Kino: Edición de vídeo en GNU/Linux

Alvaro del Castillo San Félix

acs@barrapunto.com

GNU/Linux cada vez se comporta de forma más completa y mejor en diferentes campos de la informática, y uno de los que ya lleva tiempo con herramientas que permiten su uso en campos profesionales es el de la edición de vídeo digital.

Nos vamos a centrar en este tutorial en la herramienta Kino, que si bien no es muy conocida debido a que hasta hace poco se aplicaba únicamente a la edición de DV, se está desarrollando de forma vertiginosa y acaba de publicarse la versión 0.6 de la misma, aunque nosotros nos vamos a centrar en la última versión que existe como paquete Debian, 0.51. En algunos aspectos sin embargo nos basaremos en la versión 0.6, ya que es en está cuando comienzan a funcionar de forma correcta.

A la edición y captura del vídeo desde Kino, uniremos las posibilidades de transcode a la hora de convertir formatos para terminar con una plataforma potente y flexible de edición de vídeo.

1. Introducción

GNU/Linux cada vez se comporta de forma más completa y mejor en diferentes campos de la informática, y uno de los que ya lleva tiempo con herramientas que permiten su uso en campos profesionales es el de la edición de vídeo digital.

Nos vamos a centrar en este tutorial en la herramienta Kino, que si bien no es muy conocida debido a que hasta hace poco se aplicaba únicamente a la edición de DV, se

está desarrollando de forma vertiginosa y acaba de publicarse la versión 0.6 de la misma, aunque nosotros nos vamos a centrar en la última versión que existe como paquete Debian, 0.51. En algunos aspectos sin embargo nos basaremos en la versión 0.6, ya que es en está cuando comienzan a funcionar de forma correcta.

A la edición y captura del vídeo desde Kino, uniremos las posibilidades de transcode a la hora de convertir formatos para terminar con una plataforma potente y flexible de edición de vídeo.

2. Instalación de Kino

Lo primero de todo es dejar claro que Kino es un producto que está avanzando a gran ritmo y que aún no ha sacado ninguna versión oficial. Aunque su funcionamiento por lo general sea bastante aceptable, nos encontraremos con problemas, algo normal y que será cada vez más extraño según avances las versiones. En ningún momento vamos a intentar ocultar ningún problema, más bien todo lo contrario, ya que los informes de los problemas a los desarrolladores y su descripción lo más exhaustiva posible, son la mejor forma de ayudar en el desarrollo de Kino si no se está programando en él.

La instalación de Kino es muy conveniente hacerla desde paquetes de una distribución ya que utiliza multitud de librerías para lograr el soporte para los diferentes formatos de vídeo, audio, captura de vídeo, comunicaciones por ieee1394 y otras necesidades del programa.

Dentro de Debian tenemos empaquetada la versión 0.51 de Kino, aunque acaba de salir la versión 0.6 que es previsible que esté en breve disponible.

```
      acs@linex:~/devel/web-xml/articulos/kino$ apt-cache show kino
Package: kino
Priority: extra
Section: graphics
Installed-Size: 1240
Maintainer: Daniel Kobras <kobras\@debian.org>
Architecture: i386
Version: 0.51-1
Depends: gdk-imlib1, libart2 (>= 1.2.13-5), libaudiofile0 (>= 0.2.3-4),
libavc1394-0, libc6 (>= 2.2.5-13), libdb3 (>= 3.2.9-16), libdv2,
libesd0 (>= 0.2.23-1) | libesd-alsa0 (>= 0.2.23-1), libglib1.2 (>= 1.2.0),
libgnome32 (>= 1.2.13-5), libgnomesupport0 (>= 1.2.13-5),
libgnomeui32 (>= 1.2.13-5), libgtk1.2 (>= 1.2.10-4), libraw1394-5,
libstdc++2.10-glibc2.2 (>= 1:2.95.4-0.010810), libxml2 (>= 2.4.19-4),
xlibs (>> 4.1.0)
```

```
Suggests: gnome-help, vorbis-tools, sox
Filename: pool/main/k/kino/kino_0.51-1_i386.deb
Size: 504320
MD5sum: 4570e44ced9c7d9547d3e0bdbe381f4e
Description: Non-linear editor for Digital Video data
 Kino allows you to record, create, edit, and play movies recorded with DV
 camcorders. Unlike other editors, this program uses many keyboard commands
 for fast navigating and editing inside the movie.
```

Vemos que Kino utiliza Gtk+ 1.2 para crear su interfaz gráfica, la cual veremos que es un poco peculiar y adaptada a las necesidades de edición de vídeo, con algunos widgets muy específicos que por ejemplo, complican la migración de Kino hacia la nueva Gtk+ 2.0.

Resumiendo, por el momento, hemos utilizado la siguiente orden para instalar kino en nuestro sistema:

```
apt-get install kino
```

3. Acceso al vídeo digital de nuestra cámara: ieee1394

Lo primero que necesitamos es conectar nuestra estupenda cámara de vídeo digital con el computador. Esto lo realizamos utilizando la salida digital que tienen este tipo de cámaras, y enchufando esta salida con un cable adecuado a alguna de las entradas ieee1394 de nuestro computador. En el caso de no tener ninguna, podemos adquirir por menos de 60 euros tarjetas PCI que tienen 4 puertos ieee1394 y para los portátiles, también hay tarjetas PCMCIA que ofrecen este tipo de puertos.

Aunque lo normal es que los ordenadores comprados de 2 años hacia aquí dispongan ya de este tipo de interfaz de comunicaciones. En el caso que nos atañe, tenemos una cámara digital Canon MV300 (ZR10 orginalmente) (http://es.livra.com/topic.asp?To=4873) que podéis ver junto otra gran gama de cámaras Canon (http://www.imagendv.com/canon.htm) y que ya se encuentra algo anticuada. Su resolución es de 340.000 pixels aunque da 480.000 extrapolados, es decir, 800x600.

Figura 1. Cámara Canon MV300

Podemos ya ver en la fotografía anterior lsa salida DV (Digital Video) de la cámara, que es la que tenemos que contectar al puerto ieee1394 de nuestro computador. Veamos una fotografía algo más completa de la conexión.

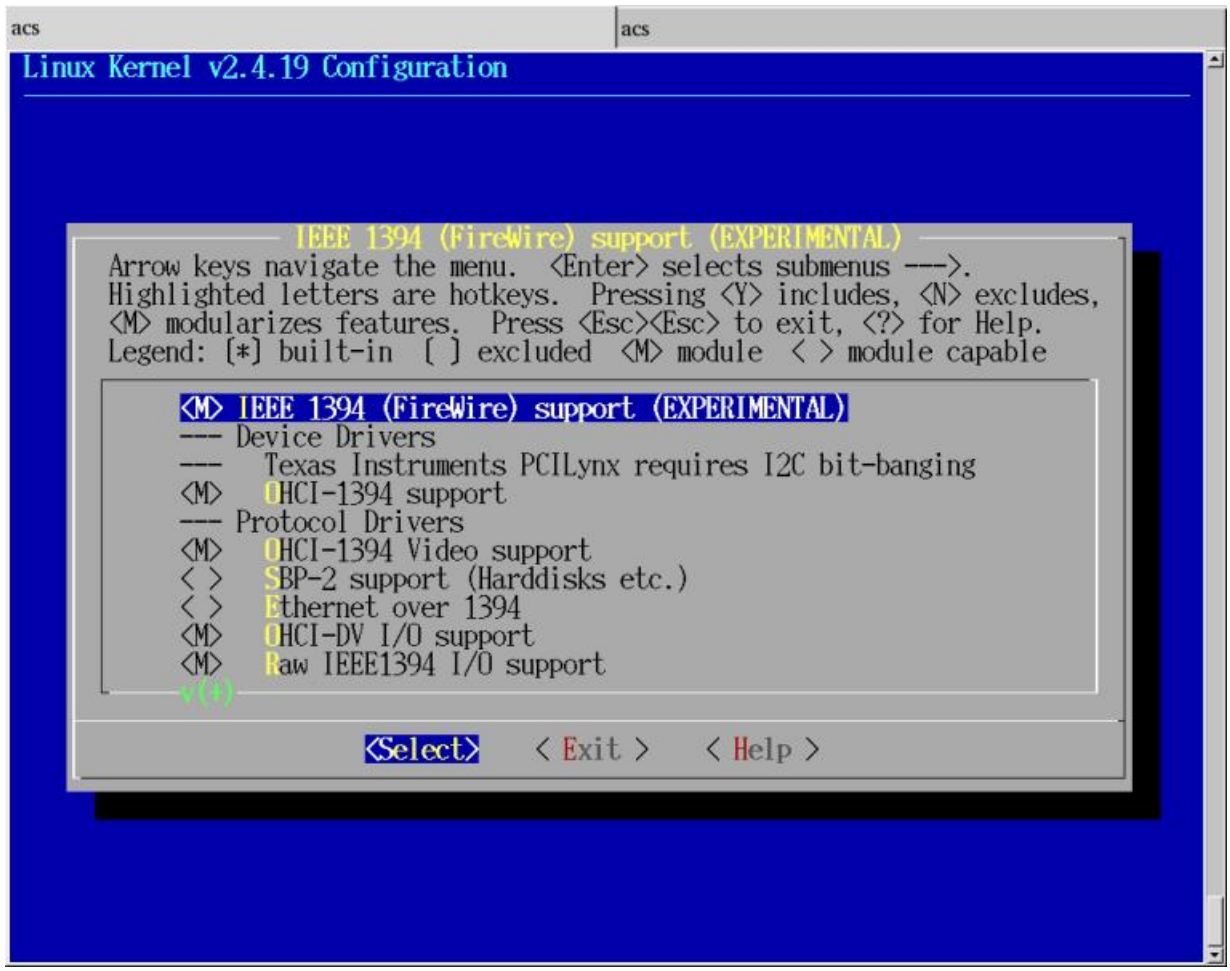
Figura 2. Conexión por ieee1394

Debemos de tener en nuestro núcleo de Linux el controlador del puerto ieee1394, algo cada vez más habitual en las nuevas versiones. Al ser aún un dispositivo experimental recomendamos compilarlo como módulo. De esta forma si tenemos algún tipo de problema, podemos descargarlo y volverlo a cargar.

En concreto, se ha detectado que con la versión actual del núcleo 2.4.19 la captura de vídeo deja de funcionar en cuanto apagamos la cámara una vez utilizada, y la volvemos a encender para realizar nuevas capturas de vídeo, estas capturas ya no funcionarán, al menos ni desde kino y desde dvgrab. Si tenemos el soporte compilado como un módulo, basta con que descarguemos el módulo y lo volvamos a cargar para que todo vuelva a la normalidad.

Las opciones a configurar como módulos en el núcleo son:

Figura 3. Configuración como módulos de la interfaz ieee1394



y tras ello pasamos a compilar e instalar nuestro nuevo núcleo:

```
linex:/usr/src/linux# make dep;make clean;make bzImage;make modules;make mod
linex:/usr/src/linux# cp arch/i386/boot/bzImage /boot/bzImage2419
linex:/usr/src/linux# cp System.map /boot/System.map
linex:/usr/src/linux# lilo
Added linux2419
linex:/usr/src/linux# reboot
```

Los módulos necesarios para tener soporte ieee1394 son:

```
linex:/home/acs# modprobe ohci1394
linex:/home/acs# lsmod
Module                  Size  Used by    Tainted: P
ohci1394               15620   0  (unused)
ieee1394               29576   0  [ohci1394]
```

Para comprobar si efectivamente lo tenemos funcionando de forma correcta, podemos hacer un "cat" del fichero "/proc/bus/ieee1394/devices" para ver sus contenidos.

```
cat: /proc/bus/ieee1394: Es un directorio
linex:/home/acs# cat /proc/bus/ieee1394/devices
Node[00:1023]  GUID[344fc00039709c10]:
  Vendor ID: ‘Linux OHCI-1394’ [0x000000]
  Capabilities: 0x0083c0
  Bus Options:
    IRMC(1) CMC(1) ISC(1) BMC(0) PMC(0) GEN(0)
    LSPD(2) MAX_REC(2048) CYC_CLK_ACC(0)
  Host Node Status:
    Host Driver     : ohci1394
    Nodes connected : 1
    Nodes active    : 1
    SelfIDs received: 1
    Irm ID          : [00:1023]
    BusMgr ID       : [00:1023]
    In Bus Reset    : no
    Root            : yes
    Cycle Master    : no
    IRM             : yes
    Bus Manager     : yes
```

Vemos que efectivamente ha detectado algo en el bus ieee1394, en este caso el módulo ohci1394 que hace de controlador del bus. Si enchufamos la cámara al puerto ieee1394 uniendo a éste con la salida DV de la cámara, tras encender la cámara deberemos de ver que el sistema ahora reconoce dos dispositivos en el bus ieee1394.

```
linex:/home/acs# cat /proc/bus/ieee1394/devices
Node[01:1023]  GUID[344fc00039709c10]:
  Vendor ID: ‘Linux OHCI-1394’ [0x000000]
  Capabilities: 0x0083c0
  Bus Options:
```

```
    IRMC(1) CMC(1) ISC(1) BMC(0) PMC(0) GEN(0)
    LSPD(2) MAX_REC(2048) CYC_CLK_ACC(0)
  Host Node Status:
    Host Driver     : ohci1394
    Nodes connected : 2
    Nodes active    : 2
    SelfIDs received: 2
    Irm ID          : [01:1023]
    BusMgr ID       : [01:1023]
    In Bus Reset    : no
    Root            : yes
    Cycle Master    : no
    IRM             : yes
    Bus Manager     : yes
Node[00:1023]  GUID[000085000025235f]:
  Vendor ID: 'Unknown' [0x000085]
  Capabilities: 0x0083c0
  Bus Options:
    IRMC(1) CMC(1) ISC(1) BMC(0) PMC(0) GEN(0)
    LSPD(0) MAX_REC(32) CYC_CLK_ACC(100)
  Unit Directory 0:

    Software Specifier ID: 00a02d
    Software Version: 010001
    Length (in quads): 0
```

Vemos efectivamente que ahora hay dos nodos en el bus iee1394, siendo el segundo la cámara digital. Veremos a lo largo del tutorial como utilizando este interfaz somos capaces de manejar la cámara, ponerla en marcha, pararla, hacer retroceder o pasar hacia delante la cinta, en fin, las opciones básicas necesarias para trabajar con la cámara.

4. La interfaz gráfica de Kino

Vamos a comenzar a intimar con Kino y en especial, con toda su interfaz gráfica. Para ello comencemos arrancando la aplicación y analizando la interfaz de inicio.

Figura 4. Pantalla de inicio de Kino

Como es de esperar disponemos de una barra de menús y una barra de herramientas con los comandos que se usan de forma más habitual. También existe una barra de estado donde se nos irá mostrando información de las acciones que lleva a cabo el programa. Hasta aquí lo típico de toda interfaz gráfica moderna.

Donde comienza lo realmente interesante es en la parte central de la ventana principal, dividida en dos regiones bien diferenciadas. A la derecha tenemos un canvas que se extiende verticalmente todo lo posible y con barra de desplazamiento. En él veremos que se irán mostrando las secuencias en las que vayamos diviendo nuestra edición.

Y a la izquierda del listado de secuencias nos encontramos con la parte principal de Kino, dominada fundamentalmente por una pantalla donde se visualizará el vídeo. Tenemos varias solapas junto a la parte superior derecha de la pantalla de vídeo. Son estas las que nos permiten ir accediendo a los diferentes modos de trabajo de Kino.

- Edit: Edición del vídeo
- Capture: captura de vídeo desde la cámara digital
- Timeline: nos muestra una secuencia de imágenes del material de forma que podemos localizar de forma rápida y sencilla escenas del video, sin necesidad de anda navegando a su búsqueda o tener que recordar en que momento se producían.
- Trim: recortes de vídeo
- Export: exportar el vídeo producido a otros formatos
- FX: efectos especiales en el vídeo como filtros de colores, filtros de distorsión y otras posibilidades que iremos viendo.

Por último pero no menos importante, debajo de la salida del vídeo tenemos la interfaz de control del vídeo, desde la que podemos poner en marcha el vídeo para visualizarlo, saltar a escenas concretas del mismo, realizar pausas, ir hacia delante y hacia atrás de forma rápida y, cabe destacar la barra de "joggle" que nos permite movernos de forma muy rápida en una zona muy concreta del vídeo.

Debajo de esta barra de control del vídeo, nos encontramos con una caja de texto para el envío de comandos de edición e información sobre el vídeo como la imagen actual (frame), la fecha de grabación del vídeo y el código de tiempos.

En resumen, la interfaz de Kino se entiende de forma bastante sencilla y veremos que su uso es cómodo y rápido. Sería bueno compararla con la interfaz de otros programas similares para ver si todos ellos siguen las mismas ideas que se nos presentan en esta interfaz.

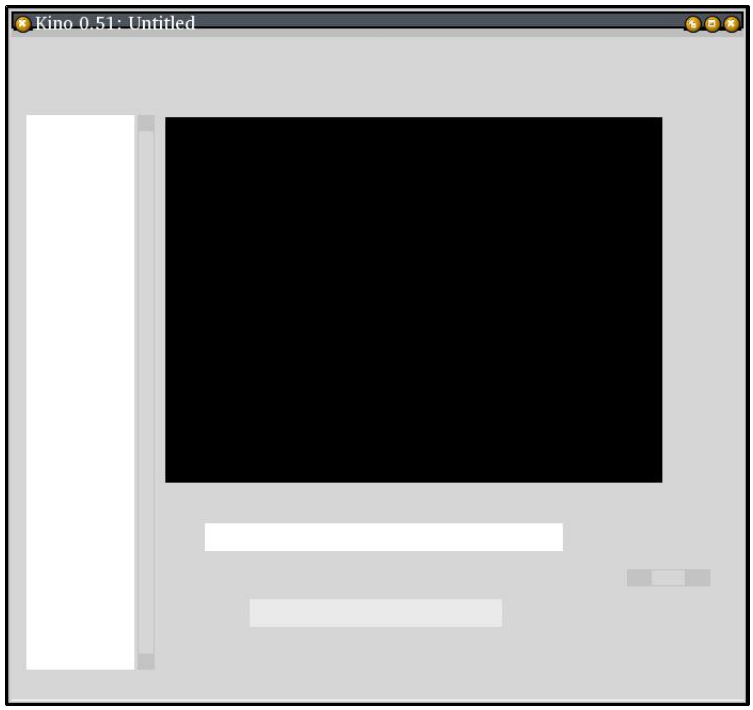
5. Captura del vídeo digital

El primer paso para poder comenzar a trabajar es el de capturar el vídeo que vamos a editar y montar. Y ello ya lo podemos hacer directamente desde la interfaz de Kino, utilizando la solapa "Capture" de la misma.

Antes de haber iniciado Kino, es conveniente tener ya la cámara de vídeo conectada y encendida ya que Kino aún no se muestra muy robusto ante cambios en caliente de la

disponibilidad de la máquina. Por ejemplo, si en la interfaz de captura encedemos la cámara de vídeo, Kino se quedará bloqueado.

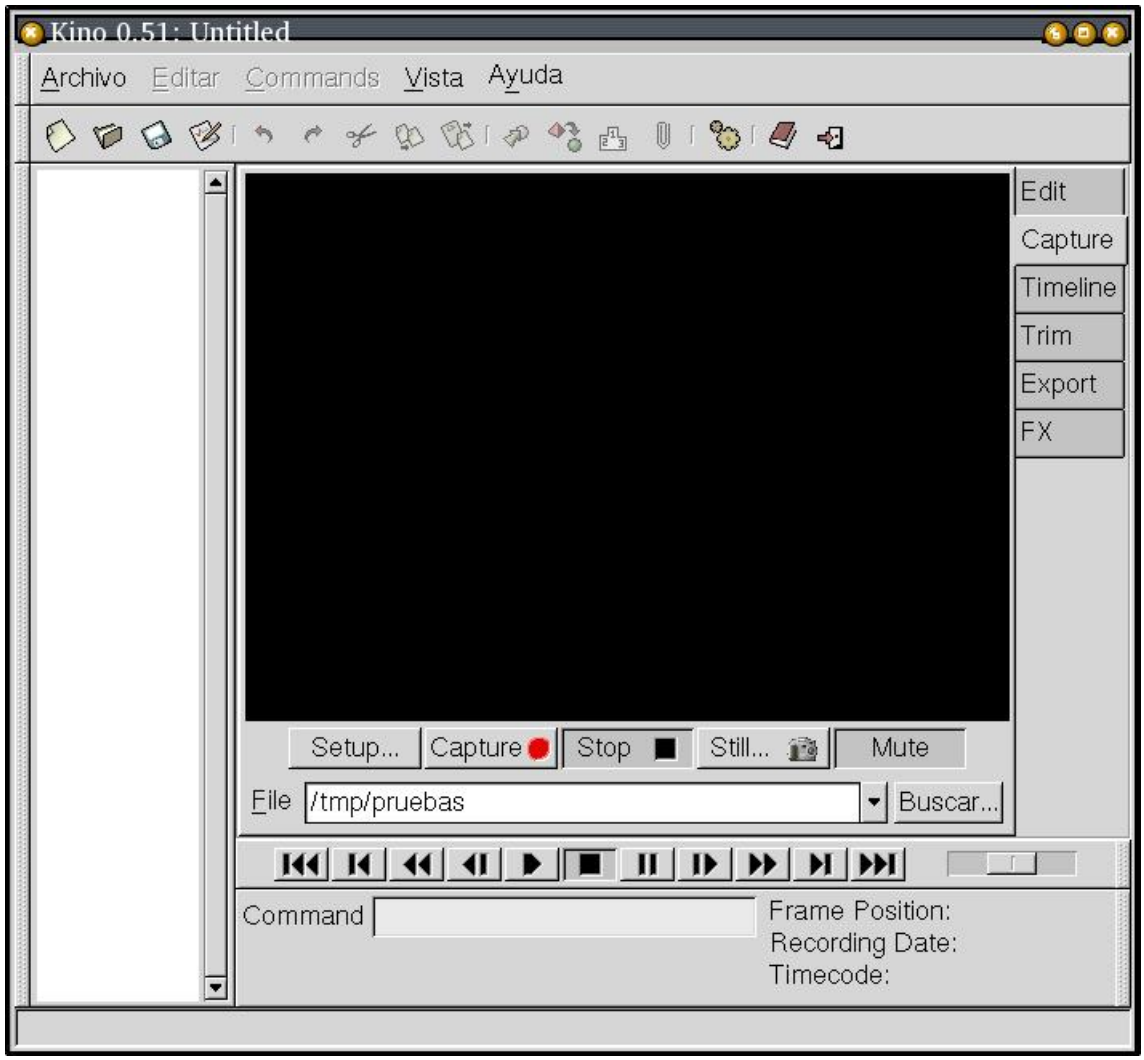
Figura 5. Kino frito



Dependiendo que cosas estemos probando y en que versión, podemos encontrarnos en ocasiones con esta situación, que nos obligará a reiniciar Kino. Por lo general Kino se muestra bastante estable, pero el fuerte desarrollo en el que aún se encuentra embarcado provoca este tipo de fallos que se irán corrigiendo de forma progresiva.

