MAKE YOUR OWN SUGAR ACTIVITIES!

TABLE OF CONTENTS

ACTIVIDADES DE SUGAR

1. INTRODUCCIÓN

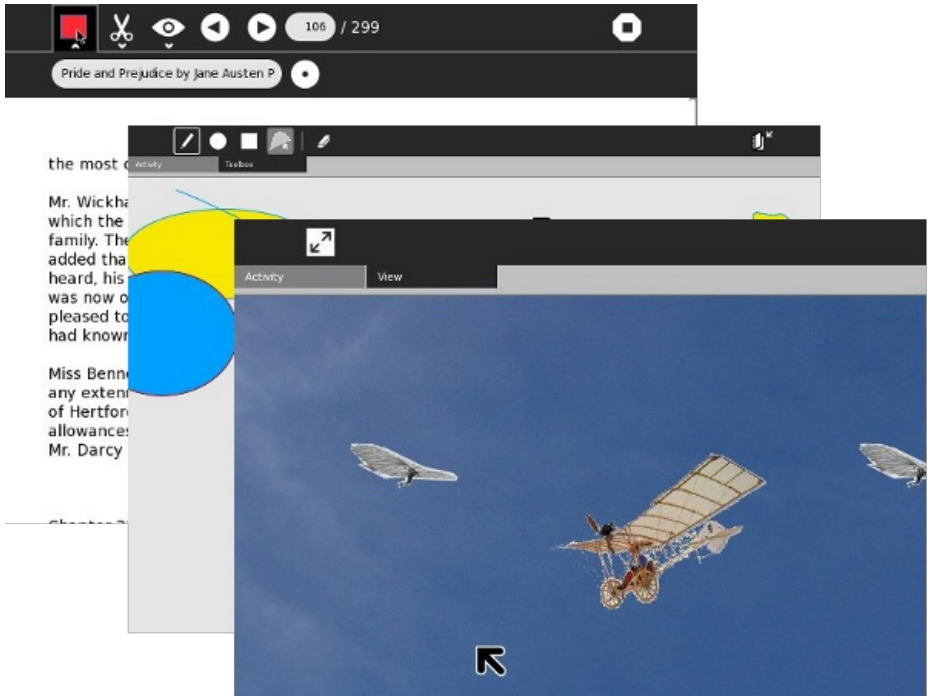
"Este libro es la historia de un viaje de placer. Si fuera la historia de una solemne expedición científica, tendría sobre ella esa gravedad, esa profundidad, y esa incomprensibilidad impresionante tan apropiada para los trabajos de ese tipo, sin embargo tan atractiva."

Del prefacio de *Los Inocentes en el extranjero*, de Mark Twain.

El propósito de este libro es enseñarle lo que necesita saber para escribir Actividades en Sugar, el sistema operativo desarrollado para el proyecto OLPC. Este libro no asume que usted sabe cómo programar una computadora, aunque aquellos que sepan programar puedan encontrar información útil en él. Mi principal objetivo al escribirlo es animar a aquellos que no son programadores, incluyendo a los niños y sus profesores, a crear sus propias Actividades para Sugar. Por esta razón, voy a incluir algunos detalles que otros libros no van a tener y voy a dejar fuera algunas cosas que otros incluyen. La *incomprensibilidad* impresionante se mantendrá al mínimo.

Si lo que desea es solo aprender a escribir programas de ordenador, Sugar proporciona muchas Actividades para ayudarle: Etoys, Turtle Art, Scratch, y Pippy. Ninguno de estos son realmente apropiados para la creación de Actividades y por eso, no voy a discutirlos en este libro, pero son una buena forma de aprender acerca de la programación. Si usted decide, después de jugar con ellos, que le gustaría probar y escribir una Actividad después de todo, va a tener una buena base de conocimiento para desarrollarla.

Una vez que haya hecho algún programa tendrá la satisfacción de que se pueda utilizar el programa que Ud. hizo, uno que funciona *exactamente* del modo que usted quiere. La creación de una Actividad para Sugar lleva ese disfrute al siguiente nivel. Una Actividad en Sugar puede ser traducida por voluntarios en todos los idiomas, puede descargarse cientos de veces por semana y ser utilizada diaramente por los estudiantes del mundo entero.



Un libro que enseñara *todo* lo que se necesita saber para escribir Actividades sería muy, muy largo y duplicaría material que ya está disponible en otros lugares. Debido a esto, voy a escribir esto como un viaje guiado a través del desarrollo de una Actividad. Eso significa, por ejemplo, que voy a enseñarle qué es Python, porqué es importante saberlo. Pero, no le enseñaré el lenguaje Python. Hay excelentes tutoriales en Internet que lo harán, y haré referencias a esas guías.

Hay muchos ejemplos de código en este libro, pero no es necesario que usted los tipee para probarlos. Todo el código está en un repositorio Git que usted podrá descargar en su propia computadora. Si usted nunca ha usado Git hay un capítulo que explica qué es y cómo usarlo.

Comencé a escribir algunas Actividades poco después que recibí mi laptop XO. Cuando empecé no sabía *nada* del material que está en este libro. Era difícil saber por dónde comenzar. Lo que yo tenía a mi favor,sin embargo, eran casi 30 años como programador profesional. Como resultado de eso, pensaba como un programador. Un buen programador puede tomar una tarea compleja y dividirla en partes manejables. Puede imaginar cómo las cosas *deben* trabajar, y a partir de ahí averiguar *cómo* funcionan. Sabe dónde y cómo pedir ayuda. Si no hay un lugar obvio por el cuál empezar, puede empezar en alguna parte y, finalmente, llegar a donde necesita ir.

Por haber realizado este proceso creo que pude hacer una guía bastante buena para la escritura de Actividades en Sugar. En el camino, espero también enseñarle a pensar como un programador.

De vez en cuando, puedo agregar capítulos a este libro. Sugar es una gran plataforma de aplicaciones y este libro sólo puede comenzar a decirle lo que es posible. Tengo la esperanza de que las futuras versiones del libro tendrán capítulos sobre temas más avanzados, escritos por otros desarrolladores de las Actividades.

2. ¿QUÉ ES "SUGAR"?

Sugar es la interfaz de usuario diseñada para la computadora portátil XO. Se puede instalar en la mayoría de las PCs, incluyendo modelos antiguos que no pueden funcionar con el software más reciente de Windows. También puede instalarse en una unidad flash (SoaS: "**S**ugar **o**n **a** **S**tick": Sugar portátil) y arranca en cualquier PC y algunos Macs.

Cuando la computadora portátil XO salió a la luz, algunas personas cuestionaron la necesidad de una nueva interfaz de usuario. ¿No sería mejor para los niños aprender algo más util como lo que utilizarían al ser adultos? ¿Por qué no darles Microsoft Windows en lugar de otra interfaz?

Esta sería una pregunta razonable, si la meta fuera entrenar a los niños a usar las computadoras y nada más. Sería aún más razonable de estar seguros que el software que utilizarán cuando sean adultos lucirá y funcionará como el Microsoft Windows de hoy. Obviamente estas suposiciones no son muy razonables.

El proyecto OLPC no se trata sólo de la enseñanza del conocimiento informático. Se trata de la enseñanza de muchas otras cosas: lectura, escritura, aritmética, historia, ciencias, artes y artesanías, programación informática, composición de música, y mucho más. No sólo esperamos que los niños utilicen las computadoras para sus trabajos escolares, sino que también esperamos que ellos las lleven a sus hogares y las utilicen para sus propias investigaciones de los temas que les interesen.

Ésto es mucho más de lo que cualquiera ha hecho con las computadoras destinadas a la educación. Por eso, es razonable repensar cómo los niños deben trabajar con las computadoras. Sugar es el resultado de esa tarea.

Sugar tiene las siguientes características únicas:

EL DIARIO

El Diario es donde el estudiante puede ver todos sus trabajos. En vez de archivos y de carpetas contiene una lista de entradas organizadas en orden decreciente por la fecha y la hora de última modificación. De alguna manera es como la opción “Documentos recientes” de Windows, excepto que en vez de contener apenas los últimos artículos, contiene a todos y es una manera natural de retomar los trabajos guardados.

El diario hace fácil organizar el trabajo. Cualquier trabajo que usted haga se guarda en el diario. Cualquier descarga del web entra en el diario. Si usted descargó algún archivo usando un browser y después pasó un buen tiempo buscándolo porque lo guardó en un directorio distinto del usual, o si usted tuvo que ayudar a sus padres en una situación similar, entonces comprenderá la utilidad del Diario.

El Diario almacena meta-datos para cada entrada contenida en él. Los meta-datos son información sobre la información. Cada entrada de diario tiene un título, una descripción, una lista de palabras claves, y una imagen de la pantalla de la última vez que la entrada fue utilizada. Mediante un código identificador hace referencia a la Actividad que lo creó, tambien puede contener el tipo MIME (esto permite que entradas del Diario no creadas por una Actividad puedan ser abiertas por una Actividad que soporte el tipo MIME declarado).

Además de los meta-datos genéricos descritos en el párrafo anterior, una entrada de Diario puede contener meta-datos propios de la Actividad que los creó. Por ejemplo, la actividad **Leer** utiliza meta-datos específicos para guardar la página que se leía cuando la actividad se dejo de utilizar. De tal manera que cuando usted vuelva a usarla, esta retornará a la página correcta.

Además de los trabajos creados por las Actividades, el diario puede contener las Actividades mismas. Para instalar una Actividad se puede utilizar la Actividad **Browser**, visitar el Web site http://activities.sugarlabs.org y descargarla. La Actividad será guardada automáticamente en el Diario y quedará lista para usarse. Si usted ya no quiere utilizar la actividad, suprímala simplemente del diario y esta *se elimina totalmente*. No existen programas de desinstalación, ni cajas de diálogo que le preguntan si quiere eliminar tal o cual DLL que al parecer no se necesitará mas. Luego de la eliminación no queda ningún archivo residual.

COLABORACIÓN

La segunda característica única de Sugar es la colaboración. La colaboración significa que las Actividades se pueden utilizar por más de una persona al mismo tiempo. Si bien no toda actividad requiere de esta característica y varias que podrian utilizarla no la soportan, una verdadera Actividad correctamente diseñada proporcionará a los estudiantes la manera de trabajar recíprocamente unos con otros en la red. Por ejemplo, todas las Actividades usadas para la lectura de e-libros proporcionan una manera de compartir una copia del libro que se esta leyendo (con cualesquiera notas que usted haya agregado en ellos) con un amigo o a la clase entera. La actividad **Escribir** deja a varios estudiantes modificar el mismo documento juntos. La actividad **Distancia** permite a dos estudiantes calcular cuan lejos están uno del otro.

Hay cinco vistas del sistema entre las que usted puede intercambiar presionando un botón (teclas F1 a F4). Son:

- La Vista de Vecindario
- La Vista de los Amigos
- El anillo de Actividades
- El Diario

De estas vistas, las primeras dos se utilizan para la colaboración.

La Vista de la Vecindad muestra iconos para cada estudiante en la red. Cada icono es una figura compuesta por una “O” sobre una “X”. Cada icono tiene un nombre, elegido por el estudiante cuando personalizó su computadora. Cada icono se exhibe en uno de dos colores, también elegidos por el estudiante. Además de estos iconos de “XO” habrá iconos que representan otras redes y/o puntos WiFi. Finalmente habrá iconos que representan las Actividades activas que sus dueños desean compartir.

Para entender cómo trabaja, considere la actividad **Conversar**. La forma usual de utilizar una aplicación de conversación consiste en que todos los participantes arrancan un programa cliente de chat e ingresan a una sala de conversación particular al mismo tiempo. Con Sugar es diferente, un estudiante comienza la actividad Conversar en su propia computadora y utiliza la Vista de Vecindario para invitar a otros usuarios en la red a que participen. Estos últimos pueden aceptar la invitación a través de un icono de charla que aparecerá en su propia Vista Vecindario. El acto de aceptar arranca una actividad Conversar propia y de esa manera quedan conectados con los otros participantes.

La Vista de los amigos es similar a la Vista de vecindario, pero contiene iconos solamente para la gente que usted ha señalado como amigos. La colaboración se puede ofrecer en tres niveles: con personas individuales, con la vecindad entera, y con los amigos. Observe que solamente el estudiante puede decidir quién son sus amigos. No hay necesidad de pedir ser amigo de alguien; es como crear una lista de correo en una aplicación email.

SEGURIDAD

La protección de las computadoras contra malos usuarios es muy importante, y si las computadoras pertenecen a los estudiantes se hace doblemente importante. Es también más difícil ya que no se puede esperar que niños recuerden contraseñas y las mantengan secretas. Puesto que Azúcar se ejecuta sobre Linux los virus no son un problema serio, en cambio si pueden serlo Actividades programadas en forma maliciosa. Si a una actividad se le ha permitido acceso irrestricto al Diario, esta podrá por ejemplo eliminar todas sus entradas. De la misma manera podría escribirse una actividad que parezca inofensiva y divertida, pero que luego de que un cierto número de ingresos elimine todo el trabajo del estudiante.

La manera más común de evitar un programa realice operaciones maliciosas, es hacer que se ejecute en un área especial que lo aislé del resto de procesos, a esta área se le llama Sandbox. En otras palabras Sandbox es una manera de limitar qué puede hacer un programa. Con el mecanismo Sandbox usted puede hacer que a un programa poco confiable no se le permita hacer mucho en el computador, o a un programa de confiabilidad verificada se le permitan todos los accesos. La confiabilidad de un programa es avalada por un tercero a través de una *firma electrónica*. La firma electrónica es una operación matemática hecha en el programa es válida solamente si el programa no ha sido modificado.

El Sandbox que provee Azúcar es aún más sofisticada ya que: No requiere que las Actividades sean firmadas. Restringe a las Actividades para que interactuen con el diario de una manera limitada e indirecta. Establece para cada Actividad directorios específicos en los que puede escribir, dando acceso solamente de lectura al resto de archivos y directorios. De esta manera ninguna Actividad puede interferir con el funcionamientos de otra. A pesar de esto, una actividad se puede hacer para hacer lo que necesite hacer.

RESUMEN

Azúcar provee un ambiente diseñado para apoyar la educación de niños. Organiza el trabajo de un niño sin la necesidad de archivos y de carpetas. Soporta la colaboración entre los estudiantes. Finalmente, proporciona un modelo de seguridad robusto para evitar que programas maliciosos dañen el trabajo de un estudiante.

No sería sorprendente ver algún día que estas características sean adoptadas por otros ambientes de escritorio.

3. ¿QUÉ ES UNA ACTIVIDAD DE SUGAR?

Una actividad Sugar es una aplicación autocontenida empaquetada en un paquete .xo.

Un paquete de .xo es un fichero de archivo en el formato .zip. Contiene:

- Un archivo MANIFEST listanto todo lo que hay en el paquete
- Un archivo **activity.info** que tiene los atributos que describen la actividad como pares nombre=valor. Estos atributos incluyen el nombre de actividad, su número de versión, un identificador, y otras cosas que discutiremos cuándo escribamos su primera actividad.
- Un archivo del icono (en formato de SVG)
- Archivos que contienen las traducciones de las secuencias de texto que la actividad utiliza en muchos idiomas
- El código del programa para que funcione la actividad

Una actividad de Sugar tendrá generalmente cierto código del Python que extiende una clase de Python llamada Activity. Puede también hacer uso del código escrito en otros lenguaje si ese código se escribe en una manera que permita que sea utilizado desde Python (se llama esto tener **enlaces de Python**). Incluso es posible escribir una actividad de Sugar sin usar Python en absoluto, pero esto está más allá del alcance de este libro.

Hay unas pocas cosas que una actividad puede depender de ser incluido con cada versión de Sugar. Éstas incluyen módulos como Evince (el pdf y de visualización de otro tipo de documentos), Gecko (representando las páginas Web) y las bibliotecas de Python como PyGTK y PyGame. Todo lo necesario para ejecutar la actividad que no es suministrada por Sugar debe entrar en el archivo del paquete. Una pregunta que a veces se escucha en las listas de correo es "¿cómo hago que Sugar instale *X* la primera vez que mi actividad de ejecuta?" La respuesta: no lo hace. Si necesita *X* tiene que ir en el paquete.

Usted puede instalar una actividad copiándola o descargándola al Diario. La desinstala quitándola del Diario. No hay ningún *protector de instalación* para lidiar con ella, ni para decidir a donde quiere usted instalar los archivos, ninguna posibilidad de que la instalación de una nueva actividad haga que una actividad previamente instalada deje de funcionar.

Una actividad generalmente crea y lee objetos en el Diario. Una actividad de primer nivel proporcionará una cierta manera para que la actividad sea compartida por varios usuarios.

4. ¿QUÉ NECESITO SABER PARA ESCRIBIR UNA ACTIVIDAD DE SUGAR?

Si usted va a escribir Actividades de Sugar debe aprender algo sobre los temas descritos en este capítulo. No es necesario que se convierta en un experto en ninguno de ellos, pero debe guardar sus sitios web y revisar sus tutoriales. Esto le ayudará a entender las muestras del código que veremos.

PYTHON

Python es el lenguaje más usado para escribir Actividades. Aunque puede utilizar otros lenguajes, la mayoría de las Actividades tienen por lo menos algún Python en ellas. Sugar proporciona un API de Python que simplifica la creación de Actividades. Aunque es posible escribir Actividades sin usar Python (como **Etoys**), es inusual.

Todos los ejemplos en este libro están enteramente escritos en Python.

Hay lenguajes compilados y lenguajes interpretados. En un lenguaje compilado el código que usted escribe se traduce al lenguaje de los chips que se ejecutará en y es esta traducción que realmente se ejecute el sistema operativo?. En un lenguaje interpretado hay un programa llamado intérprete que lee el código que usted escribe y hace lo que el código dice que debe hacer. (Esto está sobre simplificado, pero es bastante cercano a la verdad para este capítulo).

Python es un lenguaje interpretado. Hay ventajas a tener un lenguaje compilado y hay ventajas a tener un lenguaje interpretado. Las ventajas que tiene Python para desarrollar Actividades son:

- Es portable. En otras palabras, usted puede hacer que su programa funcione en cualquier procesador? y cualquier SO sin hacer una versión específica para cada uno. Los programas compilados funcionan solamente en el SO y en los procesadores para los que fueron compilados.
- Puesto que el código fuente es lo que se ejecuta, usted no puede dar a alguien un programa en Python sin darle el código fuente. Usted puede aprender mucho sobre la programación de la Actividad estudiando el código de otras personas y hay mucho en él para estudiar.
- Es un lenguaje fácil para que los nuevos programadores aprendan, pero también tiene características que los programadores experimentados necesitan.
- Es ampliamente utilizado. Uno de los usuarios más conocidos de Python es Google. Lo utilizan bastante que han comenzado un proyecto nombrado "~~Unladen Swallow~~"? para hacer que los programas de Python funcionen más rápidamente.

La mayor ventaja de un lenguaje compilado es que puede funcionar mucho más rápido que uno interpretado. Sin embargo, en práctica un programa del Python puede realizarse tan bien como un programa compilado. Para entender porqué esto es así usted necesita entender cómo se hace un programa de Python.

Python es conocido como un lenguaje "pegamento". La idea es que usted tiene componentes escritos en varios lenguajes (generalmente C y C++) y tienen vinculaciones con Python. Python se utiliza para "pegar" estos componentes al crear aplicaciones. En la mayoría de las aplicaciones la mayor parte de las funciones de la aplicación es realizada por estos componentes compilados, y se emplea relativamente poco tiempo ejecutando el código Python que pega los componentes.

Además de Actividades que usan Python la mayor parte del propio ambiente Sugar está escrito en Python.

Si usted ha programado en otros lenguajes hay buenos tutoriales para aprender Python en el sitio web de Python: http://docs.python.org/tutorial/. Si usted está comenzando en la programación usted puede ser que compruebe hacia fuera *inventa sus propios juegos de ordenador con Python*, que usted puede leer en http://inventwithpython.com/.

PYGTK

GTK+ es un sistema de los componentes para crear interfaces de usuario. Estos componentes incluyen cosas como botones, barras de desplazamiento, cajas de lista, y así sucesivamente. Es utilizado por el ambiente de escritorio de GNOME y las aplicaciones que funcionan en él. Las Actividades de Sugar utilizan un tema especial del GNOME que dan a controles de GTK+ un aspecto único.

PyGTK es un sistema de enlaces de Python que le permiten utilizar componentes de GTK+ en programas de Python. Hay un tutorial que muestra cómo utilizarlo en el sitio web de PyGTK: http://www.pygtk.org/tutorial.html.

PYGAME

La alternativa a usar PyGTK para su actividad es PyGame. PyGame puede crear imágenes llamadas sprites y moverlas alrededor en la pantalla. Pues usted puede ser que espere, PyGame se utiliza sobre todo para los juegos de la escritura. Es menos de uso general en actividades que PyGTK.

La clase particular a aprender sobre PyGame está en el Web site de PyGame: http://www.pygame.org/wiki/tutorials. El Web site también tiene un manojo de proyectos del pygame que usted puede transferir y probar.

PROGRAMMING

5. DETERMINACIÓN DE UN AMBIANTE DE DESARROLLO DE SUGAR
6. CREAR SU PRIMERA ACTIVIDAD DEL SUGAR
7. UN PROGRAMA INDEPENDIENTE DEL PYTHON PARA LEER ETEXTS
8. HEREDE DE SUGAR.ACTIVITY.ACTIVITY
9. EMPAQUETE LA ACTIVIDAD
10. AGREGUE LOS REFINAMIENTOS
11. AGREGUE SU CÓDIGO DE LA ACTIVIDAD AL CONTROL DE VERSIÓN
12. INTERNATIONAL QUE VA CON POOTLE
13. DISTRIBUYA SU ACTIVIDAD
14. ACTIVIDADES DEL SUGAR DEL DEPURACIÓN

5. DETERMINACIÓN DE UN AMBIANTE DE DESARROLLO DE SUGAR

Actualmente no es práctico desarrollar las Actividades para XO en la XO. No es tanto que usted no pueda hacerlo, pero es más fácil y más productivo hacer su desarrollo y la prueba en otra máquina que funciona con un Sistema Operativo más convencional. Esto le da el acceso a mejores herramientas y también le permite simular la colaboración entre dos computadoras que funcionan con Sugar usando solamente una computadora.

¿INSTALE EL LINUX O UTILICE UNA MÁQUINA VIRTUAL?

Aunque Sugar funciona en Linux, es posible hacerlo funcionar completamente en una máquina virtual que funcione en Windows. Una máquina virtual es una manera en que puede usar un sistema operativo encima otro. El sistema operativo que es funcionado se engaña en el pensamiento de ella tiene la computadora entera a sí mismo. (Los pandit de la industria del ordenador le dirán que ésa usando las máquinas virtuales es la más nueva nueva cosa hacia fuera allí. Los viejos como mí saben que IBM la hacía en sus ordenadores centrales detrás en los años 70).

Para esto estaba un rato realmente la manera recomendada de desarrollar Actividades. La versión de Linux que Sugar utilizó era bastante diferente de distribuciones regulares del Linux que incluso los usuarios del Linux funcionaban con Sugar en una máquina virtual encima de Linux.

La situación ha mejorado, y la mayoría de las distribuciones actuales del Linux tienen un ambiente usable del Sugar.

Si le utilizan a Windows usted puede piensar que Sugar de funcionamiento en una VM de Windows en vez de instalar Linux puede ser la opción más fácil. No está en la práctica. El Linux que funciona en una VM sigue siendo Linux, así que usted todavía va a tener que aprender algunas cosas sobre Linux para hacer el desarrollo de la Actividad. También, funcionar con un segundo OS en una VM requiere una máquina múy poderoso con gigabytes de la memoria. Por una parte, hago mi desarrollo de Sugar usando Linux en un Pentium IV que compré usado de IBM para un poco sobre cientos dólares, incluyendo envío y NetVista incluido. Es más que adecuado.

La instalación de Linux no es la prueba de machismo que estaba una vez. Cualquier persona puede hacerla. La mesa del GNOME qúe viene con Linux es múy similar al Windows y por eso usted sentirá muy comodo usando él.

Cuando usted instala Linux, usted tiene la opción para hacer un cargador dual, un Linux corriente y Windows en la misma computadora (pero no al mismo tiempo). Esto significa que usted pone una partición de disco a un lado para uso de Linux y cuando usted enciende la computadora un menú aparece que pregunta qué OS usted quiere comenzar para arriba. El Linux instala incluso creará la partición para usted, y un par de gigabytes son más que bastante espacio de disco. La distribución de una computadora con una instalación del Linux no afectará a su instalación de Windows en ninguna manera.

Sugar Labs han estado trabajando para conseguir sugar incluido con todas las distribuciones del Linux. Si usted tiene ya una distribución preferida, las ocasiones son la última versión de ella incluyen Sugar. Fedora, openSuse, Debian, y Ubuntu todo incluyen Sugar. Si usted ya utiliza Linux, vea si el Sugar se incluye en su distribución. Si no, Fedora es qué es utilizada por la computadora de XO así que Fedora 10 o más adelante pudo ser su lo mejor que se puede hacer. Usted puede transferir Fedora instala el CD o DVD aquí:

https://fedoraproject.org/get-fedora.

Vale a notar que todas las otras herramientas que estoy recomendando están incluidas en cada distribución del Linux, y pueden ser instaladas sin más esfuerzo que comprobando una caja de cheque. Las mismas herramientas funcionarán a menudo en Windows, pero la instalación de ellas allí es más trabajo que usted esperaría para los programas de Windows.

Si usted no desea instalar y aprender sobre Linux pero todavía querer desarrollar Actividades, una opción usted tiene debe desarrollar un programa independiente del Python que utilice PyGame de PyGTK y hace lo que usted quisiera que su Actividad hiciera. Usted podría entonces volcar su programa a alguín otro que podría convertirlo en una Actividad de Sugar. Usted podría desarrollar tal programa de Python sobre Windows o sobre Macintosh.

Si usted quiere hacer el desarrollo en Macintosh con Sugar corriente en una máquina virtual puede ser una opción más atractiva. Si usted quiere tratarlo los detalles serán encontrados aquí: http://wiki.laptop.org/go/Developers/Setup. También, puede ser posible instalar Linux de Fedora en Intel o Power PC Macintosh como cargador dual, apenas como usted puede hacer con Windows. Compruebe el Web site de Fedora para saber si hay detalles.

Otra opción para los usuarios del Mac es utilizar el *Sugar en un palillo* como ambiente de prueba. Usted puede aprender sobre eso aquí: http://wiki.sugarlabs.org/go/Sugar_on_a_Stick.

¿QUÉ ACERCA DEL USO DEL SUGAR-JHBUILD?

El sugar-jhbuild es una escritura que transfiere el código fuente para la última versión de todos los módulos de Sugar y lo compila en un sub-directório de su directorio casero. No instala realmente el Sugar en su sistema. En lugar, usted lo usa del directorio que usted lo instaló adentro. Debido a la manera se construye y funcionamiento que no interfiere con los módulos que componen su mesa normal. Si usted está desarrollando el Sugar sí mismo, o si usted se está convirtiendo las Actividades que dependen de las características muy últimas de Sugar usted necesitarán funcionar con el sugar-jhbuild.

Funcionar con esta escritura es un poco más difícil que apenas instalando los paquetes de Sugar que vienen con la distribución. Usted necesitará instalar Git y Subversión, funciona con un comando de Git del terminal de transferir la escritura de sugar-jhbuild, después funciona con la escritura con varias diversas opciones que transfieran más código, pide que usted instale más paquetes, y compila en última instancia todo. Puede tardarle unas par de horas para hacer todos los pasos. Cuando le hacen usted tendrá un ambiente de prueba hasta la fecha que usted pueda funcionar como alternativa al **sugar-emulator**. No hay necesidad de desinstalar Sugar-emulador; ambos pueden coexistir.

Usted puede hacerlo con éstos comandos:

```
cd sugar-jhbuild
./sugar-jhbuild run sugar-emulator
```

¿Debe usted considerar usarlo? La respuesta corta es No. Una respuesta más larga está *probablemente no todavía*.

Si usted quisiera que sus Actividades alcanzaran a la audiencia posible más ancha usted no quiere el más último Sugar. De hecho, si usted quiere un ambiente de prueba que mímico cuál es en la mayoría de las computadoras de XO ahora usted necesidad de utilizar Fedora 10. Porque la puesta al día de sistemas operativos en el campo puede ser una empresa importante para una escuela la mayoría de los XO serán usando Sugar .82 o más viejos por mucho tiempo.

Por supuesto es también importante tener programadores que quieran empujar los límites de lo que Sugar puede hacer. Si, después de desarrollar algunas Actividades, usted decider desea ser una de ellas, usted

puede aprender sobre el azúcar-jhbuild corriente aquí: http://wiki.sugarlabs.org/go/DevelopmentTeam/Jhbuild.

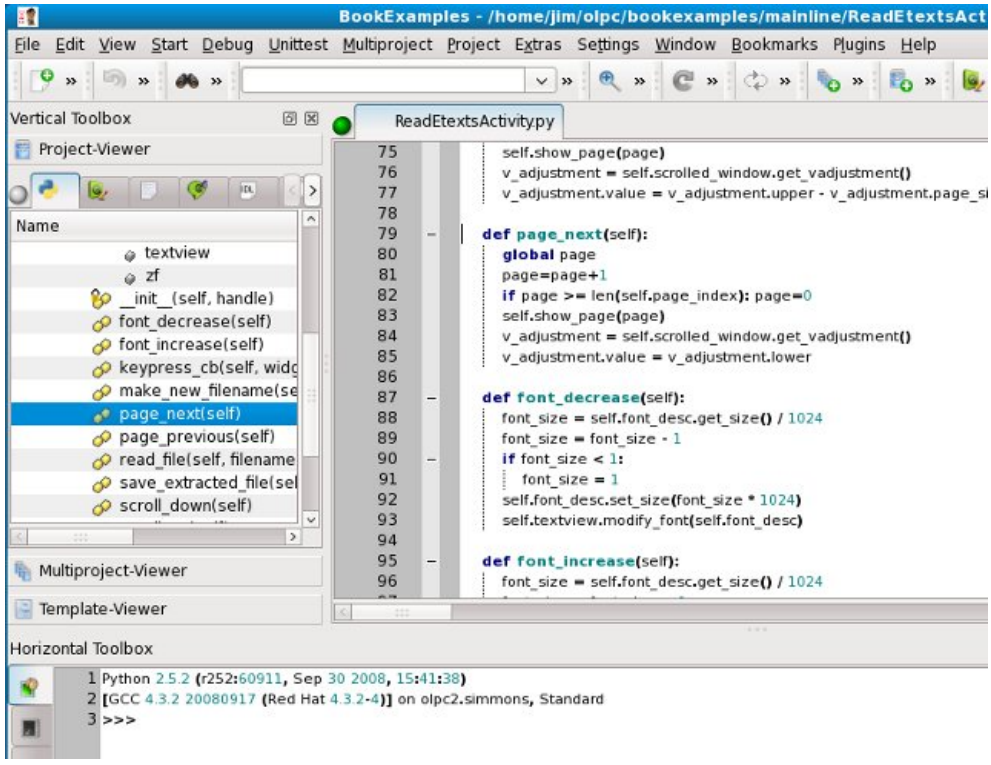
En realidad Sugar-jhbuild es solamente la escritura que transfiere y compila Sugar. Si usted quisiera estar correcto usted diría "funcionamiento la copia del **sugar-emulator** que usted hizo con el azúcar-jhbuild". La mayoría de los programadores del Sugar decen solamente que el "Sugar-jhbuild del funcionamiento" y ése es lo que diré en este libro.

PYTHON

Haremos todas las muestras del código en Python así que usted necesitará hacer el Python instalar. Python viene con cada distribución de Llinux. Usted puede transferir los instaladores para Windows y Macintosh en http://www.python.org/.

ERIC

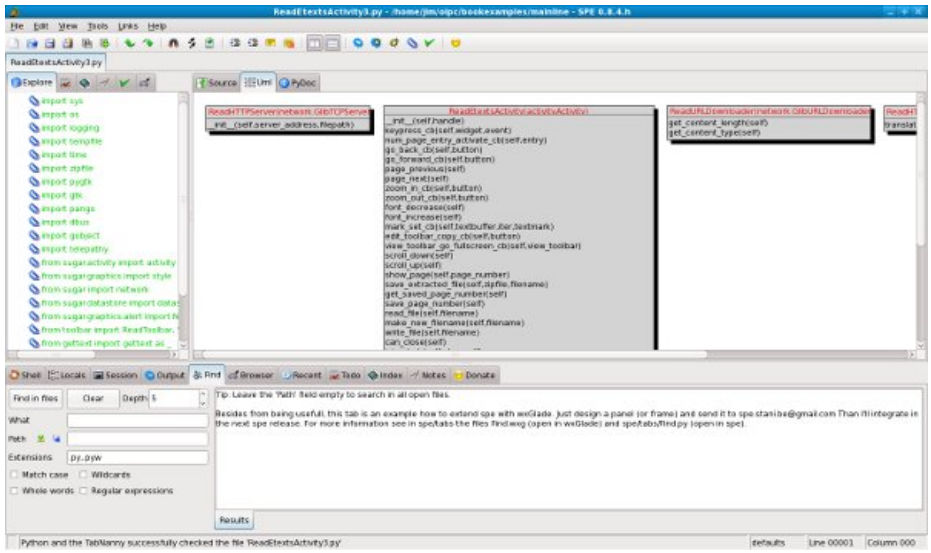
Hoy en dia los programadores esperan que sus idiomas que se apoyarán por un **entorno de desarrollo integrado (IDE)** y no hay excepción con Python. Un
IDE ayuda a organizar su trabajo y proporciona la edición y de textos construidos en el sistema de programación y de herramientas de puesta a punto.



Hay dos IDEs para Python que yo ha intentado: Eric y Idle. Eric es el más elaborado de los dos y lo recomiendo. Cada distribución del linux debe incluirlo. Parece ella puede ser que va a funcionar en Windows también. Usted puede aprender más sobre ella en el Web site de Eric: http://eric-ide.python-projects.org/.

