ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

UNIVERSIDAD DE CANTABRIA

Proyecto Fin de Carrera

Actualización de Software: GUIspec
(Software Update: GUIspec)

Para acceder al Título de

INGENIERO TÉCNICO DE TELECOMUNICACIÓN

Autor: Ángel Vidal Rodríguez Martínez

Septiembre - 2012

E.T.S. DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACION

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN

CALIFICACIÓN DEL PROYECTO FIN DE CARRERA

Realizado por: Ángel Vidal Rodríguez Martínez
Director del PFC: Francisco J. Carrera
Título: "Actualización de Software: GUIspec"
Title: "Software Update: GUIspec"

Presentado a examen el día:

Para acceder al Título de

INGENIERO TÉCNICO DE TELECOMUNICACIÓN, ESPECIALIDAD EN SISTEMAS ELECTRÓNICOS

Composición del Tribunal:
Presidente (Apellidos, Nombre):
Secretario (Apellidos, Nombre):
Vocal (Apellidos, Nombre):

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del PFC
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Proyecto Fin de Carrera Nº
(a asignar por Secretaría)

Agradecimientos

Gracias a todas las personas que han estado a mi lado desde que empecé el proyecto en el mes de Enero, sin ellos no hubiese sido posible.

A mis padres por su paciencia, aunque de vez en cuando les podía la impaciencia.
A mis hermanas, por el ruido que hacen cada vez que vienen a casa con sus hijos, ahora soy capaz de concentrarme en cualquier parte.
A mis cuñados, por sacar a mis hermanas y mis sobrinos de casa y dejarme momentos de tranquilidad.
A mis amigos, porque aunque la gran mayoría de ellos no entiende lo que estoy haciendo, me siguen apoyando.
Al Dr. López-Rasines por su insistencia y a la Dra. Ortega por sus cenas.
Y por último a Julia, mi novia, por su apoyo incondicional a lo que hago y facilitarme la vida.

Muchas gracias a todos.

Índice

Palabras Clave: GUIspec, método, unidades, error, interfaz.

SOFTWARE UPDATE: GUISPEC

Capítulo 0: Introducción

0.1. Contexto de trabajo

GUIspec es un software de análisis de espectros astronómicos creado por el Dr. Francisco J. Carrera del Instituto de Física de Cantabria en el año 2002 en lenguaje Java y fue mantenido por él mismo hasta el 2004.

Desde el 2004 el programa ha estado sin mantenimiento y en el 2012 nos encontramos con una aplicación funcional que cumple con su propósito pero no se ha adaptado a las nuevas revisiones de Java.

El software GUIspec también necesita algunas mejoras para ofrecer una mayor versatilidad y ser competitivo frente a otras aplicaciones similares que se han ido desarrollando por otras universidades recientemente, lo que hacen que GUIspec se quede obsoleto y necesite más funciones de cara al tratamiento y representación de espectros.

La última premisa es que el Dr. Francisco J. Carrera continuará el mantenimiento de GUIspec después de esta actualización por lo que la estructura del funcionamiento de la aplicación debe permanecer intacta para facilitar dicha tarea en la medida de lo posible.

0.2. Motivación

La principal motivación es el hecho de realizar como proyecto de fin de carrera un trabajo que fuese útil una vez terminado, que no fuese un mero trámite para conseguir el título.

En las asignaturas de programación que cursé a lo largo de la carrera eché en falta hacer algo "de verdad" algo que luego fuese útil para algún departamento de la Universidad de Cantabria, aunque entiendo que los conocimientos que tenemos al inicio de los laboratorios y el tiempo que hay de prácticas no facilita nada esta posibilidad y al final quedaría a medio-hacer en el mejor de los casos.

Hacer un proyecto de estas características me permitía completar esa parte de la formación haciendo una aplicación para un departamento y además complementar los conocimientos de programación utilizando un lenguaje más actual que "C" en el que realizamos las prácticas de programación a lo largo de la carrera universitaria.

Otro aspecto que me parece interesante es el reto que supone realizar una aplicación para un área del que desconoces prácticamente todo, ya que además de programar tienes que documentarte y esto es lo que sucede si trabajas como programador, los clientes pueden llegar desde muchas áreas diferentes y tienes que poder satisfacer sus necesidades.

0.3. Objetivos

- Actualización de los métodos obsoletos utilizados en el código fuente de la aplicación GUIspec.

- Actualización de la interfaz gráfica de GUIspec.

- Mejoras en las funcionalidades de GUIspec:
    - Lectura y escritura de formatos estándar: FITS, VOTables e IRAF.
    - Posibilidad de representar los espectros en diferentes magnitudes físicas y unidades.
    - Utilización de los errores en los espectros para calcular ajustes lineales.

0.4. Estructura del documento

El capítulo cero es un pequeño resumen de las herramientas utilizadas para llevar a cabo este proyecto.

El primer capítulo abarca el estudio del funcionamiento interno de GUIspec, explicando sus clases principales y su organización.

El segundo capítulo se centra en la actualización del código fuente obsoleto y la actualización de la interfaz gráfica.

El tercer capítulo trata la implementación de los métodos de lectura y escritura de formatos de archivo estándar, comentando la estructura de cada uno de estos archivos y sus particularidades.

El cuarto capítulo explica la implementación del cambio de magnitudes en los ejes de representación.

El quinto capítulo trata el ajuste lineal en los espectros y la inclusión de los errores en las medidas para calcular dichos ajustes.

Capitulo 7: últimas consideraciones.

Capítulo 1:

Herramientas Utilizadas en el Proyecto.

Para llevar a cabo este proyecto las herramientas utilizadas han sido:

- Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) NetBeans.
  Hay varios IDE's y todos cumplen su función perfectamente, así que he utilizado NetBeans porque ya estaba familiarizado con su interfaz por un trabajo anterior.

- Librerías. Además de las librerías internas de java he utilizado dos librerías externas para la entrada y salida de ficheros estándar, estas son:
  - STIL: Es un conjunto de librerías muy completo y permite leer o escribir en diferentes formatos, entre ellos FITS (utilizando la librería nom.tam.fits), pero en este caso sólo la he usado para el caso de las VOTables, ya que utiliza un paso intermedio que nos ahorramos utilizando directamente la siguiente librería.
    Librería STIL para JAVA y documentación: http://www.star.bris.ac.uk/~mbt/stil/

  - Nom.tam.fits: Incluye métodos para realizar la entrada y salida de archivos FITS de una manera rápida y eficiente. Se ha utilizado esta librería en vez de STIL para los casos de FITS y IRAF porque con nom.tam.fits trabajas directamente sobre los datos y en la librería STIL trabajas sobre una interfaz "StarTable", y siempre es más preciso trabajar sobre los datos que quieres manejar para no perder ningún paso intermedio.
    Libería nom.tam.fits para Java y documentación: http://heasarc.gsfc.nasa.gov/docs/heasarc/fits/java/v1.0/FitsLib.pdf

- Para la edición y visualización de archivos FITS y IRAF se ha usado el software fv 5.2.

Capítulo 2:

Funcionamiento de GUIspec.

2.1. Estructura interna de GUIspec

La estructura interna de GUIspec se divide en cuatro paquetes:

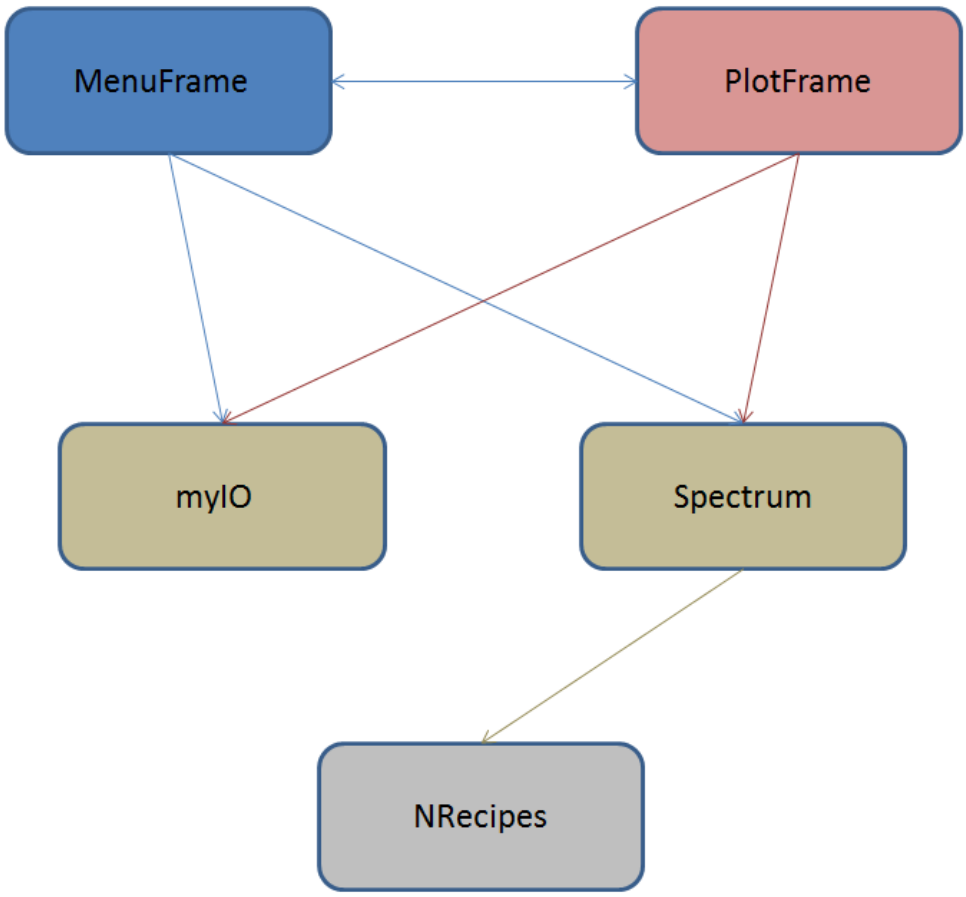
NRecipes:
Contiene los objetos necesarios para realizar operaciones matemáticas complejas.

myIO:
Apertura de archivos y búsqueda de recursos dentro del .jar.