Si entramos en Kino con nuestra cámara encendida la interfaz de captura que nos presenta es la siguiente.

Figura 6. Interfaz de captura

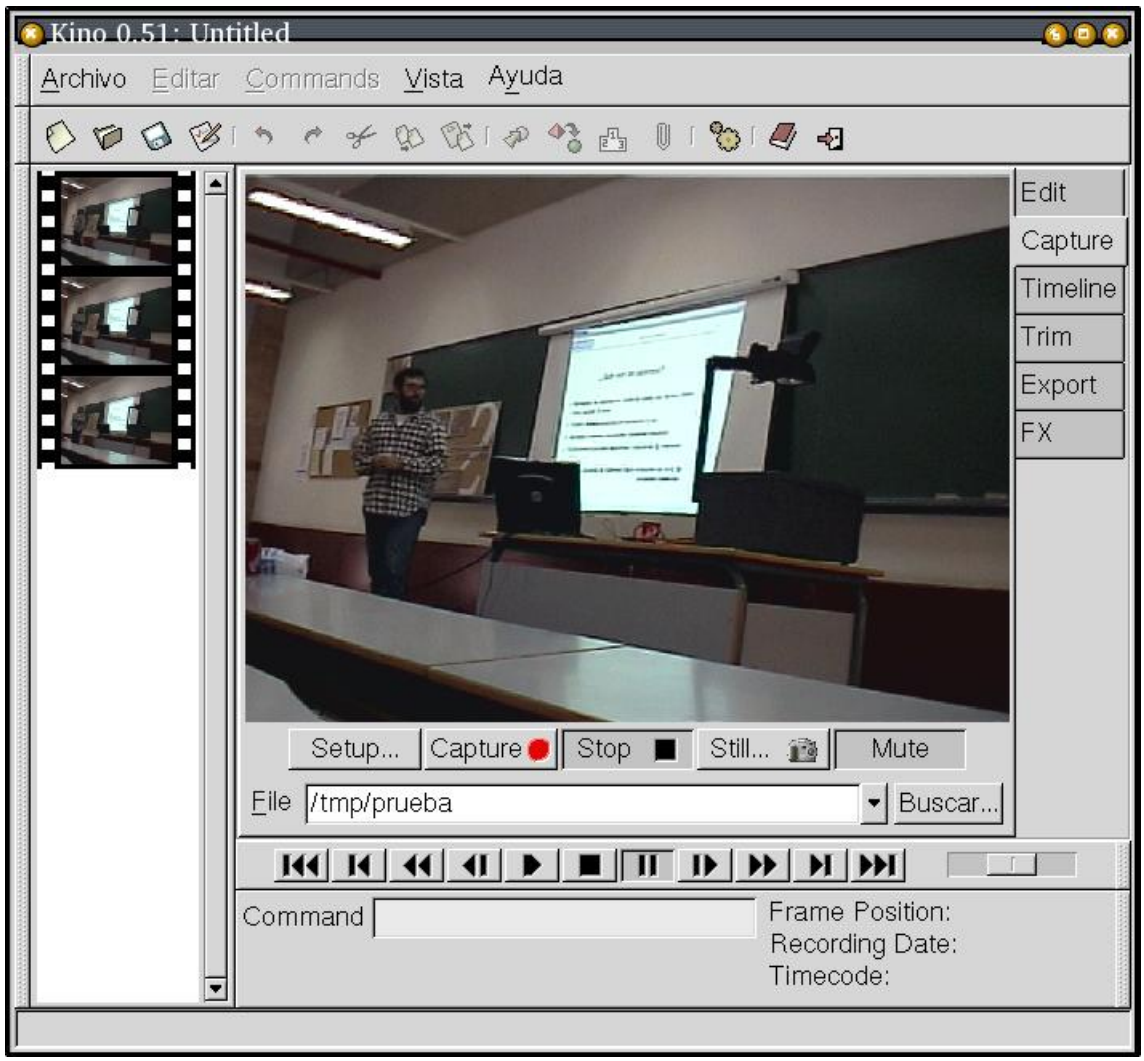


Para comenzar a capturar vídeo basta con pulsar sobre el botón rojo de "Capture" y Kino se comunicará con la cámara de vídeo, la pondrá en marcha y comenzará a capturar el vídeo que se vaya mostrando desde la cámara y emitiendo por la salido DV de la cámara. Aquí de nuevo hay que tener presente que el usuario con el que hayamos arrancado Kino debe de tener los permisos adecuados sobre los dispositivos de captura de vídeo siendo el dispositivo en cuestión "/dev/raw1394".

Podemos configurar la captura para que se vaya mostrando lo que se va grabando a la vez que se graba. Podemos incluso ir parando la captura y reanundala lo que provocará

que la captura se realice en varias secuencias diferentes. Incluso podemos quitar el "Mute" (silenciador) para oir lo que se está grabando, aunque si tenemos nuestra cámara cerca del sitio de grabación, y la cámara también emite sonido, puede no ser muy buena idea.

Figura 7. Resultado de la captura de vídeo



6. Edición del vídeo digital

Ya sabemos como capturar el vídeo desde nuestra máquina y dejarlo en ficheros en el disco duro con los que podemos trabajar desde kino. Ahora ha llegado el momento de hacer algo con ese vídeo en bruto: vamos a estudiarlo, ver que partes son las que realmente nos interesan, fijarnos un tope de duración de la secuencia final de montaje, ir delimitando las partes que nos interesan en secuencias de imágenes, unir todas estas secuencias y programar las transiciones entre dichas uniones de secuencias.

Tradicionalmente ha este proceso se le conocía como montaje. Antes de la aparición del medio digital, el proceso que se seguía era el de filmar muchas secuencias de vídeo de las que se obtendría el producto final.

A continuación se iban viendo sobre el medio físico, película de vídeo normalmente, las partes que interesaban. Estas se cortaban físicamente de la película original y luego se iban montando estos trozos seleccionados, pegándolos con algún producto, en lo que sería el montaje del producto de vídeo: una película, un documental, un reportaje o cualquier otro tipo de producto.

La técnica digital ha permitido flexibilizar mucho todo este proceso, pero la exencia sigue siendo la misma como vamos a ver: localizar trozos de vídeo (secuencias) y pegarlos para formar el video final editado.

Kino se caracteriza por realizar este proceso de edición de forma muy similar a como se usa un editor de textos: las imágenes son las letras, las líneas son las secuencias y el documento es el vídeo final editado. Esta analogía se lleva al campo del manejo de la aplicación ya que las teclas que permiten trabajar con las diferentes funciones de Kino son las mismas que las que llevan a cabo tareas similares en el editor de textos vi. Para los fanáticos de Emacs, también se está preparando el soporte para sus combinaciones de teclas.

6.1. Obtención del vídeo en bruto

El vídeo con el que vamos a trabajar fue tomado durante las II Jornadas de Software Libre en Málaga y son parte de la ponencia de Proinnova que impartió Jesús Gonzalez Barhona.

Fue una filmación con cámara fija durante unos 75 minutos. Los 75 minutos en brutos necesitarían de unos 15 GB de espacio libre para poder trabajar con ellos. Pero podemos ir capturando partes del vídeo e irlas montando de forma independiente.

El objetivo es lograr un resumen de la charla de 2 o 3 minutos en las que se presenten las ideas fundamentales de la misma. Junto a este resumen, podemos tener otro montaje

más extenso con la práctica totalidad de la charla.

El principal riesgo de una grabación como este es que resulte aburrida a las personas que la vean, ya que es una toma fija de la charla de Jesús, que al menos se mueve bastante lo que puede animar un poco la imagen.

En primer lugar, vamos a centrarnos en capturar esos 10 primeros minutos de vídeo, que finalmente han de quedarse en 1 minuto aproximadamente. La captura la podemos hacer desde Kino directamente, pero al querer que tenga una duración determinada y que tras ella se detenga, podemos utilizar mejor "dvgrab" que nos permite especificar cuanto queremos capturar. Como hemos firmado en PAL, tenemos 24 imágenes por segundo, por lo que en total queremos capturar 24*60*10 = 14400 imágenes.

```
acs@linex:~/devel/web-xml/articulos/kino$ dvgrab --format raw --frames 14400
raw1394 - couldn’t get handle: No such device
This error usually means that the ieee1394 driver is not loaded or that
/dev/raw1394 does not exist.
```

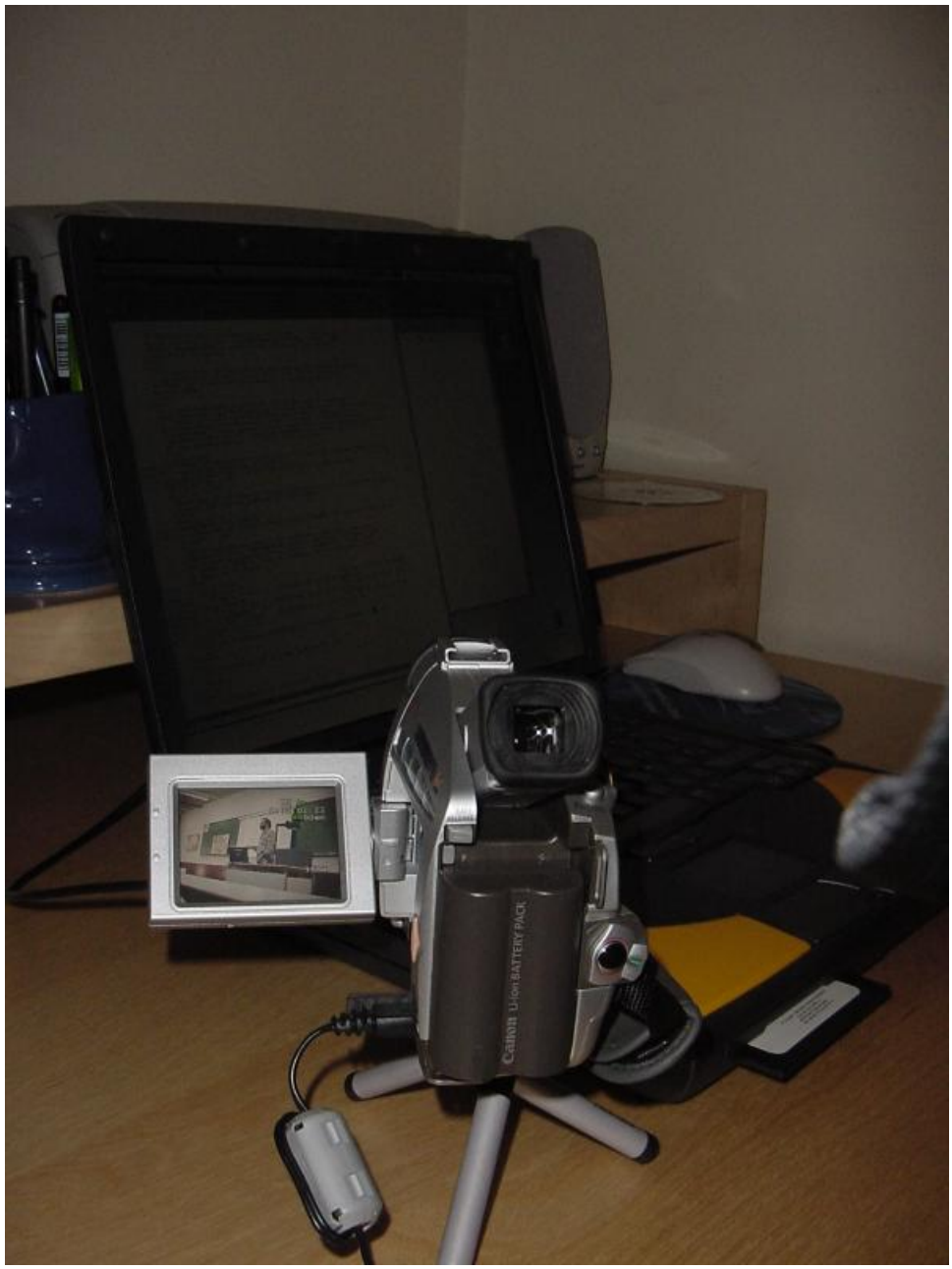
Parece que aún nos queda por cargar algún módulo antes de poder usar nuestro interfaz ieee1394 para capturar el vídeo digital.

```
linex:/home/acs# modprobe raw1394
acs@linex:~/devel/web-xml/articulos/kino$ dvgrab --format raw --frames 14400
```

Esta ya preparada la captura para llevarse a cabo, pero para que comience de verdad, deberemos de poner a reproducir la cámara, algo que no hace de forma automática "dvgrab". Podemos ir viendo como el tamaño del fichero "jesus.dv" aumenta una vez que hemos pulsado el "play" de la cámara.

```
Every 2s: ls -l jesus.dv                                  Tue Oct 22 22:16:1
-rw-r--r--    1 acs      acs      173864640 oct 22 22:16 jesus.dv
Every 2s: ls -l jesus.dv                                  Tue Oct 22 22:16:5
-rw-r--r--    1 acs      acs      327783840 oct 22 22:16 jesus.dv
Every 2s: ls -l jesus.dv                                  Tue Oct 22 22:17:1
-rw-r--r--    1 acs      acs      411207360 oct 22 22:17 jesus.dv
...
Every 2s: ls -l jesus.dv                                  Tue Oct 22 22:25:3
-rw-r--r--    1 acs      acs      2074020960 oct 22 22:25 jesus.dv
```

Figura 8. Durante el proceso de captura



Tras los 10 minutos permitentes de reproducción tenemos ya nuestro fichero de vídeo capturado y preparado para que comience la edición. Y ha llegado el momento de abrirlo con Kino y comenzar el montaje. Hemos tenido la suerte de trabajar con el

sistema de ficheros ReiserFS, que nos permite tener ficheros de más de 2 GB, una limitación que puede ser muy incómoda en el tratamiento de vídeo digital.

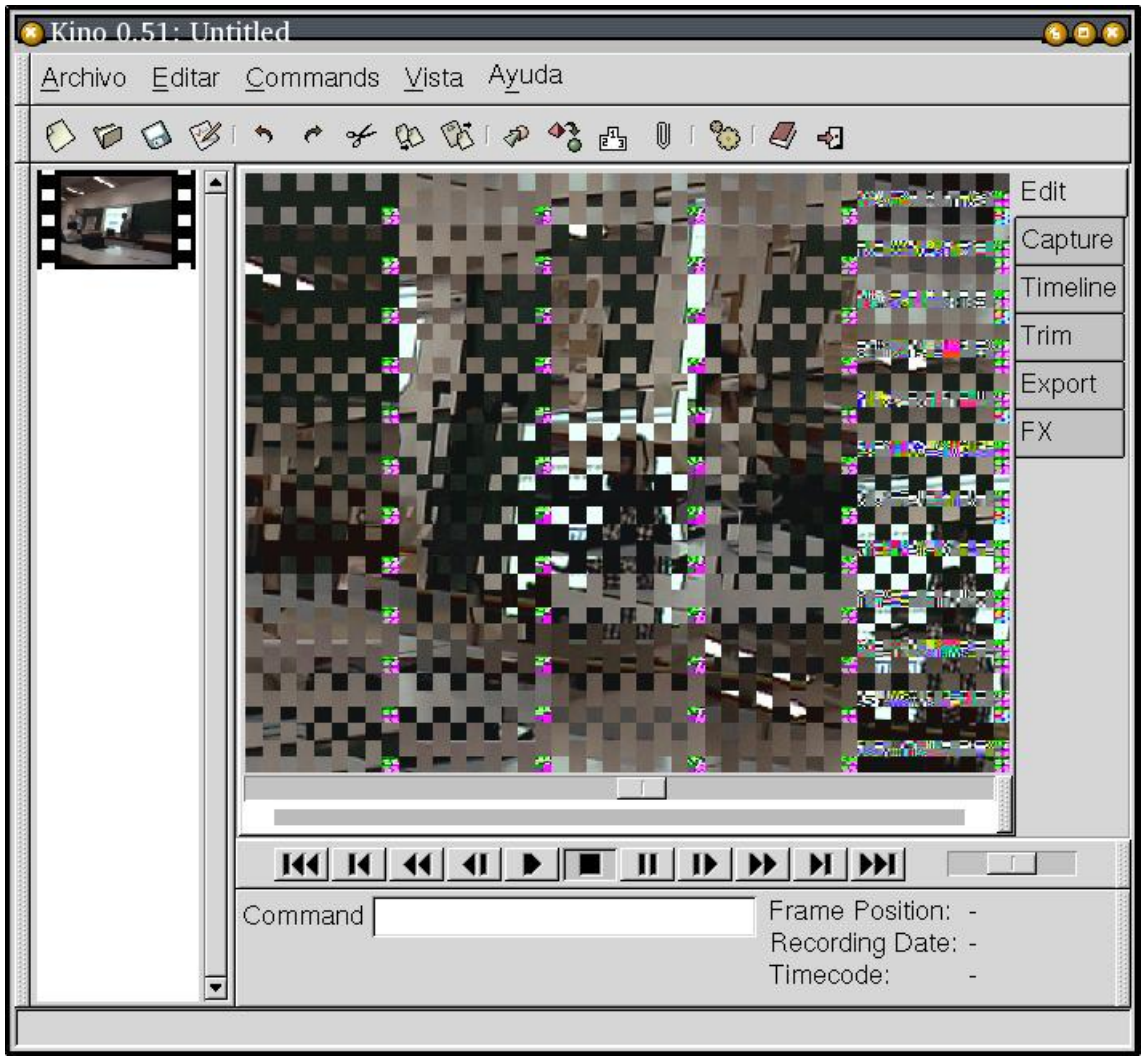
6.2. Navegación por el vídeo en bruto

Tras arrancar kino y pasar a reproducir el fichero de vídeo de 10 minutos, nos damos cuenta de que pasados siempre unos 30 segundos, kino no es capaz de trabajar bien con este fichero tan grande para la edición. Debemos de recortarlo y por ejemplo, tratar con la edición de ficheros de 1 o 2 minutos. Vamos a comenzar de forma modesta, con un fichero de 1 minutos, y ya iremos conociendo el tamaño óptimo de edición. Si lo que queremos es pasar todo el bruto a otro formato, MPEG4 por ejemplo, entonces si es útil con dvgrab capturar grandes secuencias de vídeo.

```
dvgrab --format raw --frames 1440 jesus
```

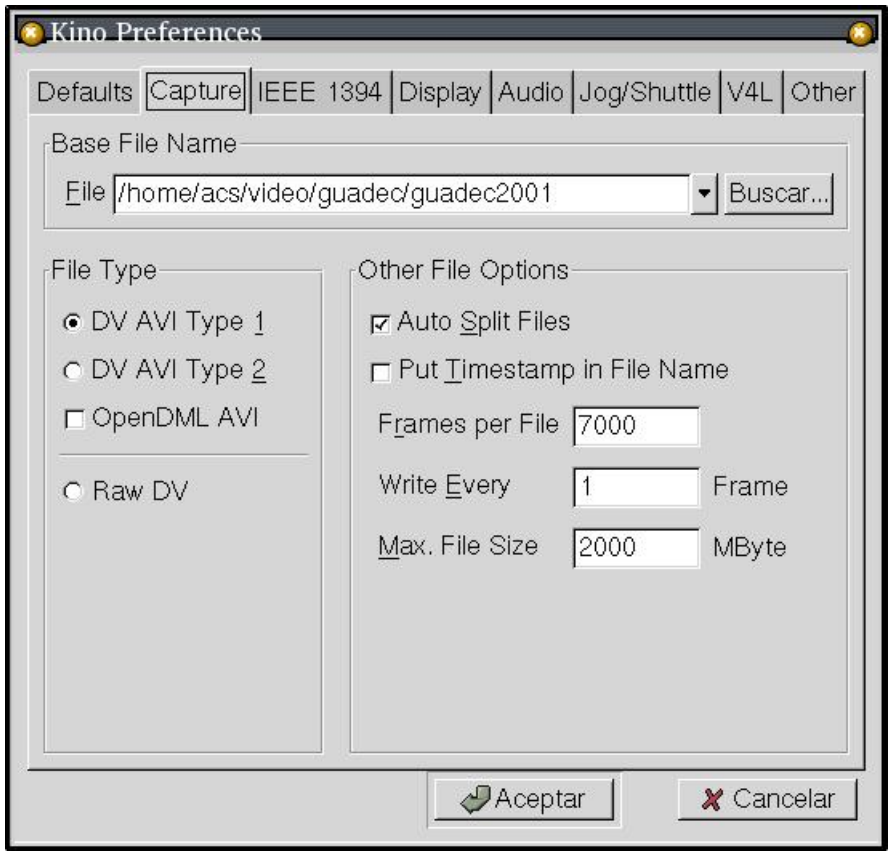
Tras 1 minutos de grabación ya tenemos nuestro nuevo fichero de trabajo el cual podemos abrir con kino y reproducirlo por completo. De nuevo los mismos problemas se repiten al intentar reproducir el vídeo. Tras unos segundos, se ensucia la imagen de vídeo y se pierde la sincronización con el sonido. El momento donde comienzan los problemas es siempre el mismo por lo que incluso el problema podría estar en la captura que hemos hecho desde dvgrab.

Figura 9. Algunos problemas en la reproducción



Capturando desde Kino en formato DV1 o RAW no tenemos ningún problema, por lo que parece que son problemas entre la captura que realiza dvgrab y Kino, por lo que por el momento, nos quedamos capturando de forma integra desde Kino. Es más, la captura desde Kino puede resultar mucho más cómoda para trabajar ya que la cámara siempre se queda detenida en el justo instante en el que dejamos de capturar. Podemos editar todo lo capturado en Kino, material que hemos podido ir visualizando según se capturaba, y una vez finalizada la edición de esa captura, volvemos a la interfaz de captura y volvemos a capturar otra parte.

