UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

instituto de astronomía
unam

Instituto de astronomía

Publicaciones Técnicas

“Comunicación Interna” **CI-2009-03**

SBIG_GUI V1.0 INTERFAZ GRÁFICA PARA EL MANEJO DE LAS CÁMARAS SBIG BAJO SISTEMA OPERATIVO LINUX .

E. Colorado, B. García.

Manual de usuario:

"SBIG_GUI V1.0, Interfaz gráfica para el manejo de las cámaras SBIG bajo sistema operativo Linux."

Enrique Colorado Ortiz, Benjamín García
Septiembre 2008

Resume

Este documento describe el funcionamiento de un sencillo programa gráfico para utilizar las cámaras de la compañía Santa Barbara Instrument Group (SBIG) bajo el sistema operativo Linux. El programa permite visualizar las imágenes usando el programa DS9, salvarlas en formato fit y controlar la temperatura de la cámara.

Contenido

1. Introducción

Las cámaras de SBIG son comúnmente utilizadas con propósitos astronómicos, además en el OAN se utilizan en el laboratorio de óptica en Ensenada, en el espectrógrafo Boller & Chiven, en el robodim y se utilizará en el espectrógrafo ESOPO, debido a esto se desarrollo un programa grafico de alto nivel en TCL / TK para el control y adquisición de estas cámaras. Este programa recibe mandos vía red por lo que está listo para el control remoto.

Este programa utiliza el programa servidor de bajo nivel "*Netsbig*" el cual se describe en el documento "Netsbig V1.0 Manejo las cámaras SBIG bajo sistema operativo Linux"

2. Objetivos

El propósito de este programa es el detener una interfaz de usuario sencilla y común para el control de todas las cámaras SBIG bajo el sistema Linux, además de permitir el control y la transferencia de imágenes vía red para que sea posible el uso vía remota del sistema.

3. Descripción de la interfaz de usuario

En esta sección se describen las funciones de la interfaz, la figura 1 muestra el menú principal.

Figura 1.- Menu principal.

Expose: Botón para iniciar una exposición de una imagen completa con el tiempo de exposición dado en “exposure time”. Al finalizar la imagen será desplegada en el programa ds9.

Movie: Botón para iniciar una serie exposiciones consecutivas hasta que sea cancelada por el usuario. Este botón cambia de nombre a “Cancel” al estar en modo movie, presionar para cancelar las exposiciones.

Test: Botón para futuras implementaciones. No se usa.

Exposure Time: Campo para dar el tiempo en segundos de exposición deseado.

Camera Model: Etiqueta que mostrara el modelo de la cámara detectada.

CCD Temp: Etiqueta que mostrara la temperatura actual del CCD de la cámara. Se actualiza en cada exposición o al presionar el botón “Actual CCD Temp” mostrado en la figura 2.

Image File: Nombre de la imagen adquirida. El nombre por defecto es sbig.tcl

Info center: Etiqueta que presenta información relevante del los procesos de la adquisición.

Presionado la pestaña “*Config*” nos mostrara el menú de la figura 2, donde se encuentras las siguientes funciones:

Figura 2.- Menú de configuración.

CCD Set Point: Menú que permite escoger la temperatura a regular por la celda peltier de la cámara.

Actual CCD Temp: Botón para actualizar el valor de la temperatura actual de la cámara.

Work Dir: Directorio de trabajo donde se grabaran las imágenes adquiridas.

Base Name: Nombre base de la cámara utilizado para construir el nombre final, se le agregan números consecutivos de 4 dígitos a los archivos finales.

Automatic Save: Opción que permite grabar cada una de las imágenes adquiridas.

Rotator Angle: Opción que actualmente no se usa.

Done: Botón que regresa al menú principal.

Finalmente, presionado la pestaña "A*bout*" nos mostrara el menú de la figura 3, donde se muestra la versión del programa y el botón de ayuda el cual mostrara este documento.