Spectrum:
En este paquete nos encontramos todo lo referente a la estructura de los espectros así como los modelos de ajuste.

Guispec:
Paquete principal del GUIspec, donde tenemos la interfaz gráfica a través de las clases MenuFrame y PlotFrame que controlan el flujo de trabajo de GUIspec

En el siguiente diagrama vemos como se comunican las dos clases principales con los demás paquetes:



2.2. Flujo de trabajo

El flujo de trabajo tiene dos puntos de partida pero con el mismo trayecto:

- Desde MenuFrame:

  Desde los menús de MenuFrame solicitamos una acción y esta podrá tomar dos caminos:

  - Petición a PlotFrame:
    PlotFrame realizará la acción solicitada, comunicándose con los objetos de los paquetes myIO y Spectrum, representará gráficamente dicha acción y devolverá el resultado a MenuFrame que los representará en su interfaz si fuese necesario.
  - Realización desde MenuFrame:
    MenuFrame se encargará de llevar a cabo la tarea y al finalizar se comunicará con PlotFrame para que este objeto realice la representación gráfica.

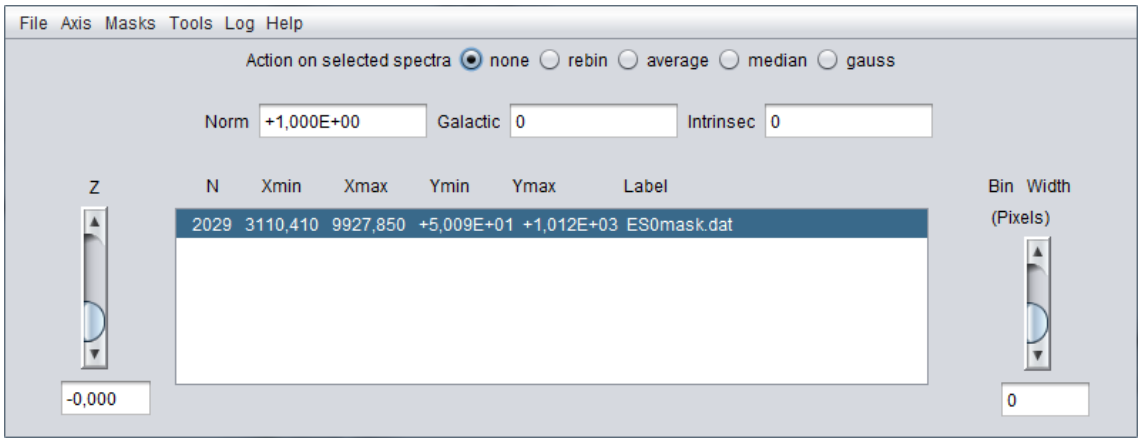
- Desde PlotFrame:

  Utilizando las teclas de función de PlotFrame solicitamos una acción que podrá ser resuelta de las dos siguientes formas:

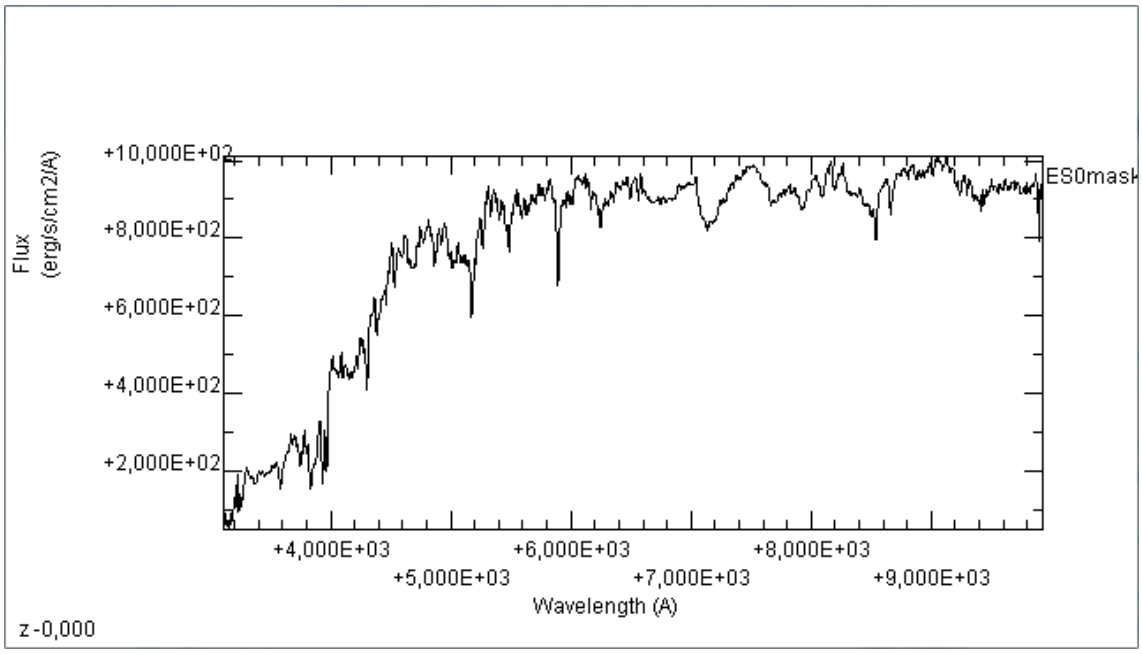
  - Petición a MenuFrame:
    MenuFrame realizará la acción solicitada, representará en su interfaz los datos que fuesen necesarios y devolverá los resultados a PlotFrame para su representación gráfica.
  - Realización desde PlotFrame:
    PlotFrame se encargará de llevar a cabo la tarea y al finalizar se comunicará con MenuFrame para que represente los datos obtenidos en su interfaz si fuese necesario.

2.3. Clases Principales

MenuFrame contiene la interfaz principal desde la que tenemos acceso a los menús básicos de funcionamiento:



PlotFrame conforma la gráfica donde se representan los espectros:



La clase Spectrum contiene la información del espectro, es decir, valores del eje X, del eje Y y los errores si los hubiera. Así como lós métodos necesarios para manejar los espectros.

Los objetos de tipo Spectrum, para que GUIspec funcione de manera coherente, han de tener los valores del espectro en Angtroms para la longitud de onda y en $\frac{erg}{s*cm^2*A}$ para el flujo.

2.4. Conclusiones

Aunque las dos clases más importantes de GUIspec son PlotFrame y MenuFrame, todo gira entorno a la clase Spectrum (lo que es lógico ya que es un software que trabaja con espectros) y los cambios que se hagan en esta clase afectan al funcionamiento de todo el programa, así que para evitar grandes cambios para las futuras actualizaciones de la aplicación, en todas las mejoras que se han incluido se ha buscado siempre dejar la clase Spectrum intacta, si bien, cuando no ha sido posible, se han implementado nuevos métodos dentro de la clase, evitando en todo momento modificar los métodos existentes y los constructores, lo que implicaría cambios en prácticamente todas las demás y su posterior mantenimiento sería más complicado, ya que habría que estudiarse el código completo y de esta forma sólo tiene que repasar los métodos nuevos.

Capitulo 3:

Actualización del Código Fuente Obsoleto e Interfaz Gráfica.

3.1. Avisos de código obsoleto

Una vez comprendido el código y visto su funcionamiento el siguiente paso fue ver los errores que contenía el código. En la primera compilación se nos mostró una lista de 32 avisos de código obsoleto, que analizando nos encontramos con dos casos:

El primero y más trivial es el caso de "show" y "hide" que han sido sustituidos por "setVisible(boolean)", true para "show" y false para "hide".

El segundo y un poco más complejo es el caso del manejo de eventos que desde la última revisión de GUIspec había cambiado sustancialmente. Tanto para los eventos generados por el ratón, el teclado o por el propio programa se deben implementar clases de escucha ("Listeners") para cada evento y ejecutar una acción para cada uno de ellos.

