

```

%!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% NeWS Forth HyperTIES
% Target Class Definitions
% Don Hopkins
%
% A Target is a subclass of LiteItem that activates when the cursor
% enters its canvas.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

systemdict begin

/Target Item
dictbegin
% Instance variables
  /Ref null def
  /TID null def
  /FilePos null def
  /Inverted? false def
  /Hilite? true def
  /old-matrix null def
dictend
classbegin % Target
% Class variables

% Class methods

/new { % Ref TID filepos parentcanvas <width height> => instance
  /new super send begin
    /FilePos exch def
    /TID exch def
    /Ref exch def
    ItemCanvas /Transparent true put
    /beye /beye_m ItemCanvas setstandardcursor
  currentdict end
} def

/destroy {
  ItemBegin
  ClientExit
  ItemEnd
} def

/makestartinterests { % - => interests (returns the start interests)
  % Note: we do NOT store the start interests in the item dict.
  % This is to avoid circularity: the interest must have self in it!
  [
    /EnterEvent {/StartItem /Self GetFromCurrentEvent send}
    null ItemCanvas eventmgrinterest
    dup /Self self PutInEventMgrInterest

    ItemButton {/StartItem /Self GetFromCurrentEvent send}
    DownTransition ItemCanvas eventmgrinterest
    dup /Self self PutInEventMgrInterest

    /Damaged {/paint /Self GetFromCurrentEvent send}
  ]
}

```

```

        null ItemCanvas eventmgrinterest
        dup /Self self PutInEventManagerInterest
    ]
} def

/maketrackinterests { % initializes the tracking interests
    /TrackInterests 10 dict dup begin
        /ClientDown load length 0 ne {
            /Down ItemButton {
                ItemBegin
                TargetDict TID known not {
                    ClientExit
                } {
                    ClientDown
                } ifelse
                ItemEnd }
            DownTransition null eventmgrinterest def
        } if
        /ClientUp load length 0 ne {
            /Up ItemButton {
                ItemBegin
                TargetDict TID known not {
                    ClientExit
                } {
                    ClientUp
                } ifelse
                ItemEnd }
            UpTransition null eventmgrinterest def
        } if
        /ClientDrag load length 0 ne {
            /Drag MouseDragged {ItemBegin ClientDrag ItemEnd}
            null null eventmgrinterest def
        } if
        /ClientExit load length 0 ne {
            /Exit /ExitEvent {ItemBegin ClientExit ItemEnd}
            null ItemCanvas eventmgrinterest def
        } if
        /ClientEnter load length 0 ne {
            /Enter /EnterEvent {ItemBegin ClientEnter ItemEnd}
            null ItemCanvas eventmgrinterest def
        } if
    end def
} def

% This is called from the event manager handling all of the
% targets. It should start another event manager tracking this
% item, and inform the application that the user has entered this
% item.
/StartItem { % activates an item
    % (Starting item %\n) [TID] dbgprintf
    ItemBegin
    ItemEventManager null eq {
        /ItemInitialValue ItemValue store
        ClientEnter
        /ItemEventManager TrackInterests forkeventmgr def
    } {
        ClientEnter
    } ifelse

```

```

        CurrentEvent /Action get /DownTransition eq {
            /button-state true def
        } if
        ItemEnd
    } def

/StopItem {
    ItemEventManager
    dup null eq {pop} {
        /ItemEventManager null store
        killprocess
    } ifelse
} def

/unit-scale { % func => -
    /old-matrix matrix currentmatrix def
    ItemWidth ItemHeight scale
    exec
    old-matrix setmatrix
} def

/SelectItemEvent {
    Ref dup length 0 gt {0 get} {pop 0} ifelse 64 gt {
        DefinedTID TID eq
        lasteventtime LastSelectionTime sub DoubleClickTime lt and
        {
            Ref TargetPileID (.pile % .articulate %\n) sprintf callback
        } {
            Ref TargetPileID (.pile % .define %\n) sprintf callback
            PileDict PileID get /DefinedTID TID put
        } ifelse
    } {
        Ref (\n) append callback
    } ifelse
    /LastSelectionTime currenttime store
} def

/DebugItemEvent {
    gsave
    ItemCanvas setcanvas
    { (Item % "%" % % % %: % @ % %\n)
        [TID Ref FilePos aload CurrentEvent begin Name XLocation YLocation end]
        dbgprintf
    } unit-scale
    grestore
} def

/ReportItemEvent {
%   systemdict /sendtoemac known {
%       (\033[T) Ref append (\n) append sendtoemac
%   } {
        SelectItemEvent
        DebugItemEvent
%   } ifelse
} def

/PopupItemRef {
    gsave framebuffer setcanvas