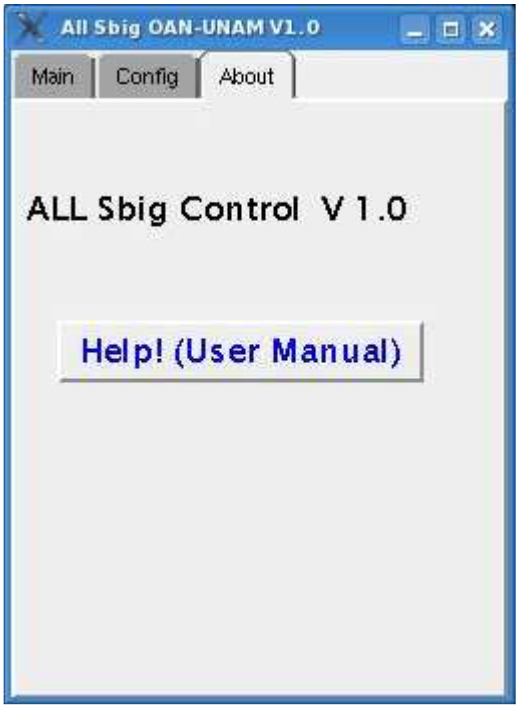

Figura 3.- Menu de ayuda

4. Dependencias de software

Para utilizar este programa es necesario que en su PC tenga instalados los programas "***SAOImage DS9***" y "**xpa**", los cuales pueden ser descargados en:

http://hea-www.harvard.edu/RD/ds9/

También los paquetes **tcl** y **tk** son necesarios, estos generalmente ya están contenidos en las distribuciones de Linux actúales.

También se debe de instalar y ejecutar el programa **Netsbig**, el cual es el servidor de mandos vía red y el cual se comunica directamente con la cámara vía USB.

5. Comunicación con el programa SaoImage DS9.

Para que el programa despliegue las imágenes, deberá estar corriendo el programa ds9, el cual debe de ejecutarse con la siguiente instrucción:

ds9 –title BUSCADOR

6. Instalación

El programa "*sbig_gui.tcl*" puede instalarse y ejecutarse donde se desee, solo verificar que tenga el permiso de archivo ejecutable de lo contrario deberá ejecutarse con el siguiente mando:

wish sbig_gui.tcl

7. Conclusiones

Este programa ha funcionado bastante bien en las pruebas realizadas en los buscadores de los telescopios de 1.5m y 0.84 del OAN ahora podemos ver estrellas de magnitud 16 con 5 segundos de exposición contra las estrellas de magnitud 12 utilizando la cámara analógica pulnix TM-72EX.

La interfaz sencilla permite que el usuario rápido se familiarice con el programa, además debido a que a cada imagen se les resta el *dark* y la calidad astronómica de la imagen, la calidad de las imágenes resultantes es muy superior en comparación de la cámara actual (pulnix TM-72EX).

8. Bibliográfica

- http://sbig.com/sbwhtmls/softpage.htm
  Linux developing Tools.

- Slackware Linux unleashed
  Bao Ha, Tina Nguyen
  Editorial Sams

- Begining Linux Programming.
  Richard Stones y Neil Matthew.
  Wrox Press.

- “Netsbig V1.0 Manejo las cámaras SBIG bajo sistema operativo Linux”
  Enrique Colorado
  Comité Editorial del instituto de de Astronomía de la UNAM.