Esta actualziación de Java se introdujo en la versión del JDK 1.1 para hacer las interfaces más eficientes, ya que en las versiones previas para manejar los eventos había sólo una clase de escucha que comprobaba la ID del componente que generaba el evento y actuaba en consecuencia. El problema es que el comprobar la "ID" consiste en comparar cadenas y esto reduce la eficiencia de las aplicaciones.

Por ejemplo, en el caso de GUIspec, el manejo del evento para abrir un archivo tenía que pasar por 11 sentencias IF, comprobando quién envía el evento y cual es su ID. Con la nueva implementación de la gestión de eventos, tenemos el siguiente código:

```
        open.setText("Open");
        open.addActionListener(new
java.awt.event.ActionListener() {
            public void
actionPerformed(java.awt.event.ActionEvent evt) {
                openActionPerformed(evt);
            }
        });
```

Este código nos permite que al pulsar la opción "Open" en la interfaz de GUIspec directamente se lance la acción que necesitamos evitando las 11 comparaciones.

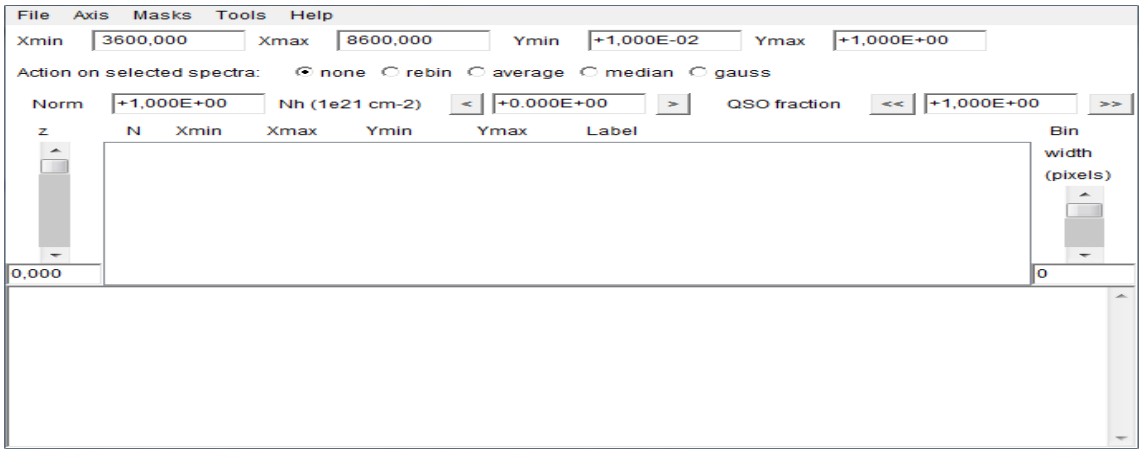
3.2. Interfaz Gráfica

El siguiente paso fue la recodificación de la interfaz gráfica, que resultó en una reconstrucción prácticamente completa, ya que se ha aprovechado la actualización para cambiar el aspecto de MenuFrame y utilizar la librería de java SWING en vez de AWT que es la que usaba.

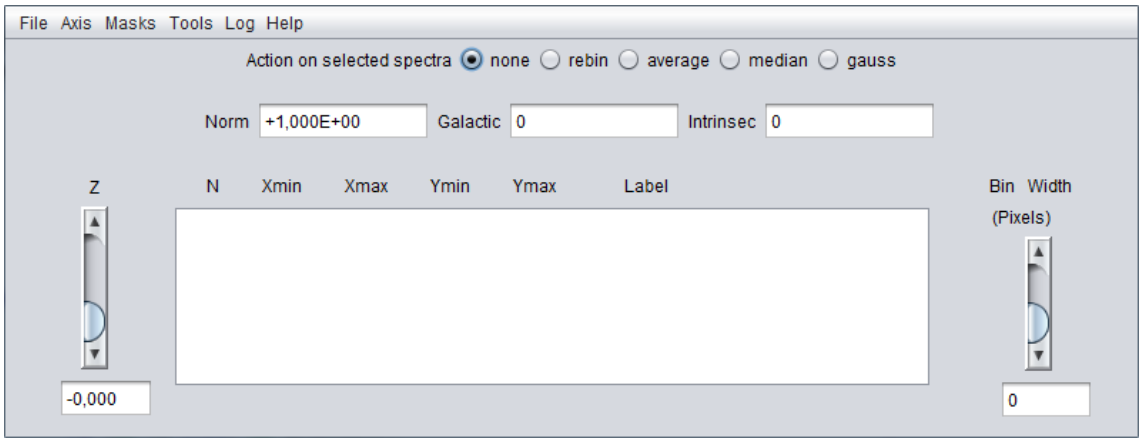
AWT a pesar de ser una libería de Java, que es multiplataforma, depende del sistema operativo en el que se ejecute para crear la interfaz gráfica es decir cuando se utilizan componentes de la librería AWT, el código utilizado para obtener los componentes visuales es generados por el sistema operativo, lo que provoca que según en qué sistema operativo ejecutes la aplicación, esta tenga un aspecto diferente.

SWING sólo necesita que el sistema operativo cree una ventana, del resto se encarga la máquina virtual de Java, con lo que es menos dependiente del sistema operativo y por tanto cumple mejor con la premisa de Java de ser un lenaguaje multiplataforma.

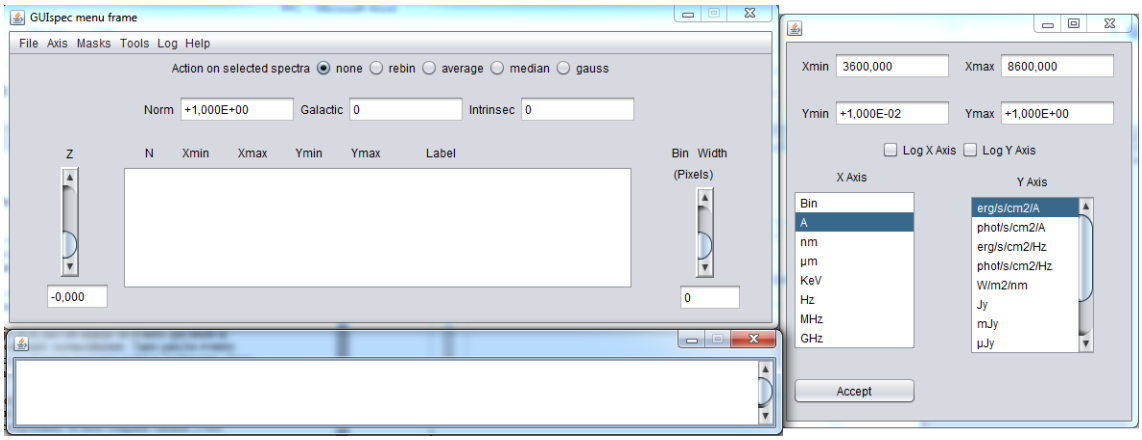
Este es el aspecto que MenuFrame tenía antes de la actualización:



Y este el aspecto que tiene ahora:



Los cambios en MenuFrame han sido siempre buscando la funcionalidad y la reducción de tamaño. La funcionalidad para que su uso sea lo más intuitivo posible y la reducción de tamaño para poder ampliar al máximo PlotFrame y poder ver las gráficas representadas con mayor claridad. Para ello se han utilizado ventanas adicionales de manera que solo se muestren cuando lo requiera el usuario, el resultado es el siguiente:



Cómo se puede observar en las imágenes, las barras de desplazamiento en la versión sin actualizar funcionaban al revés, es decir, pulsando el botón hacia abajo el contador incrementaba, ahora las barras de desplazamiento incrementan su valor al desplazarse hacia arriba.

En cuanto a PlotFrame los cambios que se han realizado han sido internos, tanto para corregir código obsoleto como para añadir funcionalidades nuevas, pero su aspecto sigue siendo el mismo al no utilizar componentes de la librería AWT.

3.3 Paquete JAR

El último cambio que se realizó en esta fase del desarrollo fue el empaquetamiento de GUIspec en un archivo .JAR.

Antes de empaquetar los componentes de GUIspec en un JAR, hubo que recodificar el acceso a ficheros que hace GUIspec, ya que antes de la actualización, para la aplicación todo eran archivos, ahora hay que diferenciar entre ficheros y recursos.

- Ficheros: Todo lo que está fuera del JAR, es decir, los espectros.
- Recursos: Todo lo que está dentro del JAR.

En la versión previa de GUIspec uno de los parámetros al ejecutarlo era la ruta donde se encontraban los archivos internos de la aplicación, ya que utilizaba rutas absolutas para su funcionamiento, ahora no es necesario, accede a los recursos mediante rutas relativas, lo que evita una posible fuente de errores y además permite que el software sea más facilmente portable.

Capítulo 4: Lectura y Escritura de Formatos Estándar.

4.1. Introducción

Antes de realizar la actualización de GUIspec, esta aplicación solo era capaz de leer ficheros de texto ASCII.