SPE (REDACTOR DEL PYTHON DE STANI)

Esto es un IDE que descubrí mientras que escribía este libro. Viene con Fedora y además de ser un redactor de Python él hará diagramas de **UML de** su código y demostrará **PyDoc** para él. Aquí está el SPE que demuestra un diagrama de UML para una de las Actividades en este libro:



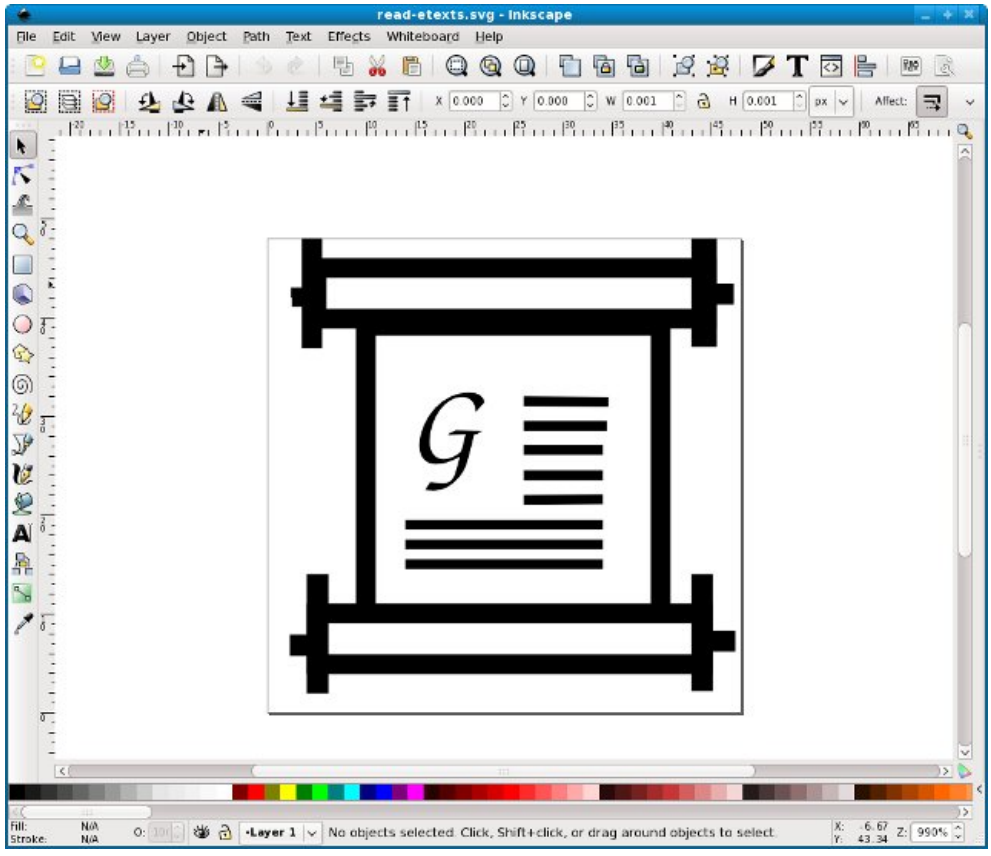
Si usted es programador experimentado usted puede preferir esto una alternativa útil a Eric. Si usted apenas está comenzando Eric debe cubrir sus necesidades bastante bien.

OTROS IDES

También hay un comercial IDE para Python llamado Wingware, que tiene una versión que usted puede utilizar para libre. Usted puede aprender más sobre él en http://www.wingware.com/.

INKSCAPE

Inkscape es una herramienta para crear imágenes en formato de SVG. Sugar utiliza SVG para los iconos de la actividad y otras clases de ilustraciones. El icono de "XO" que representa cada niño en la vista de la vecindad es un archivo de SVG que puede ser modificado.



Inkscape viene con cada distribución del linux, y se puede instalar en Windows también. Usted puede aprender más sobre él aquí: http://www.inkscape.org/.

GIT

Git es un sistema de control de versión. Guarda versiones de su código del programa en una manera que las haga fáciles volver. Siempre que usted realice cambios a su código usted pide Git para almacenar su código en su depósito. Si usted necesita mirar una vieja versión de ese código más adelante usted puede. Incluso mejore, si un cierto problema aparece en su código que usted puede comparar su último código a un viejo versión de su trabajo y considerar exactamente qué líneas usted cambió.

readtoolbar.py

```
15  15 # along with this program; if not, write to the Free Software
16  16 # Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
17  17
    18 import os
18  19 import logging
19  20 from gettext import gettext as _
20  21 import re
... ...
416 416         combotool.show()
417 417
418 418         self.pitchadj = gtk.Adjustment(0, -100, 100, 1, 10, 0)
419         self.pitchadj.connect("value_changed", self.pitch_adjusted_cb)
420 419         pitchbar = gtk.HScale(self.pitchadj)
421 420         pitchbar.set_draw_value(False)
422 421         pitchbar.set_update_policy(gtk.UPDATE_DISCONTINUOUS)
... ...
427 427         pitchbar.show()
428 428
429 429         self.rateadj = gtk.Adjustment(0, -100, 100, 1, 10, 0)
430         self.rateadj.connect("value_changed", self.rate_adjusted_cb)
431 430         ratebar = gtk.HScale(self.rateadj)
432 431         ratebar.set_draw_value(False)
433 432         ratebar.set_update_policy(gtk.UPDATE_DISCONTINUOUS)
... ...
453 453     def pitch_adjusted_cb(self, get):
454 454         speech.pitch = int(get.value)
455 455         speech.say(_("pitch adjusted"))
    456         f = open(os.path.join(self.activity.get_activity_root(), 'insta
```

Si hay dos personas que trabajan en el mismo programa independientemente un sistema de control de versión combinará sus cambios juntos automáticamente.

Suponga que usted está trabajando en una nueva versión importante de su actividad cuando alguien encuentra un bug realmente embarazoso en la versión que usted acaba de lanzar. Si usted utiliza Git usted no necesita decir a gente que nesesita vivir con él hasta el lanzamiento siguiente, que podría ser meses lejos. En lugar usted puede crear una rama de la versión previa y del trabajo sobre él junto a la versión que usted está realzando. En efecto Git trata la vieja versión que usted está corrigiendo y la versión usted está mejorando como dos proyectos separados.

Usted puede aprender más sobre Git en el Web site de Git: http://git-scm.com/.

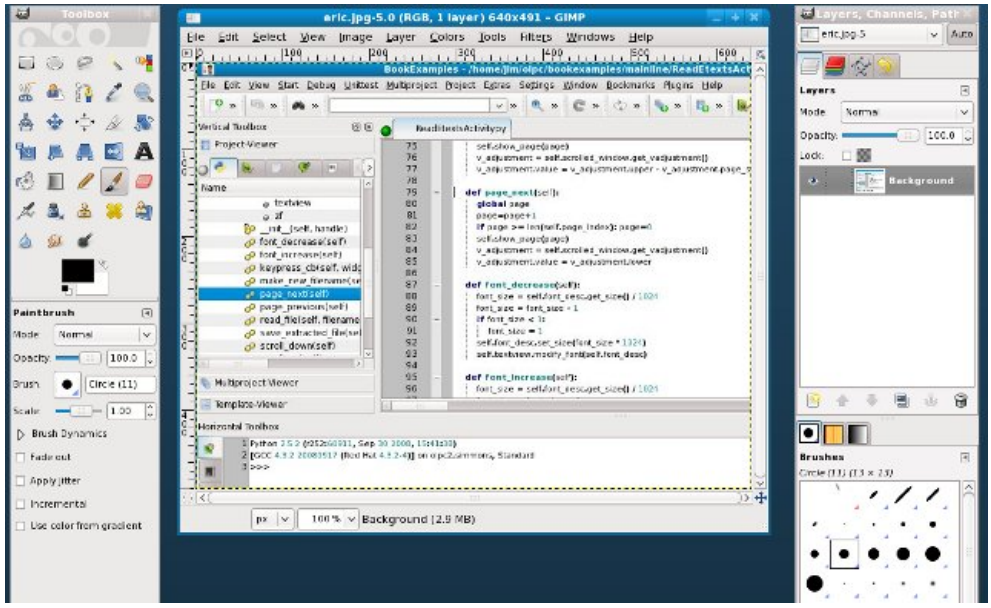
Cuando usted está listo para un depósito de Git para su proyecto usted puede instalar uno aquí: http://git.sugarlabs.org/. Diré más sobre la creación y usar un depósito de Git más adelante en este libro.

Hay un depósito de Git que contiene todos los ejemplos del código de este libro. Cuando usted tiene Git instalado, usted puede copiar el depósito a su computadora con este comando:

```
git clone git://git.sugarlabs.org/myo-sugar-activities-examples/mainline.git
```

EL GIMP

El GIMP es uno de los programas más útiles y mal nombrados que fue hecho nunca. Usted puede pensar en él como versión libre de Adobe Photoshop. Si usted necesita trabajar con los archivos de imagen (con excepción de SVG) usted necesita este programa.

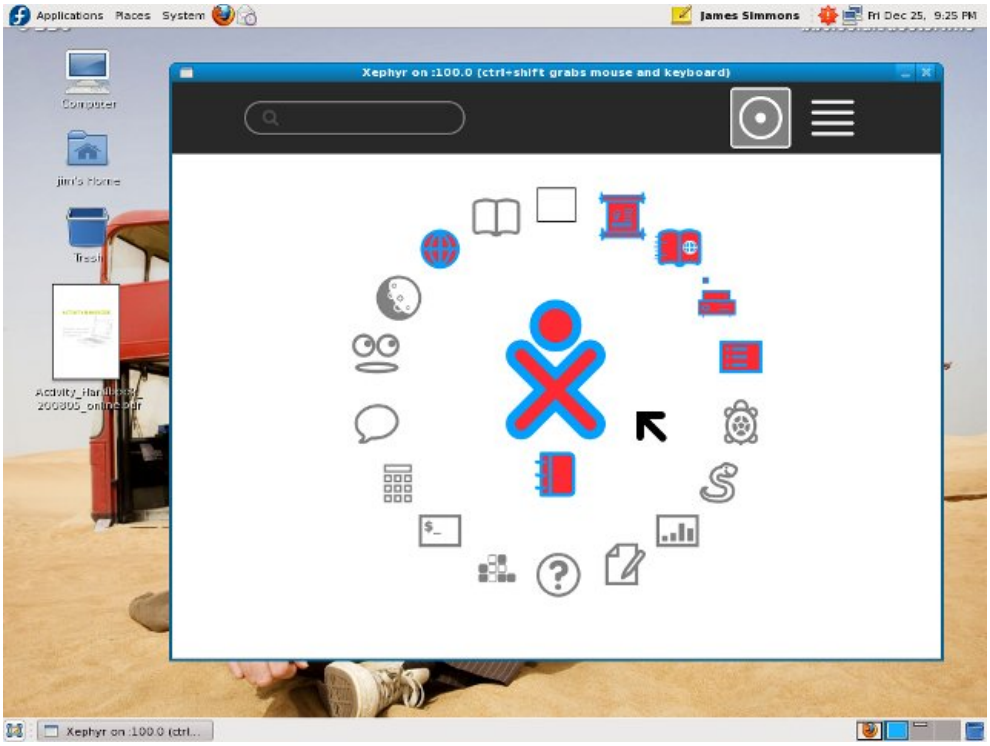


Usted puede nunca necesitar este programa desarrollar su Activad, pero cuando es hora de distribuir la Actividad usted la utilizará para crear tiros de pantalla de su Actividad en la acción. Nada vende una Actividad a un usuario potencial como buenos tiros de pantalla.

EMULACIÓN DE SUGAR

La mayoría de las distribuciones de Linux deben incluir Sugar. En Fedora usted puede funcionar con Sugar como ambiente de escritorio alternativo. Cuando usted abre una sesión a GDM Sugar aparece como selección de escritorio junto a GNOME, a KDE, a Window Maker, y a cualquier otro encargado de ventana que usted haya instalado.

Ésta no es la manera normal de utilizar Sugar para la prueba. La manera normal utiliza una herramienta llamada Xephyr para funcionar un ambiente de Sugar en una ventana en su mesa. En efecto, Xephyr funciona con una sesión de X dentro de una ventana y Sugar funciona en ése. Usted puede tomar fácilmente capturas de pantalla de Sugar en sesiones de Sugar de la acción, de la parada y del recomenzar sin el recomienzo de la computadora, y funciona con copias múltiples de sugar para probar la colaboración.



Tendré más a decir sobre esto cuando es hora de probar su primera actividad.

6. CREAR SU PRIMERA ACTIVIDAD DEL SUGAR

HAGA QUE UN PYTHON INDEPENDIENTE PROGRAMA PRIMERO

El mejor consejo que podría dar un programador del principio de una Actividad es hacer una versión de su Actividad que pueda funcionar con en sus los propios, fuera del ambiente del Sugar. Probar y depurar un Python programa que los soportes solamente son más rápidos, más fáciles y menos aburridos que haciendo la misma cosa con una Actividad similar. Usted entenderá porqué cuando usted comienza a probar su primera Actividad.

Más bugs que usted encuentra antes de que usted dé vuelta a su código en una Actividad el mejor. De hecho, es una buena idea guardar una versión independiente de su programa alrededor incluso después usted tiene la versión de la Actividad en curso. Utilicé mi versión independiente de **Read Etexts** para desarrollar el texto al discurso con destacar la característica. Esto me ahorró mucho tiempo, que era especialmente importante porque imaginaba cosas mientras que programé.

Nuestro primer proyecto será una versión de la Read Etexts Actividad que escribí.

HEREDE DEL SUGAR.ACTIVITY. CLASE DE LA ACTIVIDAD

Después, vamos a tomar nuestro programa independiente de Python y transformelo a una Actividad. Para hacer esto que necesitamos entender el concepto de *herencia*. En discurso diario la herencia significa conseguir algo de sus padres para quienes usted no trabajó. Un rey llevará a su hijo a una ventana del castillo y decir, “algún día, mijo, éste todo será el suyo." Ésa es herencia.

En el mundo de los programas de computadoras puede tener padres y heredar cosas de ellas. En vez de heredar la característica, heredan código. Hay un pedazo de Python código llamado sugar.activity.Activity que es el mejor padre que una Actividad podría esperar para tener, y nosotros vamos a convencerla adoptar nuestro programa. Esto no significa que nuestro programa nunca tendrá que trabajar otra vez, pero no tendrá que trabajar tanto.

EMPAQUETE LA ACTIVIDAD

Ahora nescesitamos empaquetar nuestro código para hacerle algo que se puede funcionar debajo de Sugar y distribuirlo como un archivo de .xo. Esto implica hacer un MANIFEST, activity.info, setup.py, y de crear un icono conveniente con Inkscape.

AGREGUE LOS REFINAMIENTOS

Cada Actividad tendrá la barra de herramientas básica de la Actividad. Para la mayoría de las Actividades éste no será bastante, así que necesitaremos crear algunas barras de herramientas especial también. Entonces necesitamos engancharlas hasta el resto del código de la Actividad para qué sucede a las acciones de los disparadores de la barra de herramientas en la Actividad y qué sucede fuera de la barra de herramientas reflejar en el estado de la barra de herramientas.

Además de barras de herramientas, miraremos algunas otras maneras al hacer su Actividad especial.

PONGA EL CÓDIGO DE PROYECTO EN CONTROL DE VERSIÓN

Para entonces tendremos bastante código escrito que vale el proteger y el compartir con el mundo. Para hacer ésto necesitamos crear un depósito de Git y agregar nuestro código a él. También pasaremos los fundamentos de usar Git.

VAMANOS INTERNATIONAL QUE VA CON POOTLE

Ahora que nuestro código está en Git podemos pedir ayuda de nuestro primer colaborador: el sistema de la traducción de Pootle. Con un pequeño trabajo de la disposición podemos conseguir a voluntarios hacer versiones traducidas de nuestra Actividad disponibles.

DISTRIBUCIÓN DE LA ACTIVIDAD

En esta tarea tomaremos nuestra Actividad y ponerla en http://activities.sugarlabs.org más nosotros empaquetaremos encima del código fuente así que puede ser incluido en distribuciones del Linux.

AGREGUE LA COLABORACIÓN

Agregaremos después código para compartir los e-books con los amigos y la vecindad.

AGREGUE EL TEXTO A VOZ

El texto a voz con destacar de la palabra es siguiente. ¡Nuestro proyecto simple se convertirá en un Kindle-asesino!

7. UN PROGRAMA INDEPENDIENTE DEL PYTHON PARA LEER ETEXTS

EL PROGRAMA

Nuestro programa del ejemplo se basa en la primera Actividad que escribí, que leí Etexts. Esto es un programa para leer los e-libros libres.

La más vieja y mejor fuente de e-libros libres es un Web site llamado Project Gutenberg (http://www.gutenberg.org/wiki/Main_Page). Crean los libros en formato de texto llano, es decir la clase de archivo que usted podría hacer si usted escribió un libro en Notepad y golpeó "Enter" en el extremo de cada línea. Tienen millares de libros que estén fuera de los derechos reservados, incluyendo algunos qué son el mejor escrito nunca. Antes de que usted lea más, vaya a ese Web site y seleccione un libro que los intereses usted. Compruebe hacia fuera la lista de los “Mejor 100” para ver los libros y a los autores más populares.

El programa que vamos a crear leerá los libros en formato de texto llano solamente.

Hay un depósito de Git que contiene todos los ejemplos del código en este libro. Una vez que usted tiene Git instalado usted puede copiar el depósito a su computadora con este comando:

```
git clone git://git.sugarlabs.org/myo-sugar-activities-examples/mainline.git
```

El código para nuestro programa independiente de Python será encontrado en el directorio **Make_Standalone_Python** en un archivo nombrado **ReadEtexts.py**. Parece esto:

```
#! /usr/bin/env python
import sys
import os
import zipfile
import pygtk
import gtk
import getopt
import pango

page=0
PAGE_SIZE = 45

class ReadEtexts():

    def keypress_cb(self, widget, event):
        "Respond when the user presses one of the arrow keys"
        keyname = gtk.gdk.keyval_name(event.keyval)
        if keyname == 'plus':
            self.font_increase()
            return True
        if keyname == 'minus':
            self.font_decrease()
            return True
        if keyname == 'Page_Up' :
            self.page_previous()
            return True
        if keyname == 'Page_Down':
            self.page_next()
            return True
        if keyname == 'Up' or keyname == 'KP_Up' \
                or keyname == 'KP_Left':
            self.scroll_up()
            return True
        if keyname == 'Down' or keyname == 'KP_Down' \
```

```
            or keyname == 'KP_Right':
        self.scroll_down()
        return True
    return False

def page_previous(self):
    global page
    page=page-1
    if page < 0: page=0
    self.show_page(page)
    v_adjustment = self.scrolled_window.get_vadjustment()
    v_adjustment.value = v_adjustment.upper - v_adjustment.page_size

def page_next(self):
    global page
    page=page+1
    if page >= len(self.page_index): page=0
    self.show_page(page)
    v_adjustment = self.scrolled_window.get_vadjustment()
    v_adjustment.value = v_adjustment.lower

def font_decrease(self):
    font_size = self.font_desc.get_size() / 1024
    font_size = font_size - 1
    if font_size < 1:
        font_size = 1
    self.font_desc.set_size(font_size * 1024)
    self.textview.modify_font(self.font_desc)

def font_increase(self):
    font_size = self.font_desc.get_size() / 1024
    font_size = font_size + 1
    self.font_desc.set_size(font_size * 1024)
    self.textview.modify_font(self.font_desc)

def scroll_down(self):
    v_adjustment = self.scrolled_window.get_vadjustment()
    if v_adjustment.value == v_adjustment.upper - \
            v_adjustment.page_size:
        self.page_next()
        return
    if v_adjustment.value < v_adjustment.upper - v_adjustment.page_size:
        new_value = v_adjustment.value + v_adjustment.step_increment
        if new_value > v_adjustment.upper - v_adjustment.page_size:
            new_value = v_adjustment.upper - v_adjustment.page_size
        v_adjustment.value = new_value

def scroll_up(self):
    v_adjustment = self.scrolled_window.get_vadjustment()
    if v_adjustment.value == v_adjustment.lower:
        self.page_previous()
        return
    if v_adjustment.value > v_adjustment.lower:
        new_value = v_adjustment.value - \
            v_adjustment.step_increment
        if new_value < v_adjustment.lower:
            new_value = v_adjustment.lower
        v_adjustment.value = new_value

def show_page(self, page_number):
    global PAGE_SIZE, current_word
    position = self.page_index[page_number]
    self.etext_file.seek(position)
    linecount = 0
    label_text = '\n\n\n'
    textbuffer = self.textview.get_buffer()
    while linecount < PAGE_SIZE:
        line = self.etext_file.readline()
        label_text = label_text + unicode(line, 'iso-8859-1')
        linecount = linecount + 1
    label_text = label_text + '\n\n\n'
    textbuffer.set_text(label_text)
    self.textview.set_buffer(textbuffer)

def save_extracted_file(self, zipfile, filename):
```

```
        "Extract the file to a temp directory for viewing"
        filebytes = zipfile.read(filename)
        f = open("/tmp/" + filename, 'w')
        try:
            f.write(filebytes)
        finally:
            f.close

    def read_file(self, filename):
        "Read the Etext file"
        global PAGE_SIZE

        if zipfile.is_zipfile(filename):
            self.zf = zipfile.ZipFile(filename, 'r')
            self.book_files = self.zf.namelist()
            self.save_extracted_file(self.zf, self.book_files[0])
            currentFileName = "/tmp/" + self.book_files[0]
        else:
            currentFileName = filename

        self.etext_file = open(currentFileName,"r")
        self.page_index = [ 0 ]
        linecount = 0
        while self.etext_file:
            line = self.etext_file.readline()
            if not line:
                break
            linecount = linecount + 1
            if linecount >= PAGE_SIZE:
                position = self.etext_file.tell()
                self.page_index.append(position)
                linecount = 0
        if filename.endswith(".zip"):
            os.remove(currentFileName)

    def destroy_cb(self, widget, data=None):
        gtk.main_quit()

    def main(self, file_path):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("destroy", self.destroy_cb)
        self.window.set_title("Read Etexts")
        self.window.set_size_request(640, 480)
        self.window.set_border_width(0)
        self.read_file(file_path)
        self.scrolled_window = gtk.ScrolledWindow(hadjustment=None, \
                                                  vadjustment=None)
        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_left_margin(50)
        self.textview.set_cursor_visible(False)
        self.textview.connect("key_press_event", self.keypress_cb)
        buffer = self.textview.get_buffer()
        self.font_desc = pango.FontDescription("sans 12")
        font_size = self.font_desc.get_size()
        self.textview.modify_font(self.font_desc)
        self.show_page(0)
        self.scrolled_window.add(self.textview)
        self.window.add(self.scrolled_window)
        self.textview.show()
        self.scrolled_window.show()
        v_adjustment = self.scrolled_window.get_vadjustment()
        self.window.show()
        gtk.main()

if __name__ == "__main__":
    try:
        opts, args = getopt.getopt(sys.argv[1:], "")
        ReadEtexts().main(args[0])
    except getopt.error, msg:
        print msg
        print "This program has no options"
        sys.exit(2)
```

FUNCIONAR CON EL PROGRAMA

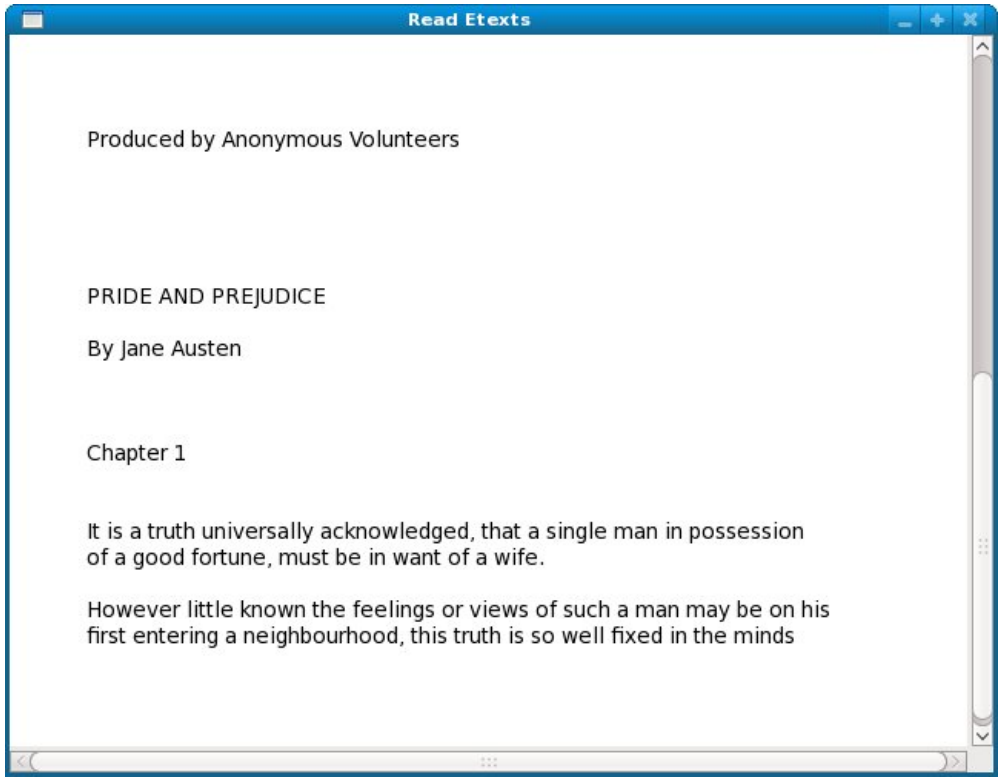
Para funcionar con el programa usted debe primero hacerlo ejecutable. Usted necesita solamente hacer esto una vez:

```
chmod 755 ReadEtexts.py
```

Para este ejemplo, yo transferido el archivo para el *orgullo y el prejudicar*. El programa trabajará con cualquiera de los formatos de texto llano, que son texto sin comprimir o un archivo de ZIP. El archivo de ZIP se nombra **1342.zip**, y podemos leer el libro funcionando esto de un terminal:

```
./ReadEtexts.py 1342.zip
```

Esto es lo que parece el programa en la acción:



Usted puede consumir la *página, la página abajo, encima, abajo, de* llaves *izquierdas*, y *correctas* para navegar a través del libro y "+" y "-" las llaves para ajustar el tamaño de fuente.

CÓMO EL PROGRAMA TRABAJA

Este programa lee a través del archivo de texto que contiene el libro y lo divide en las páginas de 45 líneas cada uno. Necesitamos hacer esto porque el componente de **gtk.TextView que** utilizamos para ver el texto necesitaría mucha memoria enrollar a través del libro entero y ése lastimaría funcionamiento. Una segunda razón es que queremos hacer la lectura del e-libro tanto cuanto sea posible como la lectura de un libro regular, y los libros regulares tienen páginas. Si un profesor asigna la lectura de un libro ella puede ser que diga las "páginas leídas 35-50 para la man¢ana". Finalmente, quisiéramos que este programa recordara qué página usted paró el seguir leyendo y le trae de nuevo a esa página otra vez cuando usted leyó el libro

la vez próxima. (El programa que tenemos no hace hasta ahora eso todavía).

Para paginar a través del libro que utilizamos **de acceso aleatorio** para leer el archivo. Para entender lo que consideran los medios de acceso aleatorio a un archivo, una cinta del VHS y un DVD. Para conseguir a cierta escena en un VHS grábele necesidad de pasar con todas las escenas que vinieron antes de ella, en orden. Aunque usted la hace en la velocidad usted todavía tiene que mirar todos para encontrar el lugar que usted quiere comenzar a mirar. Esto es **de acceso secuencial**. Por una parte un DVD tiene paradas del capítulo y posiblemente un menú del capítulo. Usando un menú del capítulo usted puede mirar cualquier escena en la película enseguida, y usted puede saltar alrededor mientras que usted tiene gusto. Esto es de acceso aleatorio, y el menú del capítulo es como un **índice** Por supuesto usted puede tener acceso al material en un DVD secuencialmente también.

Necesitamos de acceso aleatorio saltar a cualquier página tenemos gusto, y necesitamos un índice de modo que sepamos dónde cada página comienza. Hacemos el índice leyendo la línea entera del archivo uno a la vez. Cada 45 líneas anotamos cuántos caracteres en el archivo hemos conseguido y almacenamos esta información en una lista de Python. Entonces volvemos al principio del archivo y exhibimos la primera página. Cuando el usuario del programa va a la página siguiente o anterior entendemos cuál será la nueva página y miramos en la entrada de lista para esa página. Esto nos dice que la página comienza 4.200 caracteres en el archivo. Utilizamos búsqueda () en el archivo para ir a ese carácter y después leemos 45 líneas que comienzan en ese punto y las cargamos en el TextView.

Cuando usted funciona este aviso del programa cómo rápidamente está. Los programas del Python duran para funcionar con una línea de código que una lengua compilada, pero en este programa no importa porque la elevación pesada en el programa es hecha por el TextView, que fue creado en una lengua compilada. Las piezas de Python no hacen que mucha el programa no pasa tan mucha hora que las funciona.

Sugar utiliza Python mucho, no apenas para las Actividades pero para el ambiente sí mismo del Sugar. Usted puede leer en alguna parte eso usando tanto Python es “un desastre” para el funcionamiento. No lo crea.

No hay lenguajes de programación lentos, solamente programadores lentos

8. HEREDE DE SUGAR.ACTIVITY.ACTIVITY

PYTHON ORIENTADO AL OBJETO

El Python apoya dos estilos de la programación: **procesal** y **orientado al objeto**. La programación procesal es cuando usted tiene ciertos datos de entrada, hace alguno que procesa en ella, y produce una salida. Si usted quiere calcular todos los números primeros debajo de ciento o convertir un documento de la palabra en un archivo de texto llano usted utilizará probablemente el estilo procesal para hacer eso.

Los programas orientados al objeto se aumentan de las unidades llamadas los **objetos**. Un objeto se describe como colección de campos o de cualidades que contienen datos junto con los métodos para hacer cosas con eso los datos. Además de hacer el trabajo y de almacenar objetos de datos puede enviar los mensajes a uno otros.

Considere un programa de procesamiento de textos. No hace apenas uno entrado, un cierto proceso, y uno hacer salir. Puede recibir la entrada del teclado, de los botones de ratón, del ratón que viaja sobre algo, del sujetapapeles, del etc. Puede enviar salida a la pantalla, a un archivo, a una impresora, al sujetapapeles, al etc. Un procesador de textos puede corregir varios documentos al mismo tiempo también. Cualquier programa con un GUI es un ajuste natural para el estilo orientado al objeto de la programación.

Los objetos son descritos por las *clases*. Cuando usted crea un objeto usted está creando un *caso de una* clase.

Hay una otra cosa que una clase puede hacer, que es **heredar** métodos y cualidades de otra clase. Cuando usted define una clase usted puede decir que **extiende una** cierta clase, y haciendo que en efecto su clase tiene la funcionalidad de la otra clase más su propia funcionalidad. La clase extendida se convierte en su padre.

Todas las Actividades de Sugar extienden una clase del Python llamada **sugar.activity.Activity**. Esta clase proporciona los métodos que todas las Actividades necesitan. Además de eso, hay los métodos que usted puede eliminar en su propia clase que la clase de padre llame cuando necesita. Para el escritor tres de la Actividad del principio los métodos son importantes:

__init()

Se llama esto cuando su Actividad se comienza para arriba. Aquí es donde usted fijará el interfaz utilizador para su Actividad, incluyendo barras de herramientas.

read_file(uno mismo, file_path)

Se llama esto cuando usted reasume una Actividad de una entrada de diario. Se llama después de que se llame el método del *__init ()*. El parámetro del file_path contiene el nombre de un fichero temporal que sea una copia del archivo en la entrada de diario. Se suprime el archivo tan pronto como este método acabe, pero porque el Sugar funciona en Linux si usted abre el archivo para leer su programa puede continuar leyéndola incluso después se suprime y el archivo no saldrá realmente hasta que usted lo cierre.

write_file(uno mismo, file_path)

Se llama esto cuando la Actividad pone al día la entrada de diario. Apenas como con *read_file()* su actividad no trabaja con el diario directamente. En lugar abre el archivo nombrado en el file_path para la salida y le escribe. Ese archivo alternadamente se copia a la entrada de diario.

Hay tres cosas que pueden hacer *write_file()* ser ejecutado:

- Su Actividad se cierra.
- Alguien presiona el botón de la **subsistencia** en la barra de herramientas de la Actividad.
- Su actividad deja de ser la actividad activa, o alguien se mueve desde la opinión de la actividad a una cierta otra visión.

Además de poner al día el archivo en la entrada de diario *() los* métodos *read_file ()* y *write_file* se utilizan para leer y para poner al día los meta datos en la entrada de diario.

Cuando convertimos nuestro programa independiente de Python a una Actividad sacaremos mucho del código que escribimos y lo substituimos por el código heredado del sugar.activity. clase de Actividad.

EXTENDER LA CLASE DE LA ACTIVITY

Aquí está una versión de nuestro programa que amplíe Actividad. Usted la encontrará en el depósito de Git en el directorio **Inherit_From_sugar.activity.Activity** bajo el nombre **ReadEtextsActivity.py**:

```
import sys
import os
import zipfile
import pygtk
import gtk
import pango
from sugar.activity import activity
from sugar.graphics import style

page=0
PAGE_SIZE = 45

class ReadEtextsActivity(activity.Activity):
    def __init__(self, handle):
        "The entry point to the Activity"
        global page
        activity.Activity.__init__(self, handle)

        toolbox = activity.ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False
        activity_toolbar.share.props.visible = False
        self.set_toolbox(toolbox)

        toolbox.show()
        self.scrolled_window = gtk.ScrolledWindow()
        self.scrolled_window.set_policy(gtk.POLICY_NEVER, gtk.POLICY_AUTOMATIC)
        self.scrolled_window.props.shadow_type = gtk.SHADOW_NONE

        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_cursor_visible(False)
        self.textview.set_left_margin(50)
        self.textview.connect("key_press_event", self.keypress_cb)

        self.scrolled_window.add(self.textview)
        self.set_canvas(self.scrolled_window)
        self.textview.show()
        self.scrolled_window.show()
        page = 0
        self.textview.grab_focus()
        self.font_desc = pango.FontDescription("sans %d" % style.zoom(10))
        self.textview.modify_font(self.font_desc)

    def keypress_cb(self, widget, event):
        "Respond when the user presses one of the arrow keys"
        keyname = gtk.gdk.keyval_name(event.keyval)
        print keyname
```

```
        if keyname == 'plus':
            self.font_increase()
            return True
        if keyname == 'minus':
            self.font_decrease()
            return True
        if keyname == 'Page_Up' :
            self.page_previous()
            return True
        if keyname == 'Page_Down':
            self.page_next()
            return True
        if keyname == 'Up' or keyname == 'KP_Up' \
                or keyname == 'KP_Left':
            self.scroll_up()
            return True
        if keyname == 'Down' or keyname == 'KP_Down' \
                or keyname == 'KP_Right':
            self.scroll_down()
            return True
        return False

    def page_previous(self):
        global page
        page=page-1
        if page < 0: page=0
        self.show_page(page)
        v_adjustment = self.scrolled_window.get_vadjustment()
        v_adjustment.value = v_adjustment.upper - v_adjustment.page_size

    def page_next(self):
        global page
        page=page+1
        if page >= len(self.page_index): page=0
        self.show_page(page)
        v_adjustment = self.scrolled_window.get_vadjustment()
        v_adjustment.value = v_adjustment.lower

    def font_decrease(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size - 1
        if font_size < 1:
            font_size = 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def font_increase(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size + 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def scroll_down(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.upper - \
                v_adjustment.page_size:
            self.page_next()
            return
        if v_adjustment.value < v_adjustment.upper - v_adjustment.page_size:
            new_value = v_adjustment.value + v_adjustment.step_increment
            if new_value > v_adjustment.upper - v_adjustment.page_size:
                new_value = v_adjustment.upper - v_adjustment.page_size
            v_adjustment.value = new_value

    def scroll_up(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.lower:
            self.page_previous()
            return
        if v_adjustment.value > v_adjustment.lower:
            new_value = v_adjustment.value - \
                v_adjustment.step_increment
            if new_value < v_adjustment.lower:
                new_value = v_adjustment.lower
            v_adjustment.value = new_value
```

```
    def show_page(self, page_number):
        global PAGE_SIZE, current_word
        position = self.page_index[page_number]
        self.etext_file.seek(position)
        linecount = 0
        label_text = '\n\n\n'
        textbuffer = self.textview.get_buffer()
        while linecount < PAGE_SIZE:
            line = self.etext_file.readline()
            label_text = label_text + unicode(line, 'iso-8859-1')
            linecount = linecount + 1
        label_text = label_text + '\n\n\n'
        textbuffer.set_text(label_text)
        self.textview.set_buffer(textbuffer)

    def save_extracted_file(self, zipfile, filename):
        "Extract the file to a temp directory for viewing"
        filebytes = zipfile.read(filename)
        outfn = self.make_new_filename(filename)
        if (outfn == ''):
            return False
        f = open(os.path.join(self.get_activity_root(), 'instance',  outfn),  'w')
        try:
            f.write(filebytes)
        finally:
            f.close

    def read_file(self, filename):
        "Read the Etext file"
        global PAGE_SIZE

        if zipfile.is_zipfile(filename):
            self.zf = zipfile.ZipFile(filename, 'r')
            self.book_files = self.zf.namelist()
            self.save_extracted_file(self.zf, self.book_files[0])
            currentFileName = os.path.join(self.get_activity_root(),\
                'instance',  self.book_files[0])
        else:
            currentFileName = filename

        self.etext_file = open(currentFileName,"r")
        self.page_index = [ 0 ]
        linecount = 0
        while self.etext_file:
            line = self.etext_file.readline()
            if not line:
                break
            linecount = linecount + 1
            if linecount >= PAGE_SIZE:
                position = self.etext_file.tell()
                self.page_index.append(position)
                linecount = 0
        if filename.endswith(".zip"):
            os.remove(currentFileName)
        self.show_page(0)

    def make_new_filename(self, filename):
        partition_tuple = filename.rpartition('/')
        return partition_tuple[2]
```

Este programa tiene algunas diferencias significativas de la versión independiente.; Primero, observe que esta línea:

```
#! /usr/bin/env python
```

se ha quitado. Estamos funcionando con no más el programa directamente del intérprete de Python. Ahora Sugar lo está usandolo como Actividad. Note que se ha quitado mucho (pero no todo) de cuál estaba en () el método principal se ha movido al método del *__init ()* y *() al* método *principal*.