Figura 10. Preferencias de captura en Kino

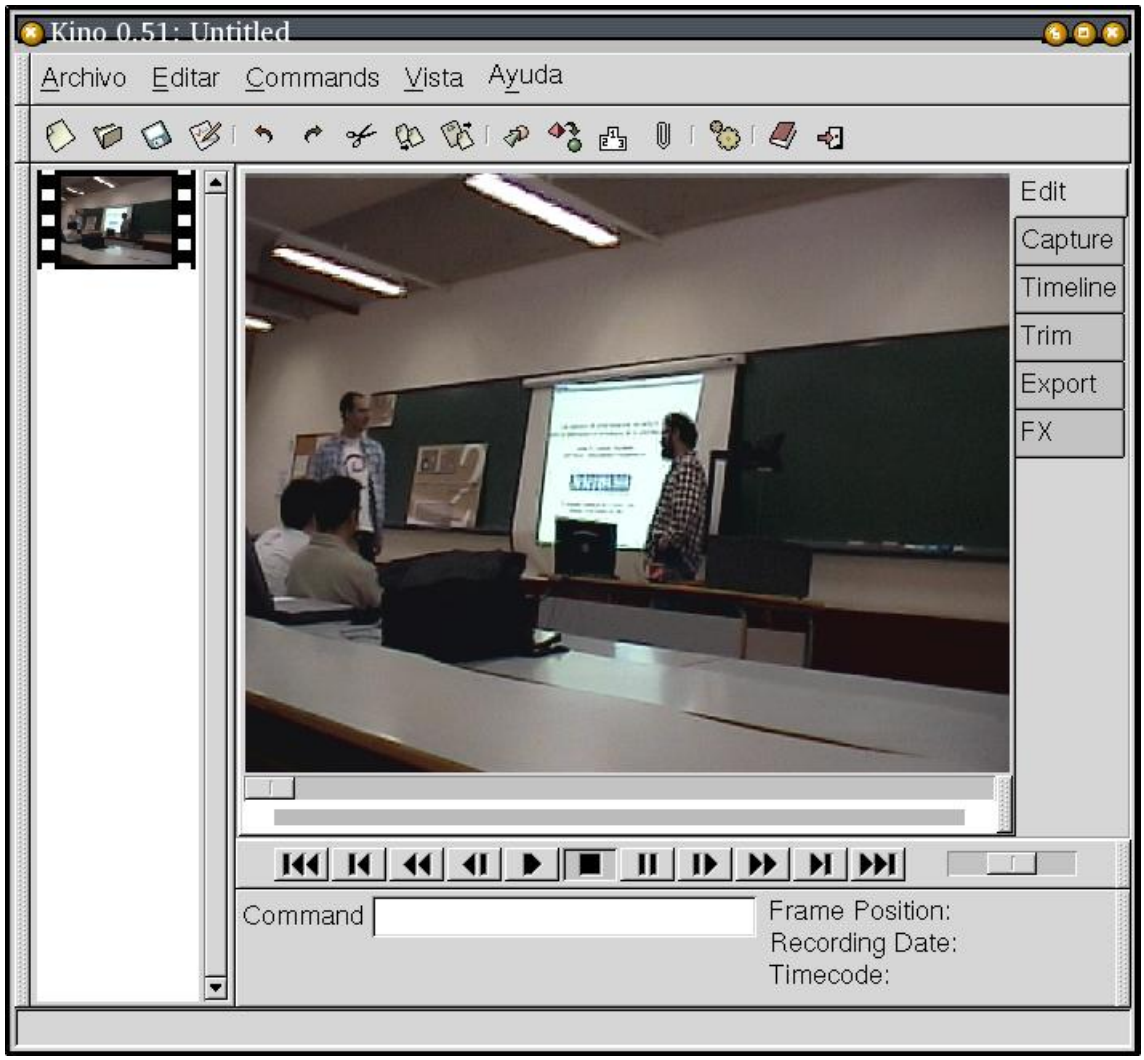


Lo primero es conocer los contenidos a editar para saber que meter y que quitar de la secuencia. Con la metodología de edición que nos hemos propuesto seguir, esto significa ir viendo lo que hemos capturado, esperar a que surjan 3 o 4 secuencias de vídeo que nos interese incluir en el producto final, parar la captura, y pasar a extraer estas partes del vídeo de la secuencia capturada.

Comenzamos capturando 1 minutos y 40 segundos, de los que hay que sacar en limpio la idea de que las patentes software afectan a todo tipo de software y en especial, suponen un grave riesgo para el software libre. Para ello nos vamos a la pantalla de edición y comenzamos con el trabajo.

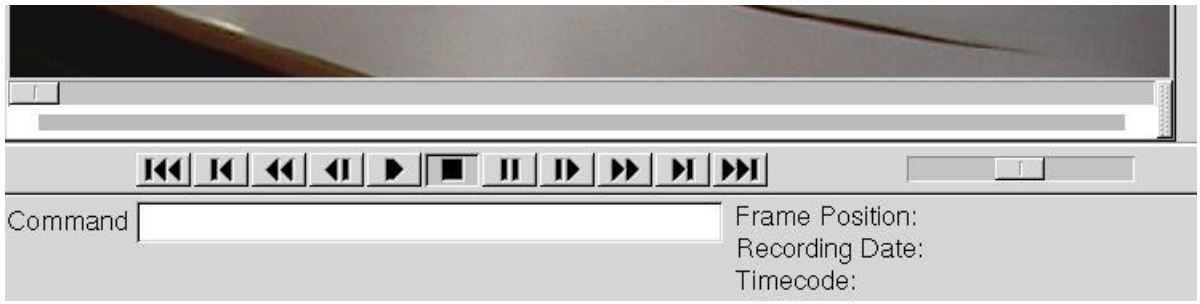
La cinta en la cámara de vídeo se queda justo tras la última imagen que hemos editado, por lo que una vez que hayamos acabado con los 1m40s que hemos capturado y ya los tenemos editados, volvemos a pulsar sobre el botón de capturar y repetimos el proceso con la nueva escena capturada. Asi hasta que cubramos toda la cinta.

Figura 11. Comienzo de edición del vídeo



Para poder navegar por los contenidos que acabamos de capturar, disponemos de la barra de navegación, similar a la que nos encontramos en cualquier equipo de reproducción, y que permite: reproducir, parar, pausar, movernos adelante o atrás una imagen o secuencia, avanzar o retroceder rápidoe ir al principio y al fin del material. Junto a este barra, nos encontramos con el control de "juggle" que nos permite movernos de forma rápida en un entorno localizado del vídeo, algo muy útil para localizar escenas exactas.

Figura 12. Barra de navegación en el vídeo



En la versión actual de Kino aún no disponemos de los datos sobre la imagen actual, la fecha de grabación y el código de tiempos, algo que sin duda nos vendría muy bien para manejarnos más rápido por la secuencia original.

6.3. Delimitación de secuencias (escenas)

Tenemos capturada una secuencia en bruto en la cual queremos delimitar algunas partes, para convertirlas en secuencias independientes, las cuales luego podremos ir pegando para obtener la composición con el video que nos interesa.

En nuestro caso al ser el vídeo siempre muy parecido, lo que vamos a utilizar para detectar cuando comienza lo que nos interesa es el audio. Aunque es justo después de un zoom cuando empieza el primer trozo interesante, por lo que nos puede servir como localizador rápido de la secuencia. Una vez en la zona, utilizamos el audio para situarnos exactamente donde queremos que empiece la nueva secuencia.

Una vez que estamos ya sobre el punto exacto, ha llegado el momento de poner algún tipo de marca que nos permita identificar este punto en el futuro.

Ha llegado el momento de comenzar a conocer todos los comandos de Kino, la mayoría de ellos accesibles sólo desde el teclado, y que nos van a permitir ir trabajando con nuestro vídeo.

6.4. Comandos fundamentales en Kino

Kino tiene un conjunto de equivalencias que es muy útil conocer y que relacionan el mundo de edición del vídeo con el mundo de la edición de textos.

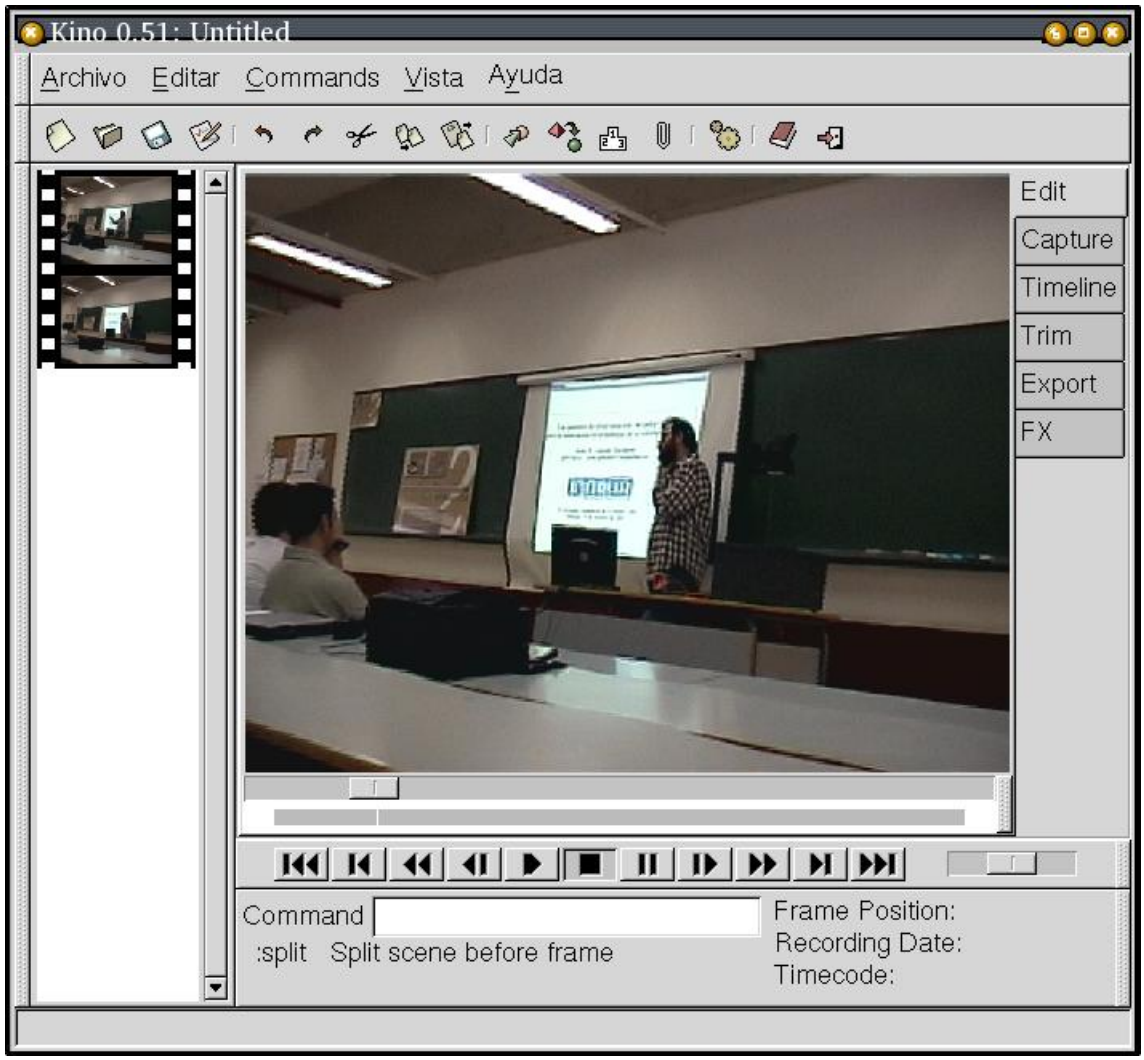
- Un caracter es una imagen
- Una palabra es un segundo
- Una línea es una escena
- El fichero es la película

Con estas ideas en mente ya podemos trabajar con la idea de que nuestro fichero de vídeo en bruto es un fichero en el cual sólo hay una gran línea sin ningún retorno de carro. Además, queremos hacer un resumen de ese documento y no incluir todo su contenido. Si estamos utilizando el "vi" una opción sería movernos por esa gran línea hasta el caracter (imagen) que comienza a tener contenido que nos interesa, pulsar retorno de carro (dividir escena con :split) para poder identificar con el comienzo de línea la zona que nos interesa incluir en el resumen (película editada), localizar el caracter (imagen) a partir del cual ya no nos interesa lo que sigue inmediatamennte, y volver a pulsar el retorno de carro (:split) para obtener en una línea (escena) parte de la información que nos interesa editar.

Lo que queda antes de esa línea (escena) que nos interesa lo podemos borrar, y lo que queda después, lo utilizaremos para repetir el proceso anterior hasta que se termine el fichero (película). Como resultado final obtendremos un conjunto de líneas (escenas), que formarán el resumen (película editada).

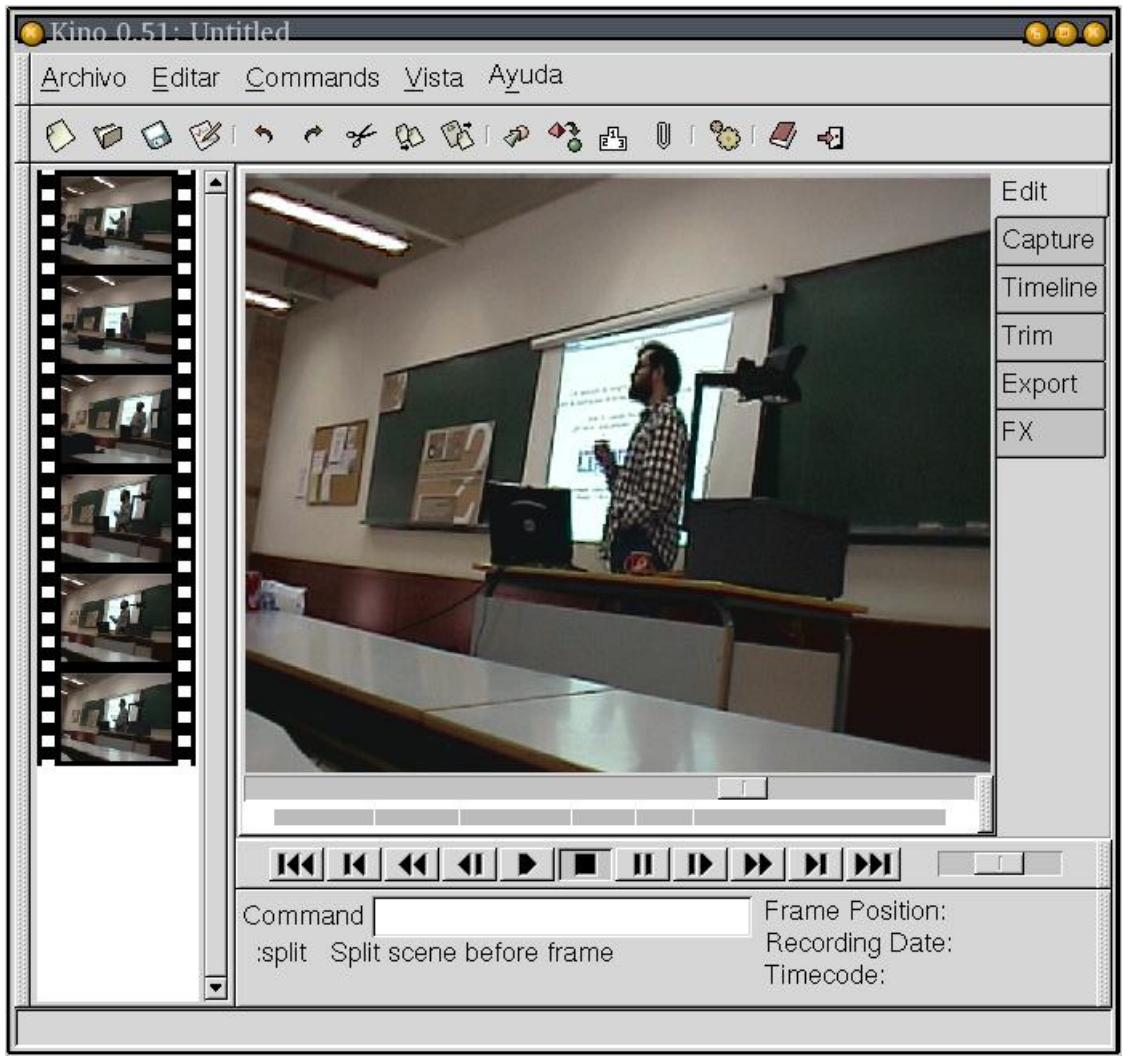
Acabamos de definir una metodología de edición. Quizá se pueda pensar en otras, pero esta nos puede valer de momento. Los comandos que vamos a utilizar son muy pocos, en realidad sólo uno, "split", que parte en dos secuencias una secuencia utilizando la imagen actual en la que nos encontramos.

Figura 13. División de la secuencia original en dos



Si repetimos este proceso, eliminando las partes que no nos interesan, finalmente nos quedará un conjunto de escenas que formarán la película editada.

Figura 14. División en 6 escenas del vídeo a editar



Lo normal es que la primera y última escenas siempre las eliminemos, ya que serán el principio hasta que llega una parte que nos interesa y el trozo que va desde la última escena interesante hasta el final de la grabación.

De las otras 4 escenas restantes, nos interesan siempre la primera y la última. Y entre las intermedias, pues la secuencia será de una que nos interesa, otra que no, asi sucesivamente y siempre igual. El que todo esto se repita así, nos puede permitir en el futuro realizar ediciones muy rápidas, en las que iremos delimitando las zonas que nos interesan siguiendo este método y que luego, mediante algún tipo de plugin, se realice

todo el proceso de forma automática.

Por lo tanto, eliminamos las secuencias 1,3 y 6. Para ello basta con que nos pongamos encima de ellas en la barra lateral y pulsemos CTRL-X o "dd". Con ello ya nos quedarán las 3 secuencias que nos interesan de estos primeros minutos capturados.

6.5. Composición de secuencias

La composición de las secuencias la obtenemos con el método anterior de edición de forma automática. Las escenas se siguen unas a otros con transiciones entre ellas que pueden resultar algo bruscas, un problema que resolveremos utilizando los efectos especiales (FX) que veremos en la siguiente sección.

6.6. Primer producto final

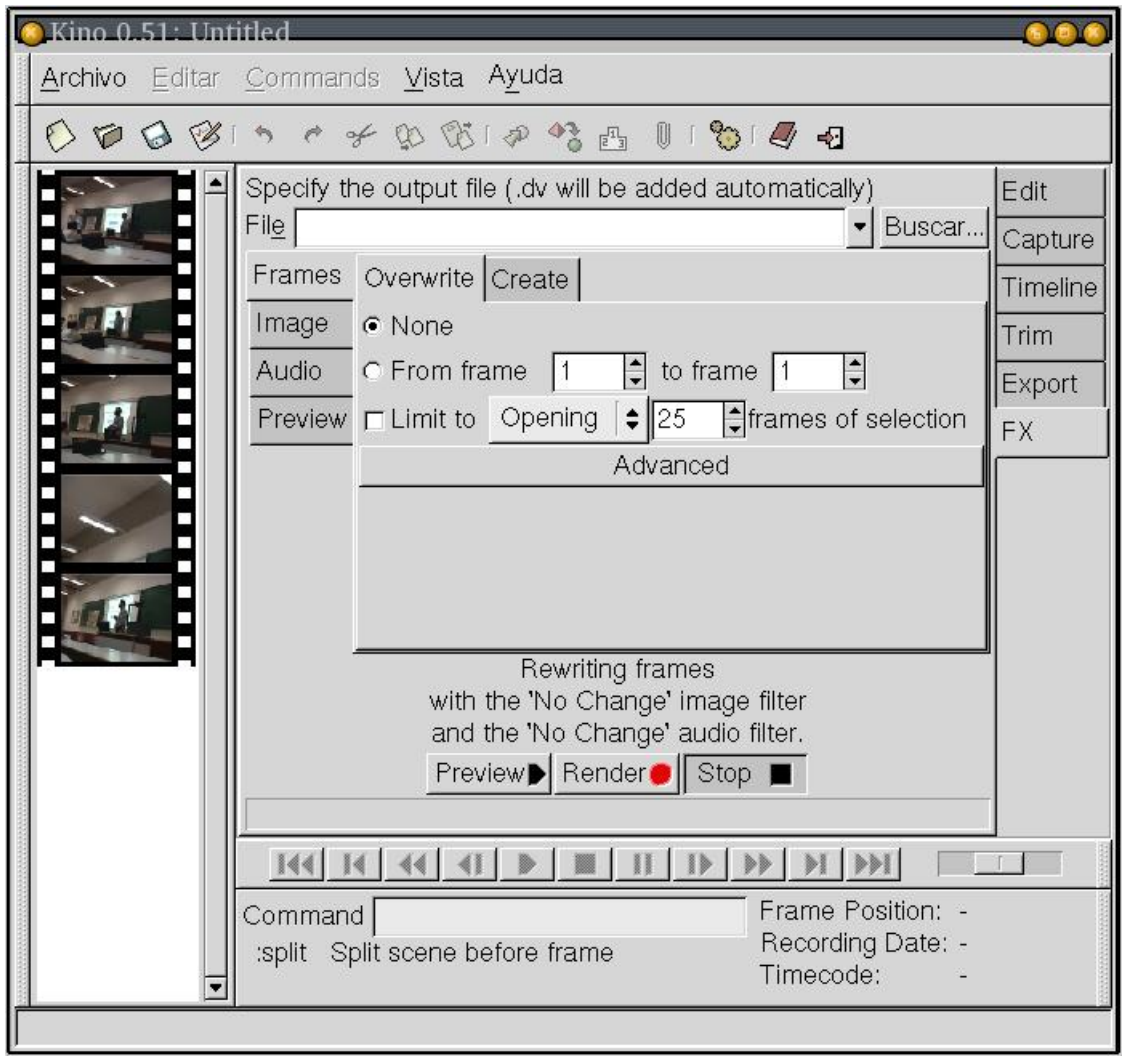
Ya hemos logrado nuestro primer montaje con Kino, el cual podemos grabar. Lo que en realidad se guarda es eun fichero en formato SMIL que hace referencia a los puntos de entrada y salida de cada una de las secuencias dentro del fichero original sobre el que estábamos trabajando. Si lo que queremos es guardar el vídeo ya montando, deberemos de exportar el montaje tal y como veremos en una sección posterior.

```
<?xml version="1.0"?>
<smil xmlns:smil2="http://www.w3.org/2001/SMIL20/Language">
  <seq>
    <video src="/home/acs/devel/web-xml/articulos/kino/jesus001.avi" clipBeg
  </seq>
  <seq>
    <video src="/home/acs/devel/web-xml/articulos/kino/jesus001.avi" clipBeg
  </seq>
  <seq>
    <video src="/home/acs/devel/web-xml/articulos/kino/jesus001.avi" clipBeg
  </seq>
  <seq>
    <video src="/home/acs/devel/web-xml/articulos/kino/jesus001.avi" clipBeg
  </seq>
</smil>
```

7. Efectos especiales en el montaje

Llegamos a una de las partes más interesantes y que más están evolucionando en la actualidad en kino: los efectos especiales. Estos efectos nos permiten realizar modificaciones sobre las secuencias de imágenes y sobre el audio, permitiendonos por ejemplo que la unión entre dos secuencias pase a ser un fundido en vez de un corte brusco, o que cierta secuencia se presente en blanco y negro o en sepia. Veamos como utilizar los efectos especiales, para lo que nos vamos a la solapa "FX" de la interfaz.