```

```

        { currentcursorlocation 20 add exch 20 add exch [Ref] popmsg
          5 60 div sleep killprocess } fork pop
    grestore
} def

/HiliteCanvas {
    Inverted? not {InvertCanvas} if
} def

/LoliteCanvas {
    Inverted? {InvertCanvas} if
} def

/InvertCanvas {
    /Inverted? Inverted? not def
    0 setgray
    5 setrasteropcode
    clipcanvaspath fill
} def

/ClientDown {
    Hilite? {HiliteCanvas} if
} def

/ClientUp {
    ReportItemEvent
} def

/ClientDrag {
} def

/ClientEnter {
    Hilite? {HiliteCanvas} if
} def

/ClientExit {
    Hilite? {LoliteCanvas} if
    StopItem
} def

/ItemPath {
    0 0 ItemWidth ItemHeight rectpath
} def
classend def % Target

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/LinkTarget Target [/MenuRef]
classbegin

/LinkMenu [
    (Define) {
        MenuRef MenuTargetPileID (.pile % .define %\n) sprintf callback
        PileDict MenuTargetPileID get /DefinedTID MenuTID put
    }
    (Left) {
        MenuRef LeftBrowser (.pile % .articulate %\n) sprintf callback
    }
]

```

```

        (Right) {
            MenuRef RightBrowser (.pile % .articulate %\n) sprintf callback
        }
    ] /new DefaultMenu send
def

LinkMenu /PieInitialAngle 270 put

/maketrackinterests { % initializes the tracking interests
/maketrackinterests super send
TrackInterests begin
    /Menu MenuButton {
        ItemBegin
            TargetDict TID known not {
                ClientExit
            } {
                userdict /MenuRef Ref put
                userdict /MenuTID TID put
                userdict /MenuTargetPileID TargetPileID put
                CurrentEvent /showat LinkMenu send
            } ifelse
        ItemEnd
    } DownTransition null eventmgrinterest def
end
} def

classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/MenuTarget Target [/Menu]
classbegin
    /ItemButton MenuButton def

    /ClientDown {
        % So MenuTargets can share menus!
        userdict /MenuRef Ref put
        userdict /MenuTID TID put
        userdict /MenuTargetPileID TargetPileID put
        Menu null ne {
            CurrentEvent /showat Menu send
        } if
    } def

    /ClientUp {
    } def

    /destroy {
        Menu null ne {
            /popdown Menu send
            /Menu null store
        } if
    } def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/PageTrackerTarget Target [/CoverCanvas]

```

```

classbegin
  /reshape { % x y w h => -
    /reshape super send
    CoverCanvas null eq {
      /CoverCanvas ItemParent newcanvas def
      CoverCanvas /Transparent false put
      CoverCanvas /Retained true put
    } if
    gsave
      ItemCanvas setcanvas clippath
      CoverCanvas reshapecanvas
      CoverCanvas setcanvas
      CoverCanvas /Mapped true put
      /ClientFillColor Win send fillcanvas
      /update-page-trackers PileDict ContentsPileID get /Win get send
    grestore
  } def

  /makestartinterests {
    /makestartinterests super send
    [
      /Page {/ClientPage /Self GetFromCurrentEvent send}
      ContentsPileID
      null
      eventmgrinterest
      dup /Self self PutInEventMgrInterest
    ] append
  } def

  % override this in Target init-function
  /ClientPage { % current-page page-count => -
  } def

  /activate {
    CoverCanvas /Mapped false put
  } def

  /deactivate {
    ItemBegin
      CoverCanvas /Mapped true put
      ClientExit
    ItemEnd
  } def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/PopupTarget LinkTarget
dictbegin
  /PoppedCanvas null def
  /PopX null def
  /PopY null def
  /PopScale null def
  /PopImage null def
  /Popped null def
  /HoleColor .25 .25 .25 rgbcolor def
  /CacheFile null def
dictend