Note también que la declaración de la *clase* ha cambiado:

```
class ReadEtextsActivity(activity.Activity)
```

Esta declaración ahora nos dice que la clase ReadEtextsActivity extiende la clase **sugar.activity.Activity**. Consecuentemente hereda el código que está en esa clase. Por lo tanto necesitamos no más un lazo principal de GTK, o definir una ventana. El código en esta clase que extendemos hará eso para nosotros.

Mientras que ganamos mucho de esta herencia, perdemos algo también: una barra de título para la ventana principal. En condiciones gráficas al pedazo de software llamado un *encargado de ventana* es responsable de poner las fronteras en ventanas, haciéndolas resizeable, reduciéndolas a los iconos, la maximización de ellos, el azúcar del etc. utiliza a un encargado de ventana nombrado Matchbox que haga que cada ventana llena la pantalla entera y no ponga ninguna frontera, la barra de título, o ningunas otras decoraciones de la ventana en las ventanas. Como resultado de eso no podemos cerrar nuestro uso chascando en el "X" en la barra de título como antes. Para compensar esto que necesitamos tener una barra de herramientas que contenga un botón cercano. Así cada Actividad tiene una barra de herramientas de la Actividad que contenga algunos controles y botones estándar. Si usted mira el código usted verá que estoy ocultando un par de controles que no tengamos ninguÌn uso para todavía.

El método *read_file()* se llama no más () del método principal y no parece ser llamado dondequiera adentro del programa. Por supuesto consigue llamado, por algo del código de la Actividad que heredamos de nuestra nueva clase de padre. Semejantemente el *__init ()* y *() los* métodos *write_file* (si teníamos *un* método *write_file*() consiguen llamados por la clase de Actividad del padre.

Si usted es especialmente observador usted puede ser que haya notado otro cambio. Nuestro programa independiente original creó un fichero temporal cuando necesitó extraer algo de un archivo de cierre relámpago. Puso que el archivo en un directorio llamó /tmp. Nuestra nueva Actividad todavía crea el archivo pero lo pone en un diverso directorio, un específico a la Actividad.

Toda la escritura al sistema de ficheros se restringe a los sub-directórios de la trayectoria dada por *self.get_activity_root ()*. Este método le dará un directorio que pertenezca a su Actividad solamente. Contendrá tres sub-directórios con diversas políticas:

data
: Este directorio se utiliza para los datos tales como archivos de configuración. Los archivos almacenados aquí sobrevivirán reinicializaciones y mejoras del OS.

tmp
: Este directorio es similar usado al directorio de /tmp, siendo movido hacia atrás por RAM. Puede ser tan pequeño como 1 MB. Se suprime este directorio cuando la Actividad sale.

instance
: Este directorio es similar al directorio del **tmp**, siendo movido hacia atrás por la impulsión de la computadora algo que por RAM. Es único por caso. Se utiliza para la transferencia a y desde el diario. Se suprime este directorio cuando la Actividad sale.

Realizar estos cambios al código no es bastante para hacer nuestro programa una Actividad.; Tenemos que hacer un cierto trabajo de empaquetado y conseguirlo fijado para funcionar del emulador de Sugar. También necesitamos aprender cómo funcionar el emulador de Sugar. ¡Eso viene después!

9. EMPAQUETE LA ACTIVIDAD

AGREGUE SETUP.PY

Usted necesitará agregar un programa del Python llamado **setup.py al** mismo directorio que usted programa de actividad está adentro. Cada setup.py es exactamente igual que cada otro setup.py. Las copias en nuestro depósito de Git parecen esto:

```
#!/usr/bin/env python

# Copyright (C) 2006, Red Hat, Inc.
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

from sugar.activity import bundlebuilder

bundlebuilder.start()
```

Esté seguro y copie el texto entero arriba, incluyendo los comentarios.

El programa de setup.py es utilizado por el azúcar para un número de propósitos. ⁞ Si usted funciona setup.py de la línea de comando que usted verá las opciones que se utilizan con él y qué lo hacen.

```
[jim@simmons bookexamples]$ ./setup.py
/usr/lib/python2.6/site-packages/sugar/util.py:25: DeprecationWarning: the sha module is deprecated;
use the hashlib module instead
  import sha
Available commands:

build                Build generated files
dev                  Setup for development
dist_xo              Create a xo bundle package
dist_source          Create a tar source package
fix_manifest         Add missing files to the manifest
genpot               Generate the gettext pot file
install              Install the activity in the system

(Type "./setup.py  --help" for help about a particular command's options.
```

Funcionaremos con algunos de estos comandos después. No se trate sobre el mensaje de **DeprecationWarning**. Ésa es apenas manera del Python que nosotros dice de que tiene una nueva manera de hacer algo que es mejor solamente la vieja manera que todavía estamos utilizando trabajos. El error está viniendo del código en azúcar sí mismo y se debe fijar en un cierto lanzamiento futuro del azúcar.

CREE ACTIVITY.INFO

Después cree un directorio dentro de el que su progam está adentro y nómbrelo **activity**. Cree un archivo nombrado **activity.info** dentro de ese directorio e incorpore las líneas abajo en él. Aquí está el que está

para nuestra primera actividad:

```
[Activity]
name = Read ETexts II
service_name = net.flossmanuals.ReadEtextsActivity
icon = read-etexts
exec = sugar-activity ReadEtextsActivity.ReadEtextsActivity
show_launcher = no
activity_version = 1
mime_types = text/plain;application/zip
license = GPLv2+
```

Este archivo dice a azúcar cómo funcionar con su actividad. Las características necesarias en este archivo son:

name	El nombre de su actividad pues aparecerá al usuario.
service_name	Un nombre único que el azúcar utilizará para referir a su actividad. Cualquier entrada de diario creada por su actividad tendrá este nombre almacenado en sus meta datos, de modo que cuando alguien reasume el azúcar de la entrada de diario sepa para utilizar el programa que lo creó para leerlo.
icon	El nombre del archivo del icono que usted ha creado para la actividad. Puesto que los iconos están siempre .svg archiva el archivo del icono en el ejemplo se nombra read-etexts.svg.
exec	Esto dice a azúcar cómo poner en marcha su actividad. Qué dice es crear un caso de la clase **ReadEtextsActivity** que encontrará en el archivo **ReadEtextsActivity.py**.
show_launcher	Hay dos maneras de poner en marcha una actividad. El primer es chascar encendido el icono en la opinión de la actividad. El segundo es reasumir una entrada en el diario. Las Actividades que no crean entradas de diario se pueden reasumir solamente del diario, tan allí no son ninguìn punto en poner un icono en el anillo de la actividad para ellas. Lea Etexts es una actividad como eso.
activity_version	Un número entero que representa el número de versión de su programa. El que la primera versión es 1, el siguiente es 2, y así sucesivamente.
mime_types	Generalmente cuando usted reasume una entrada de diario pone en marcha la actividad que lo creó. En el caso de un e-libro no fue creado por ninguna actividad, así que necesitamos otra manera de decir al diario qué actividad puede utilizar. Un tipo del MIME es el nombre de un formato de archivo común. Algunos ejemplos son texto/llanos, el texto/HTML, uso/cierre relámpago y el uso/pdf. En esta entrada somos diciendo al diario que nuestro programa puede manejar cualquier archivos de texto llano o relampagar ficheros de archivo.
licence	La posesión de un programa de computadora no es como la compra de un coche. Con un coche, usted es el dueño y usted puede hacer de lo que usted tiene gusto con él. Usted puede venderlo, lo alquila, lo hace en una barra caliente, lo que. Con un programa de computadora hay siempre una licencia que dice a la persona que recibe el programa qué a le se permite hacer con él. GPLv2+ es una licencia estándar popular que se puede utilizar para las Actividades, y puesto que éste es *mi* programa que es qué va aquí. Cuando usted está listo para distribuir una de *sus* Actividades tendré más a decir sobre licencias.

CREE UN ICONO

Necesitamos después crear un icono nombrado **read-etexts.svg** y ponerlo en el sub-directório de la **activity**. Vamos a utilizar Inkscape para crear el icono. **Del nuevo** menú en Inkscape seleccione

icon_48x48. Esto creará un área de dibujo que sea un buen tamaño.

Usted no necesita ser un experto en Inkscape para crear un icono. De hecho menos la suposición su icono es la mejor. Al dibujar su icono recuerde los puntos siguientes:

- Su icono necesita parecer bueno de tamaños que se extienden de realmente, realmente pequeño a grande.
- Necesita ser recognizeable cuando su realmente, realmente pequeño.
- Usted consigue solamente utilizar dos colores: un color del movimiento y un color del terraplén. No importa cuáles usted elige porque el azúcar necesitará eliminar sus opciones de todos modos, tan apenas utiliza movimientos negros en un fondo blanco.
- Un color del terraplén se aplica solamente a un área que se contenga dentro de un movimiento intacto. Si usted dibuja una caja y una de las esquinas no conecta absolutamente el interior del área que la caja no será llenada. El dibujo de la carta blanca está solamente para el talentoso. Los círculos, las cajas, y los arcos son fáciles de dibujar con Inkscape así que utilícelos cuando usted puede.
- Inkscape también dibujará las cajas 3D usando perspectiva de dos puntos. No las utilice. Los iconos deben ser imágenes planas. 3D apenas no parece bueno en un icono.
- El subir con las buenas ideas para los iconos es resistente. I subió una vez con un cuadro algo agradable de un cajón de catálogo de tarjeta de biblioteca para *consigue los libros del archivo del Internet*. El problema es, ninguÌn niño bajo edad de cuarenta ha visto nunca un catálogo de tarjeta y menos aún así entender su propósito.

Cuando le hacen que hace su icono usted necesita modificarlo así que puede trabajar con el azúcar. Específicamente, usted necesita hacerlo que el azúcar de la demostración puede utilizar su propia opción del color del movimiento y llenar color. El formato de archivo de SVG se basa en XML, que los medios él son un archivo de texto con algunas etiquetas especiales en él. Esto significa que una vez que hemos acabado de corregirlo en Inkscape podemos cargar el archivo en Eric y corregirlo como archivo de texto.

No voy a poner el archivo entero en este capítulo porque la mayor parte de él que usted apenas dejar solo. La primera parte que usted necesita modificarse está al principio.

Antes:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!-- Created with Inkscape (http://www.inkscape.org/) -->
<svg
```

Después:

```
<?xml version="1.0" ?><!DOCTYPE svg  PUBLIC '-//W3C//DTD SVG 1.1//EN'
'http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd' [
 <!ENTITY stroke_color "#000000">
 <!ENTITY fill_color "#FFFFFF">
]><svg
```

Ahora en el cuerpo del documento usted encontrará referencias *para llenar* y *para frotar ligeramente* como parte de una cualidad llamada *style*. Cada línea o le forma que el drenaje tendrá éstos, como esto:

```
    <rect
       style="fill:#ffffff;stroke:#000000;stroke-opacity:1"
       id="rect904"
       width="36.142857"
       height="32.142857"
       x="4.1428571"
       y="7.1428571" />
```

Usted necesita cambiar cada uno al parecer esto:

```
    <rect
       style="fill:&fill_color;;stroke:&stroke_color;;stroke-opacity:1"
```

```
       id="rect904"
       width="36.142857"
       height="32.142857"
       x="4.1428571"
       y="7.1428571" />
```

Observe ese *&stroke_color;* y *&fill_color;* ambo extremo con puntos y comas (;), y los puntos y comas también se utilizan para separar las características para el estilo. Debido a esto es error de un principiante extremadamente común a irse del punto y coma que se arrastra porque dos puntos y comas en una fila no parecen correctos. ¡Sea confiado que los dos puntos y comas en una fila son intencionales y absolutamente necesarios!

HAGA UN ARCHIVO MANIFEST

Usted debe recordar que setup.py tiene una opción para poner al día un manifesto. Intentémoslo:

```
./setup.py fix_manifest
/usr/lib/python2.6/site-packages/sugar/util.py:25: DeprecationWarning: the sha module is deprecated;
use the hashlib module instead
  import sha
WARNING:root:Missing po/ dir, cannot build_locale
WARNING:root:Activity directory lacks a MANIFEST file.
```

Esto construirá realmente un archivo MANIFEST que contiene todo en el directorio y sus sub-directórios. El directorio de /po que se está quejando alrededor se utiliza para traducir Actividades a diversas idiomas. Podemos no hacer caso de eso para ahora.

El archivo MANIFEST que crea contendrá un poco de materia adicional, así que necesitamos librarnos de las líneas adicionales usando Eric. El MANIFEST corregida debe parecer esto:

```
setup.py
ReadEtextsActivity.py
activity/read-etexts.svg
activity/activity.info
```

INSTALE LA ACTIVIDAD

Hay apenas una más cosa a hacer antes de que poder probar nuestra actividad debajo del emulador del azúcar. Necesitamos instalarlo, que en este caso los medios que nos hacen un acoplamiento simbólico entre el directorio están utilizando para nuestro código en el directorio de ~/Activities/. El ~ del símbolo refiere al directorio "casero" del usuario que estamos funcionando con el azúcar debajo, y un acoplamiento simbólico es una manera de hacer que un archivo o un directorio aparece ser situado en más de un lugar sin el copiado de él. Hacemos este acoplamiento simbólico funcionando setup.py otra vez:

```
./setup.py dev
```

FUNCIONAR CON NUESTRA ACTIVIDAD

Ahora al final podemos funcionar con nuestra actividad debajo del azúcar. Para hacer que necesitamos aprender cómo funcionar el **sugar-emulator**.

Fedora no hace una opción del menú para el emulador del azúcar, sino que es fácil agregar uno usted mismo. El comando de funcionar está simplemente

```
sugar-emulator
```

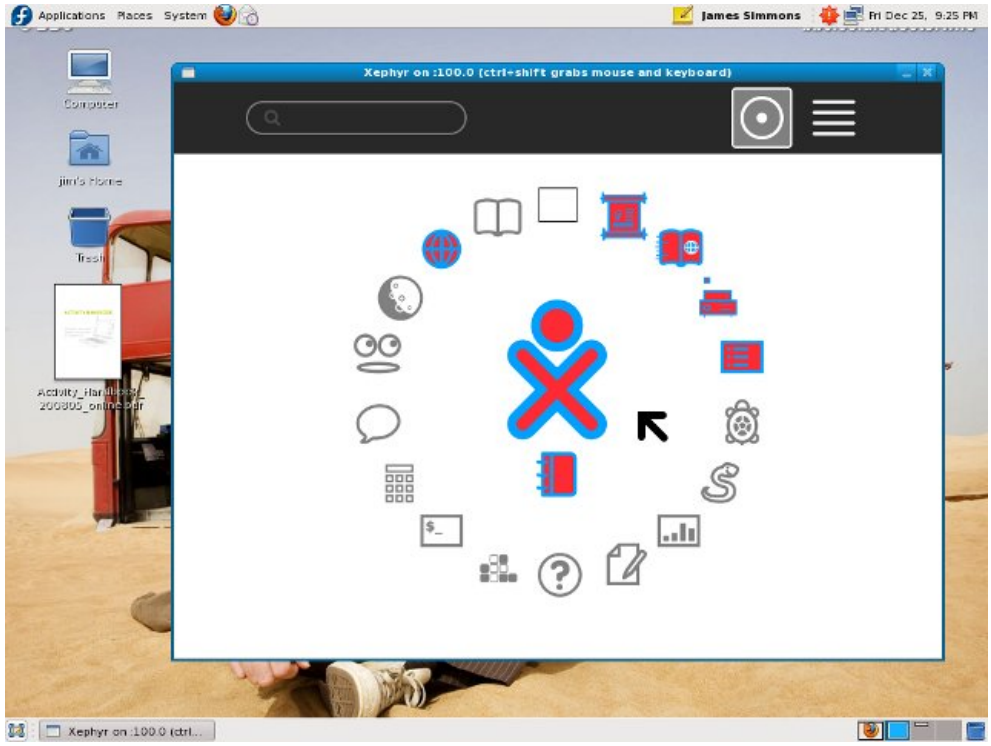
Si su resolución de la pantalla es más pequeña que los funcionamientos del azúcar-emulador del tamaño del defecto en ella funcionarán de plena pantalla. Esto no es conveniente para la prueba, así que usted puede

querer especificar su propio tamaño:

```
sugar-emulator -i 800x600
```

Observe que esta opción existe solamente en Fedora 11 y más adelante.

Cuando usted funciona el azúcar-emulador que una ventana abre y el ambiente del azúcar empieza para arriba y funciona dentro de él. Parece esto:



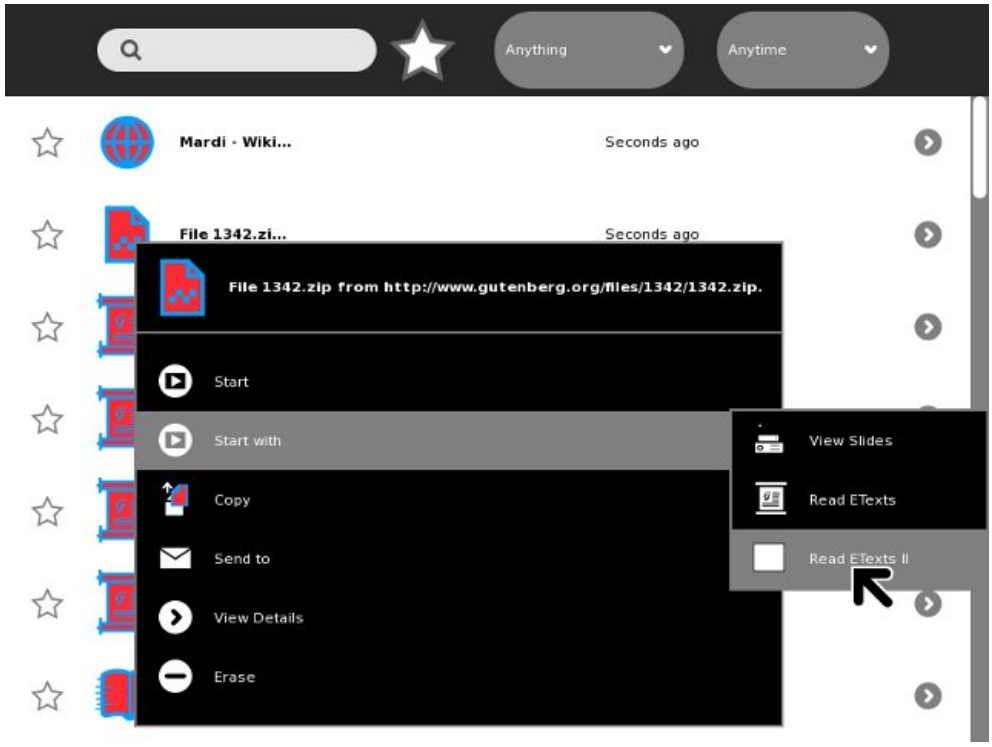
Cuando es corriente azúcar-emulador usted puede encontrar que algunas llaves no parecen trabajar en el ambiente del azúcar. Esto es causada por los insectos en el software de **Xephyr** que crea la ventana que el azúcar rueda. Tiene a veces dificultad el identificar de su teclado y consecuentemente de algunas llaves para conseguir malinterpretada. En Fedora 11 noté que mis llaves de funcionamiento no trabajaron, y mis llaves de flecha regulares no trabajaron tampoco aunque lo hicieran mis llaves de flecha del telclado numérico. Podía conseguir mis llaves de funcionamiento que trabajaban otra vez poniendo esta línea en *~/.sugar/debug*:

```
run setxkbmap
```

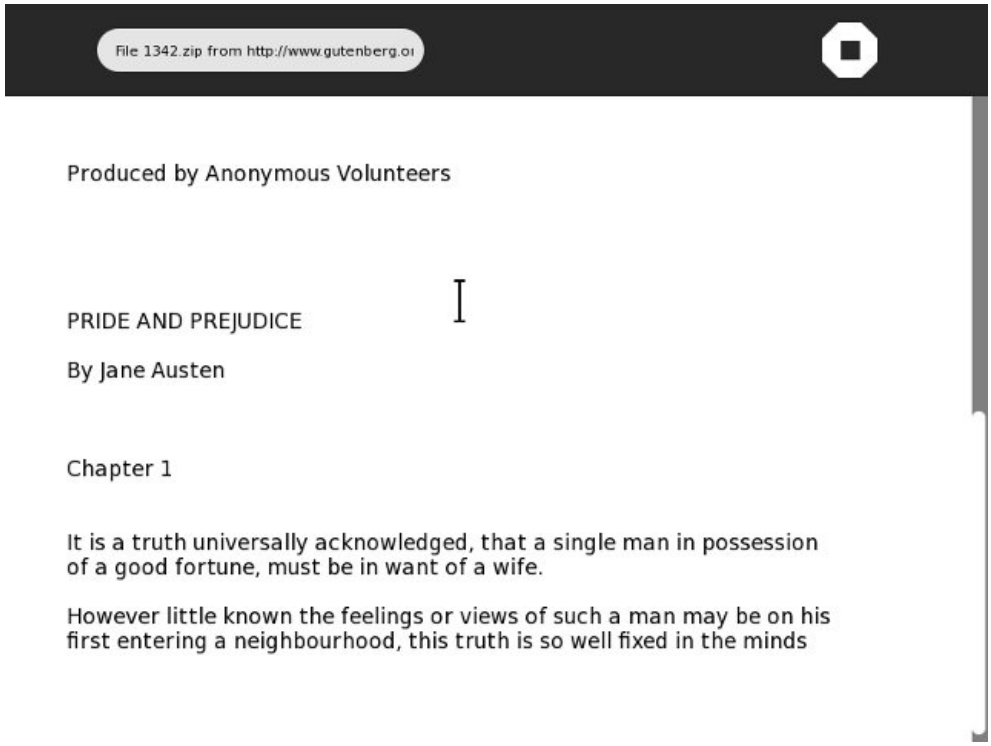
Esto necesita más explicación. Primero, el símbolo "~" refiere a su directorio casero. En segundo lugar, cualquier archivo nombrado el comenzar con un período se considera ocultado en linux, así que usted necesitará utilizar la opción para demostrar archivos ocultados y directorios en el hojeador del directorio del GNOMO para navegar a él. Finalmente, el nombre del keymap es un código de país de dos caracteres: nosotros para los Estados Unidos, franco para Francia, de para Alemania, etc.

Para probar nuestra actividad vamos a necesitar tener un libro en el diario, así que utilice la actividad de la **ojeada** para visitar el proyecto Gutenberg otra vez y para transferir el libro de su opción. Esta vez es importante transferir el libro en formato del cierre relámpago, porque hojee no puede transferir un archivo de texto llano al diario. En lugar, abre el archivo para la visión como si fuera un Web page. Si usted intenta la misma cosa con el archivo de cierre relámpago creará una entrada en el diario.

No podemos apenas abrir el archivo con un tecleo en el diario porque nuestro programa no creó la entrada de diario y hay varias Actividades que apoyan el tipo del MIME de la entrada de diario. Necesitamos utilizar el comienzo con la opción del menú como esto:



Cuando abrimos la entrada de diario esto es lo que vemos:



Técnico, ésta es la primera **iteración de** nuestra actividad. (La iteración es un significado sumamente útil de la palabra algo que usted lo hace más de una vez. En este libro estamos construyendo nuestra actividad un pedacito a la vez así que puedo demostrar principios de la escritura de la actividad, pero realmente la construcción de un programa en pedazos, la prueba de él, conseguir la regeneración, y la construcción un poco más pueden ser una manera altamente productiva de crear software. Usando la iteración de la palabra describir cada paso en el proceso hace el sonido de proceso más formal que está realmente).

Mientras que esta actividad pudo ser bastante buena demostrar a su propia madre, debemos mejorarla realmente un pedacito antes de que hagamos eso. Esa parte viene después.

10. AGREGUE LOS REFINAMIENTOS

BARRAS DE HERRAMIENTAS

Es una verdad reconocida universal que una actividad de la primera tarifa necesita buenas barras de herramientas. En este capítulo aprenderemos cómo hacerlas. Vamos a poner las clases de la barra de herramientas en un archivo separado del resto, porque hay dos estilos de la barra de herramientas (vieja y nueva) y podemos querer apoyar ambos en nuestra actividad. Si tenemos dos diversos archivos contener la barra de herramientas clasifica nuestro código puede decidir en el tiempo de pasada cuál quiere utilizar. Para ahora, este código apoya el viejo estilo, que trabaja con cada versión del azúcar. El nuevo estilo es apoyado actualmente solamente por *Sugar en un palillo.*

Hay un archivo llamado **toolbar.py** en el directorio de **Add_Refinements del** depósito de Git que parece esto:

```
from gettext import gettext as _
import re

import pango
import gobject
import gtk

from sugar.graphics.toolbutton import ToolButton
from sugar.activity import activity

class ReadToolbar(gtk.Toolbar):
    __gtype_name__ = 'ReadToolbar'

    def __init__(self):
        gtk.Toolbar.__init__(self)

        self.back = ToolButton('go-previous')
        self.back.set_tooltip(_('Back'))
        self.back.props.sensitive = False
        self.insert(self.back, -1)
        self.back.show()

        self.forward = ToolButton('go-next')
        self.forward.set_tooltip(_('Forward'))
        self.forward.props.sensitive = False
        self.insert(self.forward, -1)
        self.forward.show()

        num_page_item = gtk.ToolItem()

        self.num_page_entry = gtk.Entry()
        self.num_page_entry.set_text('0')
        self.num_page_entry.set_alignment(1)
        self.num_page_entry.connect('insert-text',
                                    self.num_page_entry_insert_text_cb)

        self.num_page_entry.set_width_chars(4)

        num_page_item.add(self.num_page_entry)
        self.num_page_entry.show()

        self.insert(num_page_item, -1)
        num_page_item.show()

        total_page_item = gtk.ToolItem()

        self.total_page_label = gtk.Label()

        label_attributes = pango.AttrList()
        label_attributes.insert(pango.AttrSize(14000, 0, -1))
```

```
        label_attributes.insert(pango.AttrForeground(65535, 65535, 65535, 0, -1))
        self.total_page_label.set_attributes(label_attributes)

        self.total_page_label.set_text(' / 0')
        total_page_item.add(self.total_page_label)
        self.total_page_label.show()

        self.insert(total_page_item, -1)
        total_page_item.show()

    def num_page_entry_insert_text_cb(self, entry, text, length, position):
        if not re.match('[0-9]', text):
            entry.emit_stop_by_name('insert-text')
            return True
        return False

    def update_nav_buttons(self):
        current_page = self.current_page
        self.back.props.sensitive = current_page > 0
        self.forward.props.sensitive = \
            current_page < self.total_pages - 1

        self.num_page_entry.props.text = str(current_page + 1)
        self.total_page_label.props.label = \
            ' / ' + str(self.total_pages)

    def set_total_pages(self, pages):
        self.total_pages = pages

    def set_current_page(self, page):
        self.current_page = page
        self.update_nav_buttons()

class ViewToolbar(gtk.Toolbar):
    __gtype_name__ = 'ViewToolbar'

    __gsignals__ = {
        'needs-update-size': (gobject.SIGNAL_RUN_FIRST,
                              gobject.TYPE_NONE,
                              ([])),
        'go-fullscreen': (gobject.SIGNAL_RUN_FIRST,
                          gobject.TYPE_NONE,
                          ([]))
    }

    def __init__(self):
        gtk.Toolbar.__init__(self)
        self.zoom_out = ToolButton('zoom-out')
        self.zoom_out.set_tooltip(_('Zoom out'))
        self.insert(self.zoom_out, -1)
        self.zoom_out.show()

        self.zoom_in = ToolButton('zoom-in')
        self.zoom_in.set_tooltip(_('Zoom in'))
        self.insert(self.zoom_in, -1)
        self.zoom_in.show()

        spacer = gtk.SeparatorToolItem()
        spacer.props.draw = False
        self.insert(spacer, -1)
        spacer.show()

        self.fullscreen = ToolButton('view-fullscreen')
        self.fullscreen.set_tooltip(_('Fullscreen'))
        self.fullscreen.connect('clicked', self.fullscreen_cb)
        self.insert(self.fullscreen, -1)
        self.fullscreen.show()

    def fullscreen_cb(self, button):
        self.emit('go-fullscreen')
```

Otro archivo en el mismo directorio del depósito de Git se nombra **ReadEtextsActivity2.py**. Parece esto:

```
import os
import zipfile
```

```
import gtk
import pango
from sugar.activity import activity
from sugar.graphics import style
from toolbar import ReadToolbar, ViewToolbar
from gettext import gettext as _

page=0
PAGE_SIZE = 45
TOOLBAR_READ = 2

class ReadEtextsActivity(activity.Activity):
    def __init__(self, handle):
        "The entry point to the Activity"
        global page
        activity.Activity.__init__(self, handle)

        toolbox = activity.ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False
        activity_toolbar.share.props.visible = False

        self.edit_toolbar = activity.EditToolbar()
        self.edit_toolbar.undo.props.visible = False
        self.edit_toolbar.redo.props.visible = False
        self.edit_toolbar.separator.props.visible = False
        self.edit_toolbar.copy.set_sensitive(False)
        self.edit_toolbar.copy.connect('clicked', self.edit_toolbar_copy_cb)
        self.edit_toolbar.paste.props.visible = False
        toolbox.add_toolbar(_('Edit'), self.edit_toolbar)
        self.edit_toolbar.show()

        self.read_toolbar = ReadToolbar()
        toolbox.add_toolbar(_('Read'), self.read_toolbar)
        self.read_toolbar.back.connect('clicked', self.go_back_cb)
        self.read_toolbar.forward.connect('clicked', self.go_forward_cb)
        self.read_toolbar.num_page_entry.connect('activate', \
                                    self.num_page_entry_activate_cb)
        self.read_toolbar.show()

        self.view_toolbar = ViewToolbar()
        toolbox.add_toolbar(_('View'), self.view_toolbar)
        self.view_toolbar.connect('go-fullscreen',
                self.view_toolbar_go_fullscreen_cb)
        self.view_toolbar.zoom_in.connect('clicked', self.zoom_in_cb)
        self.view_toolbar.zoom_out.connect('clicked', self.zoom_out_cb)
        self.view_toolbar.show()

        self.set_toolbox(toolbox)
        toolbox.show()
        self.scrolled_window = gtk.ScrolledWindow()
        self.scrolled_window.set_policy(gtk.POLICY_NEVER, gtk.POLICY_AUTOMATIC)
        self.scrolled_window.props.shadow_type = gtk.SHADOW_NONE

        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_cursor_visible(False)
        self.textview.set_left_margin(50)
        self.textview.connect("key_press_event", self.keypress_cb)

        self.scrolled_window.add(self.textview)
        self.set_canvas(self.scrolled_window)
        self.textview.show()
        self.scrolled_window.show()
        page = 0
        self.clipboard = gtk.Clipboard(display=gtk.gdk.display_get_default(), \
                                       selection="CLIPBOARD")
        self.textview.grab_focus()
        self.font_desc = pango.FontDescription("sans %d" % style.zoom(10))
        self.textview.modify_font(self.font_desc)

        buffer = self.textview.get_buffer()
        self.markset_id = buffer.connect("mark-set", self.mark_set_cb)
        self.toolbox.set_current_toolbar(TOOLBAR_READ)
```