Figura 15. Efectos especiales en Kino

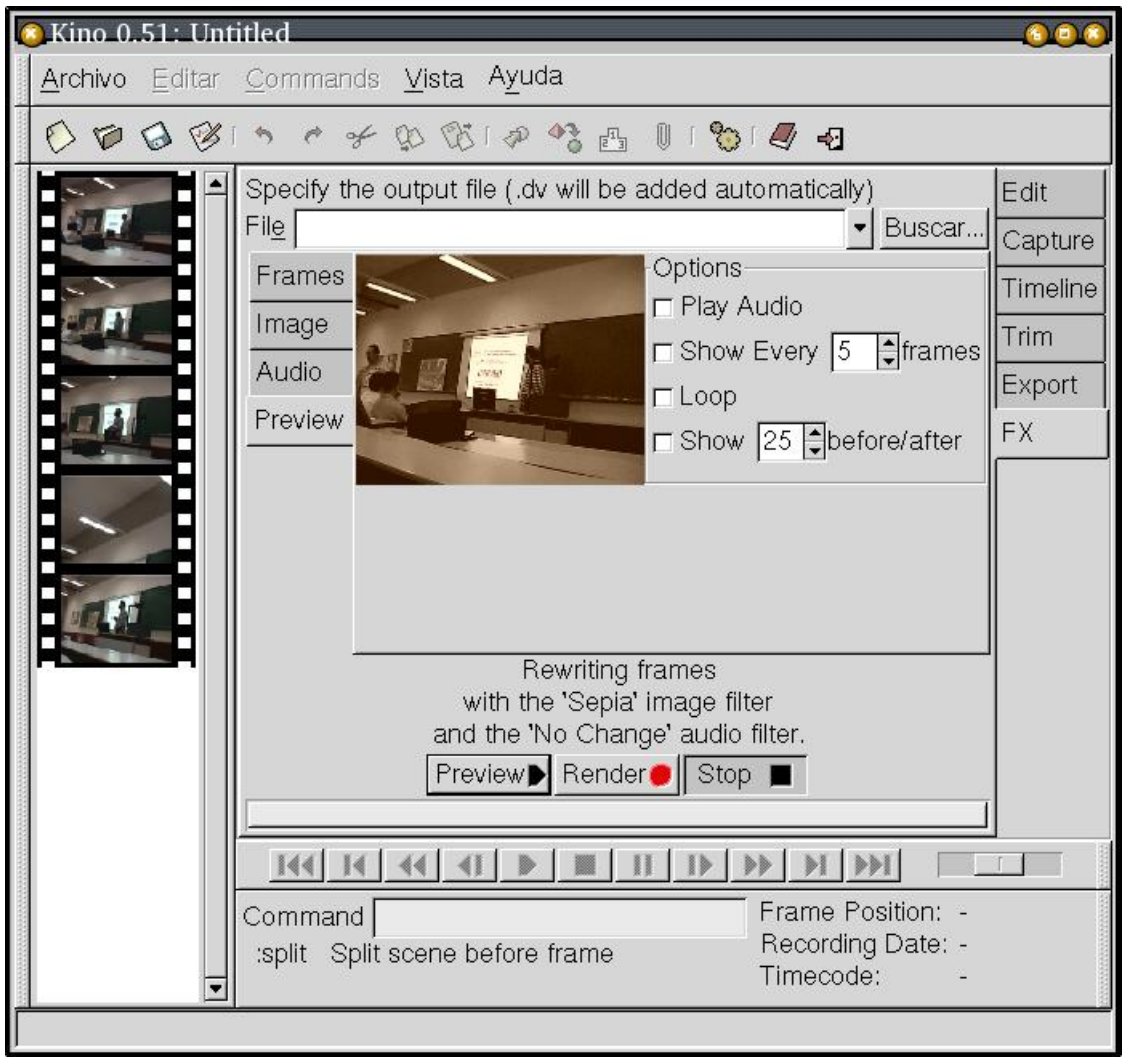


Disponemos de 4 solapas dentro de los efectos especiales. Por un lado nos permite la primera seleccionar si el resultado de los efectos va a sustituir a imágenes que ya existían o bien se va a crear una nueva secuencia en una posición determinada.

La siguiente solapa "Image" nos permite especificar el tipo de efectos que le vamos a aplicar a la imagen. Tenemos filtros que actúan por igual en toda la secuencia del efecto y que nos permiten poner la imagene en color sepia, en blanco y negro, invertir el vídeo, crear un espejo, un kaleidoscopio o hacer un swap de la imagen.

Por ejemplo, podemos poner en sepia desde la imagen 1 hasta la 50.

Figura 16. Video en color sepia

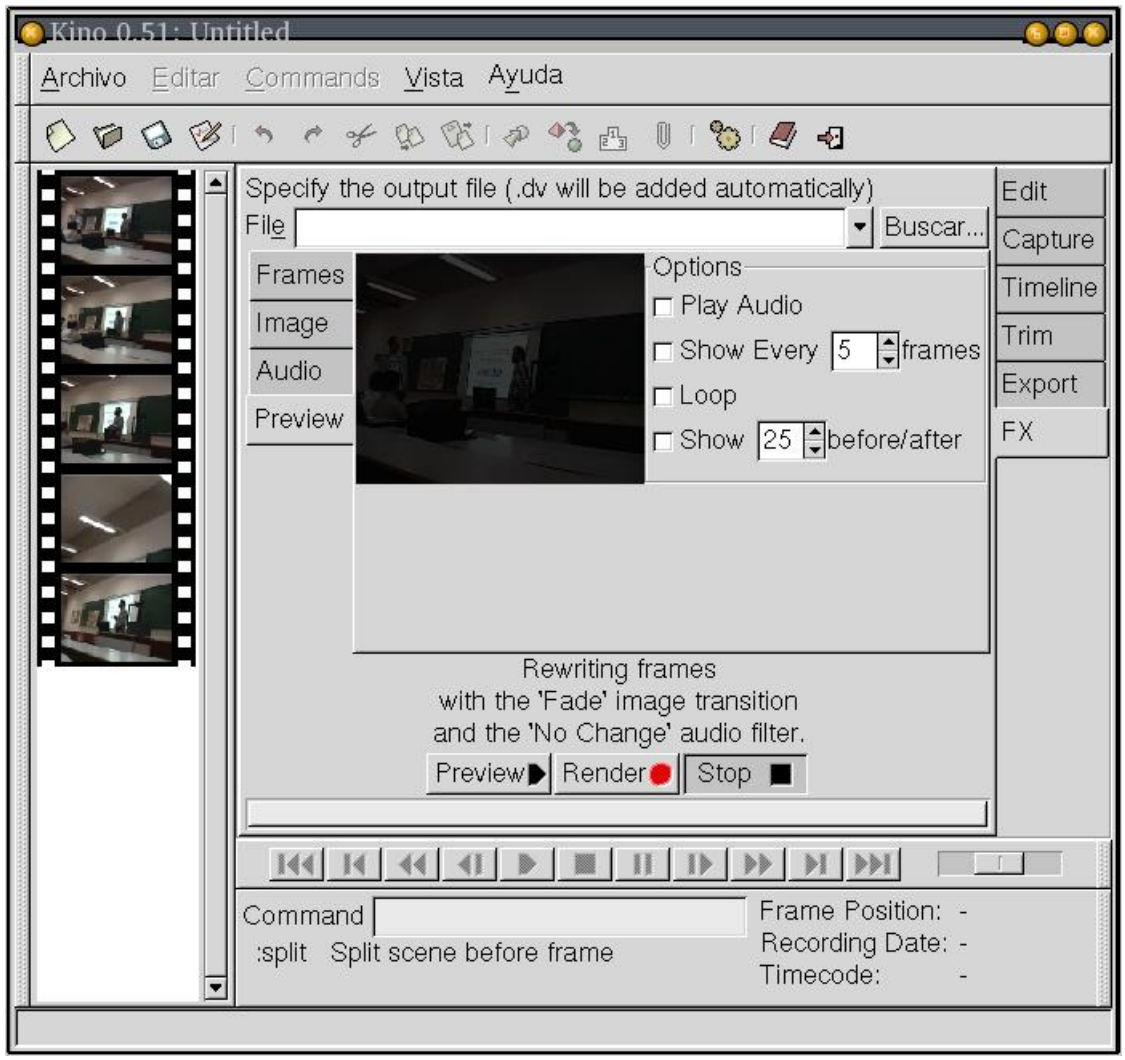


o algo más espectacular como puede ser hacer un espejo del vídeo

Figura 17. Video con espejo

Respecto a las transiciones, podemos realizar los típicos fundidos y cortinillas tan habituales en el mundo de la imagen. Podemos ver una captura momentos antes de terminar el fundido a negro.

Figura 18. Fundido a negro



Con respecto al audio, podemos silenciarlo, y fundirlo a silencio o desde silencia. Además, dentro de las transciones, disponemos de posibilidades muy interesantes como jugar con los dos canales de audio que llevan muchos vídeos DV, poder añadir un canal a modo de voz en off o incluso poder mezclar una voz en off con el propio sonido de la imagen.

hay que destacar la velocidad de las previsualizaciones que realiza Kino, que nos permiten trabajar de forma muy rápida y ajustar los efectos a nuestras necesidades.

Una vez que hemos finalizado con las previsualizaciones hacemos el "render" real del

efecto, el cual se quedará almacenado en un nuevo fichero que podremos incorporar a nuestro montaje. La verdad es que al hacer el render definitivo Kino se ha mostrado un poco inestable. Y es que esta parte de efectos lleva poco tiempo dentro de la herramienta y está poco probada.

Es de esperar que en un futuro próximo dispongamos de efectos como la rotulación de ciertas escenas.

8. Captura de vídeo analógico

En las últimas versiones de Kino se ha incorporado un nueva pestaña de captura de vídeo, en este caso utilizando los dispositivos video4linux Estos dispositivos son los que nos permiten comunicarnos con cámaras de vídeo de baja resolución, como las que se usan para videoconferencia, o con las tarjetas capturadoras de vídeo, que es el caso en el que nos vamos a centrar en este apartado.

En concreto, tenemos unas tarjeta capturadora que utiliza el chipset Bt848, que es el más común en todo este tipo de tarjetas, y que está perfectamente soportado en Linux. Al utilizar casi todos los fabricantes este chipset, tenemos la seguridad de que la gran mayoría de tarjetas van a funciona en GNU/Linux. Por otro lado, al ser las tarjetas que se usan para ver la TV en el computador, están muy extendidas y son baratas, estando su precio por debajo de los 60 euros. En la siguiente imagenes podemos observar la capturadora que vamos a utilizar.

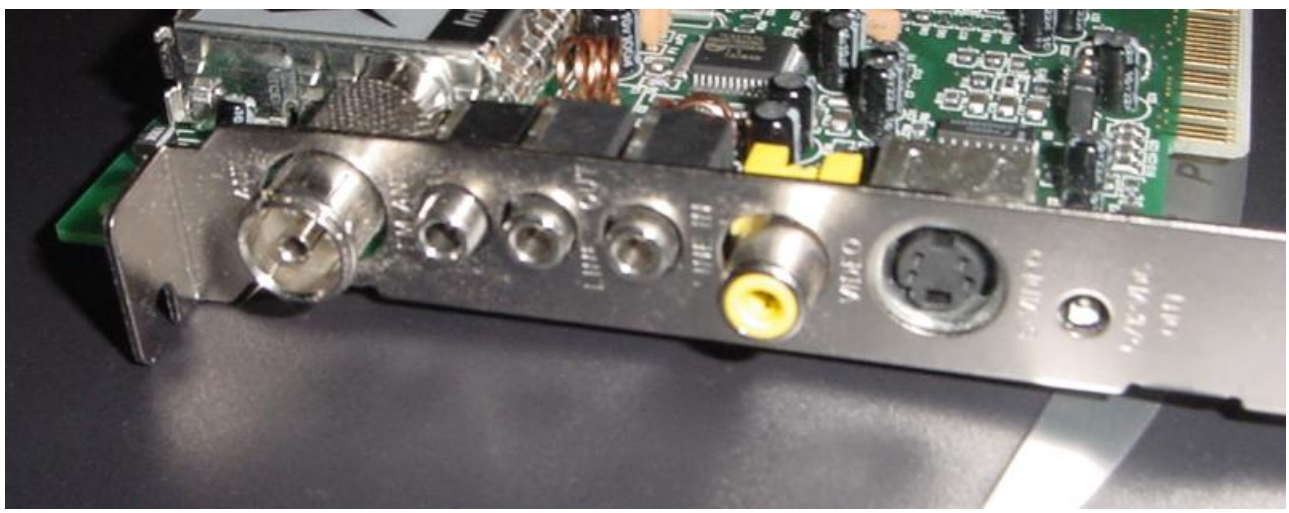
Figura 19. Tarjeta capturadora Bt848

Podemos ver el detalle del chipset Bt848

Figura 20. Chipset Bt848

Las entradas y salidas que proporciona la tarjeta de capturado que estamos usando las podemos ver en la siguiente imagen, y como vemos, dan mucho juego ya que nos permiten meter vídeo desde entrada análogica de señal de televisor tracicional por cable coaxial (no recuerdo el nombre: FIXME) o entrada de señal por SVIDEO.

Figura 21. Entradas y salidas de la tarjeta de video



Vamos a intentar utiliza la entrada de SVIDEO y volcar por ella la señal de nuestra

cámara digital de vídeo. Vemos que en el caso de no disponer ieee1394 en nuestra máquina podemos utilizar esta opción, aunque siempre la calidad será inferior y el esquema bastante más engorroso: digital-analógico-digital. Lo ideal es evitar este paso a analógico cuando ya los contenidos están grabados en digital.

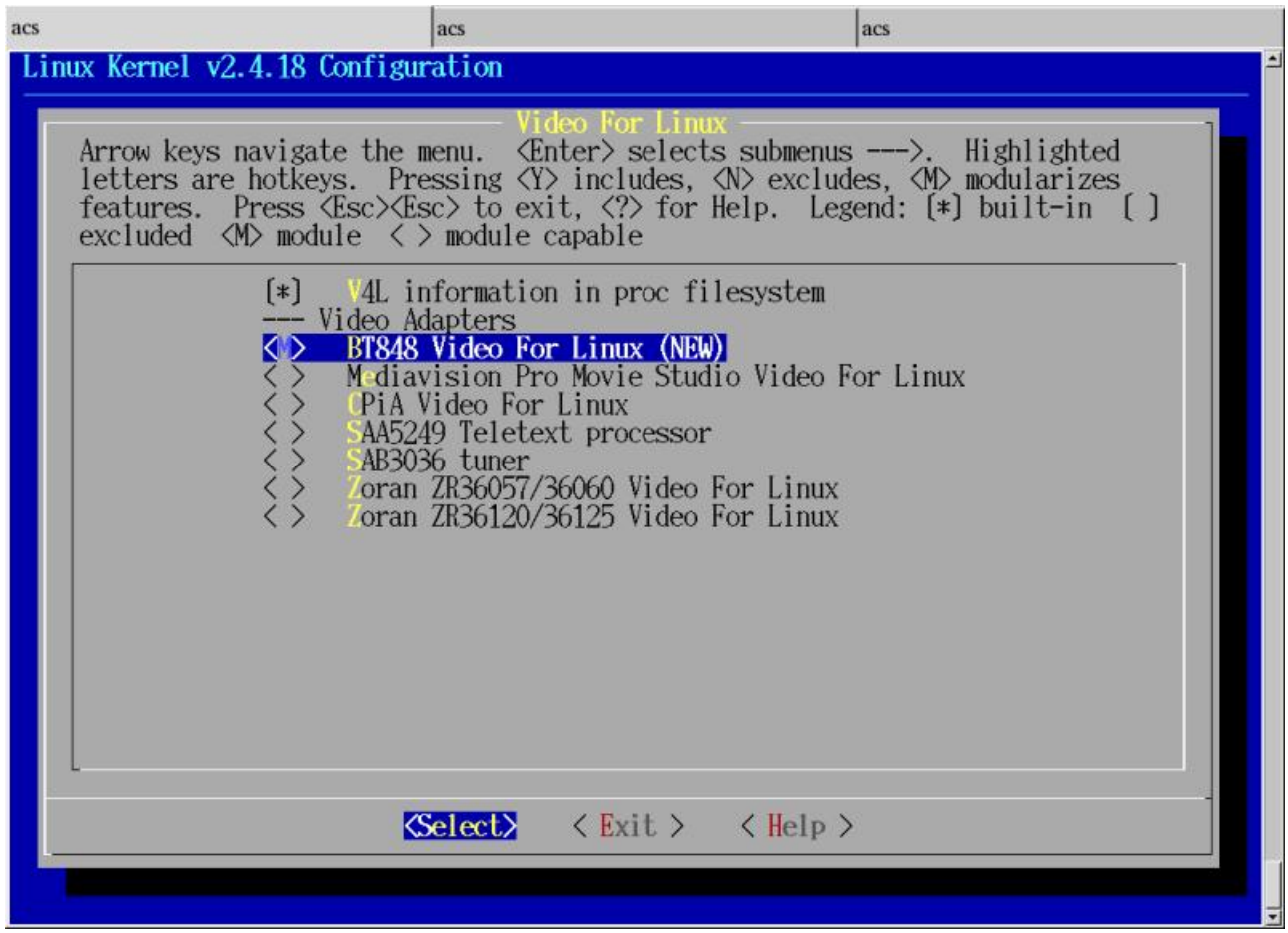
Figura 22. Conexión de la cámara de vídeo a tarjeta capturadora

Ya tenemos montado todo a nivel de hardware pero necesitamos ahora dar soporte al controlador bt848 del núcleo de GNU/Linux y ver si realmente el sistema reconoce a la tarjeta. Vemos que en principio, está bien pinchada y es reconocida como un dispositivo PCI.

```
      tunel:/usr/src# lspci
...
00:0c.0 Multimedia video controller: Brooktree Corporation Bt848 TV with DMA
h (rev 12)
...
```

Ahora nos vamos al núcleo y compilamos el núcleo bttv si es que no lo teníamos ya. Para poder compilar el módulo de bttv necesitamos ir primero a la opción de "Character Devices", dentro de ella a "I2C" y allí activar los módulos de "I2C support" e "I2C bit-banging interfaces". Tras ello nos vamos dentro del menú principal a "Multimedia devices" y allí activamos "Video For Linux" y dentro del menú de "Video For Linux" activamos "BT848 Video For Linux (NEW)".

Figura 23. Conexión de la cámara de vídeo a tarjeta capturadora




```
make dep clean bzImage module modules_install
```

Una vez finalizado el proceso configuramos el nuevo núcleo y arrancamos con él el sistema. Recordad al lector que es muy probable que ya tenga en su propio núcleo este controlador, por lo que bastará con dar el siguiente paso:

```
modprobe bttv
```

Aquí llegamos a la que es sin duda la parte más compleja del uso del controlador de bt848. Como dijimos anteriormente, tenemos la gran ventaja de que ese controlador viene en la mayoría de tarjetas de captura de vídeo, pero cada una de esta tarjetas usa

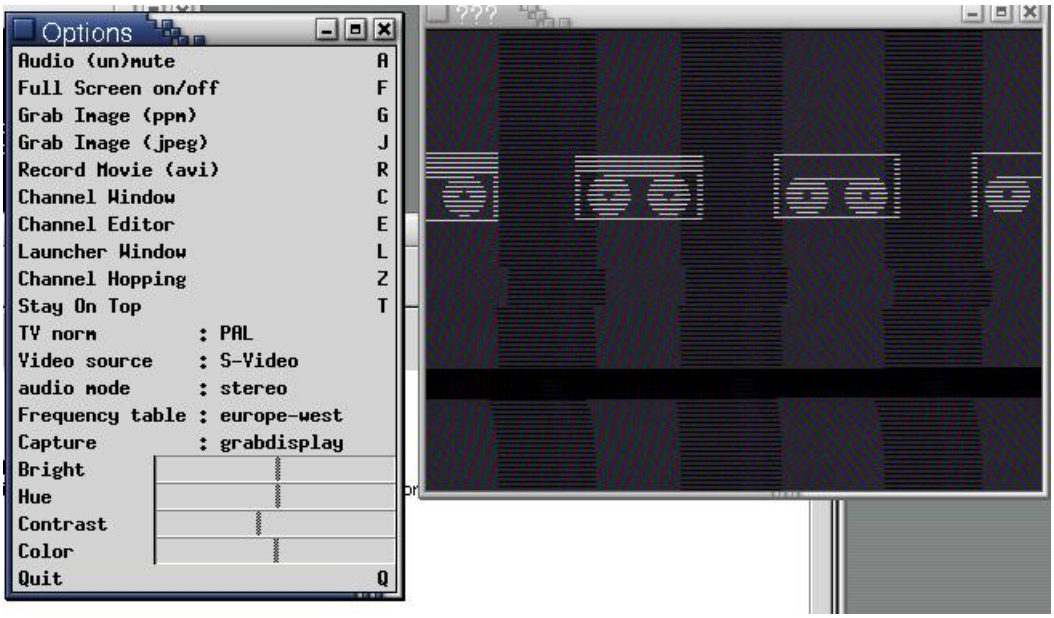
configuraciones diferentes para el sintonizador de la señal de vídeo y audio, por lo que tendremos que jugar con todas las posibles opciones hasta que logremos capturar la señal de vídeo de forma correcta.

Nada más cargar el módulo bttv dentro del fichero "/var/log/messages" tenemos información sobre el tipo de tarjeta que se ha detectado. Veámoslo:

```
Oct 26 11:22:54 tunel kernel: bttv: driver version 0.7.83 loaded
Oct 26 11:22:54 tunel kernel: bttv: using 2 buffers with 2080k (4160k total)
Oct 26 11:22:54 tunel kernel: bttv: Host bridge is Intel Corp. 440LX/EX - 82
Oct 26 11:22:54 tunel kernel: bttv: Host bridge needs ETBF enabled.
Oct 26 11:22:54 tunel kernel: bttv: Bt8xx card found (0).
Oct 26 11:22:54 tunel kernel: PCI: Found IRQ 9 for device 00:0c.0
Oct 26 11:22:54 tunel kernel: bttv0: Bt848 (rev 18) at 00:0c.0, irq: 9, late
Oct 26 11:22:54 tunel kernel: bttv0: using: BT848A( *** UNKNOWN/GENERIC **)
Oct 26 11:22:54 tunel kernel: bttv0: enabling ETBF (430FX/VP3 compatibilty)
Oct 26 11:22:54 tunel kernel: tuner: chip found @ 0xc0
Oct 26 11:22:54 tunel kernel: bttv0: i2c attach [client=(unset),ok]
Oct 26 11:22:54 tunel kernel: i2c-core.o: client [(unset)] registered to ada
Oct 26 11:22:54 tunel kernel: i2c-core.o: adapter bt848 #0 registered as ada
Oct 26 11:22:54 tunel kernel: bttv0: i2c: checking for MSP34xx @ 0x80... not
Oct 26 11:22:54 tunel kernel: bttv0: i2c: checking for TDA9875 @ 0xb0... not
Oct 26 11:22:54 tunel kernel: bttv0: i2c: checking for TDA7432 @ 0x8a... not
```

Vemos que la tarjeta la ha detectado por lo que pasamos a utilizar xawtv para ver que se recibe por la tarjeta, aunque la autodetección del tipo de tarjeta no parece que ha ido muy bien por lo que podemos tener problemas a la hora de sintonizar la tarjeta. Probemos con xawtv a ver que se recibe pues.

Figura 24. Recepción de la señal por el puerto SVIDEO en xawtv



Estupendo, la señal se está recibiendo, aunque como era de esperar, no se sintoniza de forma correcta. Pero hemos sido capaces de cambiar la entrada de vídeo en xawtv para que lo cogiera de SVIDEO y ha funciona de forma correcta. Vamos ahora a jugar con los parámetros a pasarle a bttv a ver si logramos una mejor sintonización. Además del módulo bttv, es importante también el módulo "tuner" para poder sintonizar de forma correcta los canales.