Los ficheros de texto pueden tener dos, tres o cuatro columnas:

- Dos columnas:
  La primera es el eje X y la segunda el eje Y.
- Tres columnas:
  La primera es el eje X, la segunda el eje Y y la tercera el error en el eje Y.
- Cuatro columnas:
  La primera es el eje X, la segunda el error en el eje X, la tercera el eje Y y la cuarta el error en el eje Y.

Al solo poder leer ficheros en este formato, el uso de GUIspec estaba bastante limitado, ya que hay otros tres formatos estándar: FITS, IRAF y VOTable.

El principal problema que nos encontramos a la hora de leer nuevos formatos es la clase Spectrum. Como he comentado anteriormente, para que todo funcione correctamente, los valores han de estar en Angstroms (eje X) y en $\frac{erg}{s*cm^2*A}$ (eje Y), lo que nos plantea un problema con nuevos ficheros de entrada, ya que estos ficheros pueden tener otras unidades. Para solucionarlo se comprueba que las unidades son las nativas de la clase Spectrum y si no lo son, hacer la transformación necesaria para mantener la coherencia en los espectros. Es decir, hace la operación necesaria para transformar los valores del espectro de la unidad de entrada a la nativa de GUIspec y la almacena con estos valores en un objeto Spectrum.

4.2. FITS

Para implementar la lectura y escritura del formato FITS se ha utilizado la librería "nom.tam.fits", que nos da las herramientas necesarias para poder manejar las tablas de datos y manejar las cabeceras.

Su estructura es bastante compleja ya que puede servir tanto para almacenar imágenes como tablas de datos (binarias o ASCII) y no es necesario profundizar en ella ya que no es el propósito de este proyecto, con lo que mostraré a continuación a grandes rasgos su organización y cómo almacena los datos de representación de espectros y un ejemplo de su contenido utilizando el software FV.

La estructura de los archivos FITS se organiza básicamente de forma que cada cabecera tenga sus datos asociados, así tienen una cabecera principal con sus datos y en el caso de que las necesiten tienen extensiones, es decir, cabeceras adicionales con sus datos.

La cabecera principal, por norma, es para albergar imágenes, por lo que si el archivo FITS contiene solo un espectro (o varios) y ninguna imagen, la cabecera principal estará vacía y en la siguiente cabecera comenzarán los espectros.

Los datos necesarios para representar los espectros están por un lado en la cabecera donde nos encontramos entre otras las siguientes etiquetas:
Número de puntos: NAXIS2
Nombres de las columnas: TTYPE1 y TTYPE2.
Unidades de los ejes: TUNIT1 y TUNIT2.
Tipo de tabla de datos (ASCII o binaria): XTENSION

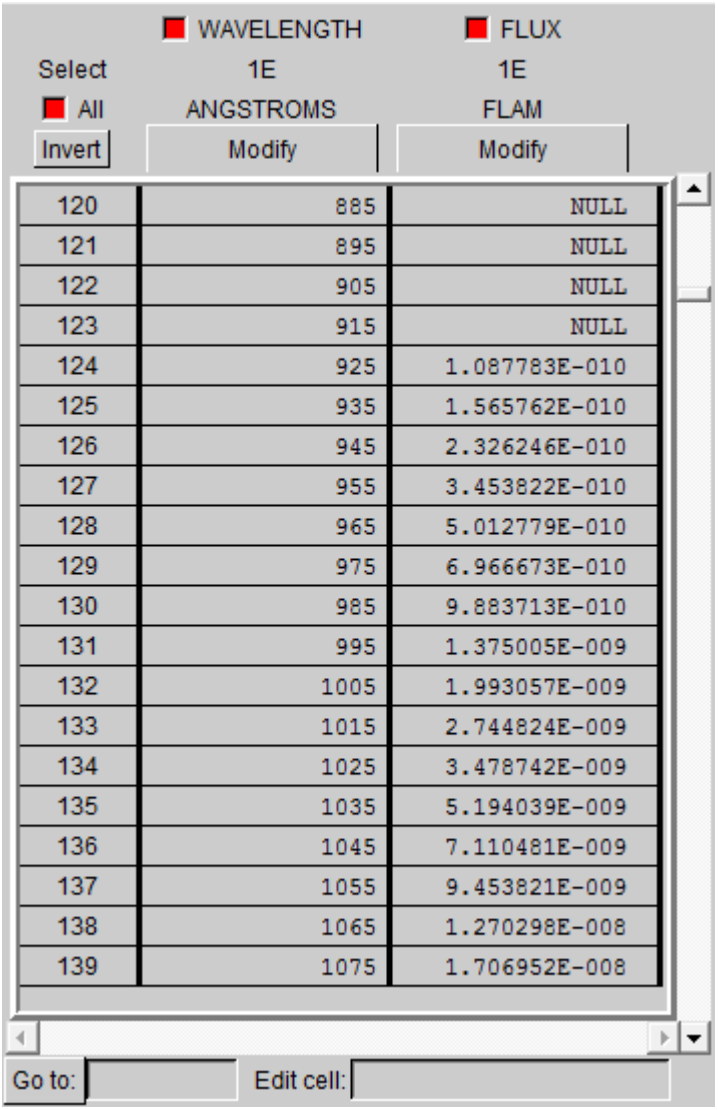
```
XTENSION= 'BINTABLE'           / binary table extension
BITPIX  =                    8 / 8-bit bytes
NAXIS   =                    2 / 2-dimensional binary table
NAXIS1  =                    8 / width of table in bytes
NAXIS2  =                 1221
PCOUNT  =                    0 / size of special data area
GCOUNT  =                    1 / one data group (required keyword)
TFIELDS =                    2
TTYPE1  = 'WAVELENGTH'         / label for field   1
TFORM1  = '1E      '           / data format of field: 4-byte REAL
TUNIT1  = 'ANGSTROMS'          / physical unit of field
TTYPE2  = 'FLUX    '           / label for field   2
TFORM2  = '1E      '           / data format of field: 4-byte REAL
TUNIT2  = 'FLAM    '           / physical unit of field
TDISP1  = 'G16.8   '           / display format
TDISP2  = 'G16.8   '           / display format
END
```

Index	Extension	Type	Dimension	View
0	NoName	Image	0	Header Image Table
1	NoName	Binary	2 cols X 1221 rows	Header Hist Plot All Select

Como se ve en este ejemplo en la imagen superior, podemos comprobar que en este caso se trata de una tabla binaria, con un espectro formado por 1221 puntos, la columna número uno contiene la longitud de onda y su unidad es “ANGSTROMS” y la columna número 2 el flujo y la unidad es “FLAM”.

En la imagen inferior vemos que el cabecero principal (index 0) es una imagen con dimensión 0 para mantener el estándar de FITS.

Por otro lado, de los datos asociados a dicha cabecera forman una tabla que representa los valores del espectro, como vemos en la siguiente imagen:



Select	WAVELENGTH	FLUX
	1E	1E
All	ANGSTROMS	FLAM
Invert	Modify	Modify
120	885	NULL
121	895	NULL
122	905	NULL
123	915	NULL
124	925	1.087783E-010
125	935	1.565762E-010
126	945	2.326246E-010
127	955	3.453822E-010
128	965	5.012779E-010
129	975	6.966673E-010
130	985	9.883713E-010
131	995	1.375005E-009
132	1005	1.993057E-009
133	1015	2.744824E-009
134	1025	3.478742E-009
135	1035	5.194039E-009
136	1045	7.110481E-009
137	1055	9.453821E-009
138	1065	1.270298E-008
139	1075	1.706952E-008

Go to: Edit cell:

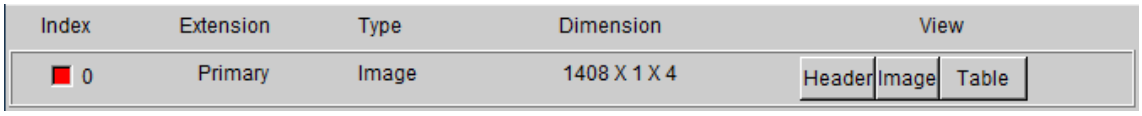
4.3. IRAF

Al estar basado en el formato FITS, hemos usado las funciones de la librería "nom.tam.fits" que aunque no está preparada para manejar el formato IRAF, si permite obtener los datos necesarios para representar el espectro.

Este formato es una estructura FITS como la que hemos explicado pero almacena los espectros de manera diferente. En IRAF el espectro va almacenado en la cabecera principal, es decir, en una imagen. Dicha imagen tiene cuatro columnas:

- La primera: valores del flujo.
- La segunda: valores del flujo.
- La tercera: valores del cielo sustraido.
- La cuarta: errores en el flujo.

En la siguiente imagen vemos un ejemplo en el que tiene 1408 filas y 4 columnas.



Para obtener los valores del eje X se utiliza esta fórmula:

$$wi = CRVAL1 + CD1\_1 * (li - CRPIX1)$$

- CRVAL1: Este valor está en la cabecera del fichero, si no está, se le asigna el valor 0.
- CD1_1: Este valor está en la cabecera del fichero, si no está, se le asigna el valor 1.
- CRPIX1: Este valor está en la cabecera del fichero, si no está, se le asigna el valor 0.
- Li: este valor se va incrementando desde 1 hasta el valor total de puntos que tiene el espectro.