```
def keypress_cb(self, widget, event):
    "Respond when the user presses one of the arrow keys"
    keyname = gtk.gdk.keyval_name(event.keyval)
    print keyname
    if keyname == 'plus':
        self.font_increase()
        return True
    if keyname == 'minus':
        self.font_decrease()
        return True
    if keyname == 'Page_Up' :
        self.page_previous()
        return True
    if keyname == 'Page_Down':
        self.page_next()
        return True
    if keyname == 'Up' or keyname == 'KP_Up' \
            or keyname == 'KP_Left':
        self.scroll_up()
        return True
    if keyname == 'Down' or keyname == 'KP_Down' \
            or keyname == 'KP_Right':
        self.scroll_down()
        return True
    return False

def num_page_entry_activate_cb(self, entry):
    global page
    if entry.props.text:
        new_page = int(entry.props.text) - 1
    else:
        new_page = 0

    if new_page >= self.read_toolbar.total_pages:
        new_page = self.read_toolbar.total_pages - 1
    elif new_page < 0:
        new_page = 0

    self.read_toolbar.current_page = new_page
    self.read_toolbar.set_current_page(new_page)
    self.show_page(new_page)
    entry.props.text = str(new_page + 1)
    self.read_toolbar.update_nav_buttons()
    page = new_page

def go_back_cb(self, button):
    self.page_previous()

def go_forward_cb(self, button):
    self.page_next()

def page_previous(self):
    global page
    page=page-1
    if page < 0: page=0
    self.read_toolbar.set_current_page(page)
    self.show_page(page)
    v_adjustment = self.scrolled_window.get_vadjustment()
    v_adjustment.value = v_adjustment.upper - v_adjustment.page_size

def page_next(self):
    global page
    page=page+1
    if page >= len(self.page_index): page=0
    self.read_toolbar.set_current_page(page)
    self.show_page(page)
    v_adjustment = self.scrolled_window.get_vadjustment()
    v_adjustment.value = v_adjustment.lower

def zoom_in_cb(self,  button):
    self.font_increase()

def zoom_out_cb(self,  button):
    self.font_decrease()
```

```
    def font_decrease(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size - 1
        if font_size < 1:
            font_size = 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def font_increase(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size + 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def mark_set_cb(self, textbuffer, iter, textmark):

        if textbuffer.get_has_selection():
            begin, end = textbuffer.get_selection_bounds()
            self.edit_toolbar.copy.set_sensitive(True)
        else:
            self.edit_toolbar.copy.set_sensitive(False)

    def edit_toolbar_copy_cb(self, button):
        textbuffer = self.textview.get_buffer()
        begin, end = textbuffer.get_selection_bounds()
        copy_text = textbuffer.get_text(begin, end)
        self.clipboard.set_text(copy_text)

    def view_toolbar_go_fullscreen_cb(self, view_toolbar):
        self.fullscreen()

    def scroll_down(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.upper - \
                v_adjustment.page_size:
            self.page_next()
            return
        if v_adjustment.value < v_adjustment.upper - \
                v_adjustment.page_size:
            new_value = v_adjustment.value + v_adjustment.step_increment
            if new_value > v_adjustment.upper - v_adjustment.page_size:
                new_value = v_adjustment.upper - v_adjustment.page_size
            v_adjustment.value = new_value

    def scroll_up(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.lower:
            self.page_previous()
            return
        if v_adjustment.value > v_adjustment.lower:
            new_value = v_adjustment.value - \
                v_adjustment.step_increment
            if new_value < v_adjustment.lower:
                new_value = v_adjustment.lower
            v_adjustment.value = new_value

    def show_page(self, page_number):
        global PAGE_SIZE, current_word
        position = self.page_index[page_number]
        self.etext_file.seek(position)
        linecount = 0
        label_text = '\n\n\n'
        textbuffer = self.textview.get_buffer()
        while linecount < PAGE_SIZE:
            line = self.etext_file.readline()
            label_text = label_text + unicode(line, 'iso-8859-1')
            linecount = linecount + 1
        label_text = label_text + '\n\n\n'
        textbuffer.set_text(label_text)
        self.textview.set_buffer(textbuffer)

    def save_extracted_file(self, zipfile, filename):
        "Extract the file to a temp directory for viewing"
        filebytes = zipfile.read(filename)
        outfn = self.make_new_filename(filename)
```

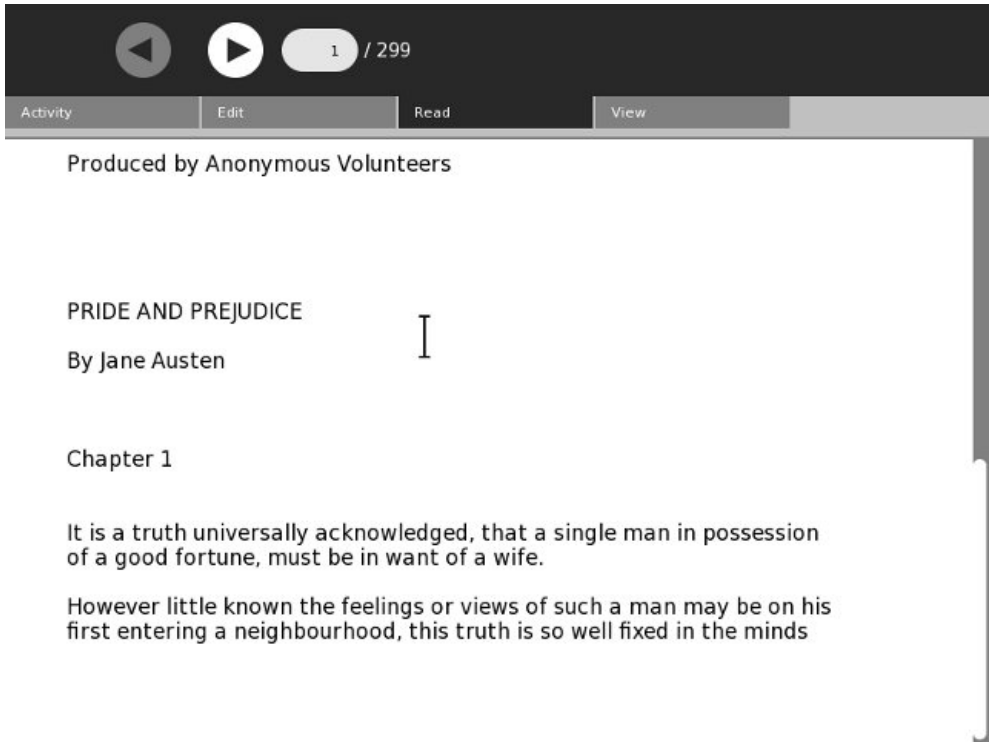
```
    if (outfn == ''):
        return False
    f = open(os.path.join(self.get_activity_root(), 'tmp',  outfn),  'w')
    try:
        f.write(filebytes)
    finally:
        f.close()

def get_saved_page_number(self):
    global page
    title = self.metadata.get('title', '')
    if title == ''  or not title[len(title)- 1].isdigit():
        page = 0
    else:
        i = len(title) - 1
        newPage = ''
        while (title[i].isdigit() and i > 0):
            newPage = title[i] + newPage
            i = i - 1
        if title[i] == 'P':
            page = int(newPage) - 1
        else:
            # not a page number; maybe a volume number.
            page = 0

def save_page_number(self):
    global page
    title = self.metadata.get('title', '')
    if title == ''  or not title[len(title)- 1].isdigit():
        title = title + ' P' +  str(page + 1)
    else:
        i = len(title) - 1
        while (title[i].isdigit() and i > 0):
            i = i - 1
        if title[i] == 'P':
            title = title[0:i] + 'P' + str(page + 1)
        else:
            title = title + ' P' + str(page + 1)
    self.metadata['title'] = title

def read_file(self, filename):
    "Read the Etext file"
    global PAGE_SIZE,  page

    if zipfile.is_zipfile(filename):
        self.zf = zipfile.ZipFile(filename, 'r')
        self.book_files = self.zf.namelist()
        self.save_extracted_file(self.zf, self.book_files[0])
        currentFileName = os.path.join(self.get_activity_root(), \
                'tmp',  self.book_files[0])
    else:
        currentFileName = filename

    self.etext_file = open(currentFileName,"r")
    self.page_index = [ 0 ]
    pagecount = 0
    linecount = 0
    while self.etext_file:
        line = self.etext_file.readline()
        if not line:
            break
        linecount = linecount + 1
        if linecount >= PAGE_SIZE:
            position = self.etext_file.tell()
            self.page_index.append(position)
            linecount = 0
            pagecount = pagecount + 1
    if filename.endswith(".zip"):
        os.remove(currentFileName)
    self.get_saved_page_number()
    self.show_page(page)
    self.read_toolbar.set_total_pages(pagecount + 1)
    self.read_toolbar.set_current_page(page)

def make_new_filename(self, filename):
```

```
        partition_tuple = filename.rpartition('/')
        return partition_tuple[2]

    def write_file(self, filename):
        "Save meta data for the file."
        self.metadata['activity'] = self.get_bundle_id()
        self.save_page_number()
```

Éste es **activity.info** por este ejemplo:

```
[Activity]
name = Read ETexts II
service_name = net.flossmanuals.ReadEtextsActivity
icon = read-etexts
exec = sugar-activity ReadEtextsActivity2.ReadEtextsActivity
show_launcher = no
activity_version = 1
mime_types = text/plain;application/zip
license = GPLv2+
```

La línea en **en negrilla** es la única que necesita cambiar. Cuando funcionamos con esta nueva versión esto es lo que veremos:



Hay algunas cosas digno de el señalamiento en este código. Primero, tenga una mirada en esta importación:

```
from gettext import gettext as _
```

Utilizaremos el módulo del *gettext del* Python para apoyar traducir nuestra actividad en otras idiomas. La utilizaremos en declaraciones como ésta:

```
        self.back.set_tooltip(_('Back'))
```

La raya actúa la misma manera que la función del gettext debido a la manera nosotros importó el gettext. ⁞ El efecto de esta declaración será mirar en un archivo especial de la traducción para una palabra o una frase que emparejen la llave "trasera" y la substituye por su traducción. Si no hay archivo de la traducción

para la lengua la queremos entonces utilizaremos simplemente la palabra “trasera”. Exploraremos la determinación de estos archivos de la traducción más adelante, pero para ahora usando el gettext para todas las palabras y frases demostraremos a nuestras endechas de los usuarios de la actividad una cierta base importante.

La segunda cosa digno de el señalamiento es que mientras que nuestra actividad revisada tiene cuatro barras de herramientas tuvimos que crear solamente dos de ellos. Los otros dos, **actividad** y **corrigen**, son parte de la biblioteca del Python del azúcar. Podemos utilizar esas barras de herramientas como es, oculte los controles que no necesitamos, o aún amplíelos agregando nuevos controles. En el ejemplo estamos ocultando la **subsistencia** y los controles de la **parte de la** barra de herramientas de la actividad y del **deshacer**, los botones **hacen de nuevo**, y de la **goma de la** barra de herramientas del corregir. No apoyamos actualmente la distribución de los libros o la modificación del texto en libros así que estos controles no son necesarios. Observe también que la barra de herramientas de la actividad es parte del **ActivityToolbox**. No hay manera de dar a su actividad una caja de herramientas que no contenga la barra de herramientas de la actividad como su primera entrada.

Otra cosa al aviso es que la clase de la actividad apenas no provee de nosotros una ventana. La ventana tiene un VBox para llevar a cabo nuestras barras de herramientas y el cuerpo de nuestra actividad. Instalamos la caja de herramientas usando *set_toolbox()* y el cuerpo de la actividad usando *set_canvas()*.

Las barras de herramientas **leída** y de la **visión** son PyGtk regular que programa, pero notan que hay un botón especial para las barras de herramientas del azúcar que pueden tener un tooltip atado a él, más la barra de herramientas de la **visión** tiene código para ocultar la caja de herramientas y **ReadEtextsActivity2** tiene código al unhide él. Esto es una función fácil a agregar a sus propias Actividades y muchos juegos y otras clases de Actividades pueden beneficiarse del área de pantalla creciente que usted consigue cuando usted oculta la caja de herramientas.

META DATOS Y ENTRADAS DE DIARIO

Cada entrada de diario representa un solo archivo más **meta datos**, o la información que describe el archivo. Hay las entradas estándar de los meta datos que todas las entradas de diario tienen y usted puede también crear sus propios meta datos de encargo.

Desemejante de ReadEtextsActivity, esta versión tiene *un* método *write_file()*.

```
def write_file(self, filename):
    "Save meta data for the file."
    self.metadata['activity'] = self.get_bundle_id()
    self.save_page_number()
```

No teníamos *() un* método *write_file* antes porque no íbamos a poner al día el archivo que el libro está adentro, y todavía no estamos. , Sin embargo, pondremos al día los meta datos para la entrada de diario. Específicamente, haremos dos cosas:

- Excepto la página nuestro usuario de la actividad paró el leer en tan de cuando él pone en marcha la actividad que podemos volver otra vez a esa página.
- Diga a entrada de diario que pertenece a nuestra actividad, de modo que en el futuro utilice el icono de nuestra actividad y pueda poner en marcha nuestra actividad con un tecleo.

La manera que la actividad **leída** ahorra página es utilizar una característica de encargo de los meta datos.

```
        self.metadata['Read_current_page'] = \
                    str(self._document.get_page_cache().get_current_page())
```

Read crea una característica de encargo de los meta datos nombrada *Read_current_page* para almacenar la página actual. Usted puede crear cualquier número de características de encargo de los meta datos apenas

esto fácilmente, así que usted puede preguntarse porqué no estamos haciendo eso con **Read Etexts**. Realmente, la primera versión de **Read Etexts** utilizó una característica de encargo, pero en el azúcar .82 o bajar allí era un insecto en el diario tales que los meta datos de encargo no sobrevivieron después de que la computadora fuera apagada. Consecuentemente mi actividad recordaría páginas mientras que la computadora funcionaba, pero las olvidaría tan pronto como fuera cerrada. Los ordenadores portátiles de XO no pueden aumentar actualmente cualquier cosa más nuevamente de .82, y cuando es posible aumentar será un trabajo grande para las escuelas.

Para conseguir alrededor de este problema creé los dos métodos siguientes:

```
    def get_saved_page_number(self):
        global page
        title = self.metadata.get('title', '')
        if title == ''  or not title[len(title)- 1].isdigit():
            page = 0
        else:
            i = len(title) - 1
            newPage = ''
            while (title[i].isdigit() and i > 0):
                newPage = title[i] + newPage
                i = i - 1
            if title[i] == 'P':
                page = int(newPage) - 1
            else:
                # not a page number; maybe a volume number.
                page = 0

    def save_page_number(self):
        global page
        title = self.metadata.get('title', '')
        if title == ''  or not title[len(title)- 1].isdigit():
            title = title + ' P' +  str(page + 1)
        else:
            i = len(title) - 1
            while (title[i].isdigit() and i > 0):
                i = i - 1
            if title[i] == 'P':
                title = title[0:i] + 'P' + str(page + 1)
            else:
                title = title + ' P' + str(page + 1)
        self.metadata['title'] = title
```

el save_page_number () mira los meta datos actuales y cualquiera del título agrega una página al extremo de él o pone al día la página ya allí. Puesto que el título es meta datos estándar para todas las entradas de diario el insecto del diario no le afecta.

Estos ejemplos demuestran cómo leer meta datos también.

```
        title = self.metadata.get('title', '')
```

Esta línea de código dice que "consiga la característica de los meta datos nombrada *título* y que póngala en el *título* nombrado variable, si no hay característica del título pone una secuencia vacía en *título*.

Usted ahorrará meta datos en *() el* método *write_file* y los leerá generalmente en *() el* método *read_file*.

En una actividad normal que pone un archivo en escrito en write_file () esta línea siguiente sería innecesaria:

```
        self.metadata['activity'] = self.get_bundle_id()
```

Cualquier entrada de diario creada por una actividad tendrá automáticamente esta característica fijada. En el caso de *orgullo y de prejudicar*, nuestra actividad no lo creó. Podemos leerla porque nuestra actividad apoya su *tipo del MIME*. Desafortunadamente, ese tipo del MIME, *uso/cierre relámpago*, es utilizado por otras Actividades. Lo encontré muy que frustraba para querer abrir un libro en **Read Etexts** y accidentalmente hacerlo abrir en **EToys** en lugar de otro. Esta línea de código soluciona ese problema. Usted necesita

solamente utilizar *comienzo usando...* la primera vez que usted lee un libro. Después que el libro utilizará el icono **Read Etexts** y se puede reasumir con un solo tecleo.

Esto afecta en absoluto al tipo del MIME de la entrada de diario, así que si usted quiso abrir deliberadamente *orgullo y prejudicar* con **Etoys** él es todavía posible.

Antes de que nos vayamos el tema de los meta datos del diario nos dejó mirar todos los meta datos estándar que cada actividad tiene. Aquí está un cierto código que crea una nueva entrada de diario y pone al día un manojo de características estándar:

```
def create_journal_entry(self,  tempfile):
    journal_entry = datastore.create()
    journal_title = self.selected_title
    if self.selected_volume != '':
        journal_title +=  ' ' + _('Volume') + ' ' +  self.selected_volume
    if self.selected_author != '':
        journal_title = journal_title  + ', by ' + self.selected_author
    journal_entry.metadata['title'] = journal_title
    journal_entry.metadata['title_set_by_user'] = '1'
    journal_entry.metadata['keep'] = '0'
    format = self._books_toolbar.format_combo.props.value
    if format == '.djvu':
        journal_entry.metadata['mime_type'] = 'image/vnd.djvu'
    if format == '.pdf' or format == '_bw.pdf':
        journal_entry.metadata['mime_type'] = 'application/pdf'
    journal_entry.metadata['buddies'] = ''
    journal_entry.metadata['preview'] = ''
    journal_entry.metadata['icon-color'] = profile.get_color().to_string()
    textbuffer = self.textview.get_buffer()
    journal_entry.metadata['description'] = \
        textbuffer.get_text(textbuffer.get_start_iter(), \
        textbuffer.get_end_iter())
    journal_entry.file_path = tempfile
    datastore.write(journal_entry)
    os.remove(tempfile)
    self._alert(_('Success'), self.selected_title + _(' added to Journal.'))
```

Este código se toma de una actividad que escribí que las transferencias directas reservan de un Web site y crean las entradas de diario para ellas. Las entradas de diario contienen un título amistoso y una descripción completa del libro.

La mayoría de las Actividades se ocuparán solamente de una entrada de diario usando *los* métodos *read_file()* y *write_file()* pero le no limitan a ése. En un capítulo posterior le demostraré cómo crear y suprimir entradas de diario, cómo enumerar el contenido del diario, y más.

Hemos cubierto mucha información técnica en este capítulo y hay más a venir, pero antes de que consigamos a ése necesitamos mirar algunos otros asuntos importantes:

- Poner su actividad en control de versión. Esto le permitirá compartir su código con el mundo y conseguir a la otra gente ayudar a trabajar en él.
- Consiguiendo su actividad traducida a otras idiomas.
- Distribución de su actividad acabada. (O su actividad no absolutamente acabada pero aún útil).

11. AGREGUE SU CÓDIGO DE LA ACTIVIDAD AL CONTROL DE VERSIÓN

¿CUÁL ES CONTROL DE VERSIÓN?

"Si he visto más lejos está solamente colocándose en los hombros de gigantes."

Isaac Newton, en una letra a Roberto Hooke.

La escritura de una actividad no es generalmente algo que usted hace por se. Usted tendrá generalmente colaboradores de una forma u otra. Cuando comencé a escribir **Etexts leído** copié mucho del código de la actividad **leída**. Cuando ejecuté el texto al discurso adapté una barra de herramientas de la actividad del **discurso**. Cuando finalmente conseguí mi código copiado del compartir archivos que trabajaba el autor del **espectador de la imagen** pensó que era bastante bueno copiar en esa actividad. Otro programador consideró el trabajo que hice para el texto al discurso y pensado él podría hacerlo mejor. Él tenía razón, y sus mejoras consiguieron combinadas en mi propio código. Cuando escribí **consiga los libros del archivo del Internet que** algún otro tomó el interfaz utilizador subí con y que hizo una actividad más de gran alcance y más versátil llamada **para conseguir los libros**. Como Newton, cada uno se beneficia del trabajo que otros han hecho antes.

Incluso si quise escribir Actividades sin ayuda todavía necesitaría a colaboradores traducirlos a otras idiomas.

Para hacer que colaboración posible usted necesita tener un lugar en donde cada uno puede fijar su código y compartirlo. Esto se llama un depósito del código. No es bastante apenas para compartir la última versión de su código. Qué usted quiere realmente hacer es compartir *cada* versión de su código. Cada vez que usted realiza un cambio significativo a su código usted quiere tener la nueva versión y la versión previa disponibles. No sólo usted quiere tener cada versión de su código disponible, usted quiere poder comparar cualquier dos versiones su código para ver qué cambió entre él. Esto es lo que lo hace el software de control de versión.

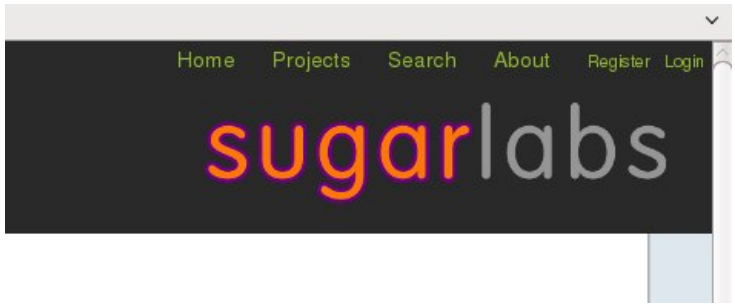
Las tres herramientas más populares del control de versión son **CVS**, **Subversion**, y **Git**. Git es el más nuevo y es el que está usado por Sugar Labs. Mientras que no cada actividad tiene su código en el depósito de Git de los laboratorios del azúcar (otros depósitos libres del código existen) allí no son ninguna buena razón para no hacerla y ventajas significativas si usted lo hace. Si usted quiere conseguir su actividad traducida a otras idiomas usando el depósito de Git de los laboratorios del azúcar es una necesidad.

GIT ALONG LITTLE DOGIES

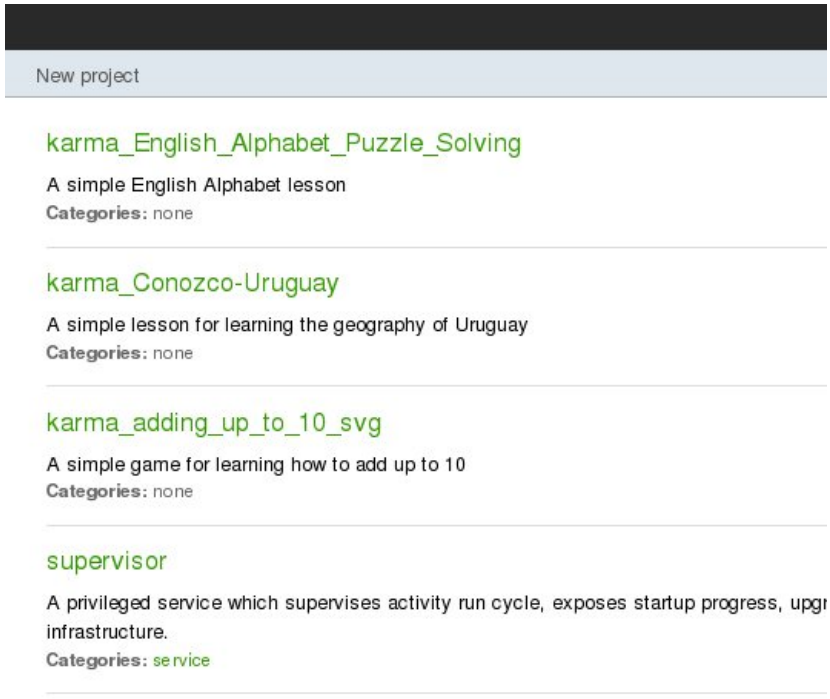
Git es un sistema de control **distribuido de** versión. Esto significa que no sólo están allí las copias de cada versión de su código en un depósito central, las mismas copias existe en la computadora de cada usuario. Esto significa que usted puede poner al día su depósito local mientras que usted no está conectado con el Internet, después no conecta y no comparte todo contemporáneamente.

Hay dos maneras que usted obrará recíprocamente con su depósito de Git: con los comandos de Git y con el Web site en http://git.sugarlabs.org/. Miraremos este Web site primero.

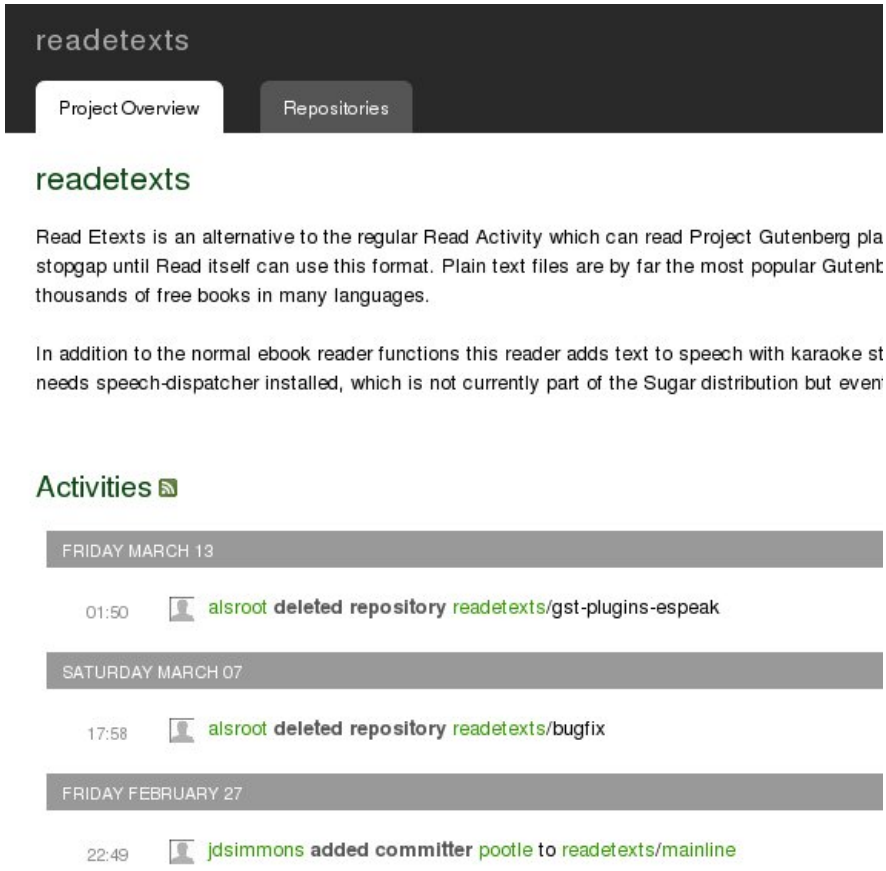
Vaya a http://git.sugarlabs.org/ y chasque encendido los **proyectos** ligan en la esquina correcta superior:



Usted verá una lista de proyectos en el depósito. Serán mencionados de la más nuevo a la más viejo. Usted también verá un **nuevo** acoplamiento del **proyecto** pero usted necesitará crear una cuenta para utilizar eso y no estamos listos para hacer eso todavía.

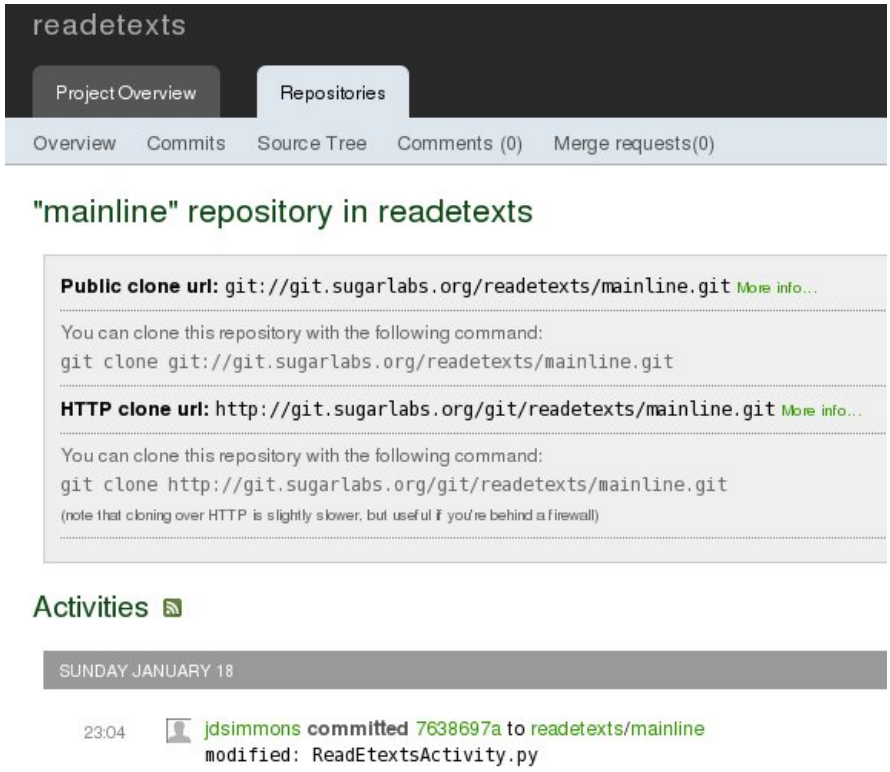


Si usted utiliza el acoplamiento de la **búsqueda** en la esquina correcta superior de la página usted conseguirá una forma de la búsqueda. Utilícela para buscar para los “etexts leídos”. Chasque encendido el acoplamiento para ese proyecto cuando usted lo encuentra. Usted debe ver algo similar:



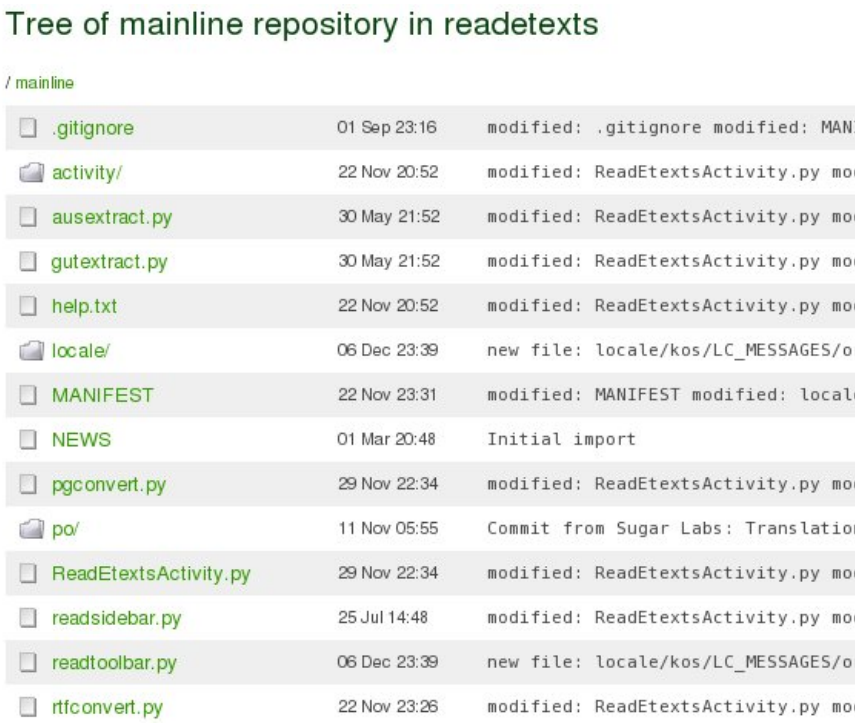
Esta página enumera *algo de la* actividad para el proyecto pero no lo encuentro particularmente útil. Para conseguir una mirada en su proyecto comience mucho mejor chascando en el nombre del depósito en el derecho de la página. En este caso el depósito se nombra **mainline**.

Usted verá algo similar en la tapa de la página:



Esta página tiene cierta información útil en ella. Primero, tenga una mirada en el **URL público de la copia** y el **URL de la copia del HTTP**. Usted necesita chascar encendido **más Info...** para ver cualquiera uno. Si usted funciona con cualquiera de estos comandos de la consola usted conseguirá una copia del depósito del git para el proyecto copiado a su computadora. Esta copia incluirá cada versión de cada pedazo de código en el proyecto. Usted necesitaría modificarla un pedacito antes de que usted podría compartir sus cambios de nuevo al depósito principal, pero todo estaría allí.

La lista bajo **Actividades** no es ésa útil, pero si usted chasca encendido el acoplamiento del **árbol de la fuente** usted considerará algo realmente bueno:

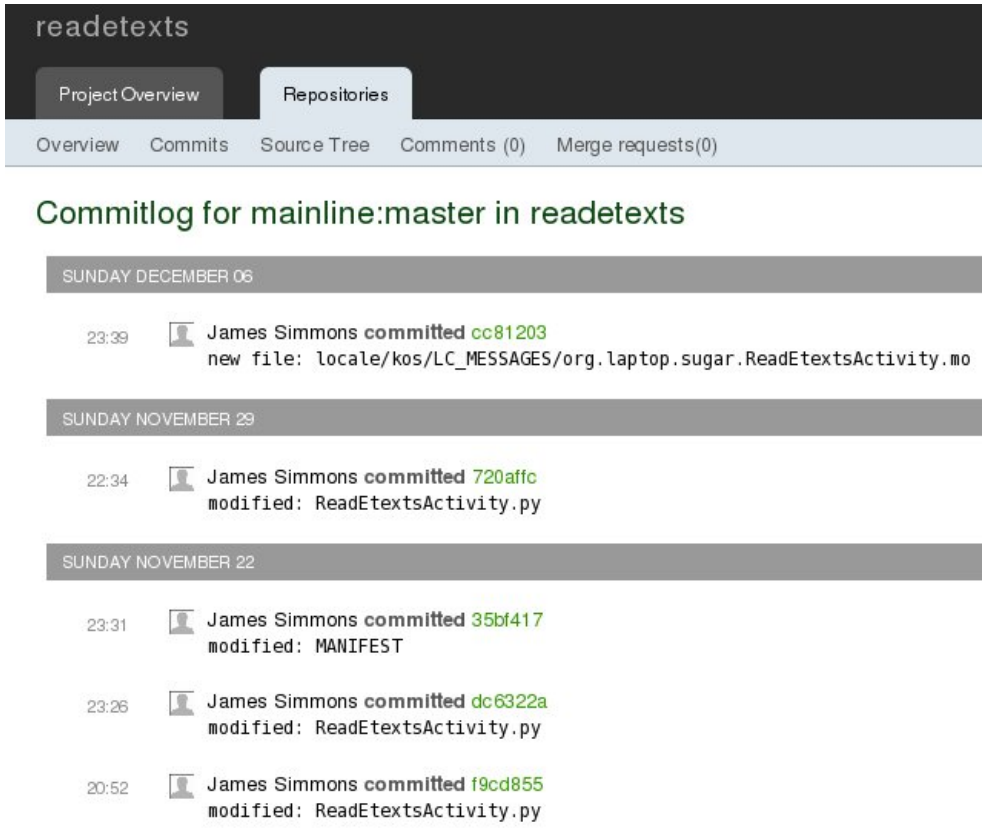


Aquí está una lista de cada archivo en el proyecto, la fecha que fue puesto al día por último, y un comentario sobre qué fue modificada. Chasque encendido el acoplamiento para **ReadEtextsActivity.py** y usted verá esto:

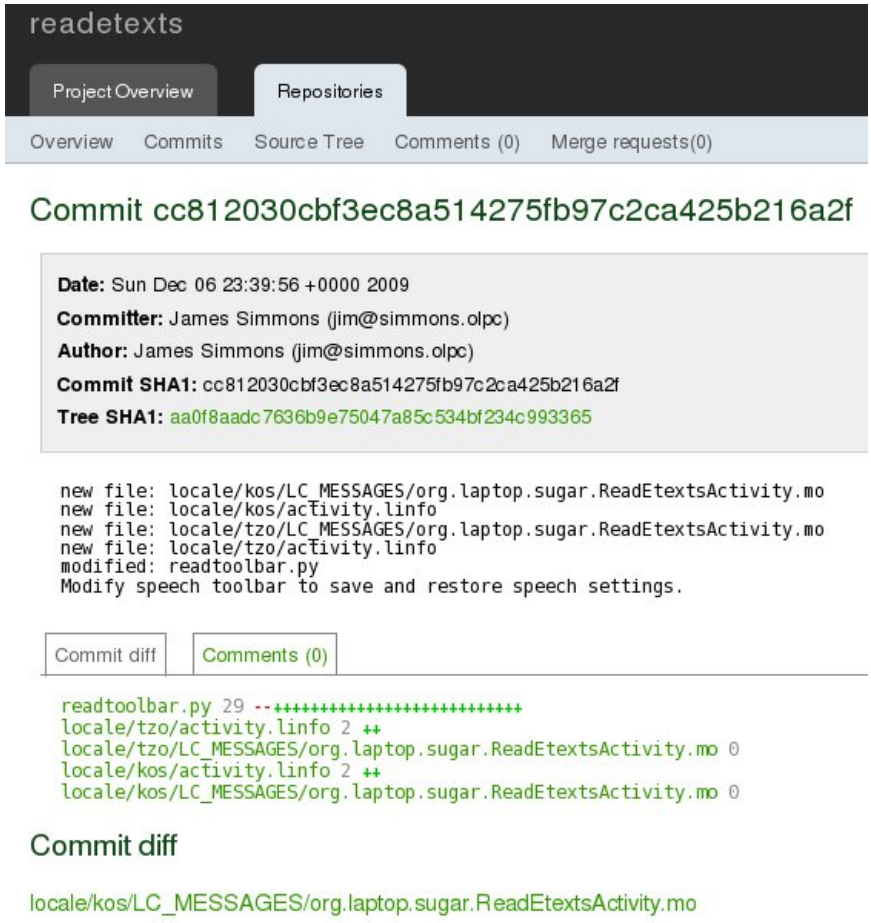
Éste es el último código de ese archivo en formato de la impresión bonita. Las palabras claves del Python se demuestran en un diverso color, allí son línea números, etc. Esto es una buena página para mirar código en la pantalla, pero no imprime bien y no es mucho bueno para copiar recortes del código en las ventanas de Eric cualquiera. Para cualquiera de esas cosas usted querrá chascar encendido **datos crudos de la gota** en la tapa del listado:

```
#! /usr/bin/env python

# Copyright (C) 2008, 2009 James D. Simmons
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
import os
import logging
import tempfile
import time
import zipfile
import pygtk
pygtk.require('2.0')
import gtk
import string
from sugar.graphics import style
from sugar import profile
from sugar.activity import activity
from sugar import network
from sugar.datastore import datastore
from sugar.graphics.alert import NotifyAlert
```

Todavía nos no hacen. Utilice el botón **trasero** para volver al listado bonito y chascar encendido **confía** acoplamiento. Esto nos dará una lista todo que nos cambió confió cada vez código en Git:



Usted pudo haber notado la combinación impar de letras y de números después de las palabras **James Simmons confiado**. Ésta es una clase de número de versión. La práctica generalmente con los sistemas de control de versión es dar cada versión del código que usted llega un número de versión, generalmente un número de serie simple. Git se distribuye, con muchas copias separadas del depósito que es modificado independientemente y después combinado. Eso hace usando apenas un número secuencial para identificar las versiones irrealizables. En lugar, Git da cada versión realmente, número al azar realmente grande. El número se expresa en la base 16, que utiliza los símbolos 0-9 y a-f. Qué usted ve en verde es solamente una pequeña parte del número completo. El número es un acoplamiento, y si usted lo chasca encendido usted verá esto:



En la tapa de la página vemos el número de versión completo usado para esto para confiar. Debajo de la caja gris vemos el comentario completo que fue utilizado para confiar los cambios. Debajo de ese es un listado de qué archivos fueron cambiados. Si miramos más lejos trague la página que vemos esto:

readtoolbar.py

```
15  15 # along with this program; if not, write to the Free Software
16  16 # Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301
17  17
    18 import os
18  19 import logging
19  20 from gettext import gettext as _
20  21 import re
... ...
416 416         combotool.show()
417 417
418 418         self.pitchadj = gtk.Adjustment(0, -100, 100, 1, 10, 0)
419         self.pitchadj.connect("value_changed", self.pitch_adjusted_cb)
420 419         pitchbar = gtk.HScale(self.pitchadj)
421 420         pitchbar.set_draw_value(False)
422 421         pitchbar.set_update_policy(gtk.UPDATE_DISCONTINUOUS)
... ...
427 427         pitchbar.show()
428 428
429 429         self.rateadj = gtk.Adjustment(0, -100, 100, 1, 10, 0)
430         self.rateadj.connect("value_changed", self.rate_adjusted_cb)
431 430         ratebar = gtk.HScale(self.rateadj)
432 431         ratebar.set_draw_value(False)
433 432         ratebar.set_update_policy(gtk.UPDATE_DISCONTINUOUS)
... ...
453 453     def pitch_adjusted_cb(self, get):
454 454         speech.pitch = int(get.value)
455 455         speech.say(_("pitch adjusted"))
    456         f = open(os.path.join(self.activity.get_activity_root(), 'insta
```

Éste es un informe del *diff* que demuestra las líneas que han cambiado entre esta versión y la versión previa. Para cada cambio demuestra algunas líneas antes y después de que el cambio para darle una mejor idea de lo que lo hace el cambio. Cada cambio demuestra la línea números también.

Un informe como esto es una ayuda maravillosa a la programación. A veces cuando usted está trabajando en un realce a su programa algo que había estado trabajando misterioso para el trabajar. Cuando sucede eso usted se preguntará apenas lo que usted cambió eso habría podido causar el problema. Un informe del diff puede ayudarle a encontrar la fuente del problema.

Ahora usted debe ser convencido de que usted quiere su código de proyecto en Git. Antes de que poder hacer que necesitamos crear una cuenta en este Web site. Eso es más difícil que creando una cuenta en cualquier otro Web site, pero necesitará un fragmento de información importante de nosotros que no tengamos todavía. Conseguir esa información es nuestra tarea siguiente.

LLAVES DE LA CREACIÓN SSH

Para enviar su código al **Gitorious** cifre el depósito que usted necesita un público de SSH/un par dominante privado. SSH es una manera de enviar datos sobre la red en formato cifrado. (Es decir utiliza un código secreto así que nadie pero la persona que consigue los datos puede leerlo). La encripción de la llave pública/privada es una manera de cifrar los datos que proporcionan una manera de garantizar que la persona que le está enviando los datos es quién él demanda ser.

En términos simples trabaja como esto: el software de SSH genera dos números muy grandes que se utilicen para codificar y para descifrar los datos que pasan la red. El primer número, llamado la **llave privada**, es secreto guardado y es utilizado solamente por usted para codificar los datos. El segundo

número, llamado la **llave pública**, se da a cualquier persona que necesite descifrar sus datos. Él puede descifrarlo usando la llave pública; no hay necesidad de él de saber la llave privada. Él puede también utilizar la llave pública para codificar un mensaje para enviarle detrás y usted puede descifrarlo usando su llave privada.

Git utiliza SSH como una firma electrónica para verificar que los cambios del código que se suponen para venir de usted están viniendo realmente de usted. El depósito de Git se da su llave pública. Sabe que cualquier cosa que descifra con esa llave se debe haber enviado por usted porque solamente usted hace la llave privada necesitar para codificarla.

Utilizaremos una herramienta llamada **OpenSSH** para generar las llaves públicas y privadas. Esto se incluye con cada versión del linux así que de usted apenas necesidad de verificar que ha estado instalada. Entonces utilice la utilidad del **ssh-keygen** que viene con OpenSSH generar las llaves:

```
[jim@olpc2 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jim/.ssh/id_rsa):
```

Por abandono el ssh-keygen genera una llave del **RSA**, que es la clase que queremos. Por abandono pone los keyfiles en un directorio llamado **/yourhome/.ssh** y queremos eso también, así que no incorporamos un nombre de fichero cuando le pregunta a. Apenas golpee la llave de **entrada** para continuar.

```
[jim@olpc2 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jim/.ssh/id_rsa):
Created directory '/home/jim/.ssh'.
Enter passphrase (empty for no passphrase):
```

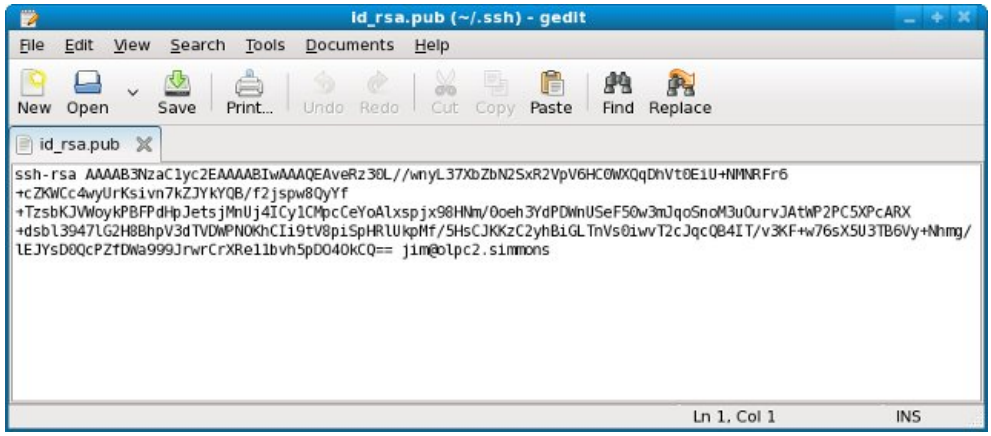
Ahora queremos un **passphrase** aquí. Un passphrase es como una contraseña que se utilice con las llaves públicas y privadas para hacer cifrar. Cuando usted mecanografía en usted no podrá ver lo que usted mecanografió. Debido a eso le preguntará que para mecanografiar la misma cosa otra vez, y la comprobará para ver que usted le mecanografió de la misma manera ambas veces.

```
[jim@olpc2 ~]$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/jim/.ssh/id_rsa):
Created directory '/home/jim/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/jim/.ssh/id_rsa.
Your public key has been saved in /home/jim/.ssh/id_rsa.pub.
The key fingerprint is:
d0:fe:c0:0c:1e:72:56:7a:19:cd:f3:85:c7:4c:9e:18 jim@olpc2.simmons
The key's randomart image is:
+--[ RSA 2048]----+
|        oo  E=.  |
|       + o+ .+=. |
|    . B +  o.oo  |
|     = O    .    |
|      . S        |
|         o       |
|           .     |
|                 |
|                 |
+-----------------+
```

Al elegir un passphrase recuerde que necesita ser algo que usted puede mecanografiar confiablemente sin verlo y sería mejor si no era una palabra usted puede encontrar en el diccionario, porque ésos están fácilmente quebrados. Cuando necesito hacerme una contraseña utilice la herramienta en http://www.multicians.org/thvv/gpw.html. Esta herramienta genera un manojo de palabras de absurdo que sean pronunciables. Escoja uno que apele a usted y utilice eso.

Ahora tenga una mirada dentro del directorio de .ssh. Por la convención cada nombre del archivo o de

directorio que comienza con un período se considera ocultado por Linux, así que él no aparecerá en una ventana de hojeador del archivo del GNOMO a menos que usted utilice la opción en el menú de la visión para demostrar archivos ocultados. Cuando usted exhibe el contenido de ese directorio usted verá dos archivos: **id_rsa** e **id_rsa.pub**. La llave pública está en id_rsa.pub. Intente abrir ese archivo con el gedit (se abren con el editor de textos) y usted verá algo similar:



Cuando usted crea su cuenta en git.sugarlabs.org habrá un lugar en donde usted puede agregar su llave del público SSH. Para hacer que el uso **selecciona todos del** menú del **corregir** en gedit, después **copia** y **pega** en el campo proporcionado en la forma de la tela.

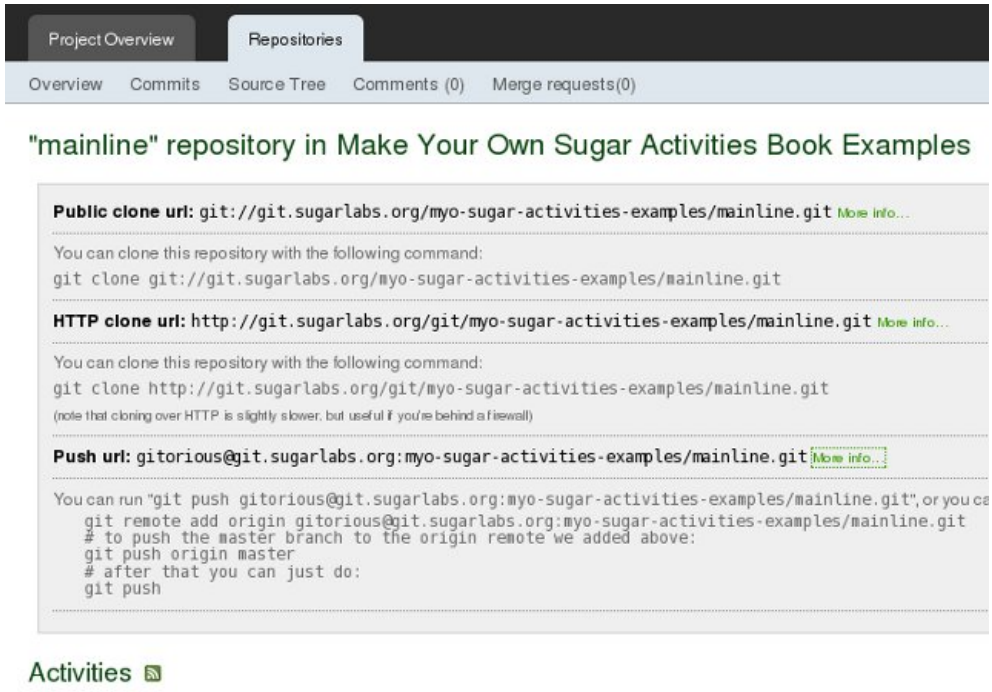
CREE UN NUEVO PROYECTO

Voy a crear un nuevo proyecto en Git por los ejemplos para este libro. Necesito abrir una sesión con mi nueva cuenta y chascar el **nuevo** acoplamiento del **proyecto que** vimos anterior. Consigo esta forma, la cual he comenzado completar:

El título se utiliza en el Web site, el **lingote** es una versión acortada del título sin los espacios usados para nombrar el depósito de Git. **Las categorías** son opcionales. **La licencia** es GLP v2 para mis proyectos. Usted puede elegir de licencias unas de los en la lista para sus propios proyectos, y usted puede cambiar la entrada de la licencia más adelante si usted quiere a. Usted también necesitará incorporar una **descripció**

n para su proyecto.

Una vez que usted tiene esta disposición usted podrá chascar encendido la entrada del mainline para el proyecto (como hicimos con Etexts leído antes) y ver algo similar:



El paso siguiente es convertir nuestros archivos de proyecto en un depósito local de Git, agregar los archivos a él, después para empujarlo al depósito en git.sugarlabs.org. Necesitamos hacer esto porque usted no puede **reproducir un** depósito vacío, y nuestro depósito alejado es actualmente vacío. Para conseguir alrededor de ese problema empujaremos el depósito local hacia fuera al nuevo depósito alejado que acabamos de crear, después reproducimos el alejado y suprimimos nuestro proyecto existente y su depósito de Git. Haremos desde entonces todo nuestro trabajo en el depósito reproducido.

Este proceso puede recordarle la cotización de Edward Albee, "una persona tiene que ir a veces un muy interurbano de su manera de volverse una distancia corta correctamente". Necesitamos afortunadamente solamente hacerlo una vez por proyecto. Incorpore los comandos demostrados abajo en **en negrilla** después de hacerle el directorio del proyecto el actual:

```
git init
Initialized empty Git repository in /home/jim/olpc/bookexamples/.git/
git add *.py
git add activity
git add MANIFEST
git add .gitignore
git commit -a -m "Create repository and load"
[master (root-commit) 727bfe8] Create repository and load
 9 files changed, 922 insertions(+), 0 deletions(-)
 create mode 100644 .gitignore
 create mode 100644 MANIFEST
 create mode 100755 ReadEtexts.py
 create mode 100644 ReadEtextsActivity.py
 create mode 100644 ReadEtextsActivity2.py
 create mode 100644 activity/activity.info
 create mode 100644 activity/read-etexts.svg
 create mode 100755 setup.py
 create mode 100644 toolbar.py
```

He hecho un depósito local vacío de Git con el **init del git**, después he utilizado el **git agrego** para agregar los archivos importantes a él. (De hecho el **git agrega** no agrega realmente cualquier cosa sí mismo; apenas dice Git agregar el archivo en el **git** siguiente **confía**). Finalmente el **git confía** con las opciones demostradas pondrá realmente la última versión de estos archivos en mi nuevo depósito local.

Para empujar este depósito local a git.sugarlabs.org utilizamos los comandos del Web page:

```
git remote add origin gitorious@git.sugarlabs.org:myo-sugar-activities-
examples/mainline.git
git push origin master
Counting objects: 17, done.
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 7.51 KiB, done.
Total 15 (delta 3), reused 0 (delta 0)
To gitorious@git.sugarlabs.org:myo-sugar-activities-examples/mainline.git
   2cb3a1e..700789d  master -> master
=> Syncing Gitorious...
Heads up: head of  changed to 700789d3333a7257999d0a69bdcafb840e6adc09 on master
Notify cia.vc of 727bfe819d5b7b70f4f2b31d02f5562709284ac4  on myo-sugar-activities-examples
Notify cia.vc of 700789d3333a7257999d0a69bdcafb840e6adc09  on myo-sugar-activities-examples
[OK]
rm *
rm activity -rf
rm .git -rf
cd ~
rm Activity/ReadEtextsII
mkdir olpc
cd olpc
mkdir bookexamples
cd bookexamples
git clone git://git.sugarlabs.org/myo-sugar-activities-examples/mainline.git
Initialized empty Git repository in /home/jim/olpc/bookexamples/mainline/.git/
remote: Counting objects: 18, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 18 (delta 3), reused 0 (delta 0)
Receiving objects: 100% (18/18), 8.53 KiB, done.
Resolving deltas: 100% (3/3), done.
```

Las líneas en **en negrilla** son los comandos de entrar, y todo es los mensajes que Git envía a la consola. No está probablemente claro qué estamos haciendo aquí y porqué, así que nos dejaron tomarlo gradualmente:

- El primer **git remote add origin** comando **agrega origen** dice a depósito alejado de Git que vamos a enviarle la materia de nuestro depósito local.
- El segundo **git push origin master** comando envía realmente su depósito local de Git el alejado y su contenido será copiado adentro. Cuando usted incorpora este comando le pedirán incorporar la frase del paso de SSH que usted creó en el pasado secciona. El GNOMO recordará esta frase para usted y la incorporará para cada comando de Git luego así que usted no necesita. Guardará el hacer de esto hasta que usted logout o apague la computadora.
- El paso siguiente es suprimir nuestros archivos existentes y nuestro depósito local de Git (que se contenga en el directorio ocultado .git). **El rm .git - los** medios del **rf** "suprimen el directorio .git y todo en él". **el rm** es un comando de Unix, no parte de Git. Si usted tiene gusto usted puede suprimir sus archivos existentes *después de que* usted cree el depósito reproducido en el paso siguiente. Observe la **actividad/el ReadEtextsII del rm del** comando, que suprime el acoplamiento simbólico a nuestro viejo proyecto que creamos funcionando con el **revelador de ./setup.py**. Necesitaremos ir a nuestro nuevo directorio reproducido del proyecto y funcionar que otra vez antes de que poder probar nuestra actividad otra vez.
- Ahora hacemos el comando de la **copia del git del** Web page. Esto toma el depósito alejado de Git que acabamos de agregar nuestro archivo MANIFESTO a y hace un nuevo depósito local en el directorio **/yourhome/olpc/bookexamples/mainline**.

Finalmente tenemos un depósito local que podemos utilizar. Bien, no absolutamente. Podemos confiar nuestro código a él pero no podemos empujar cualquier cosa de nuevo al depósito alejado porque nuestro

depósito local no se configura correctamente todavía.

Qué necesitamos hacer es corregir los **config del** archivo en el directorio **.git** en **/yourhome/olpc/bookexamples/mainline**. Podemos utilizar el gedit para hacer eso. Necesitamos cambiar la entrada del **url=** para señalar al **URL del empuje** demostrado en el Web page del mainline. Cuando nos hacen nuestro archivo de los **config** debe parecer esto:

```
[core]
 repositoryformatversion = 0
 filemode = true
 bare = false
 logallrefupdates = true
[remote "origin"]
  url = gitorious@git.sugarlabs.org:myo-sugar-activities-examples/mainline.git
 fetch = +refs/heads/*:refs/remotes/origin/*
[branch "master"]
 remote = origin
 merge = refs/heads/master
```

La línea en **en negrilla** es la única que consigue cambiada.

De ahora en adelante cualquier persona que quiere trabajar en nuestro proyecto puede conseguir una copia local del depósito de Git haciendo esto dentro del directorio donde él quisiera que fuera el depósito:

```
git clone git://git.sugarlabs.org/myo-sugar-activities-examples/mainline.git
```

Él tendrá que cambiar su archivo de **.git/config** apenas como lo hicimos, después él estará listo para ir.

USO DIARIO DE GIT

Mientras que conseguir la disposición de los depósitos para comenzar con es una tarea, el uso diario no es. Hay solamente algunos comandos que usted necesitará trabajar con. Cuando nos fuimos apagado teníamos un depósito en **/yourhome/olpc/bookexamples/mainline** con nuestros archivos en él. Necesitaremos agregar cualquier nuevo archivo que creemos también.

Utilizamos el **git add** comando de decir a Git que queremos utilizar Git para almacenar un archivo particular. Esto no almacena realmente cualquier cosa, apenas dice a Git nuestras intenciones. El formato del comando está simplemente:

```
git add file_or_directory_name
```

Hay archivos que no queremos agregar a Git, comenzar con esos archivos que terminen en **.pyc**. Si nunca hacemos un **git agregue** en ellos que nunca conseguirán agregados, pero Git constantemente nos preguntará porqué no los estamos agregando. Hay afortunadamente una manera de decir a Git que realmente, no queremos realmente agregar esos archivos. Necesitamos crear un archivo nombrado **.gitignore** usando gedit y poner en entradas como esto:

```
*.pyc
*.e4p
*.zip
.eric4project/
.ropeproject/
```

Estas entradas también no harán caso de los archivos de proyecto usados por los archivos de Eric y de cierre relámpago que contienen ebooks, una vez que tenemos este archivo creado en el directorio del mainline que podemos agregarlo al depósito:

```
git add .gitignore
git commit -a -m "Add .gitignore file"
```

De ahora en adelante Git pedirá no más que agreguemos .pyc u otros archivos indeseados que emparejan

nuestros patrones. Si hay otros archivos no queremos en el depósito que podemos agregarlo a .gitignore como lleno archivamos nombres o nombres de directorio o como patrones como *.pyc.

Además del adición archiva a Git que podemos quitarlos también:

```
git rm filename
```

Observe que esto apenas dice a Git que de ahora en adelante no no perderá de vista un nombre de fichero dado, y ése tomará efecto en el siguiente confía. Las viejas versiones del archivo todavía están en el depósito.

Si usted quiere ver qué cambios serán aplicados en el siguiente confíe el funcionamiento esto:

```
git status
# On branch master
# Changed but not updated:
#   (use "git add ..." to update what will be committed)
#
# modified:   ReadEtextsActivity.py
#
no changes added to commit (use "git add" and/or "git commit -a")
```

Finalmente, poner sus últimos cambios del depósito utilizan esto:

```
git commit -a -m "Change use of instance directory to tmp"
Created commit a687b27: Change use of instance directory to tmp
 1 files changed, 2 insertions(+), 2 deletions(-)
```

Si usted se va de - m que un redactor abrirá y usted puede mecanografiar adentro un comentario, después excepto y salir. Desafortunadamente por abandono el redactor es **VI**, un redactor viejo del modo de texto que no sea amistoso como gedit.

Cuando hacemos todos nuestros cambios hacer podemos enviarlos al depósito central usando **git push**:

```
git push
Counting objects: 5, done.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 322 bytes, done.
Total 3 (delta 2), reused 0 (delta 0)
To gitorious@git.sugarlabs.org:myo-sugar-activities-examples/mainline.git
   700789d..a687b27  master -> master
=> Syncing Gitorious...
Heads up: head of  changed to a687b27e2f034e5a17d2ca2fe9f2787c7f633e64 on master
Notify cia.vc of a687b27e2f034e5a17d2ca2fe9f2787c7f633e64  on myo-sugar-activities-examples
[OK]
```

Podemos conseguir los últimos cambios de otros reveladores haciendo **git pull**:

```
git pull
remote: Counting objects: 17, done.
remote: Compressing objects: 100% (14/14), done.
remote: Total 15 (delta 3), reused 0 (delta 0)
Unpacking objects: 100% (15/15), done.
From gitorious@git.sugarlabs.org:myo-sugar-activities-examples/mainline
   2cb3a1e..700789d  master     -> origin/master
Updating 2cb3a1e..700789d
Fast forward
 .gitignore               |    6 +
 MANIFEST                 |  244 +-----------------------------------
 ReadEtexts.py            |  182 ++++++++++++++++++++++++
 ReadEtextsActivity.py    |  182 ++++++++++++++++++++++++
 ReadEtextsActivity2.py   |  311 ++++++++++++++++++++++++++++++++++++++++++++++++
 activity/activity.info   |    9 ++
 activity/read-etexts.svg |   71 ++++++++++
 setup.py                 |   21 +++
 toolbar.py               |  136 ++++++++++++++++++++
 9 files changed, 921 insertions(+), 241 deletions(-)
 create mode 100644 .gitignore
```

```
create mode 100755 ReadEtexts.py
create mode 100644 ReadEtextsActivity.py
create mode 100644 ReadEtextsActivity2.py
create mode 100644 activity/activity.info
create mode 100644 activity/read-etexts.svg
create mode 100755 setup.py
create mode 100644 toolbar.py

cree el modo 100755 ReadEtexts.py
cree el modo 100644 ReadEtextsActivity.py
cree el modo 100644 ReadEtextsActivity2.py
cree la actividad/activity.info del modo 100644
cree la actividad/read-etexts.svg del modo 100644
cree el modo 100755 setup.py
cree el modo 100644 toolbar.py
```

12. INTERNATIONAL QUE VA CON POOTLE

INTRODUCCIÓN

La meta de los laboratorios del azúcar y de un ordenador portátil por niño es educar a todos los niños del mundo, y no podemos hacer eso con las Actividades que están solamente disponibles en una lengua. Es igualmente verdad que la fabricación de versiones separadas de cada actividad para cada lengua no va a trabajar, y la espera que de los reveladores de la actividad sean fluidos en muchas idiomas no es realista tampoco. Necesitamos una manera para que los reveladores de la actividad puedan concentrar en crear Actividades y para los que puedan traducir a apenas haga eso. Afortunadamente, esto es posible y la manera que ha hecho está usando el *gettext*.

CONSEGUIR EL TEXTO CON EL GETTEXT

Usted debe recordar que nuestro último ejemplo del código hizo uso de una importación impar:

```
from gettext import gettext as _
```

La función del "_()" fue utilizada en declaraciones como esto:

```
        self.back.set_tooltip(_('Back'))
```

Cuando expliqué que esta función de mirada impar fue utilizada para traducir la palabra "trasera" en otras idiomas, de modo que cuando alguien mira la extremidad de herramienta del botón trasero él vea el texto en su propia lengua. I también dicho que si no fuera posible traducir este texto el usuario vería el sin traducir "trasero" de la palabra. En este capítulo aprenderemos más sobre cómo éste trabaja y lo que tenemos que hacer para apoyar a los voluntarios que traducen estas secuencias de texto a otras idiomas.

La primera cosa que usted necesita aprender es cómo dar formato correctamente a las secuencias de texto que se traducirán. Esto es una edición cuando las secuencias de texto son oraciones reales que contienen el inormation. Por ejemplo, usted puede ser que escriba a tal mensaje esta manera:

```
        message = _("User ") + username + _(" has joined the chat room.")
```

Esto trabajaría, pero usted ha hecho cosas difíciles para el traductor. Él tiene dos secuencias separadas a traducir y ninguna pista que pertenecen juntas. Es mucho mejor hacer esto:

```
        message = _("User %s has joined the chat room.") % username
```

Si usted sabe que ambas declaraciones dan la misma secuencia resultante entonces usted puede ver fácilmente porqué un traductor preferiría segundo. Utilice esta técnica siempre que usted necesite un mensaje que tenga cierta información insertada en ella. Cuando usted lo utiliza, intente y limítese a solamente un código del formato (el %s) por secuencia. Si usted utiliza más de uno puede causar los problemas para el traductor.

EL IR AL POTE

Si se asume que cada secuencia del texto un usuario se pudo demostrar por nuestra actividad se pasa a través de "_ ()" que el paso siguiente es generar un archivo del pote. Usted puede hacer esto funcionando setup.py con una opción especial:

```
./setup.py genpot
```

Esto crea un directorio llamado **po** y pone un archivo ***ActivityName.pot*** en ese directorio. En el caso de nuestro proyecto del ejemplo *ActivityName* es **ReadEtextsII**. Éste es el contenido de ese archivo:

```
# SOME DESCRIPTIVE TITLE.
# Copyright (C) YEAR THE PACKAGE'S COPYRIGHT HOLDER
# This file is distributed under the same license as the PACKAGE package.
# FIRST AUTHOR , YEAR.
#
#, fuzzy
msgid ""
msgstr ""
"Project-Id-Version: PACKAGE VERSION\n"
"Report-Msgid-Bugs-To: \n"
"POT-Creation-Date: 2010-01-06 18:31-0600\n"
"PO-Revision-Date: YEAR-MO-DA HO:MI+ZONE\n"
"Last-Translator: FULL NAME \n"
"Language-Team: LANGUAGE \n"
"MIME-Version: 1.0\n"
"Content-Type: text/plain; charset=CHARSET\n"
"Content-Transfer-Encoding: 8bit\n"

#: activity/activity.info:2
msgid "Read ETexts II"
msgstr ""

#: toolbar.py:34
msgid "Back"
msgstr ""

#: toolbar.py:40
msgid "Forward"
msgstr ""

#: toolbar.py:115
msgid "Zoom out"
msgstr ""

#: toolbar.py:120
msgid "Zoom in"
msgstr ""

#: toolbar.py:130
msgid "Fullscreen"
msgstr ""

#: ReadEtextsActivity2.py:34
msgid "Edit"
msgstr ""

#: ReadEtextsActivity2.py:38
msgid "Read"
msgstr ""

