Subject: Re: READ issue
Posted by Vince Hradil on Fri, 23 Feb 2007 17:31:15 GMT
View Forum Message <> Reply to Message

On Feb 23, 11:06 am, l...@lbnc.de wrote:
> Hi there,
>
> I created a game using IDL. The idea is that a horse runs along and as
> fences come up, the user has to press "enter" for the horse to jump
> over it. The program advances the fence up to the point where it is
> directly in front of the horse. Then the READ procedure is use to wait
> for the user input. Once that is entered the horse runs along until
> the next fence comes up.
> However, if I press enter *before* the READ procedure is called, the
> keyboard input is (aparently) kept in the buffer and directly passed
> to the READ routine once is *is* called. This is clearly biasing the
> time of the run. Is there a way to avoid this?
>
> If that all does not make much sense, do the following:
> 1) save the attached source on your computer
> 2) start IDL, make sure you are using the command line version, not
> the IDE
> 3) compile it and run it with the "test" keyword set, i.e:
> Go > .r the_race
> IDL: Compiled module: THE_RACE.
> Go > the_race, /test
> iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
> iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
> iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii
> Go >
> Now adjust the size of the shell window so that you can only see the
> i's and the command input (40x4 characters). Then type "the_race".
> 4) Enjoy!
>
> I have only tested this under UNIX, I do not know if it'll work under
> WINDOWS. It does not work in the IDE because the output window does
> not use a fixed width font.
>
> Also, it would be nices to code this using the "backspace" character,
> i.e. overwriting previously written output. In order to achieve this,
> one needs to "undo" newlines. Is this possible? I tried around a
> little and apparently it is not?
>
> Cheers,
> Lasse
>
> Here's the source:
> pro the_race, test=test, fences=fences

```
>
>  length = 40
>  dist = 20
>  h_offset = 15
>  h_len = 8
>  dummy_input = ''
>  started = 0
>  if not(keyword_set(fences)) then fences = 8
>
>  horse = make_array(3, 4, /string, value=' ')
>
>  horse[0,0] = '      _ '
>  horse[1,0] = '  ===/  '
>  horse[2,0] = '_/___\__'
>
>  horse[0,1] = '      _ '
>  horse[1,1] = '  ===/  '
>  horse[2,1] = '__\_/___'
>
>  horse[0,2] = '      _ '
>  horse[1,2] = '__===/__'
>  horse[2,2] = '_________'
>
>  horse[0,3] = '      _ '
>  horse[1,3] = '  ===/  '
>  horse[2,3] = '__|__|__'
>
>  if keyword_set(test) then begin
>     print, strjoin(replicate('i', length),'')
>     print, strjoin(replicate('i', length),'')
>     print, strjoin(replicate('i', length),'')
>     return
>  endif
>
>  world = make_array(3,length, /string, value=' ')
>  world[2,0:length-2] = '_'
>  world[2,length-1] = '|'
>  act_fence = 1
>
>  top_str = strjoin(reform(world[0,*]), '')
>  mid_str = strjoin(reform(world[1,*]), '')
>  bas_str = strjoin(reform(world[2,*]), '')
>
>  hip = 0
>  dist_count = dist
>
>  while started lt 3 do begin
>
```

```
>     world[0,h_offset:h_offset+h_len-1] = strmid(horse[0,3],
>  indgen(h_len), 1)
>     world[1,h_offset:h_offset+h_len-1] = strmid(horse[1,3],
>  indgen(h_len), 1)
>     world[2,h_offset:h_offset+h_len-1] = strmid(horse[2,3],
>  indgen(h_len), 1)
>
>     if started eq 0 then begin
>        sstr = 'Ready... '
>     endif else if started eq 1 then begin
>        sstr = 'Steady...'
>     endif else if started eq 2 then begin
>        sstr = 'Go!      '
>     endif
>     world[0,2:2+strlen(sstr)-1] = strmid(sstr, indgen(strlen(sstr)),
>  1)
>
>     top_str = strjoin(reform(world[0,*]), '')
>     mid_str = strjoin(reform(world[1,*]), '')
>     bas_str = strjoin(reform(world[2,*]), '')
>
>     print, top_str
>     print, mid_str
>     print, bas_str
>     print, '', format='(a,$)'
>
>     if started eq 2 then break
>     wait, randomu(systime)*3
>
>     started = started+1
>
>  endwhile
>
>  stime = systime(/seconds)
>  atime = systime(/seconds)
>  while act_fence le fences do begin
>
>     ; print time
>     etime = atime-stime
>     etime_str = string(etime, format='(F7.3)')
>     world[0,length-7:length-1] = strmid(etime_str, indgen(7), 1)
>
>     ; find fences
>     bas_bak = strjoin(reform(world[2,*]), '')
>     npos = -1
>     pos = 0
>     pos = strpos(bas_bak, '|')
>     while pos ne -1 do begin
```

```
>         if npos[0] eq -1 then $
>            npos = pos $
>         else $
>            npos = [npos, pos]
>         pos = strpos(bas_bak, '|', pos+1)
>      endwhile
>
>      sinds = where(npos ge h_offset+1 and npos lt h_offset+h_len-2)
>      if sinds[0] ne -1 then hip=2
>      if hip eq 2 and sinds[0] eq -1 then hip=0
>
>      world[0,h_offset:h_offset+h_len-1] = strmid(horse[0,hip],
>   indgen(h_len), 1)
>      world[1,h_offset:h_offset+h_len-1] = strmid(horse[1,hip],
>   indgen(h_len), 1)
>      world[2,h_offset:h_offset+h_len-1] = strmid(horse[2,hip],
>   indgen(h_len), 1)
>
>      ; replace fences
>      if npos[0] ne -1 then begin
>         for i=0, n_elements(npos)-1 do begin
>            world[2, npos] = '|'
>         endfor
>      endif
>
>      top_str = strjoin(reform(world[0,*]), '')
>      mid_str = strjoin(reform(world[1,*]), '')
>      bas_str = strjoin(reform(world[2,*]), '')
>
>      print, top_str
>      print, mid_str
>      print, bas_str
>
>      oinds = where(npos eq h_offset+h_len-2)
>      if oinds[0] ne -1 then begin
>         read, '', dummy_input
>      endif else $
>         print, '', format='(a,$)'
>
>      ; move the world!
>      world[2,*] = shift(world[2,*], -1)
>
>      ; fill up end
>      dist_count = dist_count - 1
>      if dist_count eq 0 then begin
>         world[2,length-1] = '|'
>         act_fence = act_fence + 1
>         dist_count = dist
```

```
>         if act_fence eq fences then begin
>            dist_count = dist-2
>            world[2,length-1] = 'G'
>         endif
>      endif else begin
>         world[2,length-1] = '_'
>      endelse
>      ; change horse
>      if hip eq 0 then hip=1 else hip=0
>      atime = systime(/seconds)
>      wait, .1
>  endwhile
>
>  end
```

Have you looked at using GET_KBRD?