Las unidades de los ejes las sacamos también de la cabecera con los siguientes valores:

- BUNIT: unidades del eje Y.
- WAT1_001: unidades del eje X.

En la siguiente imagen mostramos parte del contenido de la cabecera de un fichero IRAF.

```
FILTER3 = 'OPEN    '           / Filter identifier in wheel 3
MASKNAME= 'LongSlits1.23Arcsec' / Mask name
SLITW   =                 1.23 / Slit width
SLITPA  =                   0. / Slit pa
CCDNAME = 'CCD_2: 01394_17_02' / CCD Name (1,2): Serial Number
AMPNAME = 'CCD_2: Left'
CCDSIZE = '[1:2048,1:4102]'    / CCDSize
CCDSUM  = '2 2     '           / First horizontal bin, then vertical bin
AMPSEC  = '[1:2048,1:4102]'    / AMP Section
DETSEC  = '[2049:4096,1:4102]' / Detector Section
RON     = 'TBD_L2_SPEED100_GAIN_X4_75_RON' / RON
ADU     = 'TBD_L2_SPEED100_GAIN_X4_75_ADU' / ADU
GAIN    = '1.18    '           / GAIN
GAINOLD = '4.75    '
RDNOISE =                 3.5
WCSDIM  =                   3
LTM1_1  =                  1.
LTM2_2  =                  1.
WAT0_001= 'system=equispec'
WAT1_001= 'wtype=linear label=Wavelength units=angstroms'
WAT2_001= 'wtype=linear'
TRIM    = 'Sep 24 14:36 Trim data section is [30:950,5:2000]'
OVERSCAN= 'Sep 24 14:36 Overscan section is [4:20,5:2048] with mean=894.7644'
ZEROCOR = 'Sep 24 14:36 Zero level correction image is zero'
ILLUMCOR= 'Sep 24 14:36 Illumination image is illum with scale=1.'
CCDMEANT=           969806214
CCDPROC = 'Sep 24 14:36 CCD processing done'
BIASSEC = '[4:20,5:2048]'
TRIMSEC = '[30:950,5:2000]'
BANDID1 = 'spectrum - background fit, weights variance, clean no'
BANDID2 = 'raw - background fit, weights none, clean no'
BANDID3 = 'background - background fit'
BANDID4 = 'sigma - background fit, weights variance, clean no'
APNUM1  = '1 1 267.56 277.74'
CTYPE1  = 'LINEAR  '
CTYPE2  = 'LINEAR  '
CTYPE3  = 'LINEAR  '
CRVAL1  =                5400.
CRPIX1  =                  1.
CD1_1   =     2.7007818052594
CD2_2   =                  1.
CD3_3   =                  1.
LTM3_3  =                  1.
WAT3_001= 'wtype=linear'
DC-FLAG =                   0
DCLOG1  = 'REFSPEC1 = aca71598_2'
EX-FLAG =                   0
CA-FLAG =                   0
BUNIT   = 'erg/cm2/s/A'
END
```

4.4. VOTable

Es un formato XML y a través de la librería STIL accediendo a las etiquetas necesarias podemos leer tanto los parámetros del espectro como los valores.

Mediante la etiqueta <FIELD ID = “ ” / > se define la información que contiene la columna. Habrá tantas etiquetas de este tipo como columnas haya. Una vez definidas todas las columnas, lo siguiente es completar la tabla con los valores mediante las etiquetas <TABLEDATA>, <TR> y <TD>, como podemos comprobar en el siguiente ejemplo:

```
<?xml version="1.0"?>
<!DOCTYPE VOTABLE SYSTEM "http://us-vo.org/xml/VOTable.dtd">
<VOTABLE version="v1.0">
  <RESOURCE utype="sed:SED">
    <TABLE
name="../FIX_NED_txt/2006ApJS..164...81M/UGC_08058:S:Opt:mk2006_votabl
e.xml" utype="sed:Segment">
      <GROUP utype="sed:Segment.SpectralCoord">
        <FIELDref ref="Frequency" />
      </GROUP>
      <GROUP utype="sed:Segment.Flux">
        <FIELDref ref="Flux" />
        <GROUP utype="sed:Segment.Flux.Accuracy">
          <FIELDref ref="StatErrLow" />
          <FIELDref ref="StatErrHigh" />
        </GROUP>
      </GROUP>
      <FIELD      ID="Frequency"      unit="Hz"      datatype="double"
name="Frequency" utype="sed:Segment.Points.SpectralCoord.Value" />
      <FIELD  ID="Flux"  unit="W/m^2/Hz"  datatype="double"  name="Flux"
utype="sed:Segment.Points.Flux.Value" />
      <FIELD   ID="StatErrLow"      unit="W/m^2/Hz"   datatype="double"
name="StatErrLow"   utype="sed:Segment.Points.Flux.Accuracy.StatErrLow"
/>
      <FIELD    ID="StatErrHigh"    unit="W/m^2/Hz"    datatype="double"
name="StatErrHigh"
utype="sed:Segment.Points.Flux.Accuracy.StatErrHigh" />
        <DATA>
          <TABLEDATA>
              <TR>
                <TD>8.15954958e+14</TD>
                <TD>5.39919793e-29</TD>
                <TD>6.04887040e-30</TD>
                <TD>6.04887040e-30</TD>
              </TR>
              <TR>
                <TD>8.15344692e+14</TD>
                <TD>4.47936505e-29</TD>
                <TD>5.93008145e-30</TD>
                <TD>5.93008145e-30</TD>
              </TR>
```

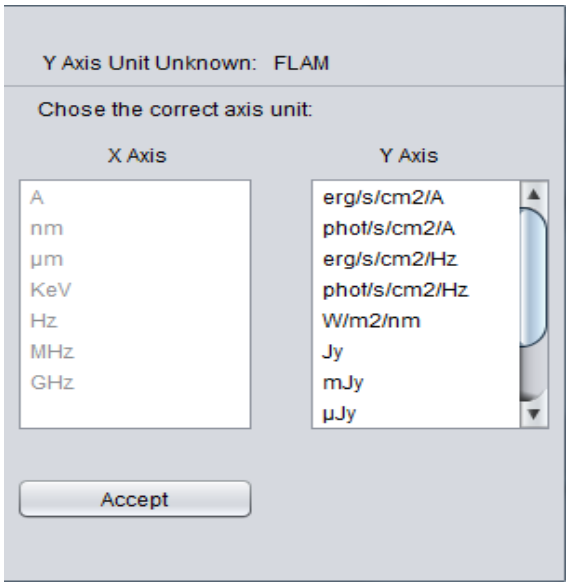
El código seguiría completando los valores de la tabla de datos.

4.5. Control de Errores:

Para evitar posibles errores y hacer que GUIspec sea capaz de abrir la mayoría de los archivos, si no reconoce la unidad de alguno de los ejes o no reconoce la columna/etiqueta de cada eje debido a que estos no tienen una nomenclatura estándar, en GUIspec se abrirá un cuadro de diálogo modal donde muestra las cadenas de texto que ha encontrado y el usuario podrá seleccionar en una lista la unidad o la columna correcta.

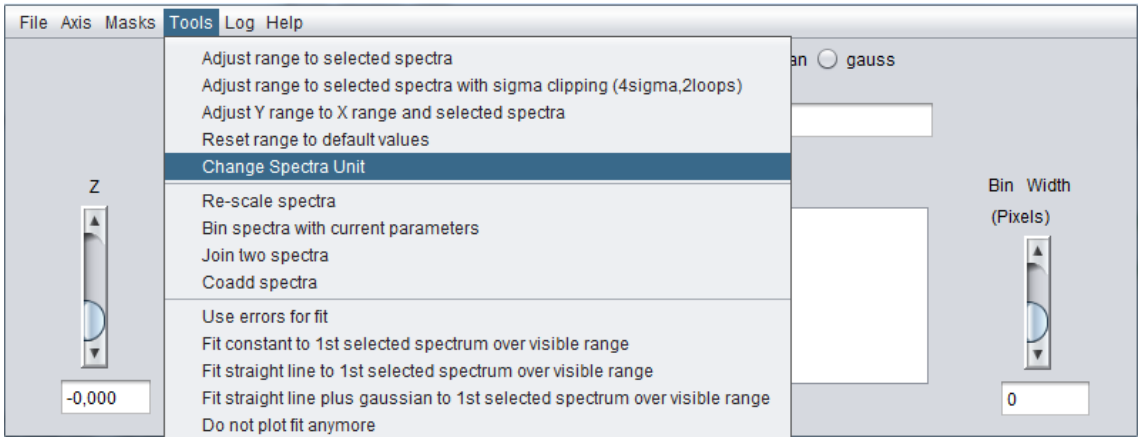
Los menús desplegables muestran todas las columnas que hay en el documento para que el usuario elija la correcta. En el ejemplo de la imagen anterior vemos que la columna correspondiente al eje X es "WAVE" pero GUIspec no lo reconoce porque el nombre estándar para la columna es: "WAVELENGTH".