```

```

classbegin
/new { % PopX PopY Scale PopImage Ref TID filepos parent <w h> => inst
/new super send begin
/PopImage exch def
/PopScale exch def
/PopY exch def /PopX exch def
/PoppedCanvas ItemParent newcanvas def
PoppedCanvas /SaveBehind true put
PoppedCanvas /Retained true put
PoppedCanvas /Transparent false put
currentdict end
} def

/ReshapePopup {
ItemCanvas setcanvas
ItemPath
pathbbox % llx lly urx ury
3 -1 roll add 2 div % llx urx cy
3 1 roll add 2 div exch % cx cy
PopX ItemWidth mul PopY ItemHeight mul translate
2 copy translate
PopScale dup scale
neg exch neg exch translate
ItemPath
PoppedCanvas reshapecanvas
} def

/reshape {
/reshape super send
gsave
/CacheFile
(%%%.%x%.cc%)
[ TiesRootDirectory FilePos 0 get FilePos 1 get
ItemWidth ItemHeight
framebuffer /Color get 8 1 ifelse ]
sprintf def

ReshapePopup
PoppedCanvas setcanvas
PoppedCanvas canvastobottom

% Is image cached on disk?
CacheFile findraster dup null ne {
clippath pathbbox points2rect 4 -2 roll
translate scale
imagecanvas
} { % generate image and write cache
pop
HoleColor fillcanvas
newpath ItemPath clip
ItemWidth ItemHeight scale
PopImage imagecanvas
newpath
PoppedCanvas /Mapped true put
CacheFile { writescreen } errored {pop} if
PoppedCanvas /Mapped false put
} ifelse
grestore

```

```

} def

/Popup {
  gsave
  /Popped true def
  PoppedCanvas /Mapped true put
  grestore
} def

/Popdown {
  /Popped false def
  PoppedCanvas /Mapped false put
} def

/ClientEnter {
  Popup
} def

/ClientExit {
  Popdown
  LoliteCanvas
  StopItem
} def

/ClientDown {
  Popdown
  Hilite? {HiliteCanvas} if
} def

/ClientUp {
  LoliteCanvas
  Popup
  ReportItemEvent
} def

classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/ScrollbarTarget SimpleScrollbar [/Ref /TID /FilePos]
classbegin
  /new { % [min max dline dpg dview] init notify ref TID filepos parent => item
    4 -3 roll 7 3 roll
    /new super send begin
      /FilePos exch def /TID exch def /Ref exch def
      ItemCanvas /Transparent true put
      /paint currentdict /send cvx
      currentdict
    end
  } def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/SliderTarget SliderItem [/Ref /TID /FilePos]
classbegin
  /new { % [min max init] loc notify Ref TID filepos parent => item
    4 -3 roll 2 index 8 4 roll

```



```

% Ref TID filepos Ref [min max init] loc notify parent
/new super send begin % Ref TID filepos
/FilePos exch def /TID exch def /Ref exch def
ItemCanvas /Transparent true put
/paint currentdict /send cvx
currentdict
end
} def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/TextEditTarget TextItem [/Ref /TID /FilePos]
classbegin
/new { % init loc notify Ref TID filepos parent => item
4 -3 roll 2 index 8 4 roll
/new super send begin
/FilePos exch def /TID exch def /Ref exch def
ItemCanvas /Transparent true put
/paint currentdict /send cvx
currentdict
end
} def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/TextCanvasTarget Item [/Ref /TID /FilePos /Text /Lines /TextLines]
classbegin
/new { % lines Ref TID filepos parent => item
/new super send begin
/FilePos exch def /TID exch def /Ref exch def /Lines exch def
/paint currentdict /send cvx
ItemCanvas /Transparent false put
currentdict end
} def

/ClientUp { 1 pop } def % so interest gets expressed ...

/destroy {
Text null ne {
/destroy Text send
% Danger: dict with errored's errordict on top dictstack!
/Text null store
} if
} def

/paint {
gsave ItemCanvas setcanvas
/fixdamage Text send
grestore
} def

/TextInit {} def

/reshape {
/reshape super send
Text null eq {

```

```

Lines ItemCanvas /new TextCanvas send
/Text exch def
{ /KeyHitCallback { % key => -
  10 dict begin
    /key exch def
    key 13 eq {
      getcuretpos
      pop /y exch 1 add def
      [( ) ( )] true writeatcaret
    }{
      /s 1 string def
      s 0 key put
      [s] true writeatcaret
    } ifelse
  end
} def
/InsertValueCallback { % string => -
  10 dict begin
    /s exch def
    /newlines 0 def
    s {10 eq {/newlines newlines 1 add def} if} forall
    /a newlines 1 add array def
    0 1 newlines 1 sub {
      /i exch def
      s (\n) search pop
      /pre exch def
      pop
      /s exch def
      a i pre put
    } for
    a newlines s put
    a true writeatcaret
  } def
/LeftMouseUpCallback { % col row => -
  10 dict begin
    /row exch def
    /col exch def
    col row movecaret
  end
} def
oncaret
} Text send
/TextInit load length 0 ne {
  /TextInit load Text send
} if
} if
TextLines null ne {
  TextLines true /writeatcaret Text send
} if
} def
classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/AnimatedTarget LinkTarget
dictbegin
  /CanvasMaker null def
  /Canvases null def