#: ReadEtextsActivity2.py:46
msgid "View"
msgstr ""
```

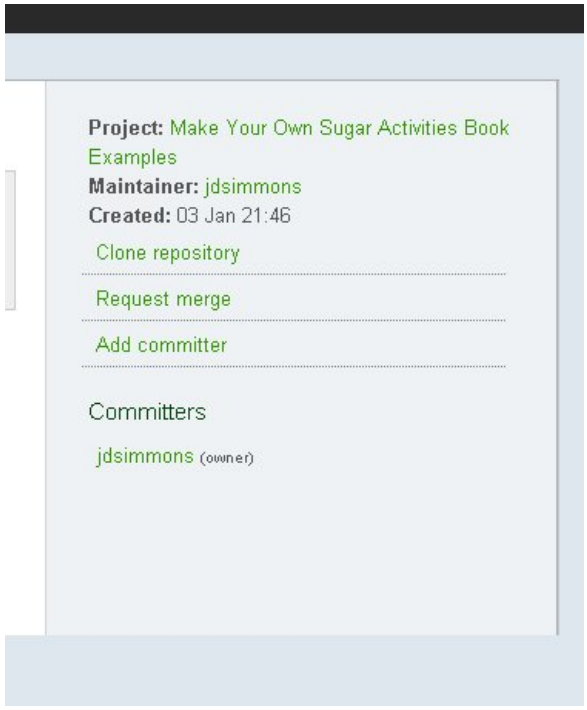
Este archivo contiene una entrada para cada secuencia de texto en nuestra actividad (como msgid) y un lugar para poner una traducción de esa secuencia (msgstr). Las copias de este archivo serán hechas por el servidor de Pootle para cada lengua deseada, y las entradas del msgstr serán completadas por los traductores voluntarios.

EL IR A POOTLE

Antes de que suceda cualquiera de esa poder necesitamos conseguir nuestro archivo del POTE en Pootle. La primera cosa que necesitamos hacer es conseguir el nuevo directorio en nuestro depósito de Git y empujarlo hacia fuera a Gitorious. Usted debe ser familiar con los comandos necesarios ahora:

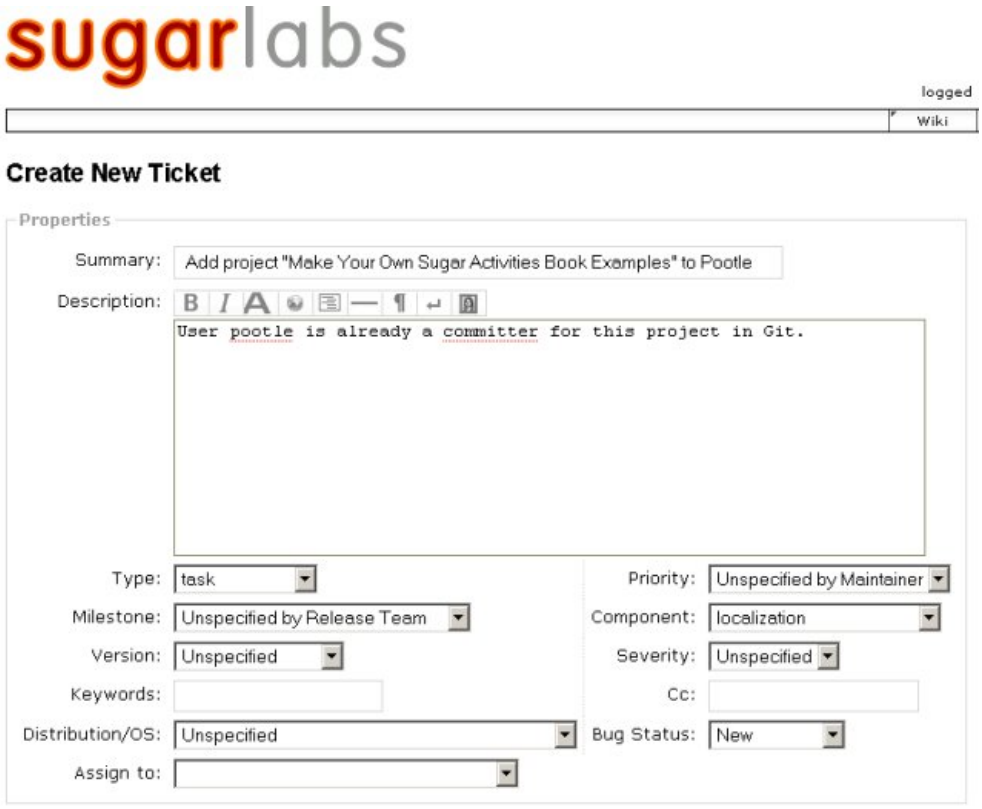
```
git add po
git commit -a -m "Add POT file"
git push
```

Necesitamos después dar al usuario “pootle” confiamos autoridad a nuestro proyecto de Git. Vaya a git.sugarlabs.org, firme adentro, y encuentre su página del proyecto y chasque encendido el acoplamiento del mainline. Usted debe ver esto en la página a la cual le toma:



Chasque encendido el acoplamiento del **Add committer** y mecanografíe adentro el **pootle** conocido en la forma a la cual le toma. Cuando usted vuelve a este **pootle de la** página sea mencionado debajo de Committers.

Su paso siguiente es ir al Web site http://bugs.sugarlabs.org y colocarse para una identificación del usuario. Cuando usted consigue que abre un boleto algo similar:



La localización de la entrada **componente** se debe utilizar, junto con el **tipo** *tarea*.

Créala o no, esto es todos lo que usted necesita hacer para conseguir su actividad para fijar para ser traducido.

NO PRESTE NINGUNA ATENCIÓN A ESE HOMBRE DETRÁS DE LA CORTINA

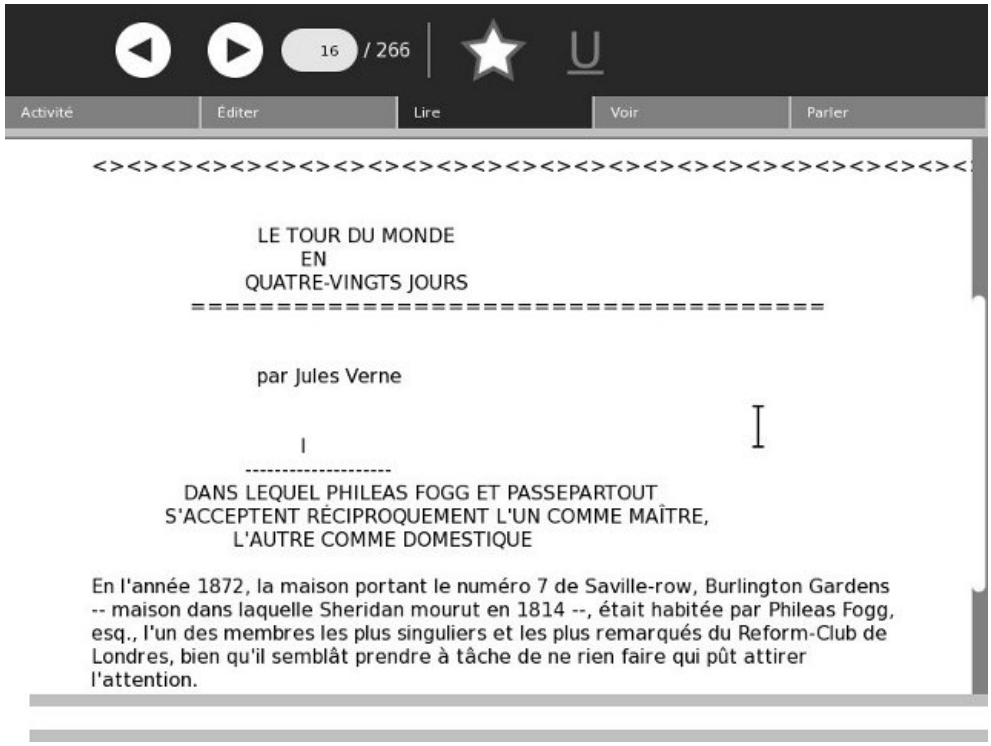
Después de que esto que usted necesitará hacer algunas cosas para conseguir traducciones de Pootle en su actividad.

- Cuando usted agrega las secuencias de texto (etiquetas, mensajes de error, etc.) a su de la actividad uso siempre la función del **_ ()** con ellas así que pueden ser traducidas.
- Después de agregar nuevas secuencias funcione siempre el **./setup.py genpot** para reconstruir el archivo del POTE.
- Eso confía y empuja después sus cambios a Gitorious.
- De vez en cuando, y especialmente antes de lanzar una nueva versión, haga un **tirón del git**. Si hay algunos archivos de la localización agregados a Gitorious éste le los traerá.
- Después de conseguir un manojo de nuevos archivos funcione **./setup.py fix_manifest** para conseguir los nuevos archivos incluidos en su archivo **MANIFEST**. Corrija luego el **MANIFEST** con el gedit para quitar cualquier entrada indeseada (que sean archivos de proyecto de Eric, etc.).

La localización con Pootle creará una gran cantidad de archivos en su proyecto, algunos en el directorio del **po** y otros en un nuevo directorio llamado **locale**. Mientras éstos se enumeren en el MANIFEST serán incluidas en el archivo de .xo que usted utilizará para distribuir su actividad.

¡C'EST MAGNIFIQUE!

Aquí está un tiro de pantalla de la versión lingüística francesa de los *jours de los quatre-vingts de du monde en del viaje del Le de la* novela de Julio Verne **leído de la** lectura de **Etexts**:



Hay razón para creer que el libro está en francés también.

13. DISTRIBUYA SU ACTIVIDAD

ELIJA UNA LICENCIA

Antes de que usted dé su actividad a cualquier persona usted necesita elegir una licencia que será distribuida debajo. El software de compra es como la compra de un libro. Ciertos correcto que usted tiene con un libro y otros usted no tiene. Si usted compra una copia del *código de DaVinci* usted tiene la derecha de leerla, de prestarla hacia fuera, de venderla a una librería usada, o de quemarla. Usted no tiene la derecha de hacer copias de ella o de hacer una película fuera de ella. El software es el la misma manera, pero a menudo peor. Esos contratos de licencia largos que aceptamos rutinario chascando un botón no pudieron permitir que usted venda el software cuando le hacen con él, o aún que lo dé lejos. Si usted vende su computadora usted puede encontrar que el software que usted compró es solamente bueno para esa computadora, y solamente mientras que usted es el dueño de la computadora. (Usted puede conseguir buenos repartos en las computadoras reacondicionadas sin el sistema operativo instalado por esa mismo razón).

Si usted está en el negocio de vender software usted puede ser que tenga que contratar a un abogado para elaborar un contrato de licencia, pero si usted está dando lejos software hay varias licencias estándar que usted puede elegir para de libre. El más popular en gran medida se llama la *licencia el público en general*, o GPL. Como las aplicaciones de Microsoft de las licencias permite la gente que consigue su programa para hacer algunas cosas con él pero no a otras. Qué hace interesante no es lo que permite que él haga (que es bonito mucho cualquier cosa que tienen gusto) pero qué los prohíbe para hacer.

Si alguien distribuye un programa autorizado bajo GPL también se requieren para hacer el código fuente del programa disponible para cualquier persona que lo quiera. Que la persona puede hacer mientras que él tiene gusto con el código, con una restricción importante: si él distribuye un programa basado en ese código él debe *también* autorizar ese código usando el GPL. Esto hace imposible para que alguien tome un trabajo autorizado GPL, lo mejore, y lo venda alguien sin el donante le del código fuente a la nueva versión.

Mientras que el GLP no es la única licencia disponible para que las Actividades sean distribuidas en http://activities.sugarlabs.org todo el las licencias requieren que cualquier persona que consigue la actividad también consiga el código fuente completo para él. Usted ha tomado ya cuidado de ese requisito poniendo su código fuente en Gitorious. Si usted utilizó cualquier código de una actividad existente autorizada con el GLP usted *debe* autorizar su propio código la misma manera. Si usted utilizó una cantidad significativa de código de este libro (que sea también GPL autorizado) usted puede ser requerido utilizar el GPL también.

¿Está autorizando algo que usted debe preocupación alrededor? No realmente. La única razón que usted querría utilizar una licencia con excepción del GLP es si usted quiso vender su actividad en vez de la da lejos. Considere lo que usted tendría que hacer para hacer eso posible:

- Usted tendría que utilizar una cierta lengua con excepción de Python así que usted podría dar a alguien el programa sin el donante les del código fuente.
- Usted tendría que tener su propio depósito del código fuente no disponible para el público en general y tomar medidas para tener los datos sostuvo regularmente.
- Usted tendría que tener su propio Web site para distribuir la actividad. El Web site tendría que ser fijado para aceptar pagos de alguna manera.
- Usted tendría que hacer publicidad de este Web site de alguna manera o nadie sabría que existió su actividad.
- Usted tendría que tener un abogado elaborar una licencia para su actividad.
- Usted tendría que subir con un cierto mecanismo para guardar a sus clientes del donante lejos copia

de su actividad.
- Usted tendría que crear una actividad tan astoundingly lista que nadie podría hacer algo similar y darle lejos.
- Usted tendría que ocuparse del hecho de que sus "clientes" serían niños sin tarjetas del dinero o de crédito.

En resumen, activities.sugarlabs.org no es el *almacén del App del iPhone*. Es un lugar en donde los programadores comparten y construyen sobre trabajo de cada uno y dan los resultados a los niños para libre. El GPL anima a eso que suceda, y recomiendo que usted elige eso para su licencia.

AGREGUE LOS COMENTARIOS DE LA LICENCIA A SU CÓDIGO DEL PYTHON

En la tapa de cada archivo de fuente del Python en su proyecto (excepto **setup.py**, que se comenta ya) ponga los comentarios como esto:

```
# filename    Program description
#
# Copyright (C) 2010 Your Name Here
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, write to the Free Software
# Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
```

Si el código se basa en algún otro código usted debe mencionar eso como cortesía.

CREE UN ARCHIVO DE .XO

Asegúrese que activity.info tenga el número de versión que usted quiere dar su actividad (actualmente debe ser un número entero positivo) y funcionar con este comando:

```
./setup.py dist_xo
```

Esto creará un directorio del **dist** si uno no existe y no pone un archivo nombrado algo como **ReadETextsII-1.xo** en él. El "1" indica la versión 1 de la actividad.

Si usted hizo todo la derecha este archivo de **.xo** debe estar listo para distribuir. Usted puede copiarlo a una impulsión del pulgar e instalarlo en un ordenador portátil de XO o sobre otro *azúcar* corriente de la impulsión del pulgar *en un palillo*. Usted debe hacer probablemente eso antes de distribuirlo más lejos. Tengo gusto de vivir con nuevas versiones de mis Actividades por una semana o tan antes de ponerlas en activities.sugarlabs.org.

Ahora sea un buen rato de agregar **dist a** su archivo de **.gitignore**, después de confiarlo y de empujarlo a Gitorious. Usted no quiere tener copias de sus archivos de .xo en Git. Otra buena cosa a hacer a este punto sería marcar su depósito de Git con etiqueta con el número de versión así que usted puede identificar qué código va con qué versión.

```
git tag -m "Release 1" v1 HEAD
git push --tags
```

AGREGUE SU ACTIVITY A ASLO

Cuando usted está listo para fijar el archivo de .xo en ASLO usted creará una cuenta como usted hizo con los otros Web site. Cuando usted ha abierto una sesión allí usted verá las **herramientas** ligar en la esquina correcta superior de la página. Chasque encendido eso y usted verá un menú móvil con una opción para el **eje del revelador**, que usted debe chascar encendido. Eso le llevará a las páginas donde usted puede agregar nuevas Actividades. La primera cosa que pide cuando la determinación de una nueva actividad es lo que utilizará la licencia usted. Después que usted no debe tener ninguìn problema el conseguir de su disposición de la actividad.

Usted necesitará crear un icono de la actividad como archivo de .gif y crear tiros de pantalla de su actividad en la acción. Usted puede hacer ambas cosas con El *GIMP* (programa de la manipulación de la imagen del GNU). Para el icono todo lo que usted necesita hacer es abrir el archivo de .svg con El GIMP y **ahorrarlo como** archivo de .gif.

Para los tiros de pantalla utilice el azúcar-emulador para exhibir su actividad en la acción, después utilice la opción de **Screenshot del** submenú del **crear del** menú de **archivo** con estas opciones:

Esto dice el GIMP esperar 10 segundos, después toma un screenshot de la ventana que usted chasca encendido con el ratón. Usted sabrá que los 10 segundos están para arriba porque el indicador de ratón desformará a a más (+) muestra. Usted también lo dice no incluir la decoración de la ventana (cuál significa la barra y la frontera de título de la ventana). Puesto que las ventanas en azúcar no tienen decoraciones el eliminar las decoraciones usadas por el azúcar-emulador le darán un screenshot que mire exactamente como una actividad del azúcar en la acción.

Cada actividad necesita un screenshot, pero usted puede tener más si usted tiene gusto. La venta de la ayuda de Screenshots la actividad y da instrucciones a los que la utilicen en lo que puede hacer la actividad. Desafortunadamente, ASLO no puede exhibir cuadros en una secuencia fiable, así que no se adapta a exhibir pasos para realizarse.

Otra cosa que usted necesitará proporcionar es un Home Page para su actividad. El que está para **Read Etexts** está aquí:

http://wiki.sugarlabs.org/go/Activities/Read_Etexts

Sí, un más Web site para conseguir un explicar. Una vez que usted hace usted puede especificar un acoplamiento con */go/Activities/some_name* y cuando usted chasca encendido que el acoplamiento el Wiki creará una página para usted. El software usado para el Wiki es MediaWiki, igual según lo utilizado para Wikipedia. Su página no necesita ser tan elaborada como es la mina, pero usted debe proporcionar definitivamente un acoplamiento a su código fuente en Gitorious.

14. ACTIVIDADES DEL SUGAR DEL DEPURACIÓN

INTRODUCCIÓN

No importa que tan cuidadoso sea usted, es probable que su actividad no le trabaje perfectamente al primer intento que intente correrlo. La depuración de errores de una actividad del Sugar es un tanto diferente a eliminar errores de un programa independiente. Cuando usted prueba un programa en realidad lo esta ejecutando. Si hay errores de sintaxis en el código que usted verá los mensajes de error en la consola en ese mismo momento, y si lo esta corriendo sobre **Eric** IDE la línea que contiene el error de código es seleccionado en el redactor así que usted puede corregirlo y continuar.

Con Sugar es un poco diferente. Es el ambiente de Sugar, no Eric, que funciona con su programa. Si hay errores de sintaxis en su código usted no los verá enseguida. En lugar de eso, el el icono de actividad que puede ver cuanto su actividad inicia comenzara a parpadear por algunos minutos, después de eso simplemente se ira y su actividad no podrá iniciar. La única manera que usted verá que error causó el problema será utilizar los **registros de actividades**. Si su programa no tiene ninguno n error de sintaxis pero tiene errores de lógica no le sera posible avanzar y depurar usando el depurador para encontrarlos. En lugar de eso, usted necesitara algún tipo de rastreador de registros para ver que esta sucediendo con su código y como ya fue mencionado, aga uso del "registro de actividades" para ver los registros en los mensajes. Es un buen momento para repetir el consejo que di anteriormente:

HAGA UNA VERSIÓN INDEPENDIENTE DE SU PROGRAMA PRIMERO

Lo que lo hace su actividad, es una buena apuesta que el 80% de él se podrían hacer por un programa independiente que sería mucho menos aburrido de eliminar errores. Si usted puede pensar en una manera de hacer su actividad runnable como una actividad o un programa independiente del Python entonces por supuesto la hace.

UTILICE PYLINT, PYCHECKER, O PYFLAKES

Una de las ventajas de una lengua compilada como **C** sobre una lengua interpretada como Python es que el recopilador hace un completo verifica del código antes de convertirlo a en lenguaje de máquina. Si hay errores de sintaxis que el recopilador le da mensajes de error informativos y para la compilación. Hay una **pelusa** para uso general de la llamada que los programador de lenguaje-c pueden utilizar para hacer cheques aún más cuidadosos que el recopilador haría y encontraría cosas cuestionables el entrar encendido en el código.

El Python no tiene un recopilador sino que tiene varios pelusa-como utilidades que usted puede funcionar con en su código antes de usted la prueba él. Estas utilidades son **pyflakes**, **pychecker**, y **pylint**. Cualquier distribución del linux debe tener tres disponibles.

PyFlakes

Aquí está un ejemplo de PyFlakes que usa:

```
pyflakes minichat.py
minichat.py:25: 'COLOR_BUTTON_GREY' imported but unused
minichat.py:28: 'XoColor' imported but unused
minichat.py:29: 'Palette' imported but unused
minichat.py:29: 'CanvasInvoker' imported but unused
```

PyFlakes parece hacer la menos comprobación de los tres, pero encuentra que los errores como éstos sobre eso que un ojo humano faltaría.

PyChecker

Aquí está PyChecker en la acción:

```
pychecker ReadEtextsActivity.py
Processing ReadEtextsActivity...
/usr/lib/python2.5/site-packages/dbus/_dbus.py:251: DeprecationWarning: The dbus_bindings module is
not public API and will go away soon.

Most uses of dbus_bindings are applications catching the exception
dbus.dbus_bindings.DBusException. You should use dbus.DBusException
instead (this is compatible with all dbus-python versions since 0.40.2).

If you need additional public API, please contact the maintainers via
.

  import dbus.dbus_bindings as m

Warnings...

/usr/lib/python2.5/site-packages/sugar/activity/activity.py:847: Parameter (ps) not used
/usr/lib/python2.5/site-packages/sugar/activity/activity.py:992: Parameter (event) not used
/usr/lib/python2.5/site-packages/sugar/activity/activity.py:992: Parameter (widget) not used
/usr/lib/python2.5/site-packages/sugar/activity/activity.py:996: Parameter (widget) not used

/usr/lib/python2.5/site-packages/sugar/graphics/window.py:157: No class attribute (_alert) found
/usr/lib/python2.5/site-packages/sugar/graphics/window.py:164: Parameter (window) not used
/usr/lib/python2.5/site-packages/sugar/graphics/window.py:188: Parameter (widget) not used
/usr/lib/python2.5/site-packages/sugar/graphics/window.py:200: Parameter (event) not used
/usr/lib/python2.5/site-packages/sugar/graphics/window.py:200: Parameter (widget) not used

ReadEtextsActivity.py:62: Parameter (widget) not used

4 errors suppressed, use -#/--limit to increase the number of errors displayed
```

PyChecker no sólo comprueba su código, él comprueba el código que usted importa, incluyendo código del azúcar.

PyLint

Aquí está PyLint, el más cuidadoso de los tres:

```
pylint ReadEtextsActivity.py
No config file found, using default configuration
************* Module ReadEtextsActivity
C:177: Line too long (96/80)
C:  1: Missing docstring
C: 27: Operator not preceded by a space
page=0
    ^
C: 27: Invalid name "page" (should match (([A-Z_][A-Z0-9_]*)|(__.*__))$)
C: 30:ReadEtextsActivity: Missing docstring
C:174:ReadEtextsActivity.read_file: Invalid name "zf" (should match [a-z_][a-z0-9_]{2,30}$)
W: 30:ReadEtextsActivity: Method 'write_file' is abstract in class 'Activity' but is not overridden
R: 30:ReadEtextsActivity: Too many ancestors (12/7)
W: 33:ReadEtextsActivity.__init__: Using the global statement
R: 62:ReadEtextsActivity.keypress_cb: Too many return statements (7/6)
C: 88:ReadEtextsActivity.page_previous: Missing docstring
W: 89:ReadEtextsActivity.page_previous: Using the global statement
C: 90:ReadEtextsActivity.page_previous: Operator not preceded by a space
        page=page-1
            ^
C: 91:ReadEtextsActivity.page_previous: Operator not preceded by a space
        if page ... A bunch of tables appear here ...

Global evaluation
-----------------
```

```
Your code has been rated at 7.52/10 (previous run: 7.52/10)
```

PyLint es el más resistente en su código y su ego. No sólo le dice de errores de sintaxis, le habla que todo alguien pudo encontrar la avería con en su código. Esto incluye las ediciones del estilo que no afectarán a cómo su código funciona pero afectarán a cómo es legible están a otros programadores.

LA ACTIVIDAD DEL REGISTRO

Cuando usted comienza a probar sus Actividades la actividad del registro será como su segunda casa. Exhibe una lista de ficheros de diario en el cristal izquierdo y cuando usted selecciona uno exhibirá el contenido del archivo en el cristal derecho. Cada vez que usted funciona con su actividad un nuevo fichero de diario se crea para él, así que usted puede comparar el registro que usted consiguió este vez con lo que usted consiguió en funcionamientos anteriores. La barra de herramientas del **corregir** es especialmente útil. Contiene un botón para demostrar el fichero de diario con las líneas envueltas (cuál no se gira por abandono pero probablemente debe estar). Tiene otro botón para copiar selecciones del registro al sujetapapeles, que será práctico si usted quiere demostrar mensajes de registro a otros reveladores.

La barra de herramientas de las **herramientas** tiene un botón para suprimir ficheros de diario. Nunca he encontrado una razón para utilizarla. Los ficheros de diario van lejos en sus los propios cuando usted cierra el azúcar-emulador.

Aquí es lo que parece la actividad del registro demostrar un error de sintaxis en su código:

REGISTRACIÓN

Sin una duda que hay la más vieja técnica del depuración sería la declaración simple de la impresión. Si usted tiene un programa corriente que comportarse mal debido a errores de lógica y usted no puede caminar con el código en una depuración para imaginar que qué está sucediendo usted puede ser que imprima declaraciones en su código. Por ejemplo, si usted no está seguro que un método está consiguiendo nunca ejecutado usted puede ser que ponga una declaración como esto como la primera línea del método:

```
def my_method():
    print 'my_method() begins'
```

Usted puede incluir datos en sus declaraciones de la impresión también. Supóngale necesitar saber cuántas veces se funciona con un lazo. Usted podría hacer esto:

```
        while linecount print 'linecount=', linecount
```

La salida de estas declaraciones de la impresión se puede considerar en la actividad del registro. Cuando le acaban que elimina errores de su programa usted quitaría estas declaraciones.

Un libro programado viejo que leí una vez hecho el caso para dejar las declaraciones en el programa acabado. ⁞ Los autores sentían que usar estas declaraciones para el depuración y ellos que los quitan es un poco como usar un paracaídas cuando el plano está en la tierra y tomarla de cuando es aerotransportado. Si el programa está hacia fuera en el mundo y tiene problemas que usted puede ser que desee bien que usted tuviera esas declaraciones en el código así que usted podría ayudar al usuario y usted mismo a imaginar qué se está encendiendo. Por una parte, las declaraciones de la impresión no están libres. Tardan tiempo para funcionar y llenan los ficheros de diario de desperdicios. Qué necesitamos son las declaraciones de la impresión que usted puede girarse apagado.

La manera que usted puede hacer esto está con la registración estándar del Python. En la forma usada por la mayoría de las Actividades parece esto:

```
        self._logger = logging.getLogger('read-etexts-activity')
```

Estas declaraciones entrarían en el método del *__init() de* su actividad. Cada vez que usted quiere hacer una declaración de la impresión () usted haría esto en lugar de otro:

```
    def _shared_cb(self, activity):
        self._logger.debug('My activity was shared')
        self.initiating = True
        self._sharing_setup()

        self._logger.debug('This is my activity: making a tube...')
        id = self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES].OfferDBusTube(
            SERVICE, {})

    def _sharing_setup(self):
        if self._shared_activity is None:
            self._logger.error('Failed to share or join activity')
            return
```

Note que hay dos clases de registración que se encienden aquí: **elimine errores** y **error**. Éstos son niveles de error. Cada declaración tiene uno, y controlan se funcionan que registran declaraciones y se no hacen caso que. Hay varios niveles de registro de errores, de la severidad más baja lo más arriba posible:

```
self._logger.debug("debug message")
self._logger.info("info message")
self._logger.warn("warn message")
self._logger.error("error message")
self._logger.critical("critical message")
```

Cuando usted fija el nivel de error en su programa a uno de estos valores usted consigue mensajes con ese nivel y más arriba. Usted puede fijar el nivel en su código del programa como esto:

```
self._logger.setLevel(logging.DEBUG)
```

Usted puede también fijar el nivel de registración fuera de su código del programa usando una **variable de entorno**. Por ejemplo, en el azúcar .82 y bájele puede comenzar el azúcar-emulador como esto:

```
SUGAR_LOGGER_LEVEL=debug sugar-emulator
```

La manera usted logra la misma cosa en el azúcar .84 y mayor es corregir el archivo **~/.sugar/debug** y el uncomment la línea que fija el SUGAR_LOGGER_LEVEL.

LA ACTIVIDAD DEL ANALIZAR

Otra actividad que usted puede encontrarse que que usa en un cierto punto es **analiza**. Esto es más probable ser utilizada para eliminar errores del azúcar sí mismo que eliminar errores de su actividad. Si, por ejemplo, su ambiente de prueba de la colaboración no parece trabajar esta actividad pudo ayudar le o algún otro a imaginar por qué.

No tengo mucho decir sobre esta actividad aquí, pero usted debe ser consciente que existe.


Activity
Interfac
Presence Service
nning: unique name :1.3
Log
INFO: Activity /org/laptop/Sugar/Presence/Activities/1 emitted NewChannel("…/MucChannel/1") or mentioned the cha
INFO: Activity /org/laptop/Sugar/Presence/Activities/1 emitted NewChannel("…/MucTubesChannel1") or mentioned th
INFO: Activity /org/laptop/Sugar/Presence/Activities/1 emitted BuddyJoined("…/keyid/7dbce07d43a5b5048e50f33588
INFO: <Buddy /org/laptop/Sugar/Presence/Buddies/keyid/7dbce07d43a5b5048e50f33588582222899c9e77>.GetPrope
INFO: Buddy /org/laptop/Sugar/Presence/Buddies/keyid/7dbce07d43a5b5048e50f33588582222899c9e77 emitted Acti
INFO: Buddy /org/laptop/Sugar/Presence/Buddies/keyid/7dbce07d43a5b5048e50f33588582222899c9e77 emitted Tele
Activities:
Object path
ID
Color
Type
Na
…/1
4c222799748a56b24e8e29c7d0c585f60e9f512c
#FF2B34,#00A0FF
net.flossmanuals.MiniChat
Min
Buddies:
Object path
Pubkey
…/keyid/7dbce07d43a5b5048e50f33588582222899c9e77
580 bytes, sha1 7dbce07d43a5b5048e50f33588582222899

ASUNTOS AVANZADOS

15. FABRICACIÓN DE ACTIVIDADES COMPARTIDAS
16. ADICIÓN DEL TEXTO AL DISCURSO
17. DIVERSIÓN CON EL DIARIO
18. CREACIÓN DE ACTIVIDADES USANDO PYGAME
19. FABRICACIÓN DE NUEVAS BARRAS DE HERRAMIENTAS DEL ESTILO

15. FABRICACIÓN DE ACTIVIDADES COMPARTIDAS

INTRODUCCIÓN:

Una de las características distintivas de Sugar es que muchas Actividades soportan ser usadas por más de una persona a la vez. Las computadoras se están utilizando cada vez más como medio de comunicaciones. Los últimos juegos de ordenador no sólo ponen el jugador contra la computadora; sino que crean un mundo donde los jugadores compiten unos contra otros. Los sitios de la Red como Facebook son cada vez más populares porque permiten que la gente obre recíprocamente uno con uno e incluso jugar los juegos. Es solamente natural que los programas informáticos educativos deben apoyar estas clases de interacciones.

Tengo una sobrina que sea un miembro entusiástico del sitio del Red "Club Pinguino" creado por Disney. Cuando le di un usb pendrive con "Sugar on a Stick, Blueberry"como un regalo adicional para la Navidad yo demostró la vista de la vecindad y le dijo que el Sugar haría su computadora entera como Club Penguino. Ella pensó que era una idea bastante genial. Con éso, yo sentía bastante genial también.

AZÚCAR CORRIENTE COMO MÁS DE UN USUARIO

Antes de que usted escriba cualquier pedazo de software usted necesita dar un cierto pensamiento a cómo usted lo probará. En el caso de una actividad compartida usted puede ser que piense you' la necesidad de d más de una computadora disponible de hacer la prueba, pero las que diseñaron el azúcar dio un cierto pensamiento a las Actividades compartidas prueba y nos dio maneras de probarlas usando solamente una computadora. Estos métodos se han estado desarrollando tan allí son variaciones leves en cómo usted prueba dependiendo de la versión del azúcar you' re usando. La primera cosa que usted tiene que saber es cómo funcionar con copias múltiples del azúcar como diversos usuarios.

Fedora 10 (Sugar .82)

En el Sugar .82 hay una manera práctica de funcionar con copias múltiples del azúcar-emulador y de hacer que cada copia sea un diverso usuario, sin tener que ser registrado en su caja del linux como más de un usuario. En la línea de comando para cada usuario adicional usted quiere agrega una variable de entorno de SUGAR_PROFILE como esto:

```
SUGAR_PROFILE=austen sugar-emulator
```

Cuando usted hace este sugar-emulator creará un directorio nombrado austen debajo de ~/.sugar para almacenar la información de perfil, el etc. Le incitarán incorporar un nombre y seleccionar los colores para su icono. Cada vez que usted lanza usando el SUGAR_PROFILE de austen le será este usuario. Si usted lanza sin SUGAR_PROFILE usted será el usuario regular que usted fijó antes.

Fedora 11 (Sugar .84)

Tan práctico como usando SUGAR_PROFILE están los reveladores del azúcar decidían que tenía limitaciones así que con la versión .84 y más adelante trabaja no más. Con .84 y más adelante usted necesita crear a un segundo usuario del linux y funcionar sus azúcar-emuladores como dos usuarios separados del linux. En el ambiente del GNOME hay usuarios y grupos de una opción en el submenú de la administración del menú de sistema que le permitirá fijar a un segundo usuario. Antes de que suba le incitará para la contraseña administrativa que usted creó cuando usted primer linux de la disposición.

¿Creando al segundo usuario es bastante simple, pero cómo usted va alrededor a ser abierta una sesión como dos diversos usuarios al mismo tiempo? It' s realmente bastante simple. Usted necesita abrir una ventana terminal y mecanografiar esto:

```
ssh -XY jausten@localhost
```

donde " jausten" es el userid del segundo usuario. Le pedirán verificar que la computadora en el "localhost" debe ser confiado en. Desde "localhost" apenas significa que usted está utilizando la red para conectar con otra cuenta en la misma computadora que es seguro contestar al " yes". Entonces le incitarán incorporar su contraseña, y desde entonces todo que usted hace en esa ventana terminal será hecha como ella. Usted puede lanzar el azúcar-emulador de ese terminal y la primera vez que usted lo hace le incitará para los colores de un nombre y del icono.

sugar-jhbuild

Con el sugar-jhbuild (la última versión del Sugar) las cosas son un pedacito diferente otra vez. Usted utilizará el método de apertura de sesión como los usuarios múltiples del linux como usted hicieron en .84, solamente de usted won' t consigue incitado para un nombre. En lugar el nombre se asoció al userid you' el re funcionamiento debajo será el nombre you' uso del ll en Sugar. Usted won' t pueda cambiarlo, pero usted podrá elegir sus colores del icono como antes.

Usted necesitará un separado instala del sugar-jhbuild para cada usuario. Este adicional instala irá rápidamente porque usted instaló todas las dependencias la primera vez.

CONEXIÓN CON OTROS USUARIOS

El azúcar utiliza a Telepathy llamada software que aplique un protocolo inmediato de mensajería llamado XMPP (protocolo extendido de la mensajería y de la presencia). Este protocolo era llamado Jabber. Esencialmente la telepatía le deja poner a un cliente de mensajería inmediato en su actividad. Usted puede utilizar esto para enviar mensajes de individual, ejecuta métodos remotamente, y hace transferencias de archivo.

Hay realmente dos maneras que azucaran a usuarios pueden ensamblar juntas en una red:

Salut

Si dos usuarios de la computadora están conectados con el mismo segmento de una red deben poder encontrarse y Actividades de la parte. Si usted tiene una red casera donde cada uno utiliza el mismo ranurador usted puede compartir con otros en esa red. Esto a veces se llama Acoplamiento-Local XMPP. El software de la telepatía que hace este posible se llama Salut.

El ordenador portátil de XO tiene el soporte físico y software especiales para apoyar el establecimiento de una red del acoplamiento, donde los ordenadores portátiles de XO que están cerca de uno a pueden comenzar automáticamente establecimiento de una red con uno a sin la necesidad de un ranurador. Por lo que el azúcar, él doesn' materia de t qué un poco red que usted tiene. Atado con alambre o sin hilos, acoplamiento o no, todos trabajan.

Servidor del Jabber

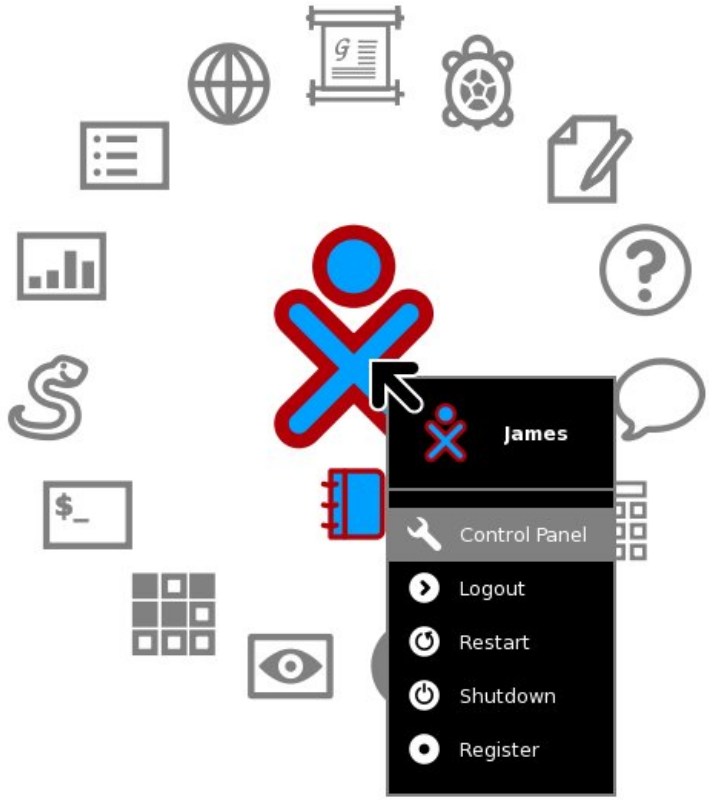
La otra manera de conectar con otros usuarios está pasando a través de un servidor del Jabber. La ventaja de usar un servidor del Jabber es usted puede entrar en contacto con y compartir Actividades con la gente fuera de su propia red. Esta gente pudo incluso estar en el otro lado del mundo. El Jabber permite que las Actividades en diversas redes conecten cuando ambas redes son protegidas por los cortafuegos. La parte

de la telepatía que trabaja con un servidor del Jabber se llama Gabble.

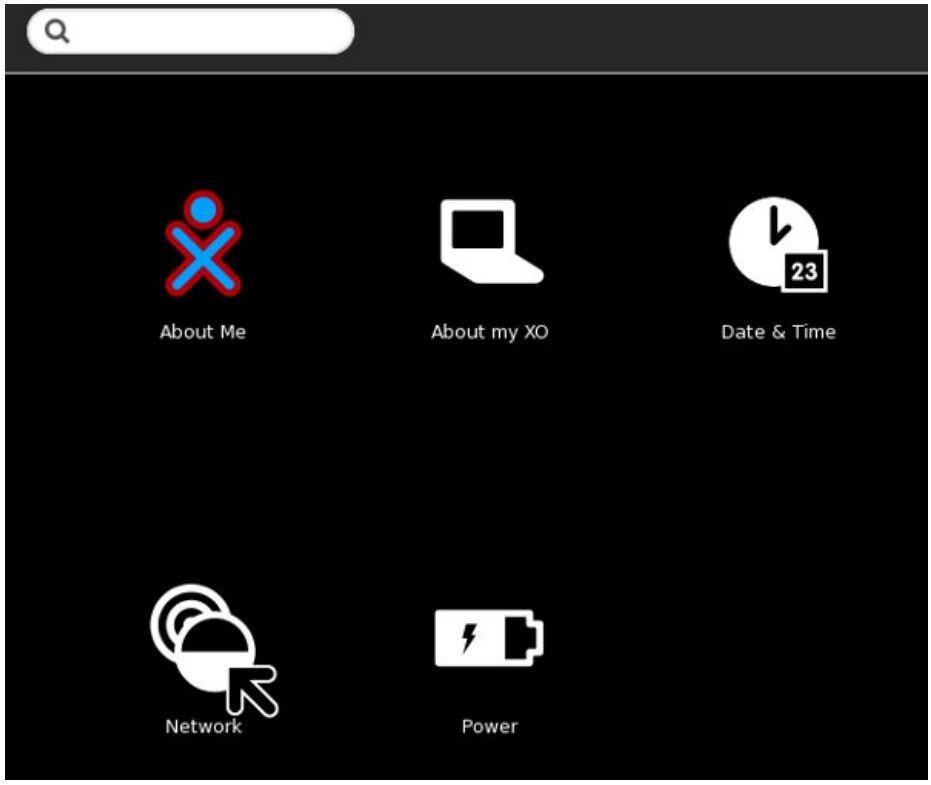
Usted debe utilizar generalmente Salut para la prueba si en todo posible. Esto simplifica la prueba y doesn' uso de t encima de recursos en un servidor del Jabber.

No importa si su actividad conecta con otras usando charla o Salut. De hecho, la actividad no tiene ninguna idea que esté utilizando. Esos detalles son ocultados de la actividad por Telepathy. Cualquier actividad que trabaje con Salut trabajará con la charla y viceversa.

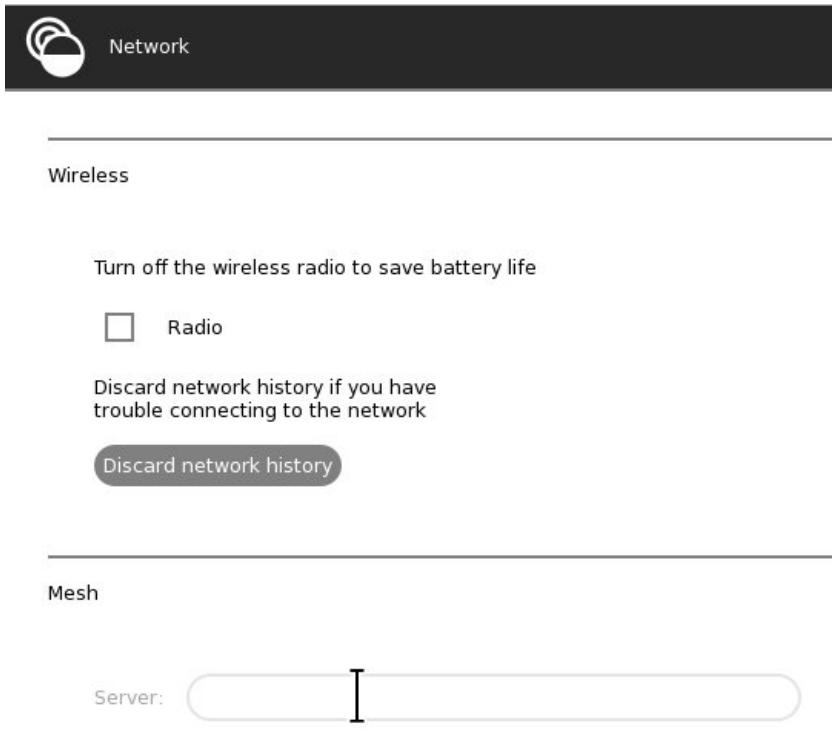
A fijar el sugar-emulador para utilizar Salut vaya al panel de control del Sugar:



En el Sugar .82 esta opción del menú es panel de control. En versiones posteriores es **My Settings**.



Click on the **Network** icon.



El campo del servidor en esta pantalla debe ser vacío utilizar Salut. Usted puede utilizar la llave de tecla de retroceso para quitar cualquier entrada allí.

Usted necesitará seguir estos pasos para cada usuario del azúcar que participe en su prueba. Si por alguna razón usted desea probar su actividad usando un servidor del Jabber que el OLPC Wiki mantiene una lista de público - los servidores disponibles en http://wiki.laptop.org/go/Community_Jabber_Servers.

Una vez que usted tiene Salut o un servidor del Jabber fija en ambos casos del azúcar que usted le esté funcionando con deba mirar la opinión de la vecindad de ambos para ver si pueden detectarse, y quizás pruebe la actividad de la charla entre los dos. Si usted tiene eso you' de trabajo; re aliste para intentar programar una actividad compartida.

LA ACTIVIDAD DE MINICHAT

Apenas como tomamos la actividad leída de Etexts y la pelamos abajo a los fundamentos we' el re ir a hacer iguales a la actividad de la charla para crear una nueva actividad llamó MiniChat. La actividad verdadera de la charla tiene un número de características esas nosotros don' necesidad de t de demostrar mensajería compartida de la actividad:

- Tiene la capacidad de cargar su código fuente en Pippy para la visión. Ésta era una característica que todas las Actividades en el XO fueron supuestas para tener, pero la charla es una del pocos que lo ejecutaron. Personalmente, si quiero ver un Activity' código de s que prefiero ir a git.sugarlabs.org donde puedo ver las viejas versiones del código así como el más último.
- La charla puede conectar uno a uno con un cliente convencional de XMPP. Esto puede ser útil para la charla pero no sería necesario o deseable para la mayoría de las Actividades compartidas.
- Si usted incluye un URL en un mensaje de la charla el interfaz utilizador le permite chascar encendido

el URL hace una entrada de diario para ese URL. Usted puede entonces utilizar el diario para abrirlo con la actividad de la ojeada. (Esto es necesario porque las Actividades no pueden lanzarse). Bastante fresco, pero no necesitado demostrar cómo hacer una actividad compartida.
- La sesión de la charla se almacena en el diario. Cuando usted reasume una entrada de la charla del diario restaura los mensajes de su anterior charla la sesión en el interfaz utilizador. Sabemos ya ahorrar cosas al diario y restaurar cosas del diario, tan MiniChat won' t hace esto.

El código resultante está sobre mitad mientras la original. Realicé algunos otros cambios también:

- El campo de entrada de texto está sobre los mensajes de la charla, en vez de abajo. Esto hace más fácil hacer los screenshots parciales de la actividad en la acción.
- Quité la nueva barra de herramientas del estilo y agregué una barra de herramientas del viejo estilo, así que podría probarla en Fedora 10 y 11 que don' ayuda de t las nuevas barras de herramientas.
- Tomé la clase TextChannelWrapper y la puse en su propio archivo. Hice esto porque la clase parecida ella pudo ser útil para otros proyectos.

El código y todos los archivos favorables para MiniChat están en el directorio de MiniChat del depósito de Git. You' necesidad del ll de funcionar

```
./setup.py dev
```

en el proyecto para hacerlo listo para probar. **activity.info** parece esto:

```
[Activity]
name = Mini Chat
service_name = net.flossmanuals.MiniChat
icon = chat
exec = sugar-activity minichat.MiniChat
show_launcher = yes
activity_version = 1
license = GPLv2+
```

Aquí está el código para **textchannel.py**:

```
import logging

from telepathy.client import Connection, Channel
from telepathy.interfaces import (
    CHANNEL_INTERFACE, CHANNEL_INTERFACE_GROUP, CHANNEL_TYPE_TEXT,
    CONN_INTERFACE_ALIASING)
from telepathy.constants import (
    CHANNEL_GROUP_FLAG_CHANNEL_SPECIFIC_HANDLES,
    CHANNEL_TEXT_MESSAGE_TYPE_NORMAL)

class TextChannelWrapper(object):
    """Wrap a telepathy Text Channel to make usage simpler."""
    def __init__(self, text_chan, conn):
        """Connect to the text channel"""
        self._activity_cb = None
        self._activity_close_cb = None
        self._text_chan = text_chan
        self._conn = conn
        self._logger = logging.getLogger(
            'minichat-activity.TextChannelWrapper')
        self._signal_matches = []
        m = self._text_chan[CHANNEL_INTERFACE].connect_to_signal(
            'Closed', self._closed_cb)
        self._signal_matches.append(m)

    def send(self, text):
        """Send text over the Telepathy text channel."""
        # XXX Implement CHANNEL_TEXT_MESSAGE_TYPE_ACTION
        if self._text_chan is not None:
            self._text_chan[CHANNEL_TYPE_TEXT].Send(
                CHANNEL_TEXT_MESSAGE_TYPE_NORMAL, text)

    def close(self):
```

```
        """Close the text channel."""
        self._logger.debug('Closing text channel')
        try:
            self._text_chan[CHANNEL_INTERFACE].Close()
        except:
            self._logger.debug('Channel disappeared!')
            self._closed_cb()

    def _closed_cb(self):
        """Clean up text channel."""
        self._logger.debug('Text channel closed.')
        for match in self._signal_matches:
            match.remove()
        self._signal_matches = []
        self._text_chan = None
        if self._activity_close_cb is not None:
            self._activity_close_cb()

    def set_received_callback(self, callback):
        """Connect the function callback to the signal.

        callback -- callback function taking buddy and text args
        """
        if self._text_chan is None:
            return
        self._activity_cb = callback
        m = self._text_chan[CHANNEL_TYPE_TEXT].connect_to_signal('Received',
            self._received_cb)
        self._signal_matches.append(m)

    def handle_pending_messages(self):
        """Get pending messages and show them as received."""
        for id, timestamp, sender, type, flags, text in \
            self._text_chan[
                CHANNEL_TYPE_TEXT].ListPendingMessages(False):
            self._received_cb(id, timestamp, sender, type, flags, text)

    def _received_cb(self, id, timestamp, sender, type, flags, text):
        """Handle received text from the text channel.

        Converts sender to a Buddy.
        Calls self._activity_cb which is a callback to the activity.
        """
        if self._activity_cb:
            buddy = self._get_buddy(sender)
            self._activity_cb(buddy, text)
            self._text_chan[
                CHANNEL_TYPE_TEXT].AcknowledgePendingMessages([id])
        else:
            self._logger.debug('Throwing received message on the floor'
                ' since there is no callback connected. See '
                'set_received_callback')

    def set_closed_callback(self, callback):
        """Connect a callback for when the text channel is closed.

        callback -- callback function taking no args

        """
        self._activity_close_cb = callback

    def _get_buddy(self, cs_handle):
        """Get a Buddy from a (possibly channel-specific) handle."""
        # XXX This will be made redundant once Presence Service
        # provides buddy resolution
        from sugar.presence import presenceservice
        # Get the Presence Service
        pservice = presenceservice.get_instance()
        # Get the Telepathy Connection
        tp_name, tp_path = pservice.get_preferred_connection()
        conn = Connection(tp_name, tp_path)
        group = self._text_chan[CHANNEL_INTERFACE_GROUP]
        my_csh = group.GetSelfHandle()
        if my_csh == cs_handle:
            handle = conn.GetSelfHandle()
```

```
        elif group.GetGroupFlags() & \
            CHANNEL_GROUP_FLAG_CHANNEL_SPECIFIC_HANDLES:
            handle = group.GetHandleOwners([cs_handle])[0]
        else:
            handle = cs_handle

            # XXX: deal with failure to get the handle owner
            assert handle != 0

        return pservice.get_buddy_by_telepathy_handle(
            tp_name, tp_path, handle)
```

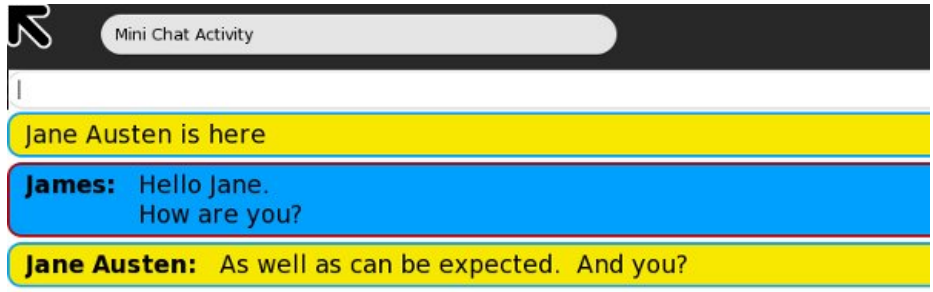
Aquí está el código para **minichat.py**:

```
from gettext import gettext as _
import hippo
import gtk
import pango
import logging
from sugar.activity.activity import Activity, ActivityToolbox, SCOPE_PRIVATE
from sugar.graphics.alert import NotifyAlert
from sugar.graphics.style import (Color, COLOR_BLACK, COLOR_WHITE,
    COLOR_BUTTON_GREY, FONT_BOLD, FONT_NORMAL)
from sugar.graphics.roundbox import CanvasRoundBox
from sugar.graphics.xocolor import XoColor
from sugar.graphics.palette import Palette, CanvasInvoker

from textchannel import TextChannelWrapper

logger = logging.getLogger('minichat-activity')

class MiniChat(Activity):
    def __init__(self, handle):
        Activity.__init__(self, handle)

        root = self.make_root()
        self.set_canvas(root)
        root.show_all()
        self.entry.grab_focus()

        toolbox = ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False
        self.set_toolbox(toolbox)
        toolbox.show()

        self.owner = self._pservice.get_owner()
        # Auto vs manual scrolling:
        self._scroll_auto = True
        self._scroll_value = 0.0
        # Track last message, to combine several messages:
        self._last_msg = None
        self._last_msg_sender = None
        self.text_channel = None

        if self._shared_activity:
            # we are joining the activity
            self.connect('joined', self._joined_cb)
            if self.get_shared():
                # we have already joined
                self._joined_cb()
        else:
            # we are creating the activity
            if not self.metadata or self.metadata.get('share-scope',
                    SCOPE_PRIVATE) == SCOPE_PRIVATE:
                # if we are in private session
                self._alert(_('Off-line'), _('Share, or invite someone.'))
            self.connect('shared', self._shared_cb)

    def _shared_cb(self, activity):
        logger.debug('Chat was shared')
        self._setup()

    def _setup(self):
        self.text_channel = TextChannelWrapper(
```

```
            self._shared_activity.telepathy_text_chan,
            self._shared_activity.telepathy_conn)
        self.text_channel.set_received_callback(self._received_cb)
        self._alert(_('On-line'), _('Connected'))
        self._shared_activity.connect('buddy-joined', self._buddy_joined_cb)
        self._shared_activity.connect('buddy-left', self._buddy_left_cb)
        self.entry.set_sensitive(True)
        self.entry.grab_focus()

    def _joined_cb(self, activity):
        """Joined a shared activity."""
        if not self._shared_activity:
            return
        logger.debug('Joined a shared chat')
        for buddy in self._shared_activity.get_joined_buddies():
            self._buddy_already_exists(buddy)
        self._setup()

    def _received_cb(self, buddy, text):
        """Show message that was received."""
        if buddy:
                nick = buddy.props.nick
        else:
            nick = '???'
        logger.debug('Received message from %s: %s', nick, text)
        self.add_text(buddy, text)

    def _alert(self, title, text=None):
        alert = NotifyAlert(timeout=5)
        alert.props.title = title
        alert.props.msg = text
        self.add_alert(alert)
        alert.connect('response', self._alert_cancel_cb)
        alert.show()

    def _alert_cancel_cb(self, alert, response_id):
        self.remove_alert(alert)

    def _buddy_joined_cb (self, activity, buddy):
        """Show a buddy who joined"""
        if buddy == self.owner:
            return
        if buddy:
            nick = buddy.props.nick
        else:
            nick = '???'
        self.add_text(buddy, buddy.props.nick+' '+_('joined the chat'),
            status_message=True)

    def _buddy_left_cb (self, activity, buddy):
        """Show a buddy who joined"""
        if buddy == self.owner:
            return
        if buddy:
            nick = buddy.props.nick
        else:
            nick = '???'
        self.add_text(buddy, buddy.props.nick+' '+_('left the chat'),
            status_message=True)

    def _buddy_already_exists(self, buddy):
        """Show a buddy already in the chat."""
        if buddy == self.owner:
            return
        if buddy:
            nick = buddy.props.nick
        else:
            nick = '???'
        self.add_text(buddy, buddy.props.nick+' '+_('is here'),
            status_message=True)

    def make_root(self):
        conversation = hippo.CanvasBox(
            spacing=0,
            background_color=COLOR_WHITE.get_int())
```

```
        self.conversation = conversation

        entry = gtk.Entry()
        entry.modify_bg(gtk.STATE_INSENSITIVE,
                        COLOR_WHITE.get_gdk_color())
        entry.modify_base(gtk.STATE_INSENSITIVE,
                          COLOR_WHITE.get_gdk_color())
        entry.set_sensitive(False)
        entry.connect('activate', self.entry_activate_cb)
        self.entry = entry

        hbox = gtk.HBox()
        hbox.add(entry)

        sw = hippo.CanvasScrollbars()
        sw.set_policy(hippo.ORIENTATION_HORIZONTAL, hippo.SCROLLBAR_NEVER)
        sw.set_root(conversation)
        self.scrolled_window = sw

        vadj = self.scrolled_window.props.widget.get_vadjustment()
        vadj.connect('changed', self.rescroll)
        vadj.connect('value-changed', self.scroll_value_changed_cb)

        canvas = hippo.Canvas()
        canvas.set_root(sw)

        box = gtk.VBox(homogeneous=False)
        box.pack_start(hbox, expand=False)
        box.pack_start(canvas)

        return box

    def rescroll(self, adj, scroll=None):
        """Scroll the chat window to the bottom"""
        if self._scroll_auto:
            adj.set_value(adj.upper-adj.page_size)
            self._scroll_value = adj.get_value()

    def scroll_value_changed_cb(self, adj, scroll=None):
        """Turn auto scrolling on or off.

        If the user scrolled up, turn it off.
        If the user scrolled to the bottom, turn it back on.
        """
        if adj.get_value()
```

Y esto es lo que parece la actividad en la acción:



Intente poner en marcha más de una copia del azúcar-emulador, con esta actividad instalado en cada uno. Si you' re usando Fedora 10 y SUGAR_PROFILE que la actividad no necesita ser instalada más de una vez, pero si you' re usando una versión posterior del azúcar que requiere las identificaciones del usuario separadas del linux para cada caso you' necesidad del ll de mantener las copias separadas del código para cada usuario. En sus propios proyectos usando un depósito central de Git en git.sugarlabs.org hará esto fácil. Usted apenas hace un empuje del git para copiar sus cambios al depósito central y a un tirón del git para copiarlos a su segundo userid. El segundo userid puede utilizar el URL del público. There' s ninguna necesidad de fijar SSH para cualquie usuario con excepción el primario.

Usted pudo haber leído en alguna parte que usted puede instalar una actividad en una máquina y parte que la actividad con otra que no tiene la actividad instaló. En tal caso la segunda máquina conseguiría una copia de la actividad de la primera máquina y la instalaría automáticamente. Usted pudo también haber leído que si dos usuarios de una actividad compartida tienen diversas versiones de esa actividad entonces la persona que tiene la más nueva versión pondrá al día automáticamente el más viejo. Ninguna de las dos declaraciones es verdades ahora o es probables ser verdades en un futuro próximo. Estas ideas se discuten en las listas de personas a quienes se mandan propaganda de vez en cuando pero hay dificultades prácticas a superar antes de que algo similar podría trabajar, teniendo que sobre todo hacer con seguridad. Para ahora ambos usuarios de una actividad compartida debe hacer la actividad instalar. Por una parte, dependiendo cómo se escribe la actividad de dos diversas versiones de una actividad puede poder comunicar el uno con el otro. Si los mensajes que intercambian están en el mismo formato allí no son ninguÌn problema.

Una vez que usted tiene ambos casos del azúcar-emulador el ir usted puede lanzar MiniChat en uno e invitar al segundo usuario que ensamble la sesión de la charla. Usted puede hacer ambos con los cristales de la vecindad de cada caso. La fabricación de la invitación parece esto:

Aceptarlo parece esto:

Después de you' VE jugó con MiniChat para vuelto un rato y we' el ll discute los secretos de usar la telepatía para crear una actividad compartida.

SEPA QUIÉN SON SUS BUDDIES

XMPP, como dijimos antes, es el **Extended Messaging and Presence Protocol**. La **Presence** es apenas como lo que suena; maneja dejarle saber quién está disponible compartir su actividad, así como cuál están disponibles otras Actividades compartir. Hay dos maneras de compartir su actividad. Primer es con cuando usted cambia la parte: la desconexión en la barra de herramientas estándar así que ella lee mi vecindad en vez de privado. Eso significa que cualquier persona en la red puede compartir su actividad. La otra manera de compartir es ir a la opinión de la vecindad e invitar alguien específico que comparta. La persona que consigue la invitación no tiene ninguna idea de la invitación estaba específicamente para él o la difusión a la vecindad. El término técnico para las personas que comparten su actividad es compinches. El lugar en donde los compinches se encuentran y colaboran se llama un MUC o un usuario multi Chatroom.

El código usado por nuestra actividad para los compinches de invitación y ensamblar la actividad como compinche está en *__init__()* method:

```
        if self._shared_activity:
            # we are joining the activity
            self.connect('joined', self._joined_cb)
            if self.get_shared():
                # we have already joined
                self._joined_cb()
        else:
            # we are creating the activity
            if not self.metadata or self.metadata.get('share-scope',
```

```
                SCOPE_PRIVATE) == SCOPE_PRIVATE:
            # if we are in private session
            self._alert(_('Off-line'), _('Share, or invite someone.'))
        self.connect('shared', self._shared_cb)

    def _shared_cb(self, activity):
        logger.debug('Chat was shared')
        self._setup()

    def _joined_cb(self, activity):
        """Joined a shared activity."""
        if not self._shared_activity:
            return
        logger.debug('Joined a shared chat')
        for buddy in self._shared_activity.get_joined_buddies():
            self._buddy_already_exists(buddy)
        self._setup()

    def _setup(self):
        self.text_channel = TextChannelWrapper(
            self._shared_activity.telepathy_text_chan,
            self._shared_activity.telepathy_conn)
        self.text_channel.set_received_callback(self._received_cb)
        self._alert(_('On-line'), _('Connected'))
        self._shared_activity.connect('buddy-joined', self._buddy_joined_cb)
        self._shared_activity.connect('buddy-left', self._buddy_left_cb)
        self.entry.set_sensitive(True)
        self.entry.grab_focus()
```

Hay dos maneras de poner en marcha una actividad: como el primer usuario de una actividad o ensamblando una actividad existente. La primera línea arriba en en negrilla determina si estamos ensamblando o somos el primer usuario de la actividad. Si pedimos tan el método del _joined_cb () ser funcionados con cuando el ' joined' el acontecimiento ocurre. Este método consigue una lista del compinche del objeto del _shared_activity y crea mensajes en el interfaz utilizador que informa al usuario que estos compinches están ya en la sala de chat. Entonces funciona con el método del _setup ().

Si no estamos ensamblando una actividad existente entonces comprobamos para ver si estamos compartiendo actualmente la actividad con cualquier persona. Si nosotros aren' t surgimos un mensaje que dice al usuario invitar alguien que charle. También pedimos eso cuando el ' shared' incluso sucede el método del *_shared_cb()* debe funcionar. Este método apenas funciona con el método del *_setup()*.

El método del _setup () crea un objeto de TextChannelWrapper usando el código en textchannel.py. También dice a objeto del _shared_activity que quiere algunos métodos de servicio repetido funcionados con cuando los nuevos compinches ensamblan la actividad y cuando los compinches existentes dejan la actividad. Todo que usted necesita saber sobre sus compinches se puede encontrar en el código arriba, excepto cómo enviarles mensajes. Para eso utilizamos el canal del texto. No hay necesidad de aprender sobre el canal del texto con gran detalle porque la clase de TextChannelWrapper hace todo you' del ll necesidad nunca de hacer con el TextChannel y las pieles los detalles de usted.

```
    def entry_activate_cb(self, entry):
        text = entry.props.text
        logger.debug('Entry: %s' % text)
        if text:
            self.add_text(self.owner, text)
            entry.props.text = ''
            if self.text_channel:
                self.text_channel.send(text)
            else:
                logger.debug('Tried to send message but text channel '
                    'not connected.')
```

El método del add_text () está de interés. Toma al dueño del mensaje e imagina qué colores pertenecen a ese dueño y exhibe el mensaje en esos colores. En el caso de los mensajes enviados por la actividad consigue al dueño como esto en el método del *__init__()*:

```
        self.owner = self._pservice.get_owner()
```

En el caso de mensajes recibidos consigue al Buddy que el mensaje vino de:

```
    def _received_cb(self, buddy, text):
        """Show message that was received."""
        if buddy:
                nick = buddy.props.nick
        else:
            nick = '???'
        logger.debug('Received message from %s: %s', nick, text)
        self.add_text(buddy, text)
```

¿Pero qué si queremos más que apenas envían los mensajes de texto hacia adelante y hacia atrás? ¿Qué utilizamos para ése?

IT'S A SERIES OF TUBES!

No, no el Internet. La telepatía tiene un concepto llamado Tubes que describa la manera que los casos de una actividad pueden comunicar juntos. Lo hace es qué telepatía toma el canal del texto y construya los tubos encima de él. Hay dos clases de tubos:

- D-Bus Tubes
- Stream Tubes

Un tubo del D-Bus se utiliza para permitir a un caso de una actividad llamar métodos en los casos del compinche de la actividad. Un tubo de corriente se utiliza para enviar datos sobre los zócalos, por ejemplo para copiar un archivo a partir de un caso de una actividad a otro. Un zócalo es una manera de comunicación sobre una red usando protocolos del Internet. Por ejemplo el protocolo del HTTP usado por el World Wide Web se aplica con los zócalos. En el ejemplo siguiente we' HTTP del uso del ll a los libros de transferencia a partir de un caso de **Read Etexts III** a otro.

¡READ ETEXTS III, AHORA CON LA DISTRIBUCIÓN DEL LIBRO!

El depósito de Git con las muestras del código para este libro tiene un archivo nombrado ReadEtextsActivity3.py en el directorio de Making_Shared_Activities que parece esto:

```
import sys
import os
import logging
import tempfile
import time
import zipfile
import pygtk
import gtk
import pango
import dbus
import gobject
import telepathy
from sugar.activity import activity
from sugar.graphics import style
from sugar import network
from sugar.datastore import datastore
from sugar.graphics.alert import NotifyAlert
from toolbar import ReadToolbar, ViewToolbar
from gettext import gettext as _

page=0
PAGE_SIZE = 45
TOOLBAR_READ = 2

logger = logging.getLogger('read-etexts2-activity')
```

```
class ReadHTTPRequestHandler(network.ChunkedGlibHTTPRequestHandler):
    """HTTP Request Handler for transferring document while collaborating.

    RequestHandler class that integrates with Glib mainloop. It writes
    the specified file to the client in chunks, returning control to the
    mainloop between chunks.

    """
    def translate_path(self, path):
        """Return the filepath to the shared document."""
        return self.server.filepath


class ReadHTTPServer(network.GlibTCPServer):
    """HTTP Server for transferring document while collaborating."""
    def __init__(self, server_address, filepath):
        """Set up the GlibTCPServer with the ReadHTTPRequestHandler.

        filepath -- path to shared document to be served.
        """
        self.filepath = filepath
        network.GlibTCPServer.__init__(self, server_address,
                                       ReadHTTPRequestHandler)


class ReadURLDownloader(network.GlibURLDownloader):
    """URLDownloader that provides content-length and content-type."""

    def get_content_length(self):
        """Return the content-length of the download."""
        if self._info is not None:
            return int(self._info.headers.get('Content-Length'))

    def get_content_type(self):
        """Return the content-type of the download."""
        if self._info is not None:
            return self._info.headers.get('Content-type')
        return None

READ_STREAM_SERVICE = 'read-etexts-activity-http'

class ReadEtextsActivity(activity.Activity):
    def __init__(self, handle):
        "The entry point to the Activity"
        global page
        activity.Activity.__init__(self, handle)

        self.fileserver = None
        self.object_id = handle.object_id

        toolbox = activity.ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False

        self.edit_toolbar = activity.EditToolbar()
        self.edit_toolbar.undo.props.visible = False
        self.edit_toolbar.redo.props.visible = False
        self.edit_toolbar.separator.props.visible = False
        self.edit_toolbar.copy.set_sensitive(False)
        self.edit_toolbar.copy.connect('clicked', self.edit_toolbar_copy_cb)
        self.edit_toolbar.paste.props.visible = False
        toolbox.add_toolbar(_('Edit'), self.edit_toolbar)
        self.edit_toolbar.show()

        self.read_toolbar = ReadToolbar()
        toolbox.add_toolbar(_('Read'), self.read_toolbar)
        self.read_toolbar.back.connect('clicked', self.go_back_cb)
        self.read_toolbar.forward.connect('clicked', self.go_forward_cb)
        self.read_toolbar.num_page_entry.connect('activate',
                                      self.num_page_entry_activate_cb)
        self.read_toolbar.show()

        self.view_toolbar = ViewToolbar()
        toolbox.add_toolbar(_('View'), self.view_toolbar)
        self.view_toolbar.connect('go-fullscreen',
```

```
            self.view_toolbar_go_fullscreen_cb)
        self.view_toolbar.zoom_in.connect('clicked', self.zoom_in_cb)
        self.view_toolbar.zoom_out.connect('clicked', self.zoom_out_cb)
        self.view_toolbar.show()

        self.set_toolbox(toolbox)
        toolbox.show()
        self.scrolled_window = gtk.ScrolledWindow()
        self.scrolled_window.set_policy(gtk.POLICY_NEVER, gtk.POLICY_AUTOMATIC)
        self.scrolled_window.props.shadow_type = gtk.SHADOW_NONE

        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_cursor_visible(False)
        self.textview.set_left_margin(50)
        self.textview.connect("key_press_event", self.keypress_cb)

        self.progressbar = gtk.ProgressBar()
        self.progressbar.set_orientation(gtk.PROGRESS_LEFT_TO_RIGHT)
        self.progressbar.set_fraction(0.0)

        self.scrolled_window.add(self.textview)
        self.textview.show()
        self.scrolled_window.show()

        vbox = gtk.VBox()
        vbox.pack_start(self.progressbar,  False,  False,  10)
        vbox.pack_start(self.scrolled_window)
        self.set_canvas(vbox)
        vbox.show()

        page = 0
        self.clipboard = gtk.Clipboard(display=gtk.gdk.display_get_default(), \
                                       selection="CLIPBOARD")
        self.textview.grab_focus()
        self.font_desc = pango.FontDescription("sans %d" % style.zoom(10))
        self.textview.modify_font(self.font_desc)

        buffer = self.textview.get_buffer()
        self.markset_id = buffer.connect("mark-set", self.mark_set_cb)

        self.toolbox.set_current_toolbar(TOOLBAR_READ)
        self.unused_download_tubes = set()
        self.want_document = True
        self.download_content_length = 0
        self.download_content_type = None
        # Status of temp file used for write_file:
        self.tempfile = None
        self.close_requested = False
        self.connect("shared", self.shared_cb)

        self.is_received_document = False

        if self._shared_activity and handle.object_id == None:
            # We're joining, and we don't already have the document.
            if self.get_shared():
                # Already joined for some reason, just get the document
                self.joined_cb(self)
            else:
                # Wait for a successful join before trying to get the document
                self.connect("joined", self.joined_cb)

    def keypress_cb(self, widget, event):
        "Respond when the user presses one of the arrow keys"
        keyname = gtk.gdk.keyval_name(event.keyval)
        print keyname
        if keyname == 'plus':
            self.font_increase()
            return True
        if keyname == 'minus':
            self.font_decrease()
            return True
        if keyname == 'Page_Up' :
            self.page_previous()
            return True
```

```
    if keyname == 'Page_Down':
        self.page_next()
        return True
    if keyname == 'Up' or keyname == 'KP_Up' \
            or keyname == 'KP_Left':
        self.scroll_up()
        return True
    if keyname == 'Down' or keyname == 'KP_Down' \
            or keyname == 'KP_Right':
        self.scroll_down()
        return True
    return False

def num_page_entry_activate_cb(self, entry):
    global page
    if entry.props.text:
        new_page = int(entry.props.text) - 1
    else:
        new_page = 0

    if new_page >= self.read_toolbar.total_pages:
        new_page = self.read_toolbar.total_pages - 1
    elif new_page = len(self.page_index): page=0
    self.read_toolbar.set_current_page(page)
    self.show_page(page)
    v_adjustment = self.scrolled_window.get_vadjustment()
    v_adjustment.value = v_adjustment.lower

def zoom_in_cb(self,  button):
    self.font_increase()

def zoom_out_cb(self,  button):
    self.font_decrease()

def font_decrease(self):
    font_size = self.font_desc.get_size() / 1024
    font_size = font_size - 1
    if font_size  v_adjustment.upper - v_adjustment.page_size:
            new_value = v_adjustment.upper - v_adjustment.page_size
        v_adjustment.value = new_value

def scroll_up(self):
    v_adjustment = self.scrolled_window.get_vadjustment()
    if v_adjustment.value == v_adjustment.lower:
        self.page_previous()
        return
    if v_adjustment.value > v_adjustment.lower:
        new_value = v_adjustment.value - \
            v_adjustment.step_increment
        if new_value  0):
            newPage = title[i] + newPage
            i = i - 1
        if title[i] == 'P':
            page = int(newPage) - 1
        else:
            # not a page number; maybe a volume number.
            page = 0

def save_page_number(self):
    global page
    title = self.metadata.get('title', '')
    if title == ''  or not title[len(title)- 1].isdigit():
        title = title + ' P' +  str(page + 1)
    else:
        i = len(title) - 1
        while (title[i].isdigit() and i > 0):
            i = i - 1
        if title[i] == 'P':
            title = title[0:i] + 'P' + str(page + 1)
        else:
            title = title + ' P' + str(page + 1)
    self.metadata['title'] = title

def read_file(self, filename):
    "Read the Etext file"
```

```
        global PAGE_SIZE,  page

        tempfile = os.path.join(self.get_activity_root(),  'instance', \
                'tmp%i' % time.time())
        os.link(filename,  tempfile)
        self.tempfile = tempfile

        if zipfile.is_zipfile(filename):
            self.zf = zipfile.ZipFile(filename, 'r')
            self.book_files = self.zf.namelist()
            self.save_extracted_file(self.zf, self.book_files[0])
            currentFileName = os.path.join(self.get_activity_root(), \
                    'tmp',  self.book_files[0])
        else:
            currentFileName = filename

        self.etext_file = open(currentFileName,"r")
        self.page_index = [ 0 ]
        pagecount = 0
        linecount = 0
        while self.etext_file:
            line = self.etext_file.readline()
            if not line:
                break
            linecount = linecount + 1
            if linecount >= PAGE_SIZE:
                position = self.etext_file.tell()
                self.page_index.append(position)
                linecount = 0
                pagecount = pagecount + 1
        if filename.endswith(".zip"):
            os.remove(currentFileName)
        self.get_saved_page_number()
        self.show_page(page)
        self.read_toolbar.set_total_pages(pagecount + 1)
        self.read_toolbar.set_current_page(page)

        # We've got the document, so if we're a shared activity, offer it
        if self.get_shared():
            self.watch_for_tubes()
            self.share_document()

    def make_new_filename(self, filename):
        partition_tuple = filename.rpartition('/')
        return partition_tuple[2]

    def write_file(self, filename):
        "Save meta data for the file."
        if self.is_received_document:
            # This document was given to us by someone, so we have
            # to save it to the Journal.
            self.etext_file.seek(0)
            filebytes = self.etext_file.read()
            f = open(filename, 'wb')
            try:
                f.write(filebytes)
            finally:
                f.close()
        elif self.tempfile:
            if self.close_requested:
                os.link(self.tempfile,  filename)
                logger.debug("Removing temp file %s because we will close", \
                             self.tempfile)
                os.unlink(self.tempfile)
                self.tempfile = None
        else:
            # skip saving empty file
            raise NotImplementedError

        self.metadata['activity'] = self.get_bundle_id()
        self.save_page_number()

    def can_close(self):
        self.close_requested = True
        return True
```

```
def joined_cb(self, also_self):
    """Callback for when a shared activity is joined.