En la siguiente imagen, vemos que GUIspec ha reconocido las unidades del eje X, pero no las del eje Y. Lo que ha leído como unidad es "FLAM" y nos habilita la lista del eje Y para seleccionar la unidad apropiada.

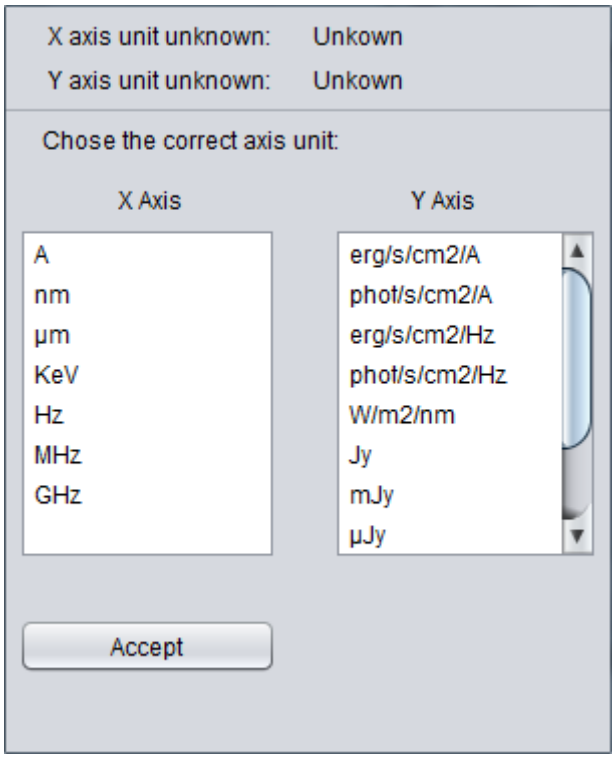


Aprovechando la estructura del control de errores en el momento de abrir un fichero, se ha implementado también una opción en el menú para cambiar la unidad de entrada del espectro una vez esté abierto.

Si un usuario se equivoca al seleccionar las unidades de entrada al abrir un fichero, podrá rectificar a través del siguiente menú:



Y obtenemos el siguiente cuadro de diálogo modal donde elegimos las unidades correctas de entrada:



Esta implementación realiza dos transformaciones en el objeto de tipo Spectrum:

1. Cambia las unidades de las nativas de la clase Spectrum a las originales del espectro.

2. Una vez tenemos los valores del espectro originales, se seleccionan las unidades de entrada y se hace la transformación correspondiente para convertirlas en las nativas de la clase Spectrum.

Para ello se ha añadido una variable que almacena las unidades en las que se han leído los espectros para poder cambiarlos a través de esta opción si es necesario.

4.6. Flujo de trabajo:

En los tres casos, el flujo de trabajo es identico tanto en apertura como en escritura.

Apertura:

1. Se comprueba el formato en el que está el fichero.

2. Se comprueba si cumple el estándar.

3. Se comprueba si se reconocen las etiquetas de las columnas y las unidades.

Si las reconoce

- Si son las unidades nativas de la clase Spectrum lo representa.
- Si no lo son, hace la transformación y lo representa en las unidades nativas.

Si no las cumple pide al usuario los datos necesarios de entrada:

- Si los datos de entrada coinciden con las unidades nativas de la clase Spectrum lo representa.
- Si no coinciden, hace la transformación y lo representa en las unidades nativas.

Escritura:

1. Se crea una estructura que cumpla con el estándar del formato seleccionado.

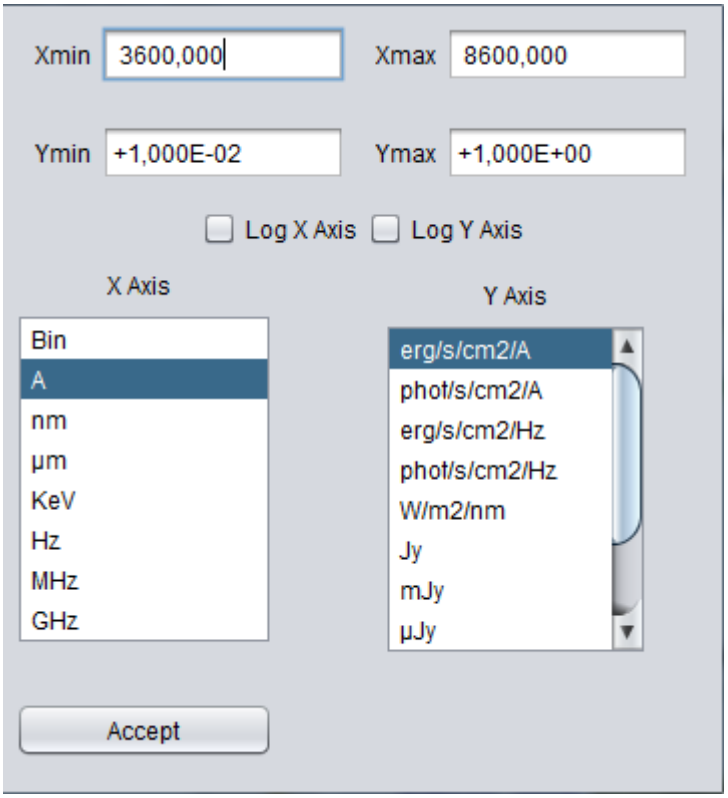
2. Se completa la estructura con los valores del espectro.

3. Se escribe el fichero de salida con los datos de la estructura completa.

Capitulo 5.- Cambio de Unidades en los Ejes

5.1. Cambio de unidades

Como ya se ha comentado anteriormente, antes de la actualización sólo representaba la lóngitud de onda en Angstroms y el flujo en $\frac{erg}{s*cm^2*A}$. En esta actualización hemos añadido un menú que permite elegir la unidad de cada eje y además, en dicho menú se han incluido los valores de X e Y máximos y mínimos que antes estaban en la interfaz principal de MenuFrame:

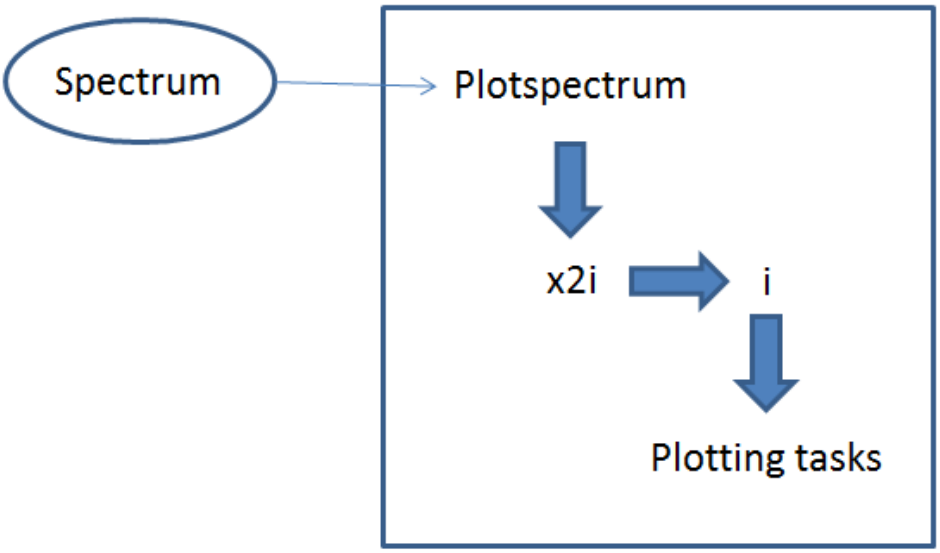


En total se han añadido 7 unidades para el eje X y 10 para el eje Y, lo que da una mayor versatilidad a GUIspec para poder representar y en definitiva, estudiar mejor los espectros.

Para esta mejora de GUIspec decidimos implementar la solución menos invasiva para evitar modificaciones en la clase Spectrum.

Para poder explicarlo primero hay que entender como representa las gráficas GUIspec:

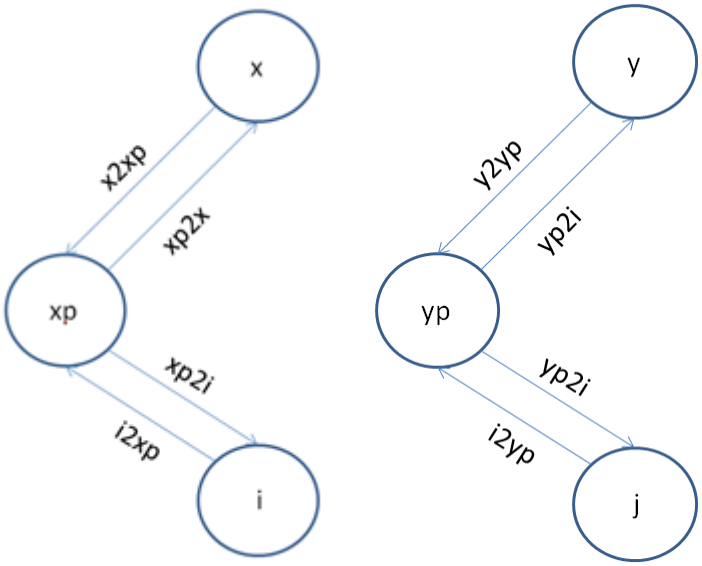
El método "Plotspectrum" de la clase PlotFrame, se encarga de dibujar la gráfica comprobando que no se sale de los márgenes de representación. Para ello transforma cada valor del espectro en el eje X y en el eje Y mediante una función lineal (implementada en los métodos x2i e y2j), en unos valores "i" y "j" que representan un pixel en la gráfica. Si este punto se encuentra entre los valores máximos y mínimos de "i" y de "j" lo utiliza para representar la recta que lo une con el punto anterior.