```

```

/CanvasesParent null def
/AnimateEvent null def
/AnimateDelay .02 60 div def
/CanvasNum 0 def
/AnimateLock null def
/AnimateMgr null def
/UnmapWhenStopped? false def
dictend
classbegin
/new { % Ref TID filepos parentcanvas <width height> => instance
/new super send
begin
%   ItemCanvas /Transparent false put %DANGER! cv_get?
/CanvasesParent ItemParent newcanvas def
CanvasesParent /Transparent false put
%   CanvasesParent /SaveBehind true put
CanvasesParent /Retained true put
CanvasesParent canvastotop
ItemCanvas canvastotop
/AnimateLock createmonitor def
currentdict
end
} def

/get-canvas { % - => can
gsave
CanvasesParent newcanvas
dup /Transparent false put
dup /Retained true put
ItemCanvas setcanvas clippath
dup reshapecanvas
pause pause
dup MapToTop
grestore
10 {pause} repeat
} def

/MapToTop { % can => -
dup canvastotop
dup /Mapped true put
{ /CanvasBelow get dup null eq {pop exit} if
dup /Mapped false put } loop
} def

/MakeCanvases {
AnimateLock {
gsave
ItemCanvas setcanvas
ItemPath CanvasesParent reshapecanvas
/Canvases [/CanvasMaker load exec] def
/CanvasNum Canvases length 1 sub def
grestore
} monitor
} def

/DoAnimate {
AnimateLock {
Canvases null ne {

```

```

    ItemBegin
    Canvases CanvasNum get MapToTop
    /CanvasNum
    CanvasNum 1 add
    dup Canvases length ge { pop 0 } if
    def
    AnimateEvent /TimeStamp currenttime AnimateDelay add put
    AnimateEvent recallevnt
    AnimateEvent sendevent
    ItemEnd
  } if
} monitor
} def

/StopAnimating {
  AnimateLock {
    AnimateEvent null ne {
      AnimateEvent recallevnt
    } if
    AnimateMgr null ne {
      AnimateMgr killprocess
      /AnimateMgr null def
    } if
    UnmapWhenStopped? {
      CanvasesParent /Mapped false put
    } if
  } monitor
} def

/StartAnimating {
  StopAnimating
  AnimateLock {
    CanvasesParent /Mapped true put
    Canvases null eq {
      {MakeCanvases} fork pop
    } if
    AnimateEvent null eq {
      /AnimateEvent createevent def
      AnimateEvent begin
        /Name /Animate def
        /Action TID def
      end
    } if
    /AnimateMgr [
      /Animate /DoAnimate TID null eventmgrinterest
    ] forkeventmgr def
    AnimateEvent sendevent
  } monitor
} def

/ClientEnter {
  StartAnimating
} def

/ClientExit {
  StopItem
} def

```

```

/ClientDown {
} def

/StartItem { % activates an item
  AnimateLock {
    ItemBegin
    ItemEventManager null eq {
      /ItemInitialValue ItemValue store
      /ItemEventManager TrackInterests forkeventmgr def
      ClientEnter
    } {
      ClientEnter
    } ifelse
    ItemEnd
  } monitor
} def

/StopItem {
  StopAnimating
  /StopItem super send
} def

classend def

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

/EmacsTarget Item [/Ref /TID /FilePos /Text /Lines /TextLines]
classbegin
  /new { % lines Ref TID filepos parent => item
    /new super send begin
      /FilePos exch def /TID exch def /Ref exch def /Lines exch def
      /paint currentdict /send cvx
      ItemCanvas /Transparent false put
      currentdict end
    } def

  /ClientUp { 1 pop } def % so interest gets expressed ...

  /destroy {
    Text null ne {
      /destroy Text send
      % Danger: dict with errored's errordict on top dictstack!
      /Text null store
    } if
  } def

  /paint {
    gsave ItemCanvas setcanvas
    /fixdamage Text send
    grestore
  } def

  /TextInit {} def

  /reshape {
    /reshape super send
    Text null eq {
      Lines ItemCanvas /new TextCanvas send

```

[illegible]