9. Apéndice A.- listado del programa.

Debido a la longitud del programa solo se muestra las rutinas de las funciones y se omiten las rutinas gráficas.

```
###################################
# USER DEFINED PROCEDURES
#
#############################################################################
## Procedure:  main

proc ::main {argc argv} {
wm protocol .top71 WM_DELETE_WINDOW {salida}
global mensaje



red_enri "init"
#set mensaje "Espera a encontrar la Camara....!"
after 1000

update
catch {exec xpaset -p BUSCADOR regions shape box } pp
if {$pp>3} {
   campanita
   puts $pp
   despliega_mensaje "Ds9 Communications Problems: xpaset" 5000;
   after 2000
   }

red_enri "get_temp"
}
#############################################################################
## Procedure:  done_config

proc ::done_config {} {
global widget
global ccd
global files


#poner set point
red_enri "set_temp $ccd(t)"

puts "work dir $files(imagenes)"
 .top71.tix72 raise page1
```

```
}
##############################################################################
## Procedure:  red_enri

proc ::red_enri {mensaje} {
global widget

global compu_enri socket_enri

   set sock [socket $compu_enri $socket_enri]
   fconfigure $sock -blocking 1 -buffering line
   puts $sock $mensaje
        puts "valor de s $sock"

   fileevent $sock readable [lee_red $sock]

#Cerrando socket
after 10
close $sock

}
##############################################################################
## Procedure:  lee_red

proc ::lee_red {s} {
global widget
global ccd

global mensaje

puts "en lee_red..... $s"
update
gets $s line
puts "Recibi de red: $line"



switch -glob -- $line {

        "Temp*" {
                        scan $line { %s %s} a ccd(rt)
                        puts "t $ccd(rt)"
      }
        "DISPLAY*" {
                        scan $line { %s %s} a ccd(rt)
                        puts "DISPLAY and Temp $ccd(rt)"
                        do_display
      }
        "Camera*" {
                        scan $line { %s %s %s} a myerror ccd(model)
                        puts "Camara $ccd(model)"
                        if {[string compare $myerror "Error"] ==0 } {
```

```
                        campanita
                    campanita
                    despliega_mensaje "Error No encontre la camara" 5000;
                        update
                        campanita
                    after 2000
                        campanita
                    exit

                        }
                    set mensaje "Camara Encontrada e inicializada $ccd(model)"

        }

}


puts "saliendo de lee_red....."
}
##############################################################################
## Procedure:  salida

proc ::salida {} {
global widget

red_enri "close"
after 100
red_enri "salir"
exit
}
##############################################################################
## Procedure:  do_display

proc ::do_display {} {
global widget
global files
global ccd
global mensaje

puts "en do display"
update

if { [ expr !$ccd(autosave)]} {
                    puts "no vamos a grabar la imagen"
                    set files(name) [concat $files(basename).fits]

} else {
        puts "SI vamos a grabar la imagen"

        #verificar si ya existe el archivo
   if {[busca_archivo $files(basename) pp] > 0} {
                set files(name) [file rootname $pp]
                set files(name) [concat $files(name).fits]
```

```
                puts "encontre $files(name) en $files(imagenes)"

        } else {putes "escoje otro nombre"}
}




if {[string compare $files(name) "sbig.fits"] != 0} {

        puts "no estamos usando nombre standar"
        file copy -force [file join $files(imagenes) "sbig.fits"]  [file join $files(imagenes)  $files(name) ]
} else {
puts "aqui toy"
}


  #mandar a ds9
  set aa [file join $files(imagenes)  $files(name)]

  catch {exec xpaset -p BUSCADOR file $aa } pp
  if {$pp>3} {
   campanita
   puts $pp
   despliega_mensaje "Ds9 Communications Problems: xpaset" 4000;
        }
        set mensaje "$files(name) Done!"


        if {[string compare $ccd(imagetype) "Movie"] == 0} {
                puts "seguimos en movie"
                do_expose

        }

        ds9_txt
}
##########################################################################################
## Procedure:  get_temp

proc ::get_temp {} {
global widget

red_enri "get_temp"
}
##########################################################################################
## Procedure:  do_expose

proc ::do_expose {} {
global widget
global ccd
global mensaje
```

```
campanita

#set ccd(imagetype) image
set mensaje "get_full $ccd(time)"
red_enri "get_full $ccd(time)"


}
##############################################################################