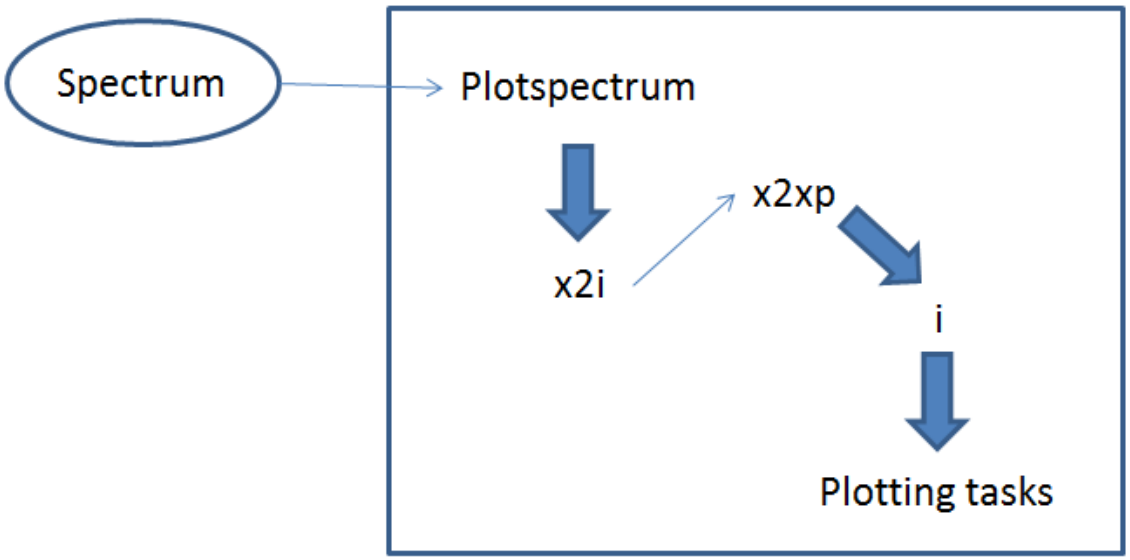
Lo que hemos hecho es crear unas variables intermedias "Xp" e "Yp" dentro de PlotFrame que contienen el valor de X y de Y en la unidad que el usuario quiere que sea representado.

Por ejemplo, tenemos un valor X[1] = 10 Å, ya que es la unidad que se almacena en el objeto Spectrum y queremos que nos lo represente en nm. Internamente, GUIspec hará todos los cálculos con X[1] = 10 Å, pero en aquellas partes de la interfaz donde tenga que aparecer visible este valor para el usuario (en la gráfica y en los datos del espectro que aparen en MenuFrame), GUIspec utilizará Xp[1] = 1nm.

De modo que donde antes el paso de X a i y de Y a j y viceversa era directo, ahora necesitamos un paso intermedio que llevan a acabo los métodos implementados en PlotFrame que podemos ver en el siguiente diagrama:



Así, a través de estos nuevos métodos podemos cambiar la unidad en la que se muestran los ejes sin tener que modificar la clase Spectrum, con lo que la estructura principal de GUIspec sigue siendo la misma y el funcionamiento de Plotspectrum queda de la siguiente forma:



5.2. Flujo de trabajo

A través del menu “Axis” se seleccionan las unidades que se necesiten, esto un valor a dos variables de PlotFrame que regulan el uso de x2xp y de y2yp respectivamente
.
Se fuerza a representar de nuevo los espectros.

Al llegar a PlotSpectrum, el método x2i llama a x2xp y y hace el cambio a la unidad seleccionada. Lo mismo con y2j e y2yp. Con esto tenemos redibujados los espectros en las unidades seleccionadas.

Por último se fuerza a rescribir la descripción de los espectros en MenuFrame y para que los valores de xp e yp máximos y mínimos también cambien de acuerdo a las unidades seleccionadas.

Capítulo 6: Utilización de los Errores

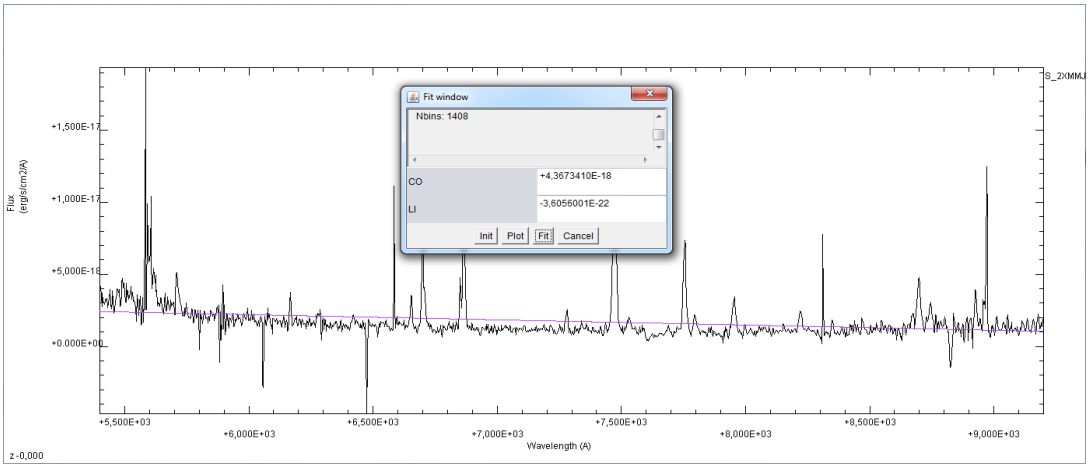
6.1. Errores en los ajustes

El tratamiento de los errores en los espectros era algo que el Dr. Francisco J. Carrera tenía pensado desde hace mucho tiempo y había comenzado su implementación, pero sin llegar a finalizarla, con lo que tras el estudio de código simplificó mucho el trabajo.

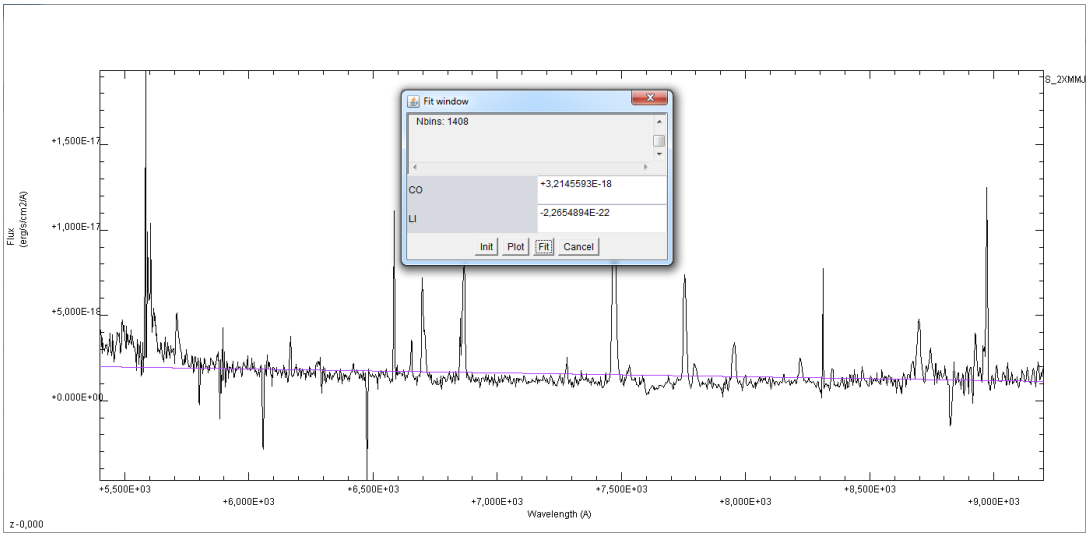
Nos encontramos con que en la clase Spectrum está preparada para que los objetos instanciados puedan albegar errores, pero en el resto de las clases de GUIspec, no lo utilizan, con lo que lo primero que se hizo fue completar el trabajo y hacer que el resto de objetos comprueben si el espectro contiene errores.

Los ajustes lineales sin errores se hacen mediante mínimos cuadrados. Para utilizar los errores se han añadido las condiciones necesarias para comprobar si el ajuste se debe de hacer con errores o sin errores y encaso de que los errores se tengan en cuenta, dividir cada término del sumatorio de mínimos cuadrados entre el error en el flujo.