Leyendo la lista de posibles tarjetas, vemos que la 16 es la "16: Aimslab VHX" que es precisamente el nombre que aparecía en al caja de la tarjeta por lo que parece que es la elección correcta.

```
insmod bttv card=16
```

Si miramos ahora el fichero de registro "/var/log/syslog"

```
Oct 26 11:39:55 tunel kernel: bttv: driver version 0.7.83 loaded
Oct 26 11:39:55 tunel kernel: bttv: using 2 buffers with 2080k (4160k total)
Oct 26 11:39:55 tunel kernel: bttv: Host bridge is Intel Corp. 440LX/EX - 82
Oct 26 11:39:55 tunel kernel: bttv: Host bridge needs ETBF enabled.
Oct 26 11:39:55 tunel kernel: bttv: Bt8xx card found (0).
```

```
Oct 26 11:39:55 tunel kernel: PCI: Found IRQ 9 for device 00:0c.0
Oct 26 11:39:55 tunel kernel: bttv0: Bt848 (rev 18) at 00:0c.0, irq: 9, late
Oct 26 11:39:55 tunel kernel: bttv0: using: BT848A(Pixelview PlayTV (bt878)
Oct 26 11:39:55 tunel kernel: bttv0: enabling ETBF (430FX/VP3 compatibilty)
Oct 26 11:39:55 tunel kernel: i2c-core.o: adapter bt848 #0 registered as ada
Oct 26 11:39:55 tunel kernel: bttv0: i2c: checking for MSP34xx @ 0x80... not
Oct 26 11:39:55 tunel kernel: bttv0: i2c: checking for TDA9875 @ 0xb0... not
Oct 26 11:39:55 tunel kernel: bttv0: i2c: checking for TDA7432 @ 0x8a... not
Oct 26 11:39:55 tunel kernel: tvaudio: TV audio decoder + audio/video mux dr
Oct 26 11:39:55 tunel kernel: tvaudio: known chips: tda9840,tda9873h,tda9874
...
```

Vemos que se ha cargado el módulo con la opción que le hemos proporcionado. Ahora si nos vamos a xawtv a ver si se está capturando de forma correcta la señal

Figura 25. Captura correcta por SVIDEO de vídeo

Y podemos ya incluso probar a meterle vídeo de verdad a ver que tal es la calidad del mismo.

Figura 26. Captura de vídeo por SVIDEO

Podemos ahora intentar utilizar kino para capturar la señal de vídeo utilizando video4linux. Vamos a ver que tal funciona, ya que es algo muy nuevo que ha sido incorporado en las últimas versiones.

Y vamos por último con las pruebas de captura de vídeo analógico emitido por un VCR, para lo que nos hace falta enchufar el vídeo a la capturadora y sintonizarlo.

Vemos a continuación las conexiones que hemos tenido que establecer entre el vídeo y la tarjeta capturada

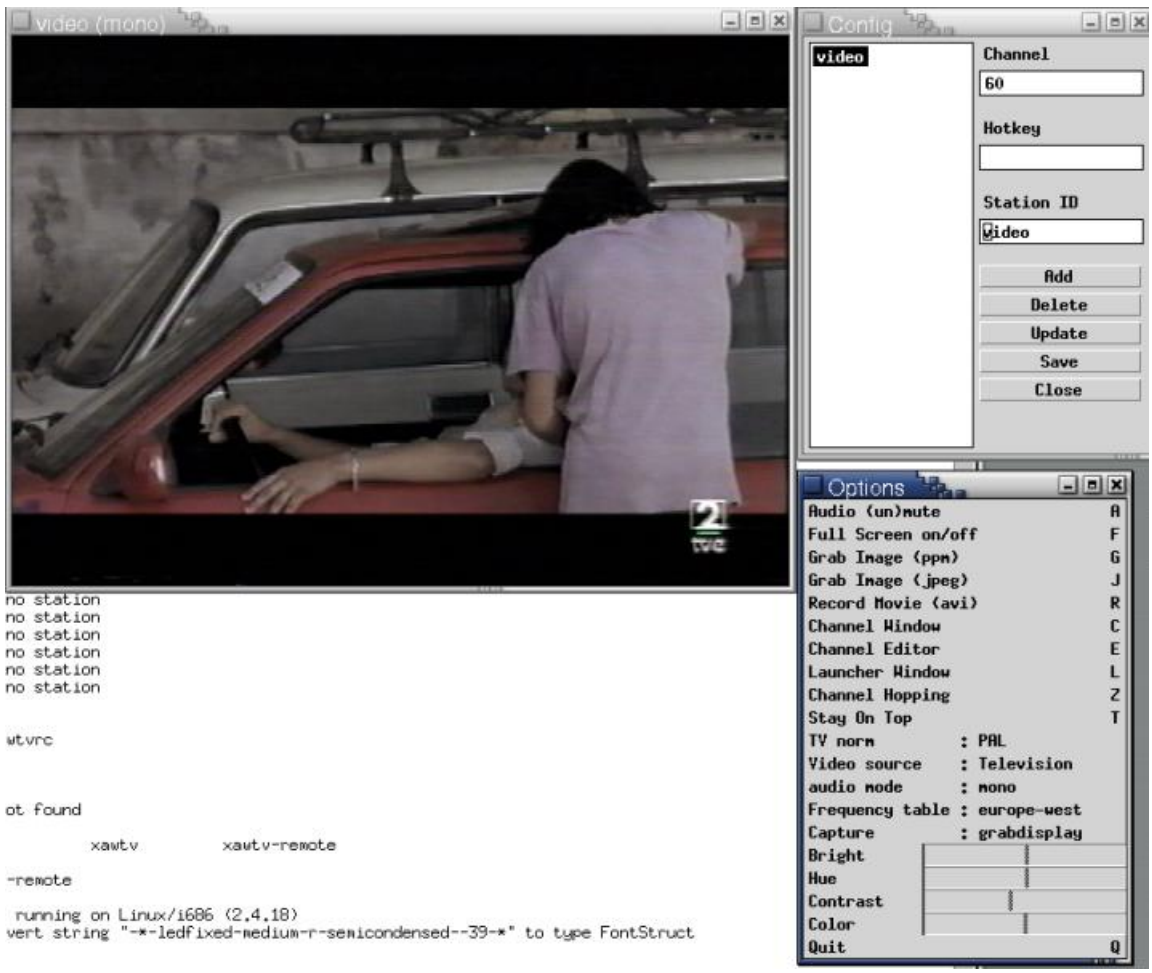
Figura 27. Conexión entre vídeo y capturadora

Lo siguiente es lograr sintonizar de forma correcta el canal del vídeo, algo que hemos logrado con un poco de pruebas ya que el "tuner" por defecto no era el de la tarjeta capturadora. Tras modificar el módulo "tuner" y cargarlo con:

```
modprobe tuner type=1
modprobe bttv card=16
```

ya hemos podido sintonizar el canal del vídeo VCR, que ha resultado ser el canal 60, como podemos ver en la siguiente captura donde se ve un fotograma de la película de "Barrio", emitida por la TVE2.

Figura 28. Captura desde un VCR



Para capturar los canales de televisón, bastará con enchufar el cable coaxial con la señal con las canales que queramos ver, y luego vamos barriendo todos los canales y les vamos poniendo nombres :)

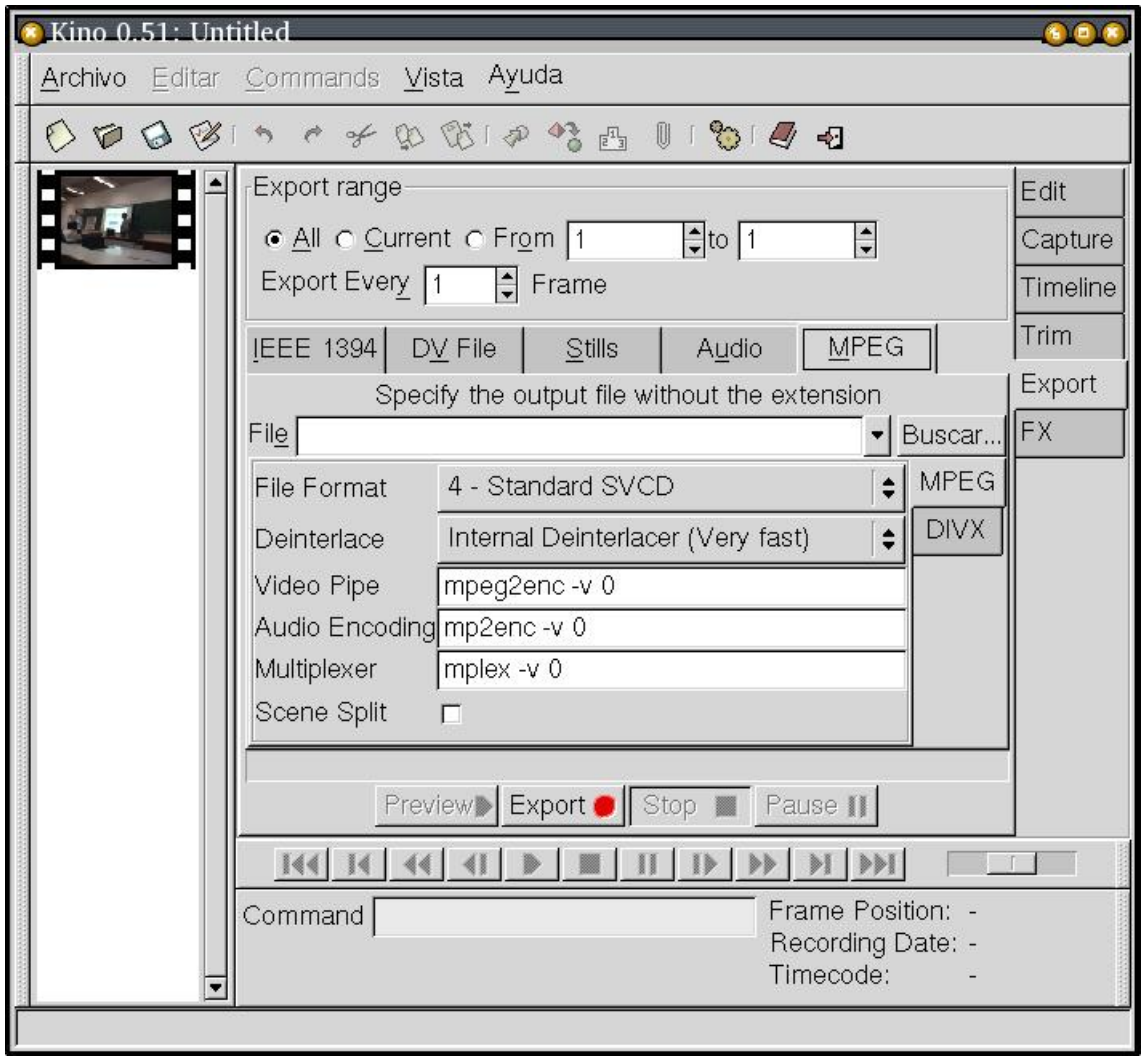
Para terminar, sería muy interesante disponer de algún programa que nos permitiera grabar la señal que se captura en la tarjeta para luego poderla editar. Desde kino lo podemos hacer, pero existen programas similar a dvgrab pero que capturan de la interfaz de vídeo. Las utilidades que podemos probar son "bttvgrab", "vstream", "ffmpeg" o "vcr".

Tras una corta investigación, y viendo que existe paquete pare debian, he comenzando las pruebas con "vcr".

9. Expotar un montaje a un fichero de vídeo

El propio kino dispone de herramientas para exportar el trabajo realizado a formatos de consumo, como puede ser MPEG2, audio en formato wav o Ogg o imágenes estáticas de toda la secuencia.

Figura 29. Exportación de secuencias



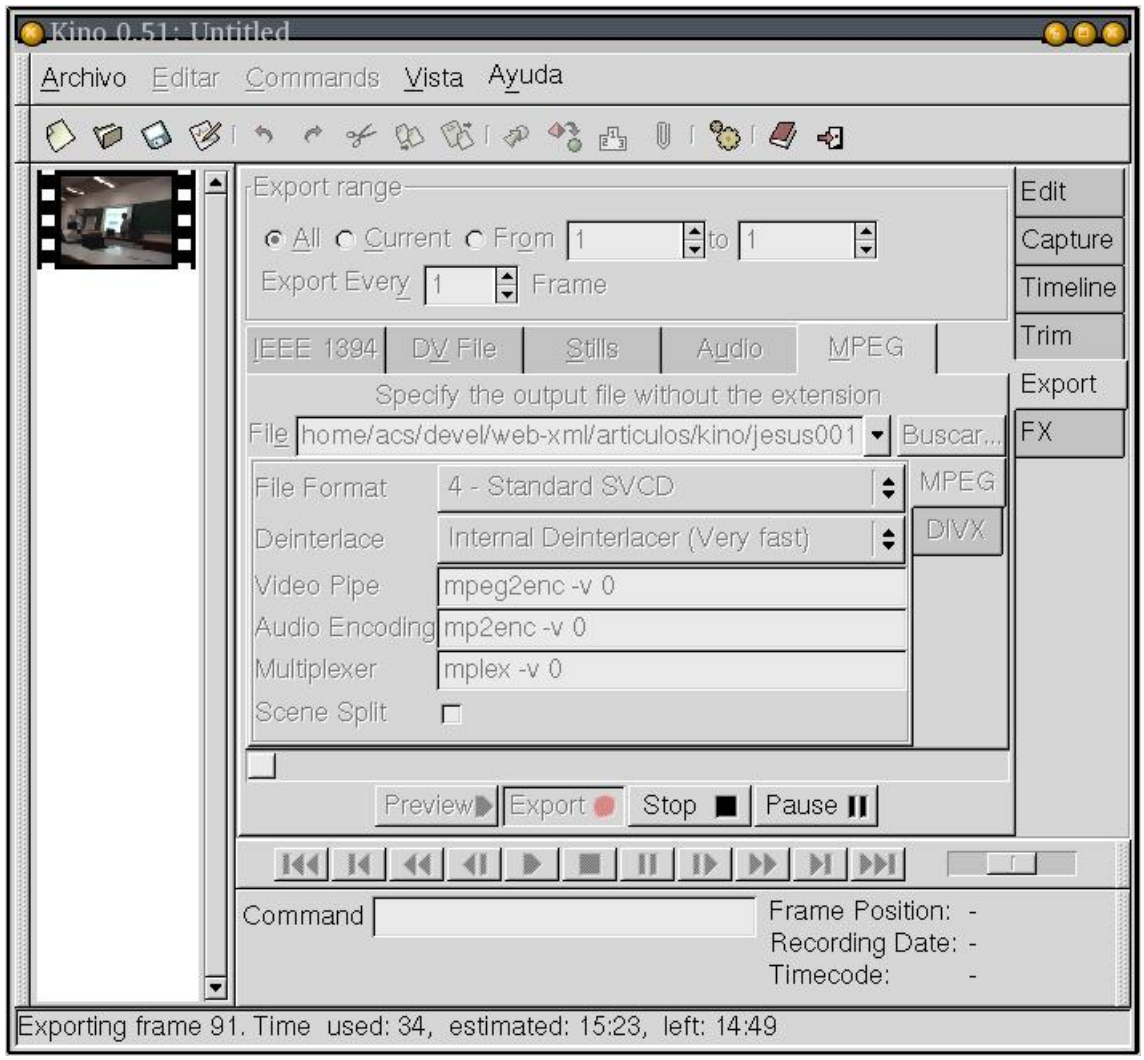
Vemos que por ejemplo, utiliza las herramientas mpeg2enc para llevar a cabo la codificación a MPEG2. Estas herramientas son parte del paquete mjpegtools. Hay que andar con mucho ojo con los formatos de vídeo que utilizamos ya que hay muchas patentes software que los afectan de forma directa, y son un riesgo para el futuro de dichos contenidos.

```
linex:/home/acs# apt-get install mjpegtools
Reading Package Lists... Done
Building Dependency Tree... Done
```

```
The following extra packages will be installed:
  libmjpegtools0
The following NEW packages will be installed:
  libmjpegtools0 mjpegtools
...
```

Si probamos por ejemplo a exportar nuestro vídeo a SVCD de trabajo que en formato DV ocupa 35 MB pasamos a tener un fichero de . Eso sí, el proceso le ha llevado más de 20 minutos el llevarlo a cabo. Es interesante el formato final ya que la mayoría de reproductores de DVD actuales son capaces de reproducirlo.

Figura 30. Exportación de secuencia a SVCD



```
acs@linex:~/devel/web-xml/articulos/kino$ ls -l jesus001*.mpeg jesus*.avi
-rw-r--r--    1 acs      acs      32308248 oct 23 21:04 jesus001001.mpeg
-rw-r--r--    1 acs      acs      216785408 oct 23 06:24 jesus001.avi
```

La tasa de compresión ha sido de 7:1 por lo que un CD de 640 MB nos deberían de entrar 640*7 MB = 4480 MB, es decir, 4.5 GB que son algo más de 20 min de vídeo. La duración en concreto de este vídeo es de 100 segundos por lo que el ancho de banda que consume SVCD en este caso es de 320 KB/s, lo que nos lleva a unos 2000 segundos en un CD cd 640 MB, que son unos 33 minutos.

Este formato de vídeo ya podemos reproducirlo por ejemplo con mplayer. Aunque vemos que la relación de alto y ancho resulta un poco extraña.

Figura 31. MPlayer reproduciendo el SVCD

En cambio xine hace un estupendo trabajo

Figura 32. XINE reproduciendo el SVCD

El sonido de este formato es realmente bueno, mucho mejor que el que podíamos oir desde kino.

COmo prueba de concepto, vamos a probar a exportar a formato DVD y ver que tamaños resultan y el tiempo que se consume en la compresión. En este caso ha tardado la compresión 30 minutos y el tamaño final ha sido de:

```
acs@linex:~/devel/web-xml/articulos/kino$ ls -l jesus001001.mpeg
-rw-r--r--    1 acs      acs      54525952 oct 23 22:20 jesus001001.mpeg
```

La resolución de salida de la imagen ha sido los 720x576 originales de la captura del vídeo, con un excelente sonido estereo de 44100Hz 2 canales 16-bit.

Es de destacar que en la visualización de ambnos formatos, xine se ha mostrado más robusto y ha dado mejor sonido y señal que el mplayer, que utilizaba un codec del proyecto ffmpeg.

10. Conversión de DV a otros formatos: transcode

Una vez que hemos editado el vídeo en el formato DV con buena calidad, 720x576 por fotograma a 25 fps con 24 bits por color y sonido en formato AC3 con 48000 muestras por segundo, 16 bits por muestra y 2 canales de audio, ha llegado el momento de convertir este formato de edición en un formato de distribución. Ha dia de hoy el que 10 minutos de vídeo ocupen 2 GB de datos aún es demasiado por lo que vamos a codificar dicho vídeo en un formato comprimido. Podemos utilizar los compresores divx5, opendivx o compresores mpeg o mjpeg.

Vamos a mostrar en este ejemplo como se convierte de DV a DivX5.

10.1. Compresión del vídeo

De momento una linea que hace que se codifique el DV de forma correcta a DivX.

```
acs@linex:~/video/jesus$ transcode -i jesus.dv -x dv  -I 3 -C 2 -z -k
```

10.2. Compresión del audio

En muchas ocasiones, por mucho que comprimamos, el distribuir una charla de 1 hora con los contenidos de vídeo por Internet es algo que aún es costoso para un gran número de usuarios, para los cuales podemos distribuir sólo el audio de la charla, que quizá sea lo más interesante.

Con transcode podemos obtener el audio de la señal de vídeo y comprimir sólo esta a formato Ogg Vorbis, el cual tiene una excelente calidad y relación de compresión. Veamos como hacerlo.

```
acs@linex:~/video/jesus$ transcode -i jesus.avi -x divx5  -I  -o jesus.ogg -
```

Vamos a obtener varios ficheros de audio en formato ogg, pero unir ficheros ogg es tan sencillo como concatenar ficheros de texto:

```
acs@linex:~/video/jesus$ cat f1.ogg f2.ogg > f.ogg
```

11. Otras herramientas de trabajo con ieee1394

11.1. Cinelarra

11.2. Coriander

Esta aplicación GNOME (http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/coriander/) nos permite controlar nuestra cámara digital y realizar muchas de las funciones que específica el protocolo IIDC (http://carol.wins.uva.nl/~fransve/literature/1394/IIDC_Spec_v1_30.pdf) de comunicaciones con las cámaras ieee1394.

Dentro de la interfaz Glade que se nos muestra, aparecerán activadas o desactivadas funciones de IIDC según las soporte nuestra cámara o no.

Para cámaras como las que estamos utilizando, basadas en el uso de cintas MiniDV, no se usa el protocolo IIDC si no el AV/C (http://carol.wins.uva.nl/~fransve/literature/1394/tape22_Final_1.pdf), por lo que no se pueden controlar desde Coriander.

12. Compilación de kino-0.6: Intimando con kino

Justo en el momento de comenzar a escribir este artículo se anunció la disponibilidad de la versión 0.6 de kino, la cual para probarla vamos a tener que compilarla desde las fuentes ya que aún no ha sido empaquetada en Debian. De paso, nos va a servir esta

experiencia para poder conocer un poco mejor las fuentes de kino, que librerías usa y sus dependencias.

Es necesario tener instalado el entorno de desarrollo de GNOME 1.4 ya que kino aún no ha sido migrado a GNOME 2. Tenemos la suerte de que los entornos de desarrollo de GNOME 1.4 y 2.0 puede convivir, por lo que nos bastará con instalar los paquetes de 1.4 adecuados.

```
linex:/home/acs/src/kino-0.6# apt-get install libgnome-dev
Reading Package Lists... Done
Building Dependency Tree... Done
The following extra packages will be installed:
  gdk-imlib-dev libart-dev libgnorba-dev liborbit-dev libungif4-dev


acs@linex:~/src/kino-0.6$ ./configure
...
checking for libdv/dv.h... no
configure: error: dv.h not found, install libdv-devel
linex:/home/acs/devel/web-xml/articulos/kino# apt-get install libdv2-dev
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  libdv2-dev
...
acs@linex:~/src/kino-0.6$ ./configure
...
checking for libraw1394/raw1394.h... no
configure: error: raw1394.h not found install libraw1394-devel
linex:/home/acs/devel/web-xml/articulos/kino# apt-get install libraw1394-dev
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  libraw1394-dev
...
linex:/home/acs# apt-get install libavc1394-dev
Reading Package Lists... Done
Building Dependency Tree... Done
The following NEW packages will be installed:
  libavc1394-dev
...
acs@linex:~/src/kino-0.6$ ./configure
...
config.status: executing default-1 commands
acs@linex:~/src/kino-0.6$
```

De momento hemos tenido que instalar las versiones de desarrollo de las librerías de libdv, tratamiento de vídeo digital, y raw1394, acceso directo al puerto 1394 y libavc1394 para el protocolo de comunicaciones con la cámara. Una vez hecho esto, ya pasamos a compilar kino y compila todo sin ningún problema.

```
make[2]: Leaving directory ‘/home/acs/src/kino-0.6/src’
Making all in help
make[2]: Entering directory ‘/home/acs/src/kino-0.6/help’
make[2]: No se hace nada para ‘all’.
make[2]: Leaving directory ‘/home/acs/src/kino-0.6/help’
make[2]: Entering directory ‘/home/acs/src/kino-0.6’
make[2]: Leaving directory ‘/home/acs/src/kino-0.6’
make[1]: Leaving directory ‘/home/acs/src/kino-0.6’
linex:/home/acs/src/kino-0.6# make install
```

Y ahora dentro de "/usr/local/bin" tenemos la nueva versión de kino a la espera de ser ejecutada. Parace que a nivel de interfaz de usuario los cambios no son muchos, pero si en cuestión de estabilidad y mejoras generales en la captura, los efectos especiales (FX), la línea de tiempos (timeline) o la interoperabilidad con otros formatos de AVI, como los que se usan en mplayer.