## Procedure:  campanita

proc ::campanita {} {
global widget

puts -nonewline stdout "\007"
}
##############################################################################
## Procedure:  despliega_mensaje

proc ::despliega_mensaje {mymensaje delay} {
global widget
global mensaje

puts "despliega_mensaje: "
set mensaje $mymensaje

puts "----- $mensaje"

Window show .mensaje
focus .mensaje
raise .mensaje
update


 after $delay { Window hide .mensaje}
 update
}
##############################################################################
## Procedure:  busca_archivo

proc ::busca_archivo {prefijo archivo} {
global widget
global files
global ccd

        set files(root) "/"
        set files(directorio) $files(imagenes)
        set ccd(imgid) 0

   upvar $archivo kk
```

```
  set i 1
  set pp [file join $files(root) $files(directorio) [format "%s%04do.fits" $prefijo $i]]
         puts "buscando $pp"
   while {$i < 1000} {


          #para todo tipo de imgid
    set pp [file join $files(root) $files(directorio) [format "%s%04do.fits" $prefijo $i]]
          set si1 [file exists $pp]

          #para todo tipo de imgid
    set pp [file join $files(root) $files(directorio) [format "%s%04db.fits" $prefijo $i]]
          set si2 [file exists $pp]

          #para todo tipo de imgid
    set pp [file join $files(root) $files(directorio) [format "%s%04df.fits" $prefijo $i]]
          set si3 [file exists $pp]

          #para todo tipo de imgid
    set pp [file join $files(root) $files(directorio) [format "%s%04da.fits" $prefijo $i]]
          set si4 [file exists $pp]

          #para todo tipo de imgid
    set pp [file join $files(root) $files(directorio) [format "%s%04dd.fits" $prefijo $i]]
          set si5 [file exists $pp]

          set si [expr $si1 | $si2 | $si3 | $si4 | $si5]
          if {$si==0} {

            #nombre archivo final
      set kk1 [file tail $pp]
                  set a [string length $kk1]
                  set kk2 [string range $kk1 0 [expr $a-7]]
                  set kk [concat $kk2$ccd(imgid).fits]
                  puts "creando: $kk1 $a $kk2 $kk"
      return $i
     }

     incr i
   }; #exit while
   puts "no encontramos archivo"
   return 0
    update idletasks
}
##############################################################################
## Procedure:  do_movie

proc ::do_movie {} {
global widget

global ccd
global boton
```

```
	switch $boton {

	Movie {
			puts "comenzamos movie"
			#.top71.tix72.nbframe.page1.but74 config -state disabled
			.top71.tix72.nbframe.page1.but74 config -background RED
			set boton Cancel
			set ccd(imagetype) Movie
			do_expose
		}

	Cancel {
			puts "Cancelamos movie"

			.top71.tix72.nbframe.page1.but74 config -background GREEN
			set boton Movie
			set ccd(imagetype) image
			after 1000
		}
	}



}
##############################################################################
proc ::ds9_txt {} {


#color
set a "xpaset -p BUSCADOR regions color green"
catch {exec sh -c $a} pp

set a "echo \"image; text 750 20 # text={North}\" | xpaset BUSCADOR regions"

#code
puts $a
catch {exec sh -c $a} pp
puts $pp

set a "echo \"image; text 750 1000 # text={South}\" | xpaset BUSCADOR regions"
catch {exec sh -c $a} pp

#84
#set a "echo \"image; text 40 500 # text={West}\" | xpaset BUSCADOR regions"
set a "echo \"image; text 40 500 # text={East}\" | xpaset BUSCADOR regions"
catch {exec sh -c $a} pp

#84
#set a "echo \"image; text 1480 500 # text={East}\" | xpaset BUSCADOR regions"
set a "echo \"image; text 1480 500 # text={West}\" | xpaset BUSCADOR regions"
catch {exec sh -c $a} pp
```

```
#regiol del ccd marconi
#color red
set a "xpaset -p BUSCADOR regions color red"
catch {exec sh -c $a} pp
set a "echo \"box 800 500 800 400 0\" | xpaset BUSCADOR regions"
catch {exec sh -c $a} pp

set a "xpaset -p BUSCADOR regions color yellow"
catch {exec sh -c $a} pp
#poner centro del ccd
set a "echo \" x point 750 500\" | xpaset BUSCADOR regions"
catch {exec sh -c $a} pp

}

##############################################################################
## Initialization Procedure:  init

proc ::init {argc argv} {
global ccd
global mensaje
global files

set mensaje "info center"

set files(imagenes) "/imagenes/"
set files(name) sbig.fits
set files(basename) sbig

set ccd(time) 0.2
set ccd(rt) "?"
set ccd(t) 0
set ccd(autosave) 0
set ccd(imagetype) image

global socket_enri;
        set socket_enri 9700
global compu_enri;
        set compu_enri {localhost}

global boton
        set boton Movie

}

init $argc $argv
main $argc $argv
```

Comité Editorial de Publicaciones Técnicas
Instituto de Astronomía
UNAM

M.C. Urania Ceseña
Dr. Carlos Chavarria
M.C. Francisco Murillo

Observatorio Astronómico Nacional
Km. 103 Carretera Tijuana-Ensenada
22860 Ensenada B.C.
editorial@astrosen.unam.mx