Ejemplo de ajuste lineal SIN tener en cuenta los errores.



Ejemplo de ajuste lineal teniendo en cuenta los errores.



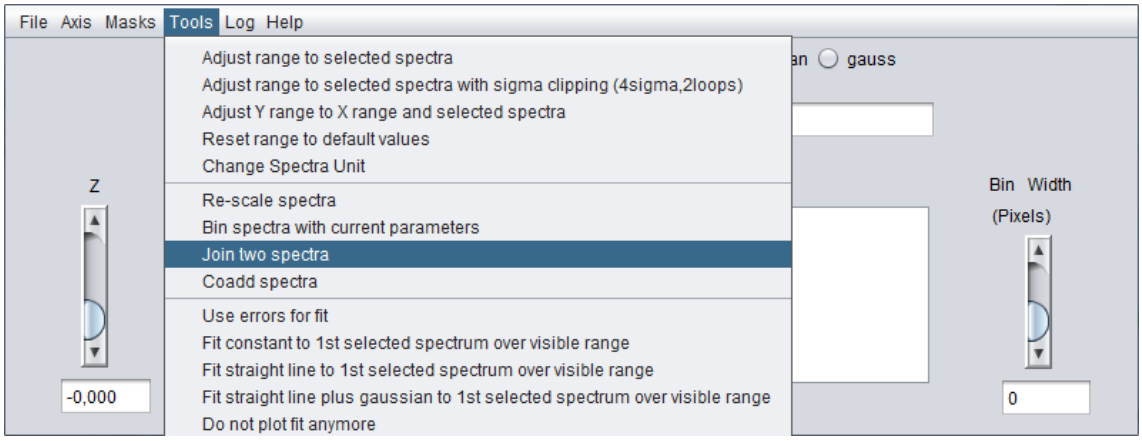
Flujo de trabajo:

Realiza el ajuste lineal según el modelo de ajuste indicado y comprueba el *toggle* de uso de errores.

Si el uso de errores está activado y el espectro tiene errores, hará el ajuste teniendo en cuenta los errores.

Si una de las dos condiciones anteriores no se cumple, el ajuste lo realizará sin tener en cuenta los errores.

6.2. Operaciones con espectros



A través de las opciones del menú "Tools" tenemos acceso a dos funciones, "Join two spectra" y "Coadd spectra" para sumar y solapar espectros respectivamente.

Estas dos funciones funcionaban correctamente, pero no tenían en cuenta los errores, es decir, de dos o más espectros generan uno nuevo, pero si los dos primeros tenían errores, estos se perdían en el proceso y el nuevo espectro no tenía errores.

Se ha añadido a los dos métodos la condición de que si los espectros de entrada contienen errores, el espectro generado también los contenga obtenidos mediante la siguiente fórmula:

$$Error = \sqrt{(error\ espectro\ A)^2 + (error\ espectro\ B)^2}$$

Capitulo 7: Últimas Consideraciones

Las premisas de este proyecto se dividen en dos partes: la primera consiste actualización del código, actualización de la interfaz gráfica y ordenar el código dentro de un paquete JAR. La segunda contempla la implementación de mejoras.

La primera parte fue más compleja de lo que me esperaba debido en gran parte al acceso a recursos, ya que GUIspec utiliza un objeto de entrada y salida propio y hubo que modificarlo de manera muy cuidadosa para que todo funcionase correctamente.

De la segunda parte destacar la implementación de las variables intermedias para los cambios de unidad de representación, que fue bastante complicada de idear pero con las ideas claras su implementación no fue tan compleja.

Aunque esta aplicación aún tiene margen de mejora, como por ejemplo más operaciones con los espectros o el acceso al "Observatorio Virtual" para descargar espectros directamente desde GUIspec, a través de este proyecto se han cubierto unos objetivos fundamentales que sientan una base que por un lado amplían las funciones de GUIspec haciendo de esta aplicación una herramienta mucho más potente y versátil, y por otra parte permiten futuras ampliaciones, ya que ahora está implementado lo necesario para poder crear el programa más completo de su área de trabajo.

Podemos concluir que los objetivos del proyecto se han cumplido y esta aplicación es ahora mas completa y permite en futuras revisiones ampliar sus funciones aprovechando el trabajo realizado.

Anexo 1

Métodos implementados:

MenuFrame:

Void writeFits() llama a *getfilename()*, obtiene el nombre del fichero donde queremos guardar el espectro y llama a *writeFits(String specfile)* que se encarga de crear la estructura fits y guardarla en el archivo especificado.

Void writeIraf() llama a *getfilename()*, obtiene el nombre del fichero donde queremos guardar el espectro y llama a *writeIraf(String specfile)* que se encarga de crear la estructura IRAF y guardarla en el archivo especificado.

Void writeVOtables() llama a *getfilename()*, obtiene el nombre del fichero donde queremos guardar el espectro y llama a *writeVOtables(String specfile)* que se encarga de crear la estructura XML y guardarla en el archivo especificado.

Boolean openIraf(String specfile), obtiene los datos necesarios para crear un objeto de tipo Spectrum, utiliza un método de *PlotFrame (addfits)* para ello.

Boolean openFits(String specfile), obtiene los datos necesarios para crear un objeto de tipo Spectrum, utiliza un método de *PlotFrame (addfits)* para ello.

Boolean openVOtable(String specfile), obtiene los datos necesarios para crear un objeto de tipo Spectrum, utiliza un método de *PlotFrame (addfits)* para ello.

Boolean axisUnit(boolean xknown, boolean yknown, String xKnownS, String yKnownS), recibe los parámetros de *openIraf*, *openFits* u *openVOtables*. Los parametros son lo siguiente:

- Xknown = verdadero si ha reconocido la unidad del eje X, falso si no.
- Yknown = verdadero si ha reconocido la unidad del eje Y, falso si no.
- xKnownS = contiene la cadena de la etiqueta de la unidad del eje X.
- yKnownS = contiene la cadena de la etiqueta de la unidad del eje Y.

Con esto edita un cuadro de diálogo modal y lo muestra si fuese necesario.

Double[] changeUnitX(double[] xx) recibe el vector con los valores primitivos del eje X del espectro leído y los transforma a la unidad nativa de GUIspec.

Double[] changeUnitY(double[] xx, double[] yy) recibe el vector con los valores del eje X en la unidad nativa de GUIspec y los valores del eje Y en la unidad primitiva del espectro y transforma estos últimos a la unidad nativa de GUIspec.

Void correctunit(int[] wrong) se encarga de corregir la unidad si el usuario se equivoca al abrir un fichero. Wrong es un par que contiene las unidades

tanto del eje X como del eje Y que se tomaron erróneamente como primitivas del espectro y este método deshace la operación que se hizo para pasar el espectro de la unidad primitiva equivocada a Angstroms.

PlotFrame:

String addfits(double[] xx, double[] yy, String label) recibe los datos leídos de un fichero de tipo IRAF, FITS o VOTable y genera el objeto de tipo Spectrum sin errores. Devuelve la cadena que se representa en MenuFrame con los valores del espectro.

String addfits(double[] xx, double[] yy,double[] ey, String label) recibe los datos leídos de un fichero de tipo IRAF, FITS o VOTable y genera el objeto de tipo Spectrum con errores. Devuelve la cadena que se representa en MenuFrame con los valores del espectro.

Cambios de unidades:

Eje X:

Double x2xp(x): recibe el valor de X y lo transforma a la unidad que se requiera.

Double xp2x(xp): recibe el valor de XP y lo transforma a la unidad nativa.

Double i2xp(i): recibe el valor de I y devuelve el valor equivalente de XP.

Int xp2i(xp): recibe el valor de XP y devuelve el valor equivalente de I.

Eje Y:

Double y2yp (y): recibe los valores de Y y de X y transforma el valor de Y a la unidad que se requiera.

Double yp2y(yp): recibe el valor de XP y de YP y transforma el valor de YP a la unidad nativa.

Double j2yp(j): recibe el valor de J y devuelve el valor equivalente de YP.

Int yp2j(yp): recibe el valor de YP y devuelve el valor equivalente de J.

Corrección de unidad de entrada:

Void addwrongx(int val) recibe la unidad que se seleccionó al abrir el fichero y la añade al final del vector wrongX.

Void setwrongx(int val, int pos) recibe una unidad y una posición y coloca la unidad en la posición del vector wrongX seleccionada. Se utiliza cuando corrige un error.

Void delwrongx(int pos) borra la posición indicada del vector wrongX.

Void addwrongy(int val) recibe la unidad que se seleccionó al abrir el fichero y la añade al final del vector wrongY.

Void setwrongy(int val, int pos) recibe una unidad y una posición y coloca la unidad en la posición del vector wrongY seleccionada. Se utiliza cuando corrige un error.

Void delwrongx(int pos) borra la posición indicada del vector wrongY.

Int getwrongs(int pos) devuelve los valores de wrongX y wrongY en la posición indicada.

Void Setfiterror(boolean val) pone el toggle del ajuste con errores a valor val y se lo comunica a los modelos de ajuste.