La estructura del código fuente no parece muy elaborada. Todas las fuentes se encuentran en "src", siendo los ficheros de cabecera que existen:

```
acs@linex:~/src/kino-0.6/src$ ls *.h
audio_filters.h      gtkenhancedscale.h   message.h            page_magick.
audio_transitions.h  ieee1394io.h         oss.h                page_timelin
avi.h                image_create.h       page_bttv.h          page_trim.h
bar.h                image_filters.h      page_capture.h       page_undefin
callbacks.h          image_transitions.h  page_editor.h        playlist.h
commands.h           interface.h          page_export_1394.h   preferences_
displayer.h          jogshuttle.h         page_export_audio.h  preferences.
error.h              kino_av_pipe.h       page_export_avi.h    riff.h
export.h             kino_common.h        page_export.h        support.h
filehandler.h        kino_extra.h         page_export_mjpeg.h  v4l.h
framedisplayer.h     magick_callbacks.h   page_export_stills.h
frame.h              magick_interface.h   page.h
```

El nombre de los ficheros está bastante bien elegido y permite localizar de forma rápida el código que se encarga de cada cosa. Por ejemplo, la implementación de efectos

especiales (FX) se basa en "magick_interface.h", que es un fichero generado de forma automática por glade. Parece pues que se usa la capacidad de glade de generación de código fuente en C++. Del código de este fichero podemos ver rápidamente los diferentes efectos que tenemos disponibles.

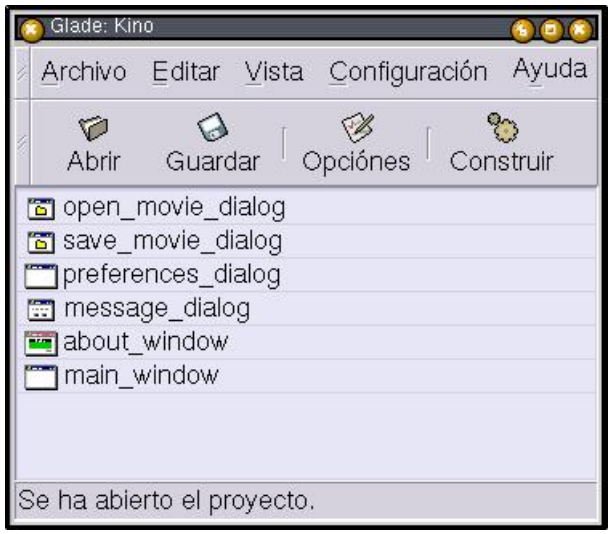
```
/*
 * DO NOT EDIT THIS FILE - it is generated by Glade.
 */

GtkWidget* create_window_magick (void);
GtkWidget* create_window_dub (void);
GtkWidget* create_window_mix (void);
GtkWidget* create_window_switch (void);
GtkWidget* create_image_filter_mirror (void);
GtkWidget* create_image_filter_kaleidoscope (void);
GtkWidget* create_image_filter_swap (void);
GtkWidget* create_image_create_gradiate (void);
GtkWidget* create_image_transition_bar_wipe (void);
GtkWidget* create_image_transition_differences (void);
GtkWidget* create_image_transition_barndoor (void);
```

El fichero "kino.glade" lo tenemos en el raíz de las fuentes de kino:

```
acs@merry:~/src/kino-0.6$ ls -l kino.glade
-rw-r--r--    1 acs      acs        227270 oct 15 21:28 kino.glade
```

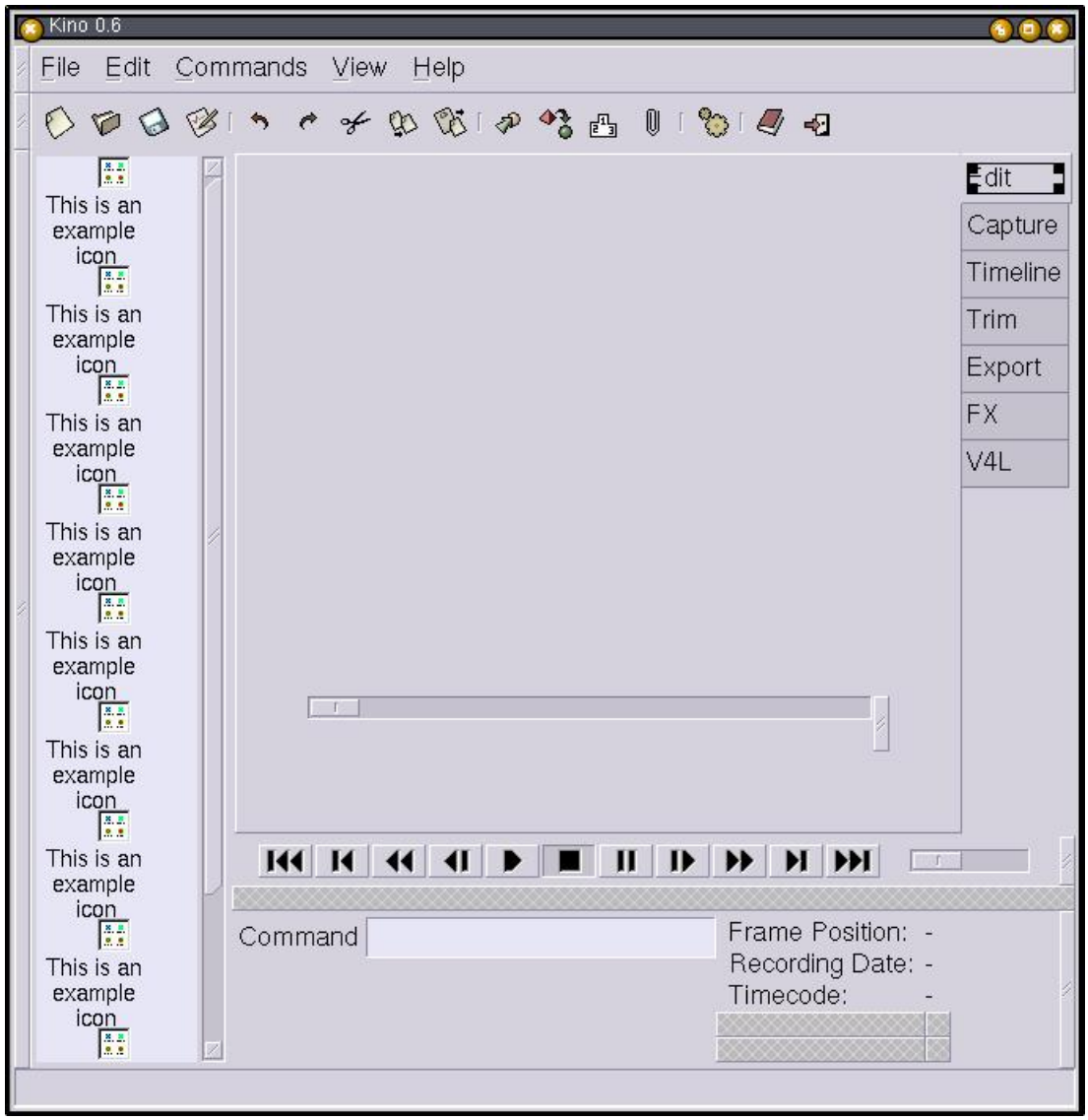
No es habitual encontrar ficheros tan grandes de glade, y podemos ver en la siguiente captura de pantalla que efectivamente, se usa glade de forma intensiva.

Figura 33. Ventanas de Kino definidas en Glade



Sin duda la ventana más espectacular es la ventana principal, que podemos ver en la siguiente captura de Glade.

Figura 34. Ventana principal de Kino definida en Glade



El código que se encarga del tratamiento de todos los eventos que provoca el usuario dentro de la interfaz Glade es "callbacks.c". Veamos un poco del código de este fichero:

```
...
void
on_new_activate (GtkWidget *menuitem,
                 gpointer user_data)
```

```
{
    newFile( );
}


void
on_open_activate (GtkWidget *menuitem,
                  gpointer user_data)
{
    openFile( );
}


void
on_save_activate (GtkWidget *menuitem,
                  gpointer user_data)
{
    savePlayList( );
}
...
void
on_move_one_frame_forward_activate (GtkMenuItem *menuitem,
                                    gpointer user_data)
{
    processMenuCommand( "l" );
}

void
on_move_one_frame_backwards_activate (GtkMenuItem *menuitem,
                                      gpointer user_data)
{
    processMenuCommand( "h" );
}
...
```

Todos los comandos que existen en kino se encuentran casi totalmente definidos en "commands.h", siendo algunos de ellos:

```
...
void videoStart( void );
void videoPreviousScene( void );
void videoStartOfScene( void );
void videoRewind( void );
```

```
void videoBack( void );
void videoPlay( void );
...
```

Los cuales ya están muy cerca de utilizar las librerías de comunicación con la cámara de vídeo. Por ejemplo, el rebobinado de vídeo:

```
...
KinoCommon *common;
...
void kinoInitialise( GtkWidget *widget )
{
...
common = new KinoCommon( widget );
...
    void videoRewind( )
    {
        common->videoRewind();
    }
...
```

que finalmente termina llamando al método de la clase "kino_common.cc"

```
/** Trigger the rewind action of the current page.
*/

void KinoCommon::videoRewind() {
    if (! is_component_state_changing ) {

        getCurrentPage()->videoRewind();

        commitComponentState();
    }
}
```

Ya hemos cumplido el objetivo de mostrar un poco la metodología de programación en Kino, la cual es aparentemente muy sencilla y que exporta de forma directa las funcionalidades de las librerías de bajo nivel a la interfaz gráfica de usuario.

13. Plugins para Kino

Hablando con Charles Yates en el canal "#kino" de "irc.openprojects.net" nos comunicó la existencia de plugins que no eran parte de la distribución oficial de Kino, y que aún estaban bastante verdes. Uno de ellos es el de rotulación, que lo necesitamos para la edición del documental de Hispalinux por lo que vamos a intentar probarlo.

Lo primero es obtenerlo del proyecto DV Titler (http://dvtitler.sourceforge.net/plugin.html). Una vez hecho esto pasamos a intentarlo compilar. Ya estamos prevenidos de que es un código muy verde aún, y que podemos tener todo tipo de problemas pero, cuando se necesita algo, se necesita :)

```
acs@linex:~/src$ tar xfz dvtitlerplug-0.0.1.tar.gz
acs@linex:~/src$ cd dvtitlerplug-0.0.1
acs@linex:~/src/dvtitlerplug-0.0.1$ ./configure
...
creating po/Makefile
acs@linex:~/src/dvtitlerplug-0.0.1$
```

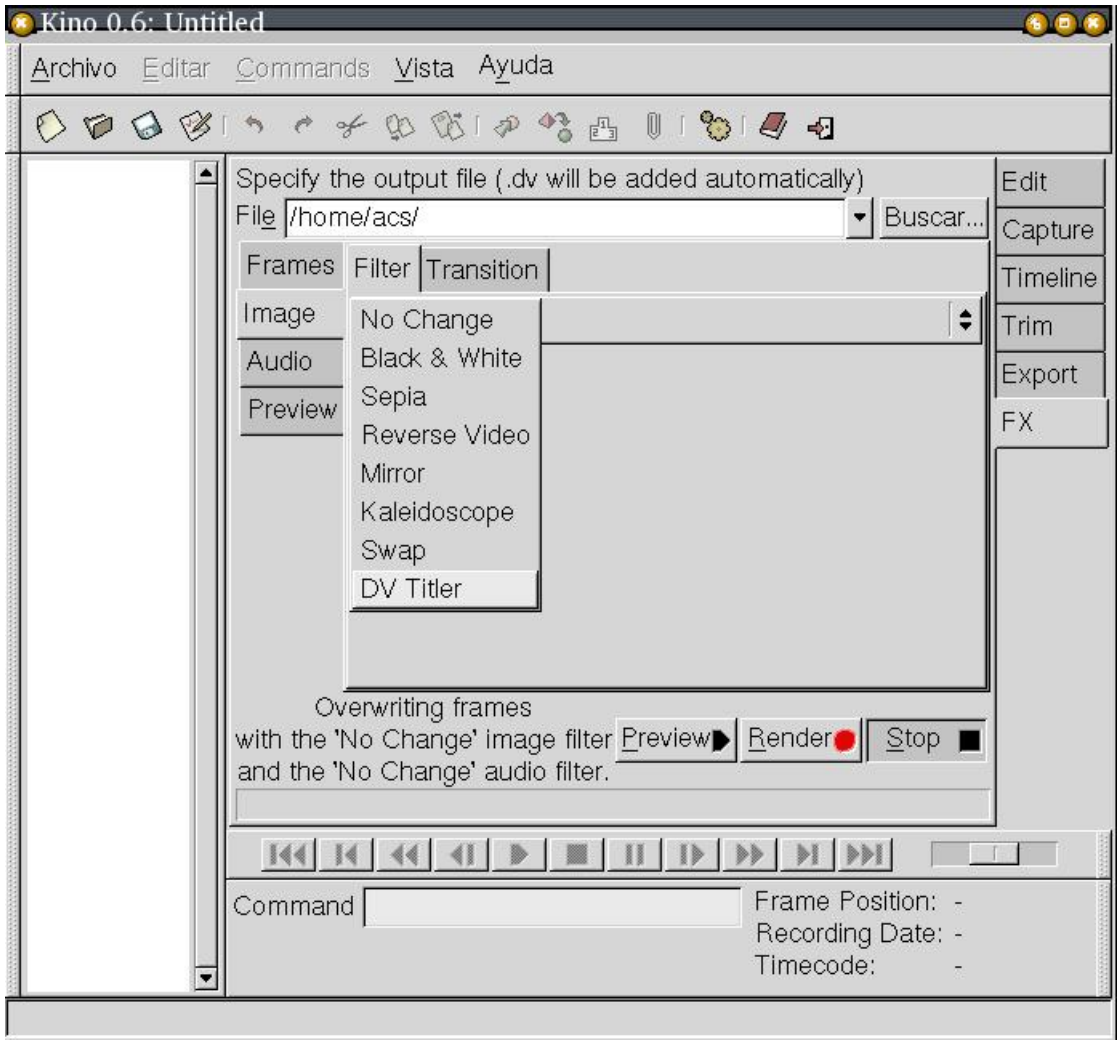
Pues de momento parece que lo vamos a poder compilar sin más problemas.

```
acs@linex:~/src/dvtitlerplug-0.0.1$ make
make  all-recursive
...
make[1]: Leaving directory ‘/home/acs/src/dvtitlerplug-0.0.1’
acs@linex:~/src/dvtitlerplug-0.0.1$
```

Lo tenemos ya compilado y casi instalado. Probemos a instalarlo y a arrancar Kino, a ver si aparece de forma automática, aunque sería demasiado bonito para un plugin alpha. Aunque vamos a ver que la experiencia nos ha vuelto demasiado desconfiados. Vamos de momento a probar con la versión de kino 0.6 que tenemos dentro de "/usr/local", al igual que este plugin.

```
linex:/home/acs/src/dvtitlerplug-0.0.1# make install
acs@linex:~/src/dvtitlerplug-0.0.1$ /usr/local/bin/kino
...
 Searching /usr/local/lib/kino for plugins
      Registering plugin /usr/local/lib/kino/libdvtitler.so
      Image Filter: DV Titler
...
```

Figura 35. Opción de rotulación incluída dentro de los FX



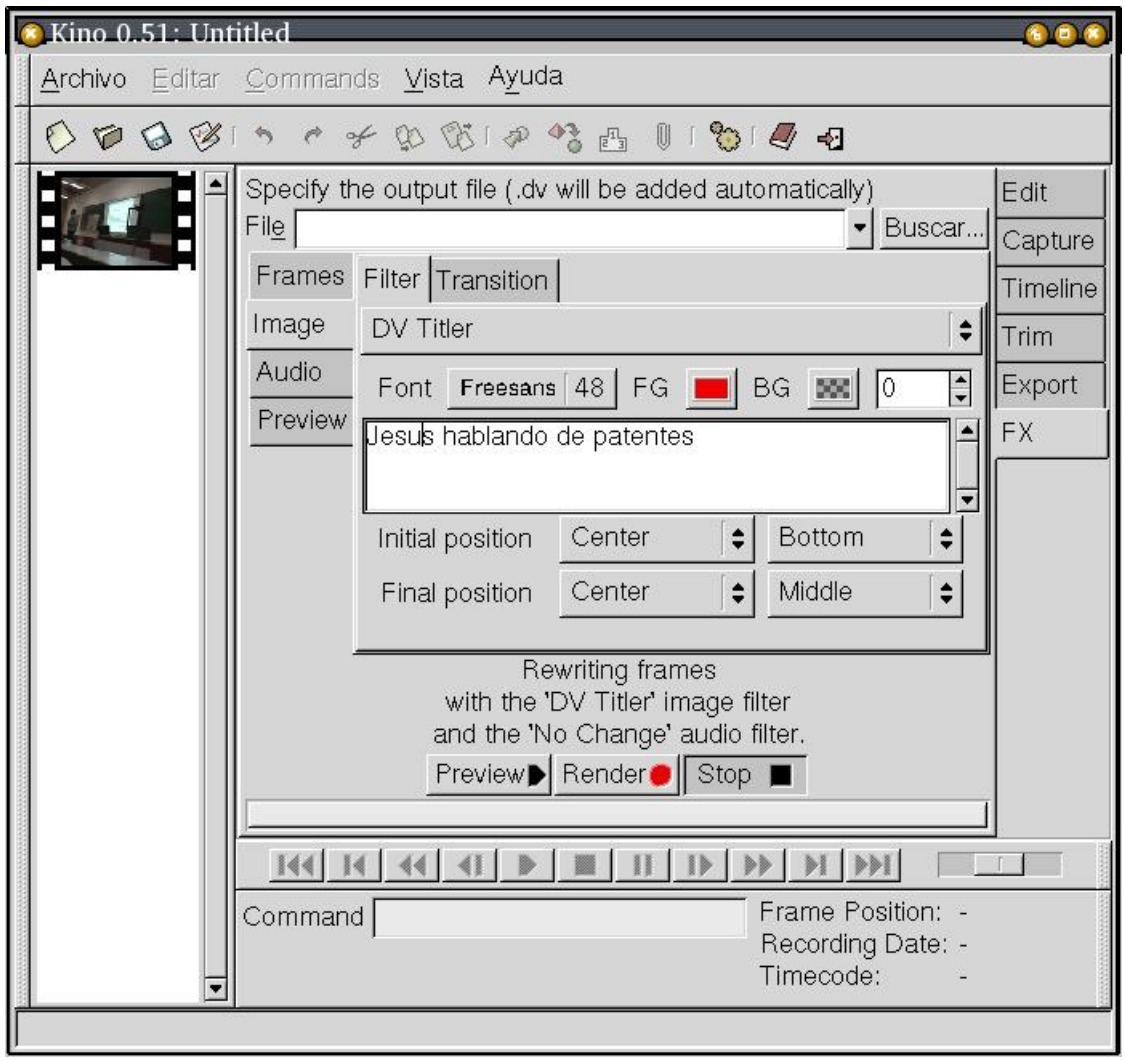
Pero eso no es todo, para pasar a tener integrada dentro de kino 0.51, el que hemos instalado desde los paquetes, nos basta con incluir el siguiente enlace que va a gustar muy poco a los puristas (a mi tampoco me gusta).

```
ln -s /usr/local/lib/kino/ /usr/lib/kino
```

Arrancamos también kino 0.51 y la opción esta de nuevo allí presente, en los famosos ya FX de Kino. Bueno, dejemos ya de jugar y pasemos a intentar utilizar el rotulador. Para ello capturamos una pequeña secuencia de la cámara digital e intentamos ponerla

un título. Para ello nos vamos a la solapa de FX, y dentro de ella seleccinamos dentro de "Image" el efecto "DV Titler", lo que nos abre una cajita en la que podemos poner el texto, el tipo de letra, los colores de fondo y frente de la caja de texto y la posición inicial y final del texto.

Figura 36. Datos sobre la rotulación

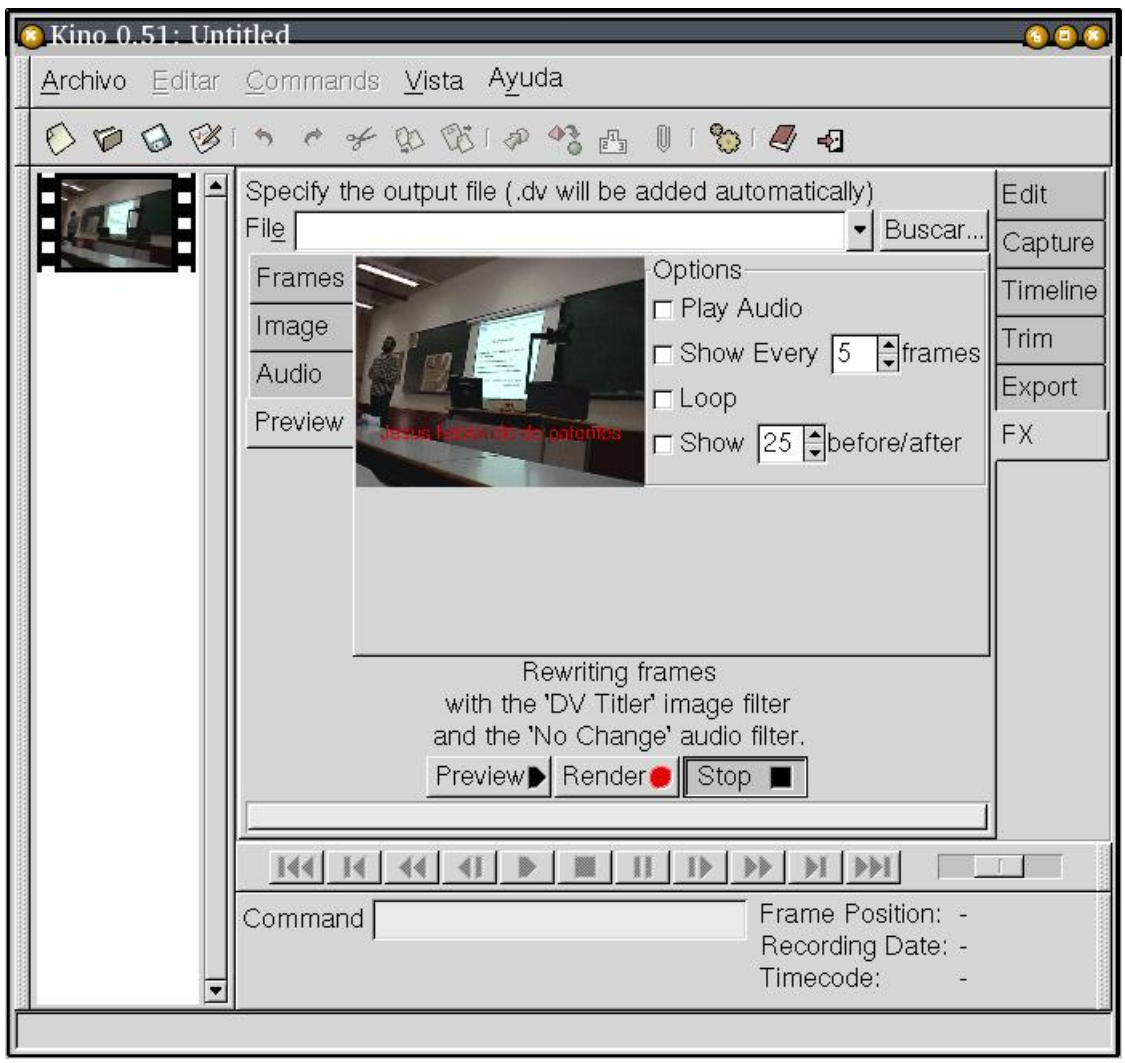


El tipo de fuente utilizado para rotular tiene que ser True Type, y además, el directorio donde este esa fuente tiene que estar incluido dentro del fichero de configuración de las X. Vamos a aprovechar el paquete de fuentes TrueType "ttf-freefont" que está

disponible en Debian, y a utilizar la fuente "FreeSans.ttf" que se encuentra dentro del directorio "/usr/share/fonts/truetype/freefont". Ojo, hay que tener mucho cuidado de tener en este directorio el fichero "fonts.dir" para que la librería que utiliza este plugin, FreeType, sea capaz de localizar el fichero con las fuentes. Para ello, y dentro del directorio "/usr/share/fonts/truetype/freefont" ejecutamos la orden "mkfontdir" la cual nos gentea el fichero "fonts.dir" con la información de las fuentes que hay en ese directorio.

