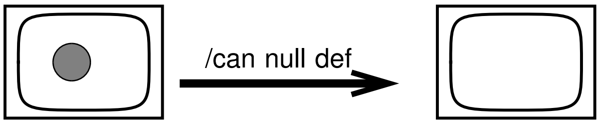
  - o Interpreter



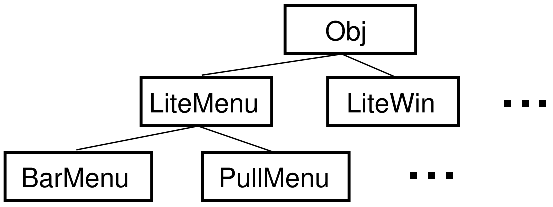
  - o Light Weight Processes



  - o Garbage Collection



  - o SmallTalk-like classes



NeWS Programming

10/14/87