SetClikLoop Install a routine for custom mouse dragging

#include <TextEdit.h> **TextEdit**

void **SetClikLoop(***clikProc*, *hTE* **)**;
ProcPtr *clikProc* ; address of your custom routine
TEHandle *hTE* ; handle of an edit record

SetClikLoop lets you get control as a user drags the mouse around the screen. Use this to provide "auto-scrolling" (i.e., when the user drags outside of the viewing rectangle).

clikProc is the address of your custom drag-processing routine. Use NIL (0) to revert to the standard handler.

hTE is a handle obtained via **TENew** or **TEStylNew**. It leads to a variable-length TERec structure and identifies the edit record to be affected by this change.

Returns: none

Notes: By default, **TextEdit** does not perform "auto-scrolling". **SetClikLoop** lets you install a routine that is called by **TEClick** and will be called repeatedly while the mouse button is pressed.

Note: For 128K ROMs, you may call **TEAutoView** to partially implement this feature (however, that will not update your scroll bars).

Your click-loop routine receives no parameters and must always return the pascal-version of TRUE. It should be declared as:

```
pascal Boolean myClikLoop(void)
{
    Point   mousePt;

    GetMouse( &mousePt );
    if ( ! PtInRect( mousePt, &(*hTE)->viewRect  ) {
        /*... scroll the text via TEScroll or TEPinScroll ...
        ... update the control value of your scroll bars ...
        ... it is normal to reuse your TrackControl procedure ...
        */
    }
    return( TRUE );   /* ALWAYS return TRUE */
}
```

Install the routine via:

```
SetClikLoop( myClikLoop, hTE );
```

Or, just store the address into the TERec structure:

(*hTE)->wordBreak=myClikLoop;

Note that when your click loop gets control, the clip region will have been set to the size of the viewRect, so before attempting to update scroll bars, you will want to save the clip region (**GetClip**) and set a larger one (**ClipRect**). Then restore the original (**SetClip**) before exiting.