Una vez hecho todo esto, podemos ya hacer un "Preview" del efecto el cual nos mostrará como el titular se desplaza desde la posición original abajo al centro hasta el posición final, en el centro mismo de la imagen, tanto horizontal como verticalmente.

Figura 37. La rotulación en acción



Una vez que ya tenemos el efecto, tenemos que guardarlo en un fichero para luego cargarlo en Kino y poderlo incorporar a nuestra edición. Esta parte aún no está muy bien integrada en Kino y es algo engorrosa, pero nos podemos apañar. El único problema real es que tras especificar el fichero donde se guardará el efecto (render) y lanzar el efecto real para que se almacene en el fichero, Kino se muere en el intento en la versión 0.51 y en la 0.60, aunque sea un "render" sin ningún tipo de efecto. Pero después de hablar con Charles Yates, el problema se resuelve instalando la última versión de "libdv", que la que viene actualmente con Debian (0.98-2) tiene fallos que afectan a esta exportación.

14. Kinoplus, más funcionalidades

Si analizamos otros programas de edición de vídeo como Cinelerra, nos puede parecen que Kino aún dispone de pocas opciones a la hora de realizar efectos sobre el vídeo, en las transiciones de las imágenes o a la hora de introducir otro tipo de formatos dentro de kino.

Pero a toda esta carencia da respuesta el plugin Kinoplus desarrollado por Charles Yates, y que como vamos a ver, incluye una gran cantidad de nuevas posibilidades, algunas de ellas realmente espectaculares.

Al igual que el plugin de rotulación, la instalación del mismo va sobre ruedas, aunque parece que hay algunos problemas con la versión 3.2 de gcc, asi que si estás a loa última puede que tengas algún problemilla aunque hoy mismo los están arraglando.

Una vez que nos hemos bajado el plugin Kinoplus (http://users.pandora.be/acp/kino/) pasamos a instalarlo tal y como hicimos con dvtitler. Es importante destacar que este plugin ya sólo funciona con la versión 0.6 de Kino por lo que a partir de este momento vamos a trabajar con esta.

```
acs@linex:~/src$ tar xfz kinoplus-0.2.0.tar.gz
acs@linex:~/src/kinoplus-0.2.0$ ./configure
checking for a BSD compatible install... /usr/bin/install -c
...
config.status: creating config.h
config.status: config.h is unchanged
acs@linex:~/src/kinoplus-0.2.0/make
acs@linex:~/src/kinoplus-0.2.0$ make
cd . \
...
```

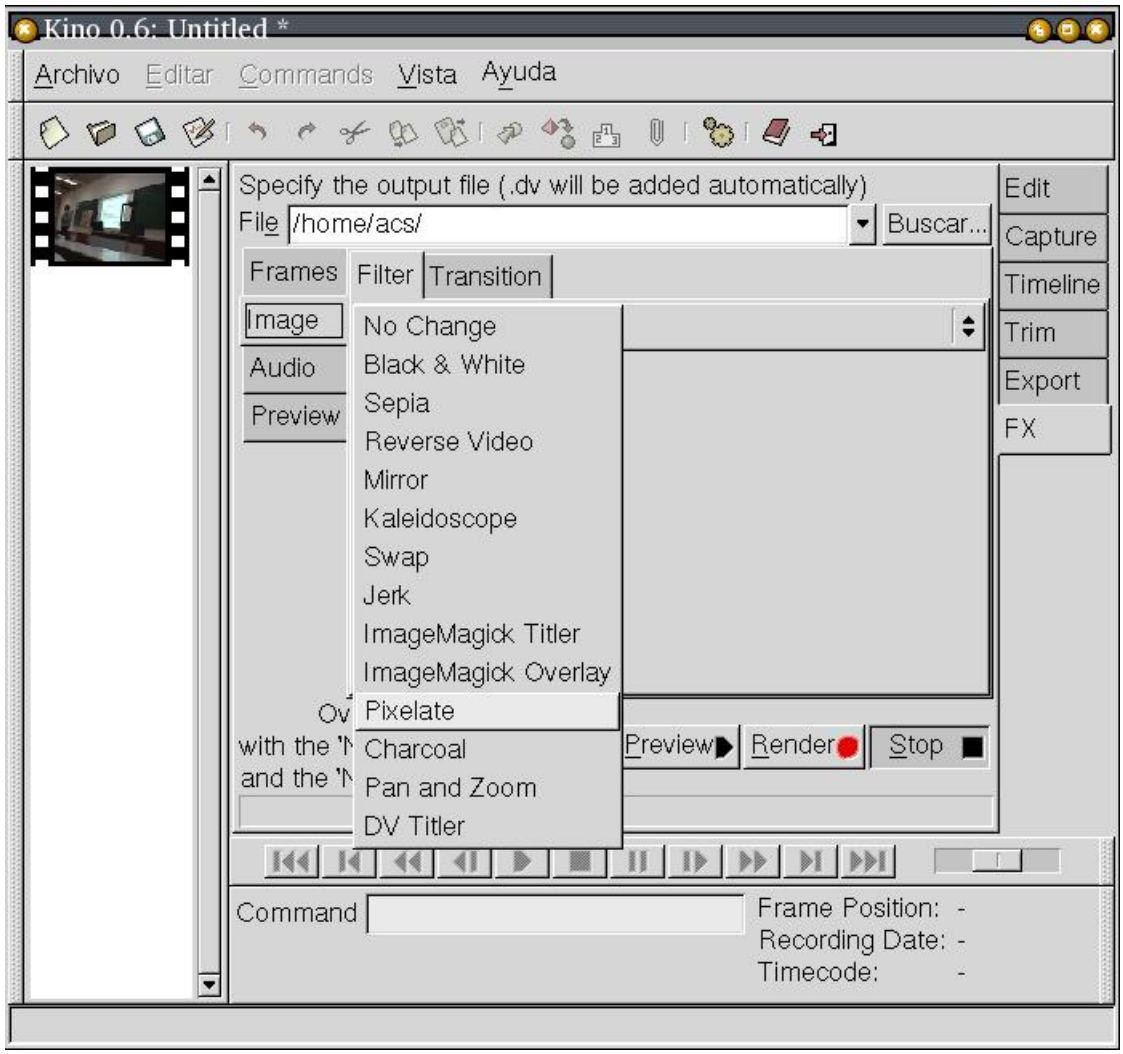
```
make[1]: Leaving directory ‘/home/acs/src/kinoplus-0.2.0’
acs@linex:~/src/kinoplus-0.2.0$
linex:/home/acs/src/kinoplus-0.2.0# make install
Making install in macros
...
make[1]: Leaving directory ‘/home/acs/src/kinoplus-0.2.0’
linex:/home/acs/src/kinoplus-0.2.0#
```

Ahora ya tenemos instalado el plugin y cuando arranquemos Kino podremos ver como lo reconoce.

```
>> Searching /usr/local/lib/kino for plugins
>>> Registering plugin /usr/local/lib/kino/libkinoplus.so
>>> Registering plugin /usr/local/lib/kino/libdvtitler.so
>>> Image Create: Multiple Image Import
>>> Image Filter: Jerk
>>> Image Filter: ImageMagick Titler
>>> Image Filter: ImageMagick Overlay
>>> Image Filter: Pixelate
>>> Image Filter: Charcoal
>>> Image Filter: Pan and Zoom
>>> Image Transition: Chroma key on blue
>>> Image Transition: Tweenies
>>> Image Filter: DV Titler
```

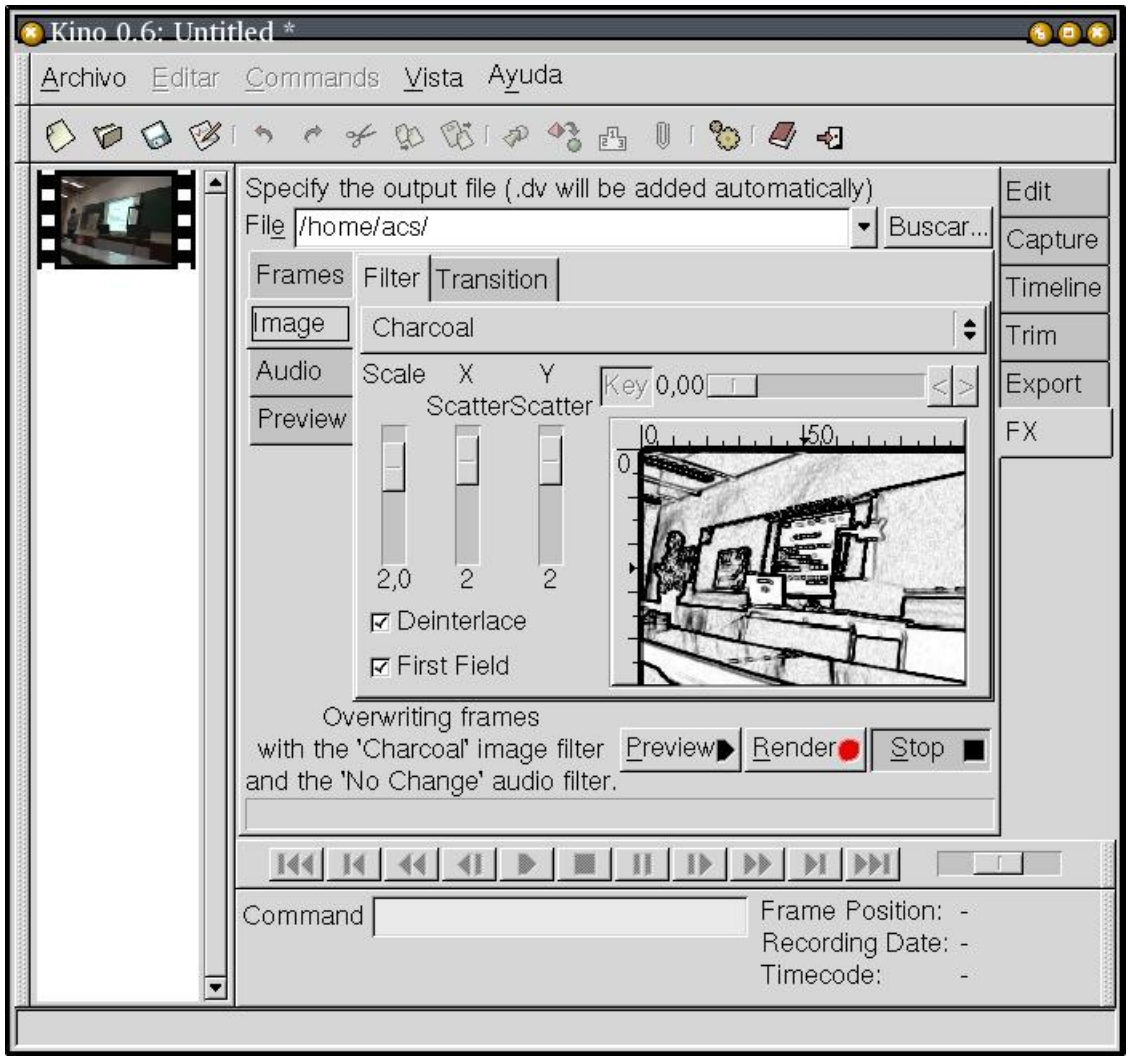
Vemos que este nuevo plugin incorpora muchas nuevas funcionalidades: la de importanción de múltiples archivos de imágenes para construir una secuencia, las de filtrado de imágenes que incluyen también un rotulador y un transformador de imágenes a tiza (Charcoal), y dos que ayudan en las transiciones entre las secuencias, destacando Tweenies que funde dos imágenes expandiendo una de ellas desde el centro utilizando una zona rectangular creciente. Pero veamos las nuevas funcionalidades en acción para ver cuanto pueden dar de sí.

Figura 38. Nuevas posibilidades gracias a Kinoplus



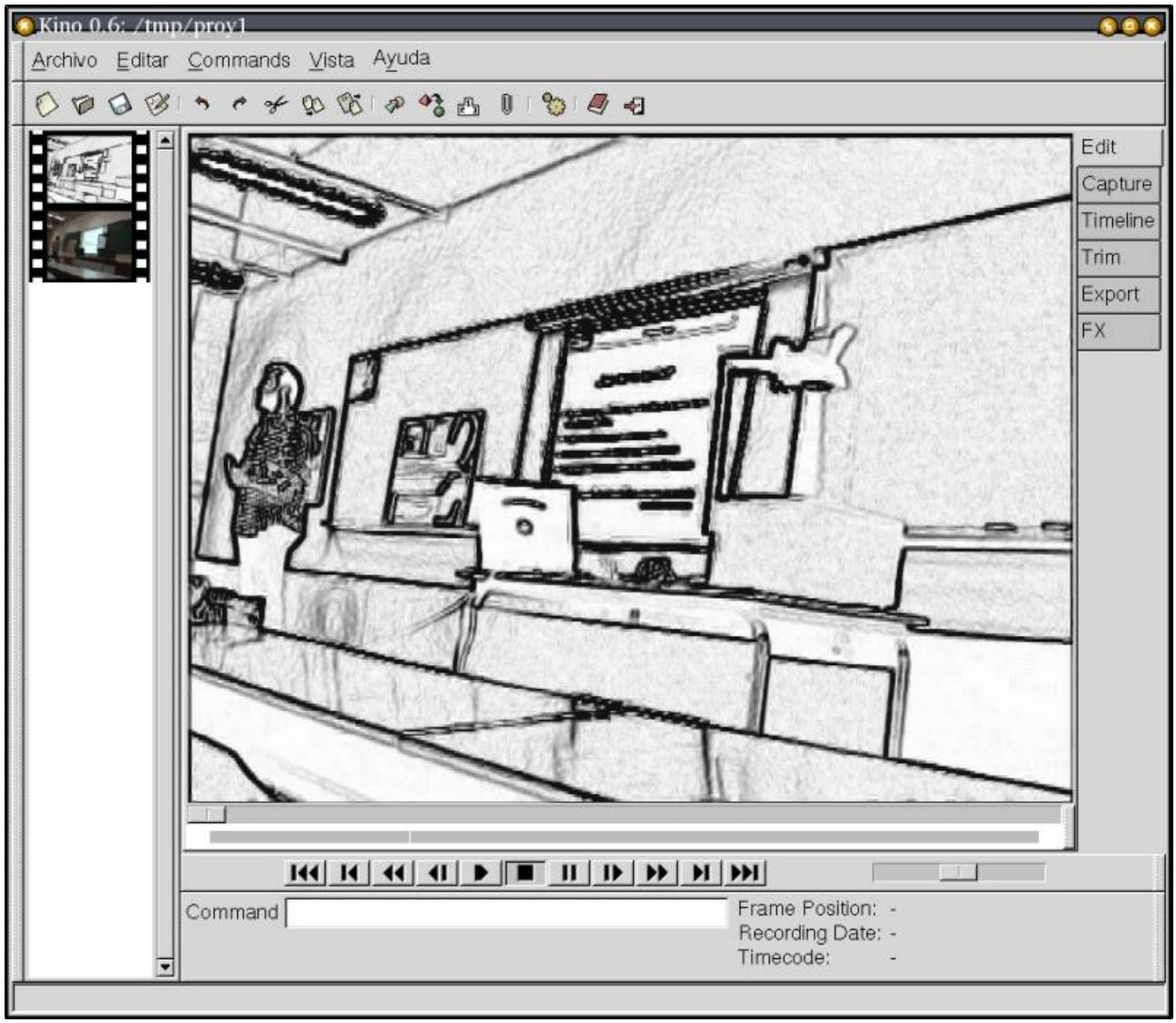
Vamos a comenzar probando "Charcoal" que es sin duda el filtro de imágen más impresionante. Por defecto los efectos especiales se aplican en 0.6 sobre las primeras 200 imágenes de la secuencia, algo que modificamos para hacerlo sólo sobre las primeras 48, los dos primeros segundos. Dentro de los filtros de imagen seleccionamos "Charcoal" y el resultado que obtenemos es el siguiente.

Figura 39. El filtro Charcoal de Kino



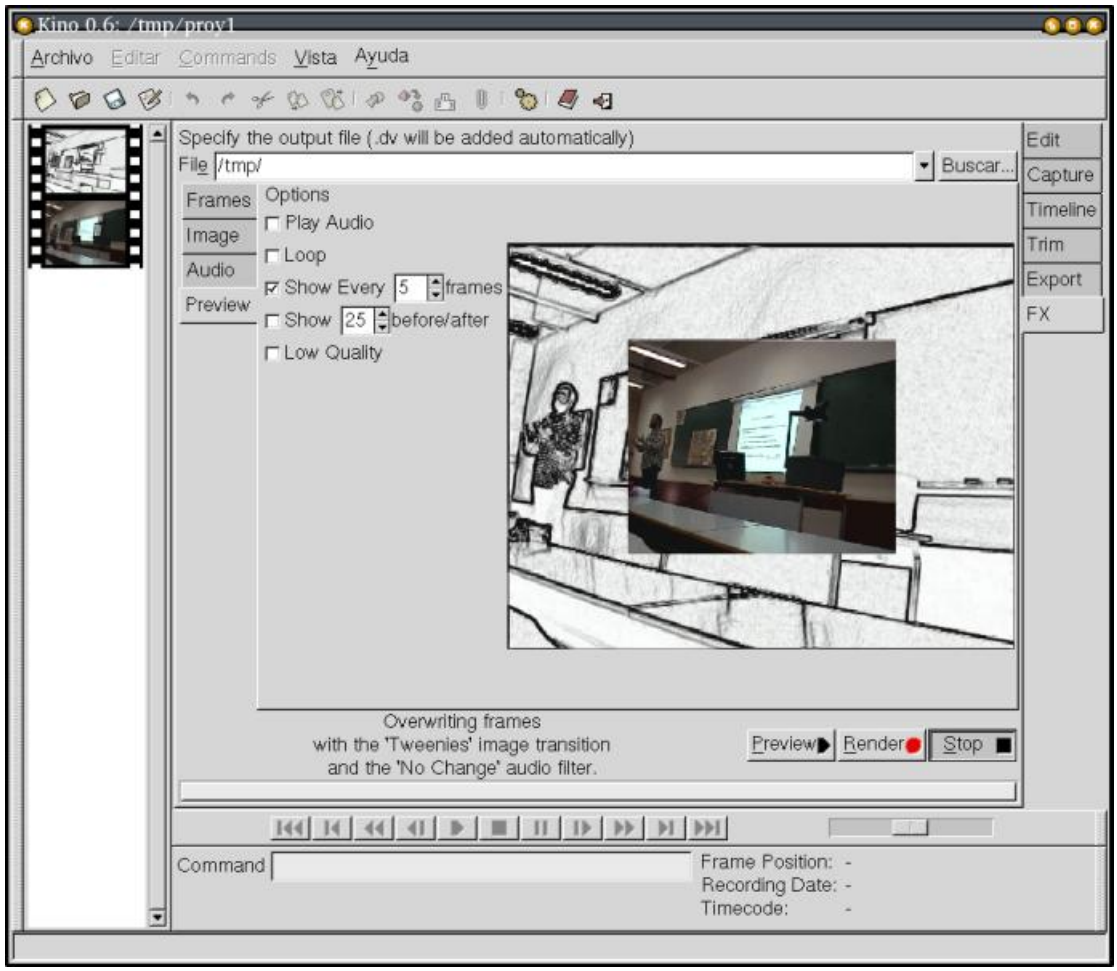
Vemos que el filtro transforma la imagen como si fuera una pintura a tiza, una especie de dibujos animados hechos a tiza. Si hacemos un "render" con esta configuración, y añadimos la nueva secuencia a nuestro proyecto, podemos observar el fuerte contraste entre la secuencia original y la modificada con el filtro.

Figura 40. Nueva secuencia con charcoal



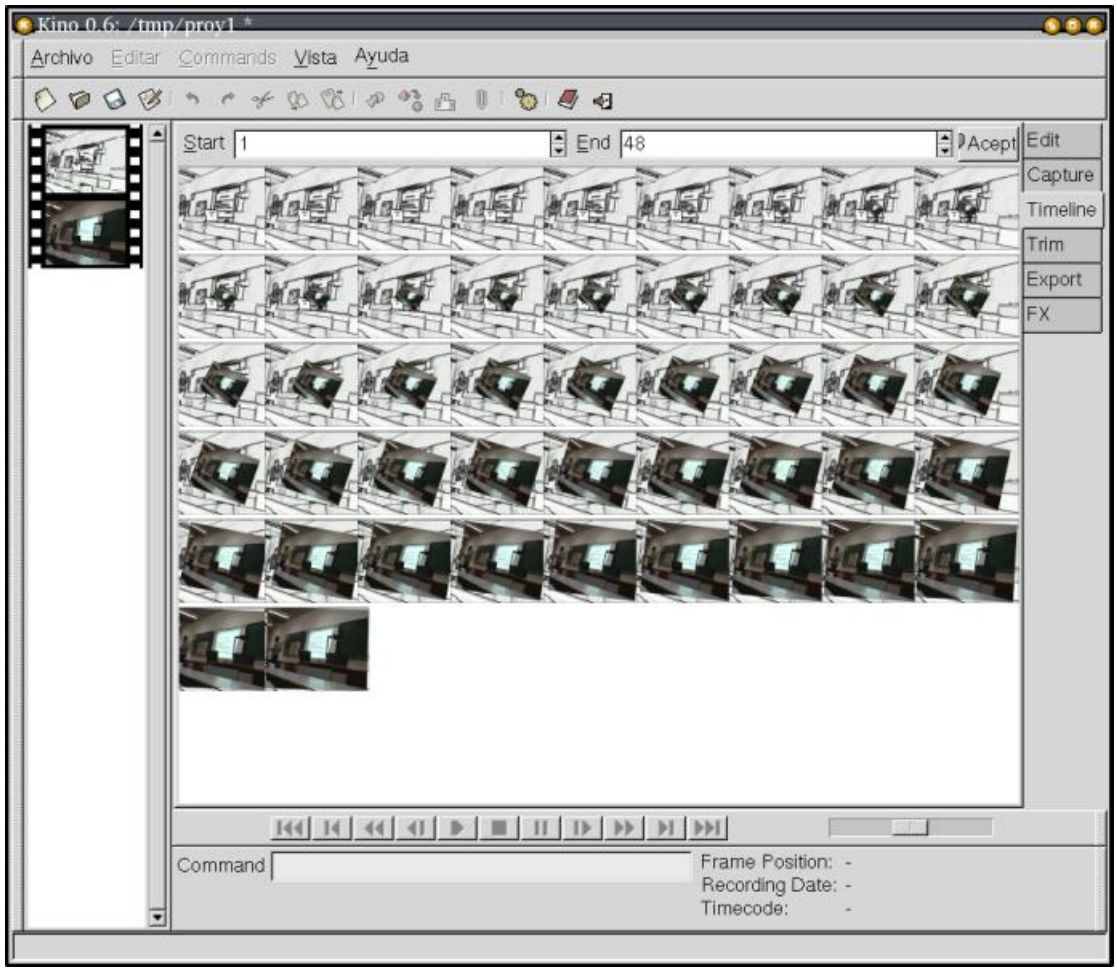
Ahora tenemos el problema de que la transición entre la secuencia con filtro y la original es que el cambio es muy brusco. Para resolverlo, vamos a utilizar otra de las ventajas de este plugin, la transición Tweenies. Para ello hay que especificar entre que imágenes hay que hacer la transición. En nuestro caso, debe de ser entre el segundo 2 y 3 que es cuando se unen las dos secuencias.

Figura 41. Transición entre dos secuencias



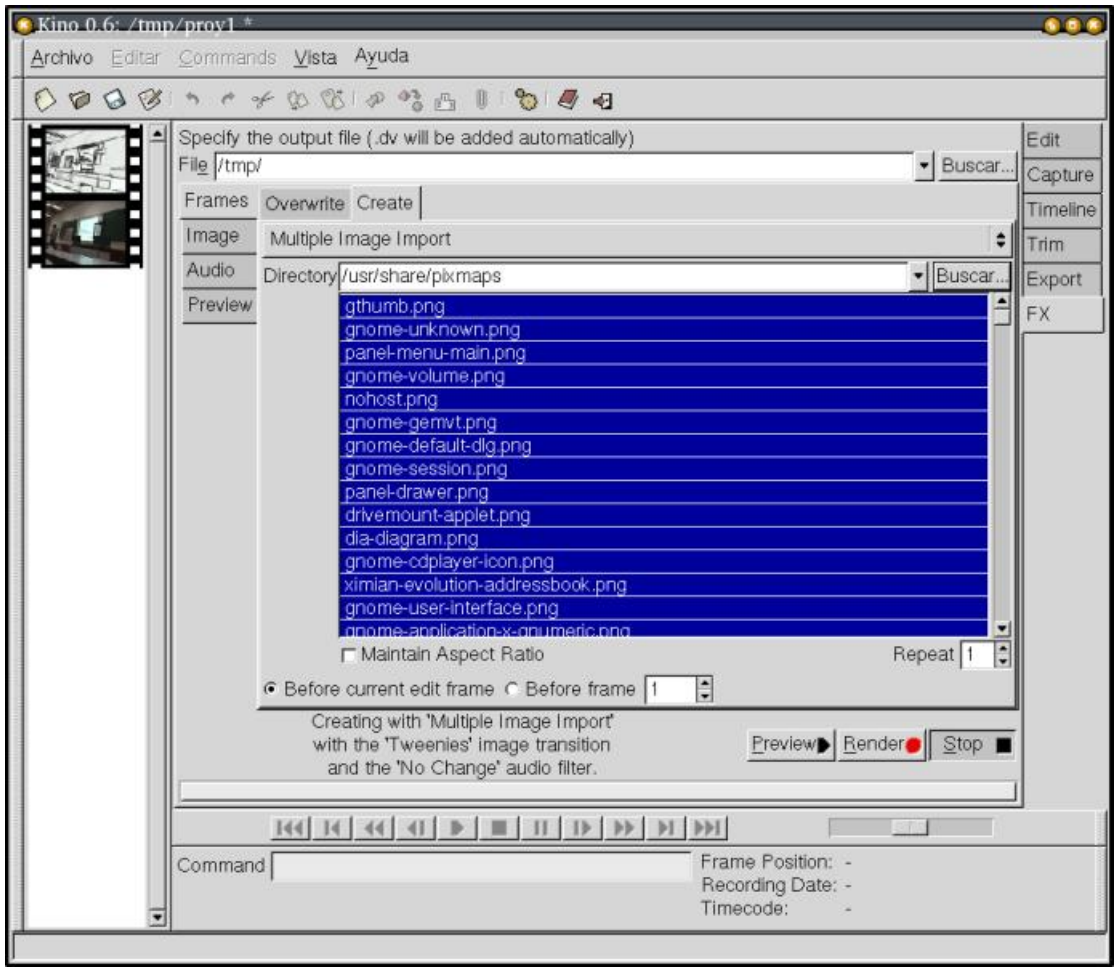
Las posibilidades a la hora de unir dos secuencias por Tweenies son enormes, ya que disponemos de diferentes tipos de fusión de una imagen en la otra junto con efectos combinados de rotaciones o que la fusión comience con un centro diferente. En fin, que tenemos un agran libertad por parte del editor para lograr trasmitir su dia artística de muchas formas diferentes.

Figura 42. Línea de tiempos de la transición



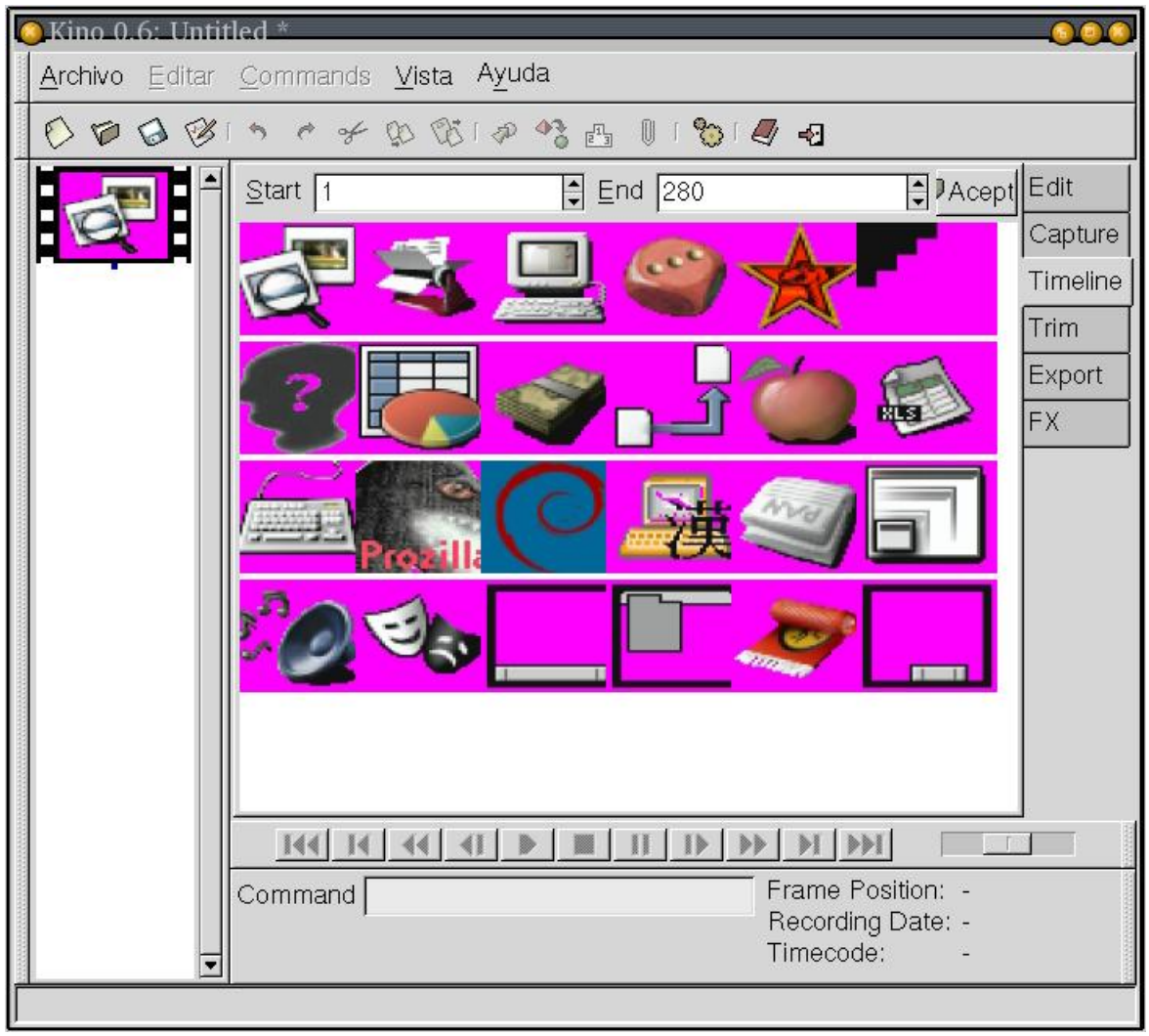
Para terminar de mostrar algunas de las características de Kinoplus, veamos como permite importar un conjunto de imágenes y convertirlas en una secuencia que podemos intergrar en nuestro proyecto. Para ello basta con selecccciona la solapa de "Frames" dentro de "FX" y allí, seleccionar la solapa de crear contenidos "Create". Una de las opciones que nos aparece es la de "Multiple Image Import". Basta con que especifiquemos un directorio con imágenes, aunque sea con diferentes formatos, y cada una de las imágenes pasará a ser parte de una nueva secuencia que formará parte de nuestro proyecto.

Figura 43. Importadas imágenes como un vídeo



La mejor forma de poder ver la estructura de la nueva secuencia es analizar su linea de tiempos.

Figura 44. Línea de tiempos del vídeo de imágenes



Por último destacar la velocidad con la que trabajamos con Kino. Todos los "renders" que hemos hecho en esta sección siempre han sido de duración inferior al minuto, es decir, que suelen llevar siempre segundos. Unido esto a la veloz previsualización de los ejemplos, podemos trabajar de forma cómoda y rápida con los efectos especiales en Kino.

15. Creación de contenidos en Kino

16. Edición de audio en Kino

17. Hacer plugins para kino

Después de la grata experiencia con el plugin de rotulación seguro que a más de uno se le ha pasado por la cabeza el crear un plugin para kino que haga ese efecto tan impresionante que ha visto en alguna ocasión en montajes profesionales.

¿Será complicado hacer un plugin? ¿Qué hay que saber? Si el lector desconoce la programación en C++ y en general, no es un desarrollador de software, le prevenimos contra lo que viene a continuación, ya que es código en C++ en su mayoría que puede resultar incomprensible para mucha gente. Si a pesar de este aviso no hemos logrado que deje de leer, quien sabe, quizá ha llegado el momento de que aprenda a programar :)

Uno de los primeros conceptos a tener claros es que Kino funciona totalmente orientado a imágenes, es decir, que al final nuestro efecto será un filtro que iremos aplicando sobre cada una de las imágenes que formen la secuencia de vídeo. Por lo tanto, no parece que en principio sea especialmente complicado, ya que tan sólo tenemos que conocer como aplicar los filtros para lograr el efecto deseado, y aplicar ese filtro de forma repetida a un conjunto de imágenes, las que forman precisamente la secuencia.

Una vez que tenemos claro conceptualmente como llevar a cabo los efectos, hay que ver que API nos ofrece Kino para acceder a la secuencia de imágenes y poder irlas transformando.

Vamos a utilizar el plugin de rotulación como referencia de trabajo. Veamos que tamaño tiene finalmente este plugin:

```
acs@linex:~/src/dvtitlerplug-0.0.1/src$ wc *.cc *.h
    177     658    5242 dvtitler.cc
    288    1022    7961 ft_titler.cc
     19      28     656 callbacks.h
     59     251    1636 dvtitler.h
     91     319    2163 ft_titler.h
      5      17      97 interface.h
     34     115     955 support.h
```

```
    673    2410   18710 total
```

Con 673 líneas totales de código, con comentarios incluídos, se logra todo el trabajo del plugin de rotulación, el cual es bastante completo. Y en esas líneas están incluídos los comentarios. Si utilizamos un contador de líneas de código algo más preciso como sloccount:

```
acs@linex:~/src/dvtitlerplug-0.0.1$ sloccount src/
...
Totals grouped by language (dominant language first):
cpp:            391 (90.30%)
ansic:           42 (9.70%)
...
Total Physical Source Lines of Code (SLOC)                = 433
```

Vemos que tenemos 433 líneas de código reales, la mayoría hechas en C++, que es el lenguaje dominante dentro de Kino.

La clase que parece que es la que usa Kino para incluir el plugin dentro de la interfaz de efectos especiales "FX", es DVTitle, cuya interfaz pública es:

```
class DVTitler : public GDKImageFilter
{
 public:
   DVTitler();
   virtual ~DVTitler();
   char *GetDescription( ) const;
   void AttachWidgets( GtkBin *bin );
   void DetachWidgets( GtkBin *bin );
   void InterpretWidgets( GtkBin *bin );
   void FilterFrame( uint8_t *io, int width, int height, double position, do
```

Vemos que con esta interfaz kino ya puede utilizar el plugin ya que por un lado, es capaz de mostrar la GUI del plugin (AttachWidgets) y de obtener los datos del GUI (InterpretWidgets), y por otro, es capaz de aplicar el efecto necesario a cada una de las imágenes utilizando el método "FilterFrame". La curiosidad nos empuja a ver como se filtra cada una de las imágenes, y aprovechando que estamos en software libre, vemos que como se dibuja el rotulo dentro de cada imagen.

```
void DVTitler::FilterFrame( uint8_t *io, int width, int height, double posit
{
```

```
   int w, h;
   int x, y;
   float ph, pv;
   int row_w = width * 3;

//   FT_Titler titler(64, text);
   titler->getMetrics(w, h);

   ph = position*(fh - ih) + ih;
   pv = position*(fv - iv) + iv;
   x = (int)(ph*(width - w)/2 + (1 - ph)*pad);
   y = (int)(pv*(height - h)/2 + (1 - pv)*pad + h);
   titler->setPixels(io, row_w);
   titler->setColors(colorfg, colorbg);

   if (pad > 1 && colorbg.a > 50)
     drawRectangle(io, x-pad, y-h-pad, w+2*pad, h+2*pad, row_w);
   titler->drawText(x, y);
}
```

y para terminar de saciar nuestra curiosidad, veamos el método "drawText" de la clase FT_Titler:

```
void FT_Titler::drawText(int start_x, int start_y)
{
   FT_Error      error;
   FT_Vector pen;

   for ( int n = 0; n < num_glyphs; n++ ) {
      FT_Glyph  image;

      image = glyphs[n];

      pen.x = start_x + pos[n].x;
      pen.y = start_y + pos[n].y;

      error = FT_Glyph_To_Bitmap( &image, ft_render_mode_normal,
                                  &pen, 0 );
      if (!error) {
         FT_BitmapGlyph  bit = (FT_BitmapGlyph)image;

         drawBitmap( bit->bitmap,
                     pen.x + bit->left,
```

```
                    pen.y - bit->top );
          FT_Done_Glyph( image );
       }
    }
}
```

Vemos como se transforman los Glyph (caracteres) en bitmaps que son los que luego se dibujan sobre las imágenes.

Como hemos visto la clase que realiza el trabajo real de incluir en las imágenes el texto es la clase FT_Titler, que utiliza la librería FreeType para llevar a cabo el trabajo.

Si nos vamos a Kino a ver como usa los plugins de efectos especiales, tenemos que pasar a consultar la clase "PluginCollection" dentro de "page_magick.cc". Allí tenemos por ejemplo el método que se encarga durante el inicio de registrar todos los plugins que se encuentran en el directorio adecuado.

```
void PluginCollection::Initialise( char *directory )
{
        char *filename;
        char *extension;
        DIR *dir;
        struct dirent *entry;
        struct stat statbuf;

        dir = opendir( directory );

        if ( dir )
        {
                while ( ( entry = readdir( dir ) ) != NULL )
                {
                        filename = g_strdup_printf( "%s/%s", directory, entr
                        extension = strrchr( entry->d_name, '.' );
                        if ( extension != NULL && !stat( filename, &statbuf
                        {
                                if ( !strcmp( extension, ".so" ) )
                                {
                                        RegisterPlugin( filename );
                                }
                        }
                        g_free(filename);
                }
                closedir(dir);
        }
```

```
}
```

Vemos pues que todos los ficheros que terminen en ".so" y que estén en el directorio adecuado, se intentan cargar como plugins utilizando el método "RegisterPlugin", el cual podemos ver a continuación:

```
void PluginCollection::RegisterPlugin( char *filename )
{
        Plugin *plugin = new Plugin;
        if ( plugin->Open( filename ) )
        {
                cout << ">>> Registering plugin " << filename << endl;
                collection.push_back( plugin );
        }
        else
        {
                cout << ">>> Rejecting plugin " << filename << " : " << plug
                delete plugin;
        }
}
```

Durante el inicio de kino ya vimos estos mensajes que nos indicaban que se había cargado un plugin nuevo.

```
>> Searching /usr/local/lib/kino for plugins
>>> Registering plugin /usr/local/lib/kino/libdvtitler.so
>>> Image Filter: DV Titler
```

Vemoas por último el método Open que es el que decide si un plugin es válido o no.

```
bool Plugin::Open( char *file )
{
        ptr = dlopen( file, RTLD_NOW );
        return ptr != NULL;
}
```

Vemos que aquí lo único que hacemos es utilizar la biblioteca de carga dinámica de módulos, y no comprobamos para nada la interfaz de lo que estamos cargando. Donde realmente se analiza de verdad el plugin que se ha cargado y de que tipo es (puede ser para filtrar imágenes, sonido o realizar transiciones de vídeo o de audio) es en el

método InstallPlugins de cada una de las 4 clases que representan a los 4 posibles tipos de plugins que podemos realizar.

En nuestro caso el filtro que tenemos es de imagen, algo que ya vimos por la forma como declaramos la clase

```
    class DVTitler : public GDKImageFilter
```

por lo que la instalación de este plugin se llevará a cabo en

```
void PluginImageFilterRepository::InstallPlugins( Plugin *plugin )
{
        GDKImageFilter *( *func )( int ) = ( GDKImageFilter *( * )( int ) )p
        if ( func != NULL )
        {
                int index = 0;
                GDKImageFilter *entry = func( index ++ );
                while ( entry != NULL )
                {
                        Register( entry );
                        entry = func( index ++ );
                }
        }
}
```

que pasa a incorporar a la clase cargada de forma dinámica a los plugins de tratamiento de imágenes, junto con los demás estándares que se encuentran en "image_filters.cc".

Para finalizar, veamos un filtro mucho más sencillo, el que pasa la imagen a Sepia, que se incorpora directamente dentro de Kino.

```
/** Converts images to sepia.
*/

class ImageFilterSepia : public ImageFilter
{
        public:
                char *GetDescription( ) const
                {
                        return "Sepia";
                }

                void FilterFrame( uint8_t *pixels, int width, int height, do
```

```
                {
                        uint8_t r, g, b;
                        uint8_t *p = pixels;
                        while ( p < ( pixels + width * height * 3 ) )
                        {
                                r = *( p );
                                g = *( p + 1 );
                                b = *( p + 2 );
                                r = (uint8_t)( 0.299 * r + 0.587 * g + 0.114
                                *p ++  = r < 225 ? (int)( r + 30 ) : 0xff;
                                *p ++  = r;
                                *p ++  = r > 30 ? (int)( r - 30 ) : 0;
                        }
                }
};
```

Vemos que la interfaz a implementar por "ImageFilter" es más sencilla que la que tenemos que implementar con "GDKImageFilter", ya que esta última incluye una interfaz gráfica para llevar a cabo la acción, mientra que la de "ImageFilter" tan sólo tiene la posibilidad de activación y desactivación.

Hemos podido ver a lo largo de todo este capítulo final como programar filtros para Kino, algo que no es tan complejo, y que nos puede permitir extender las funcionalidades de Kino y adaptarlas a nuestras necesidades.

18. Conclusiones

A lo largo de todo este tutorial hemos aprendido todas las características que nos ofrece Kino a la hora de editar vídeo. Ha llegado el momento de ponerlas en práctica en nuestro proyectos y de mostrar como el software libre dispone de un excelente editor de vídeo para el usuario final.

19. Referencias

- Página principal de Kino (http://kino.schirmacher.de/)
- Página Debian de kino (http://packages.debian.org/unstable/graphics/kino.html)

- Enlaces de Linux y vídeo digital (http://www.schirmacher.de/cgi-bin/dclinks.cgi)
- Coriander: control de la cámara de vídeo (http://www.tele.ucl.ac.be/PEOPLE/DOUXCHAMPS/ieee1394/coriander/)
- Transcode (http://www.theorie.physik.uni-goettingen.de/~ostreich/transcode/)
- Proceso de creación de películas en GNU/Linux (http://www.netjunki.org/articles/makingmovieswithlinux.html)
- Cinelarra (http://heroinewarrior.com/cinelerra.php3)
- Historias con éxito en edición de vídeo digital (http://slashdot.org/article.pl?sid=02/10/02/2359254&mode=thread&tid=167)
- Tabla comparativa de cámaras digitales (http://www.imagendv.com/tabla.htm)
- IEEE1394 Trade Center (http://www.1394ta.org/)
- Enlaces a temas de multimedia (http://carol.wins.uva.nl/~fransve/mm/multimedia3.html). Interesantes enlaces de IEEE1394.
- Linux IEEE 1394 (http://www.linux1394.org/)
- Biblioteca basada en GStreamer para edicion no lineal de video (http://gnonlin.sourceforge.net/)
- Utilizar GNU/Linux como un VCR (http://www.digitaltoad.net/docs/VCR-HOWTO.htm)
- Titulos en Kino (http://dvtitler.sourceforge.net/plugin.html)
- Titulos en Kino (http://dvtitler.sourceforge.net/plugin.html)
- Kinoplus, filtros y efectos avanzados de Kino (http://users.pandora.be/acp/kino/)
- Efectos de desenfoque y lupa para Kino (http://www.k-3d.com/kino/)
- Uso de transcode para codificar video (http://www.bunkus.org/dvdripping4linux/en/separate/index.html#toc)