    Get the shared document from another participant.
    """
    self.watch_for_tubes()
    gobject.idle_add(self.get_document)

def get_document(self):
    if not self.want_document:
        return False

    # Assign a file path to download if one doesn't exist yet
    if not self._jobject.file_path:
        path = os.path.join(self.get_activity_root(), 'instance',
                            'tmp%i' % time.time())
    else:
        path = self._jobject.file_path

    # Pick an arbitrary tube we can try to download the document from
    try:
        tube_id = self.unused_download_tubes.pop()
    except (ValueError, KeyError), e:
        logger.debug('No tubes to get the document from right now: %s',
                     e)
        return False

    # Avoid trying to download the document multiple times at once
    self.want_document = False
    gobject.idle_add(self.download_document, tube_id, path)
    return False

def download_document(self, tube_id, path):
    chan = self._shared_activity.telepathy_tubes_chan
    iface = chan[telepathy.CHANNEL_TYPE_TUBES]
    addr = iface.AcceptStreamTube(tube_id,
            telepathy.SOCKET_ADDRESS_TYPE_IPV4,
            telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0,
            utf8_strings=True)
    logger.debug('Accepted stream tube: listening address is %r', \
                 addr)
    assert isinstance(addr, dbus.Struct)
    assert len(addr) == 2
    assert isinstance(addr[0], str)
    assert isinstance(addr[1], (int, long))
    assert addr[1] > 0 and addr[1]  0:
        logger.debug("Downloaded %u of %u bytes from tube %u...",
                     bytes_downloaded, self.download_content_length,
                     tube_id)
    else:
        logger.debug("Downloaded %u bytes from tube %u...",
                     bytes_downloaded, tube_id)
    total = self.download_content_length
    self.set_downloaded_bytes(bytes_downloaded,  total)
    gtk.gdk.threads_enter()
    while gtk.events_pending():
        gtk.main_iteration()
    gtk.gdk.threads_leave()

def set_downloaded_bytes(self, bytes,  total):
    fraction = float(bytes) / float(total)
    self.progressbar.set_fraction(fraction)
    logger.debug("Downloaded percent",  fraction)

def clear_downloaded_bytes(self):
    self.progressbar.set_fraction(0.0)
    logger.debug("Cleared download bytes")

def download_error_cb(self, getter, err, tube_id):
    self.progressbar.hide()
    logger.debug("Error getting document from tube %u: %s",
                 tube_id, err)
    self.alert(_('Failure'), _('Error getting document from tube'))
    self.want_document = True
```

```
        self.download_content_length = 0
        self.download_content_type = None
        gobject.idle_add(self.get_document)

    def download_result_cb(self, getter, tempfile, suggested_name, tube_id):
        if self.download_content_type.startswith('text/html'):
            # got an error page instead
            self.download_error_cb(getter, 'HTTP Error', tube_id)
            return

        del self.unused_download_tubes

        self.tempfile = tempfile
        file_path = os.path.join(self.get_activity_root(), 'instance',
                                 '%i' % time.time())
        logger.debug("Saving file %s to datastore...", file_path)
        os.link(tempfile, file_path)
        self._jobject.file_path = file_path
        datastore.write(self._jobject, transfer_ownership=True)

        logger.debug("Got document %s (%s) from tube %u",
                     tempfile, suggested_name, tube_id)
        self.is_received_document = True
        self.read_file(tempfile)
        self.save()
        self.progressbar.hide()

    def shared_cb(self, activityid):
        """Callback when activity shared.

        Set up to share the document.

        """
        # We initiated this activity and have now shared it, so by
        # definition we have the file.
        logger.debug('Activity became shared')
        self.watch_for_tubes()
        self.share_document()

    def share_document(self):
        """Share the document."""
        h = hash(self._activity_id)
        port = 1024 + (h % 64511)
        logger.debug('Starting HTTP server on port %d', port)
        self.fileserver = ReadHTTPServer(("", port),
            self.tempfile)

        # Make a tube for it
        chan = self._shared_activity.telepathy_tubes_chan
        iface = chan[telepathy.CHANNEL_TYPE_TUBES]
        self.fileserver_tube_id = iface.OfferStreamTube(READ_STREAM_SERVICE,
                {},
                telepathy.SOCKET_ADDRESS_TYPE_IPV4,
                ('127.0.0.1', dbus.UInt16(port)),
                telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0)

    def watch_for_tubes(self):
        """Watch for new tubes."""
        tubes_chan = self._shared_activity.telepathy_tubes_chan

        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].connect_to_signal('NewTube',
            self.new_tube_cb)
        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].ListTubes(
            reply_handler=self.list_tubes_reply_cb,
            error_handler=self.list_tubes_error_cb)

    def new_tube_cb(self, tube_id, initiator, tube_type, service, params,
                    state):
        """Callback when a new tube becomes available."""
        logger.debug('New tube: ID=%d initator=%d type=%d service=%s '
                     'params=%r state=%d', tube_id, initiator, tube_type,
                     service, params, state)
        if service == READ_STREAM_SERVICE:
            logger.debug('I could download from that tube')
            self.unused_download_tubes.add(tube_id)
```

```
            # if no download is in progress, let's fetch the document
            if self.want_document:
                gobject.idle_add(self.get_document)

    def list_tubes_reply_cb(self, tubes):
        """Callback when new tubes are available."""
        for tube_info in tubes:
            self.new_tube_cb(*tube_info)

    def list_tubes_error_cb(self, e):
        """Handle ListTubes error by logging."""
        logger.error('ListTubes() failed: %s', e)

    def alert(self, title, text=None):
        alert = NotifyAlert(timeout=20)
        alert.props.title = title
        alert.props.msg = text
        self.add_alert(alert)
        alert.connect('response', self.alert_cancel_cb)
        alert.show()

    def alert_cancel_cb(self, alert, response_id):
        self.remove_alert(alert)
        self.textview.grab_focus()
```

The contents of **activity.info** are these lines:

```
[Activity]
name = Read Etexts III
service_name = net.flossmanuals.ReadEtextsActivity
icon = read-etexts
exec = sugar-activity ReadEtextsActivity3.ReadEtextsActivity
show_launcher = no
activity_version = 1
mime_types = text/plain;application/zip
license = GPLv2+
```

Para probarlo, transfiera un libro de Gutenberg del proyecto al diario, ábralo con esto Etexts lo más tarde posible leído lll, después compártalo con un segundo usuario que haga el programa instalar pero el funcionamiento. Ella debe aceptar la invitación a ensamblar que aparece en su opinión de la vecindad. Cuando ella lee Etexts ll comenzará para arriba y copiará el libro del primer usuario sobre la red y lo cargará. La actividad primero demostrará una pantalla en blanco, pero por otra parte una barra del progreso aparecerá apenas debajo de la barra de herramientas e indicará el progreso del copiado. Cuando se acaba la primera página del libro aparecerá.

¿Tan cómo trabaja? Let' mirada de s en el código. Los primeros puntos del interés son las definiciones de clase que aparecen al principio: ReadHTTPRequestHandler, ReadHTTPServer, y ReadURLDownloader. Estas tres clases extienden (es decir, herede el código de) las clases proporcionadas por el paquete de trabajo de sugar.net. Estas clases proporcionan a un cliente del HTTP para recibir el libro y un servidor de HTTP para enviar el libro.

Éste es el código usado para enviar un libro:

```
    def shared_cb(self, activityid):
        """Callback when activity shared.

        Set up to share the document.

        """
        # We initiated this activity and have now shared it, so by
        # definition we have the file.
        logger.debug('Activity became shared')
        self.watch_for_tubes()
        self.share_document()

    def share_document(self):
        """Share the document."""
        h = hash(self._activity_id)
        port = 1024 + (h % 64511)
```

```
        logger.debug('Starting HTTP server on port %d', port)
        self.fileserver = ReadHTTPServer(("", port),
            self.tempfile)

        # Make a tube for it
        chan = self._shared_activity.telepathy_tubes_chan
        iface = chan[telepathy.CHANNEL_TYPE_TUBES]
        self.fileserver_tube_id = iface.OfferStreamTube(READ_STREAM_SERVICE,
                {},
                telepathy.SOCKET_ADDRESS_TYPE_IPV4,
                ('127.0.0.1', dbus.UInt16(port)),
                telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0)
```

Usted notará que un picadillo del _activity_id está utilizado para conseguir un número de acceso. Que el puerto está utilizado para el servidor de HTTP y pasado a la telepatía, que lo ofrece como tubo de corriente. En el lado de recepción tenemos este código:

```
    def joined_cb(self, also_self):
        """Callback for when a shared activity is joined.

        Get the shared document from another participant.
        """
        self.watch_for_tubes()
        gobject.idle_add(self.get_document)

    def get_document(self):
        if not self.want_document:
            return False

        # Assign a file path to download if one doesn't exist yet
        if not self._jobject.file_path:
            path = os.path.join(self.get_activity_root(), 'instance',
                                'tmp%i' % time.time())
        else:
            path = self._jobject.file_path

        # Pick an arbitrary tube we can try to download the document from
        try:
            tube_id = self.unused_download_tubes.pop()
        except (ValueError, KeyError), e:
            logger.debug('No tubes to get the document from right now: %s',
                         e)
            return False

        # Avoid trying to download the document multiple times at once
        self.want_document = False
        gobject.idle_add(self.download_document, tube_id, path)
        return False

    def download_document(self, tube_id, path):
        chan = self._shared_activity.telepathy_tubes_chan
        iface = chan[telepathy.CHANNEL_TYPE_TUBES]
        addr = iface.AcceptStreamTube(tube_id,
                telepathy.SOCKET_ADDRESS_TYPE_IPV4,
                telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0,
                utf8_strings=True)
        logger.debug('Accepted stream tube: listening address is %r', \
                     addr)
        assert isinstance(addr, dbus.Struct)
        assert len(addr) == 2
        assert isinstance(addr[0], str)
        assert isinstance(addr[1], (int, long))
        assert addr[1] > 0 and addr[1]  0:
            logger.debug("Downloaded %u of %u bytes from tube %u...",
                         bytes_downloaded, self.download_content_length,
                         tube_id)
        else:
            logger.debug("Downloaded %u bytes from tube %u...",
                         bytes_downloaded, tube_id)
        total = self.download_content_length
        self.set_downloaded_bytes(bytes_downloaded,  total)
        gtk.gdk.threads_enter()
        while gtk.events_pending():
            gtk.main_iteration()
```

```
        gtk.gdk.threads_leave()

    def download_error_cb(self, getter, err, tube_id):
        self.progressbar.hide()
        logger.debug("Error getting document from tube %u: %s",
                     tube_id, err)
        self.alert(_('Failure'), _('Error getting document from tube'))
        self.want_document = True
        self.download_content_length = 0
        self.download_content_type = None
        gobject.idle_add(self.get_document)

    def download_result_cb(self, getter, tempfile, suggested_name, tube_id):
        if self.download_content_type.startswith('text/html'):
            # got an error page instead
            self.download_error_cb(getter, 'HTTP Error', tube_id)
            return

        del self.unused_download_tubes

        self.tempfile = tempfile
        file_path = os.path.join(self.get_activity_root(), 'instance',
                                 '%i' % time.time())
        logger.debug("Saving file %s to datastore...", file_path)
        os.link(tempfile, file_path)
        self._jobject.file_path = file_path
        datastore.write(self._jobject, transfer_ownership=True)

        logger.debug("Got document %s (%s) from tube %u",
                     tempfile, suggested_name, tube_id)
        self.is_received_document = True
        self.read_file(tempfile)
        self.save()
        self.progressbar.hide()
```

La telepatía nos da la dirección y el número de acceso asociado un tubo de corriente y a nosotros fijó al cliente del HTTP para leer en él. El cliente lee el archivo en download_progress_cb de los pedazos y de las llamadas () después de que cada pedazo así que nosotros poder poner al día una barra del progreso para demostrar al usuario cómo está progresando la transferencia directa. Hay también métodos de servicio repetido para cuando hay un error de la transferencia directa y para cuando se acaba la transferencia directa.

La clase de **ReadURLDownloader** es no sólo útil para transferir archivos sobre los tubos de corriente, puede también ser utilizada para obrar recíprocamente con Web site y servicios de tela. Mi actividad consigue a aplicaciones de los libros del archivo del Internet esta clase para ese propósito.

El un pedazo restante es el código de el cual maneja conseguir los tubos de corriente para transferir el libro. En este código, adaptado de la actividad leída, tan pronto como un caso de una actividad reciba un libro lo vuelve y ofrece compartir, así la actividad puede tener varios tubos posibles que podría conseguir el libro de:

```
READ_STREAM_SERVICE = 'read-etexts-activity-http'

    ...

    def watch_for_tubes(self):
        """Watch for new tubes."""
        tubes_chan = self._shared_activity.telepathy_tubes_chan

        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].connect_to_signal('NewTube',
            self.new_tube_cb)
        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].ListTubes(
            reply_handler=self.list_tubes_reply_cb,
            error_handler=self.list_tubes_error_cb)

    def new_tube_cb(self, tube_id, initiator, tube_type, service, params,
                    state):
        """Callback when a new tube becomes available."""
        logger.debug('New tube: ID=%d initator=%d type=%d service=%s '
                     'params=%r state=%d', tube_id, initiator, tube_type,
                     service, params, state)
```

```
        if service == READ_STREAM_SERVICE:
            logger.debug('I could download from that tube')
            self.unused_download_tubes.add(tube_id)
            # if no download is in progress, let's fetch the document
            if self.want_document:
                gobject.idle_add(self.get_document)

    def list_tubes_reply_cb(self, tubes):
        """Callback when new tubes are available."""
        for tube_info in tubes:
            self.new_tube_cb(*tube_info)

    def list_tubes_error_cb(self, e):
        """Handle ListTubes error by logging."""
        logger.error('ListTubes() failed: %s', e)
```

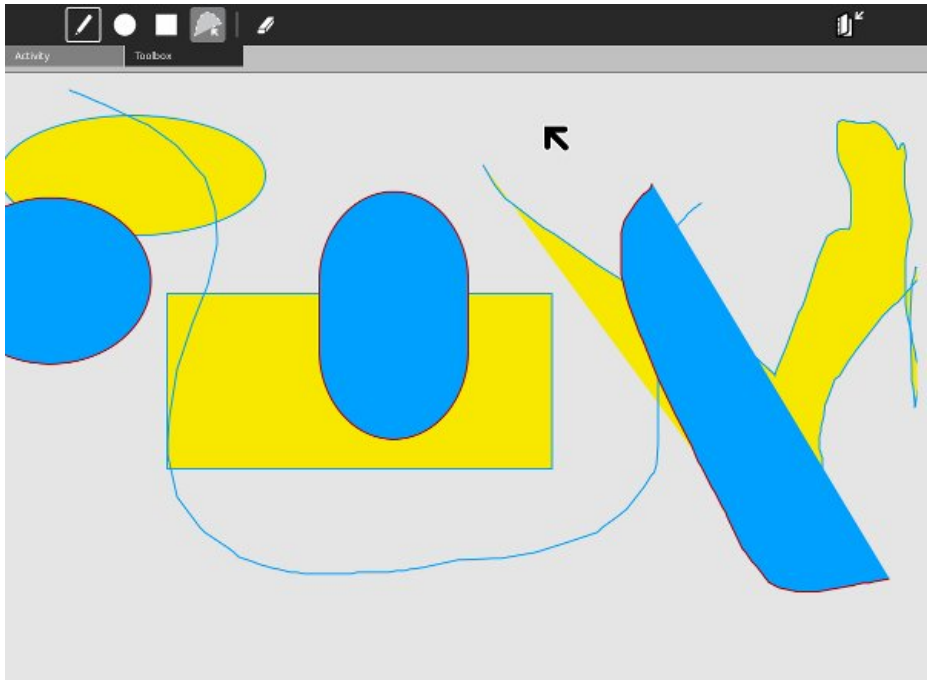
El constante de **READ_STREAM_SERVICE** se define cerca de la tapa del archivo de fuente.

USANDO LOS TUBOS DEL D-BUS

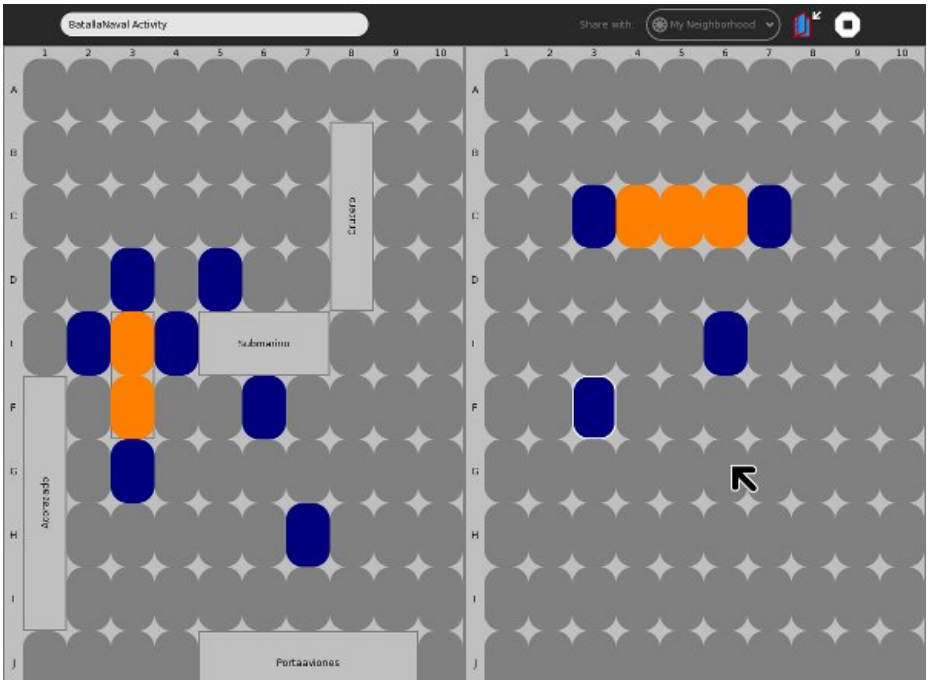
El D-Bus es un método del apoyo IPC, o la comunicación entre procesos, que fue creada para el ambiente de la mesa del GNOME. La idea del IPC es permitir que dos programas corrientes comuniquen con uno a y se ejecuten ' código de s. El GNOME utiliza el D-Bus para proporcionar la comunicación entre el ambiente de escritorio y los programas que funcionan en él, y también entre el GNOMO y el sistema operativo. Un tubo del D-Bus es cómo la telepatía hace posible para un caso de una actividad que funciona en una computadora para ejecutar métodos en otro caso de la misma actividad que funciona en una diversa computadora. En vez apenas de enviar mensajes de texto simples hacia adelante y hacia atrás o de hacer transferencias de archivo, sus Actividades pueden ser compartidas verdad. Es decir, su actividad puede permitir que mucha gente trabaje en la misma tarea junta.

Nunca he escrito una actividad que utiliza los tubos mismo del D-Bus, pero muchas otras tienen. We' el re ir a hechar una ojeada el código a partir del dos de ella: Garabatee por Sayamindu Dasgupta y Batalla naval, por Gerard J. Cerchio y Andrés Ambrois, que fue escrito para el atasco de Ceibal.

El garabato es un programa de dibujo que permite que mucha gente trabaje en el mismo dibujo al mismo tiempo. En vez de permitir que usted elija con qué colores usted dibujará, utiliza los colores de fondo y del primero plano de su icono del compinche (la figura del palillo de XO) para dibujar con. Esa manera, con el dibujo de mucha gente forma it' s fácil saber quién dibujó lo que. Si usted ensambla el garabato en curso de la actividad pondrá al día su pantalla que su dibujo empareja tan cada uno else' pantalla de s. El garabato en la acción parece esto:



Batalla naval es una versión del acorazado clásico del juego. Cada jugador tiene dos rejillas: uno para colocar sus los propios envía (realmente los lugares de la computadora las naves para usted) y otra rejilla en blanco que representa el área donde su opponent' las naves de s son. Usted can' t considera sus naves y lo can' t considera el suyo. Usted chasca encendido el opponent' rejilla de s (a la derecha) para indicar donde usted quiere apuntar un proyectil de artillería. Cuando usted hace el cuadrado correspondiente se encenderá para arriba en su rejilla y su opponent' s los propios rejilla de la nave. Si el cuadrado que usted escogió corresponde a un cuadrado donde su opositor ha colocado una nave que el cuadrado aparecerá en un diverso color. El objeto es encontrar los cuadrados el contener de su opponent' naves de s antes de que él encuentre el suyo. El juego en la acción parece esto:



Sugiero que usted transfiera el último código para estas dos Actividades de Gitorious usando estos comandos:

```
mkdir scribble
cd scribble
git clone git://git.sugarlabs.org/scribble/mainline.git
cd ..
mkdir batallanaval
cd batallanaval
git clone git://git.sugarlabs.org/batalla-naval/mainline.git
```

You' necesidad del ll de hacer un cierto trabajo de la disposición para conseguir éstos que funcionan en azúcar-emulador. El garabato requiere el componente de los goocanvas GTK y los atascamientos del Python que vayan con él. Éstos no fueron instalados por abandono en Fedora 10 pero podía instalarlos usando agrego/quito programas del menú de sistema en GNOMO. Batalla naval está faltando setup.py, pero that' s fijado fácilmente puesto que cada setup.py es idéntico. Copie el que está de los ejemplos del libro en el directorio del mainline/BatallaNaval.activity y funcione con el revelador de ./setup.py en ambas Actividades.

Estas estrategias del uso de las Actividades diversas para la colaboración. El garabato crea las líneas de código del Python que pasa a todos los compinches y los compinches utilizan el exec para funcionar con los comandos. Éste es el código usado para dibujar un círculo:

```
    def process_item_finalize(self, x, y):
        if self.tool == 'circle':
            self.cmd = "goocanvas.Ellipse(parent=self._root, center_x=%d, \
                center_y=%d, radius_x = %d, radius_y = %d, \
                fill_color_rgba = %d, stroke_color_rgba = %d, \
                title = '%s')" % (self.item.props.center_x, \
                self.item.props.center_y, self.item.props.radius_x, \
                self.item.props.radius_y, self._fill_color, \
                self._stroke_color, self.item_id)
...

    def process_cmd(self, cmd):
        #print 'Processing cmd :' + cmd
        exec(cmd) #FIXME: Ugly hack, but I'm too lazy to do this nicely

        if len(self.cmd_list) > 0:
```

```
            self.cmd_list += (';' + cmd)
        else:
            self.cmd_list = cmd
```

La variable del cmd_list se utiliza para crear una secuencia larga que contiene todos los comandos ejecutados hasta ahora. Cuando un nuevo compinche ensambla la actividad le envían esta variable a ejecutar de modo que su área de dibujo tenga el mismo contenido que los otros compinches tienen.

Esto es un acercamiento interesante pero usted podría hacer la misma cosa con el TextChannel tan él isn' t el mejor uso de los tubos del D-Bus. Batalla Naval' el uso de s del D-Bus es más típico.

CÓMO LOS TUBOS DEL D-BUS FUNCIONAN, MÁS O MENOS

El D-Bus le permite hacer que dos programas corrientes envíen mensajes el uno al otro. Los programas tienen que funcionar en la misma computadora. El envío de un mensaje es clase de una manera de cruce giratorio de tener un código del funcionamiento del programa en otro. Un programa define la clase de mensajes que está dispuesto a recibir y a actuar encendido. En el caso de Batalla naval define un " del mensaje; dígame en qué cuadrado usted quiere encender una cáscara y un l' el ll imagina si la parte de una de mis naves está en ese cuadrado y dice you." El primer programa doesn' t funciona con realmente código en segundo, pero el resultado final es similar. Los tubos del D-Bus son una manera de hacer el D-Bus capaz de enviar mensajes como esto a un programa que funciona en otra computadora.

Piense por un minuto en cómo usted puede ser que haga un programa en un código del funcionamiento de la computadora en un programa corriente sobre una diversa computadora. You' d tiene que utilizar la red, por supuesto. Cada uno es familiar con el envío de datos sobre una red, pero en este caso usted tendría que enviar código del programa sobre la red. Usted necesitaría poder decir el programa corriente sobre la segunda computadora qué código usted quisiera que funcionara con. Usted tendría que enviar le una llamada del método y todos los parámetros que usted necesitó pasar en el método, y you' necesidad de d una manera de conseguir un valor de vuelta detrás.

Isn' ¿t que un poco como qué garabato nos está haciendo en el código apenas miraba? ¿Podríamos hacer quizá nuestro código hacemos algo similar?

Por supuesto si usted hiciera que entonces cada programa que usted quiso funcionar con el código adentro tendría que remotamente ser escrito para tratar de eso. Si usted tenía un manojo de programas usted quiso hacer eso con you' d tiene que tener cierta manera de dejar los programas saber qué peticiones fueron significadas para ella. Sería agradable si había un programa que funcionaba en cada máquina que se ocupó de hacer las conexiones de red, convirtiendo llamadas del método a los datos que se podrían enviar sobre la red y después convertir los datos nuevamente dentro de llamadas y del funcionamiento del método ellos, más la devolución de cualquier valor de vuelta. Este programa debe poder saber qué programa usted quiso para rodar código y para ver que la llamada del método está funcionada con allí. El programa debe funcionar todo el tiempo, y sería realmente bueno si hizo funcionando con un método en un programa alejado tan simple como el funcionamiento un método en mi propio programa.

Como usted puede ser que conjeture, qué l' VE apenas descrita es más o menos son qué tubos del D-Bus. Hay artículos que explican cómo trabaja detalladamente pero no es necesario saber trabaja para utilizarlo. Usted necesita saber sobre algunas cosas, aunque. Primero, usted necesita saber utilizar los tubos del D-Bus para hacer objetos en su actividad disponible para uso de otros casos de esa actividad que funciona a otra parte.

Una actividad que necesita utilizar los tubos del D-Bus necesita definir qué clases de mensajes está dispuesto a actuar encendido, en efecto qué métodos específicos adentro en el programa están disponibles para este uso. Todas las Actividades que utilizan los tubos del D-Bus tienen constantes como esto:

```
SERVICE = "org.randomink.sayamindu.Scribble"
```

```
IFACE = SERVICE
PATH = "/org/randomink/sayamindu/Scribble"
```

Éstos son los constantes usados para la actividad del garabato. El primer SERVICIO constante, nombrado, representa el nombre del autobús de la actividad. Esto también se llama un nombre bien conocido porque utiliza un Domain Name invertido como parte del nombre. En este caso Sayamindu Dasgupta tiene un Web site en http://sayamindu.randomink.org que él invierte tan las palabras punto-separadas de ese URL para crear la primera parte de su nombre del autobús. No es necesario a propio un Domain Name antes de que usted pueda crear un nombre del autobús. Usted puede utilizar org.sugarlabs.ActivityName si usted tiene gusto. El punto es que el nombre del autobús debe ser único, y por la convención esto es hecha más fácil comenzando con un Domain Name invertido.

El constante de la PATH representa la trayectoria del objeto. Parece el nombre del autobús con las rayas verticales que separan las palabras algo que períodos. Para la mayoría de las Actividades que es exactamente lo que debe ser, solamente él es posible para que un uso exponga más de un objeto al D-Bus y en ese caso cada objeto expuesto tendría su propio nombre único, por las palabras de la convención separadas por rayas verticales.

El tercer constante es IFACE, que es el nombre del interfaz. Un interfaz es una colección de métodos y de señales relacionados, identificada por un nombre que utilice a la misma convención usada por el nombre del autobús. En el ejemplo arriba, y probablemente en la mayoría de las Actividades usando un tubo del D-Bus, el nombre del interfaz y el nombre del autobús son idénticos.

¿Cuál es tan una señal? Una señal es como un método pero en vez de un programa corriente que llama un método en un otro programa corriente, una señal es difusión. Es decir en vez de ejecutar un método en apenas un programa ejecuta el mismo método en muchos programas corrientes, de hecho en cada programa corriente que tenga ese método que está conectado con a través del D-Bus. Una señal puede pasar datos en una llamada pero ella del método can' t recibe cualquier cosa trasero como un valor de vuelta. It' s tiene gusto de una estación de radio que difunda música a cualquier persona que se temple adentro. El flujo de información es unidireccional solamente.

Por supuesto una estación de radio recibe a menudo llamadas de teléfono de sus oyentes. Un jinete de disco pudo jugar una nueva canción e invitar a oyentes que llamen la estación y digan lo que pensaron en él. La llamada de teléfono es comunicación de dos vías entre el jinete de disco y el oyente, pero fue iniciada por una petición que era difusión a todos los oyentes. Su actividad pudo utilizar de la misma manera una señal de invitar a todos los oyentes (compinches) que utilicen un método para llamarlo detrás, y ese método puede suministrar y recibir la información.

En D-Bus los métodos y las señales tienen firmas. Una firma es una descripción de los parámetros pasajeros en un método o una señal incluyendo sus tipos de datos. El Python no es una lengua fuertemente mecanografiada. En una lengua fuertemente mecanografiada cada variable tiene un tipo de datos que limite lo que puede hacer. Los tipos de datos incluyen las cosas tales como las secuencias, los números enteros, los números enteros largos, los números de la coma flotante, los booleans, el etc. cada una se pueden utilizar para un propósito específico solamente. Por ejemplo un boleano puede solamente llevar a cabo los valores verdades y falsos, nada otro. Una secuencia se puede utilizar para sostener cadenas de caracteres, pero incluso si esos caracteres representan un número usted no puede utilizar una secuencia para los cálculos. En lugar usted necesita convertir la secuencia en uno de los tipos de datos numéricos. Un número entero puede llevar a cabo números enteros hasta cierto tamaño, y un número entero largo puede llevar a cabo números enteros mucho más grandes, número de la coma flotante de A es un número con una coma en la notación científica. Es casi inútil para la aritmética del negocio, que requiere resultados redondeados.

En Python usted puede poner cualquier cosa en variable y la lengua sí mismo imaginará cómo ocuparse de ella. Para hacer que el Python trabaja con el D-Bus, que requiere variables fuertemente mecanografiadas ese Python doesn' t tiene, usted necesita una manera de decir a D-Bus qué tipos deben tener las variables que usted pasa en un método. Usted hace esto usando una secuencia de la firma como discusión al método

o a la señal. Los métodos tienen dos secuencias: un in_signature y un out_signature. Las señales apenas tienen un parámetro de la firma. Algunos ejemplos de las secuencias de la firma:

ii	Dos parámetros, ambos números enteros
sss	Tres parámetros, todas las secuencias
ixd	Tres parámetros, un número entero, un número entero largo, y un número de precisión doble de la coma flotante.
a(ssiii)	Un arsenal donde está un tuple cada elemento del arsenal que contiene dos secuencias y tres números enteros.

Hay más información sobre secuencias de la firma en la clase particular del dbus-python en http://dbus.freedesktop.org/doc/dbus-python/doc/tutorial.html.

PRESENTAR AL ACOPLAMIENTO Y A AMIGOS DEL HOLA

ISi usted estudia el código fuente de algunas Actividades compartidas you' el ll concluye que muchas de ellas contienen métodos casi idénticos, como si todos fueran copiadas de la misma fuente. De hecho, que no ellos estaban más probablemente. El acoplamiento de la actividad hola fue creado para ser un ejemplo de cómo utilizar los tubos del D-Bus en una actividad compartida. Es tradicional en libros de textos programados hacer que el primer programa del ejemplo sea algo que apenas imprime el " de las palabras; Hola World" a la consola o a las exhibiciones las mismas palabras en una ventana. En esa tradición hola el acoplamiento es un programa que doesn' t hace todos que mucho. Usted puede encontrar el código en Gitorious en http://git.sugarlabs.org/projects/hello-mesh.

Hello Mesh se copia extensamente porque demuestra cómo hacer las cosas que todas las Actividades compartidas necesitan hacer. Cuando usted tiene una actividad compartida usted necesita poder hacer dos cosas:

- Envíe la información o los comandos a otros casos de su actividad.
- Dé a los compinches que ensamblan su actividad una copia del estado actual de la actividad.

Hace esto usando dos señales y un método:

- Una señal llamó hola () que alguien que ensambla la actividad envía a todos los participantes. Hola () el método no toma ninguìn parámetro.
- Un método llamó a World () que los casos de la actividad que recibe hola () envían detrás al remitente. Este método toma una secuencia de texto como discusión, que se significa para representar el estado actual de la actividad.
- Otra señal llamó SendText () que envía una secuencia de texto a todos los participantes. Esto representa la puesta al día del estado de la actividad compartida. En el caso de garabato esto estaría informando al otros que este caso acaba de dibujar una nueva forma.

Algo que el acoplamiento sí mismo l' del estudio hola; d tiene gusto de mirar el código derivado de él utilizó en Batalla naval. He tomado la libertad de funcionar los comentarios, originalmente en español, a través de Google traduzco para hacer todo en inglés. También he quitado algunas líneas de código comentadas-hacia fuera.

Esta actividad hace algo listo para permitir funcionarlo como actividad del azúcar o como programa independiente del Python. El programa independiente no apoya la distribución en absoluto, y funciona en una ventana. La actividad de la clase es una subclase de la ventana, así que cuando el código es independiente funcionado el init () la función en BatallaNaval.py consigue una ventana, y cuando se funciona con el mismo código mientras que una actividad el caso de la clase BatallaNavalActivity se pasa al init ():

```
from sugar.activity.activity import Activity, ActivityToolbox
import BatallaNaval
from Collaboration import CollaborationWrapper

class BatallaNavalActivity(Activity):
    ''' The Sugar class called when you run this program as an Activity.
        The name of this class file is marked in the activity/activity.info file.'''

    def __init__(self, handle):
        Activity.__init__(self, handle)

        self.gamename = 'BatallaNaval'

        # Create the basic Sugar toolbar
        toolbox = ActivityToolbox(self)
        self.set_toolbox(toolbox)
        toolbox.show()

        # Create an instance of the CollaborationWrapper so you can share the activity.
        self.colaboracion = CollaborationWrapper(self)

        # The activity is a subclass of Window, so it passses itself to the init function
        BatallaNaval.init(False, self)
```

La otra cosa lista que se enciende aquí es que todo el código de la colaboración está puesto en su propia clase de CollaborationWrapper que tome el caso de la clase de BatallNavalActivity en su constructor. Esto separa el código de la colaboración del resto del programa. Aquí está el código en CollaborationWrapper.py:

```
import logging

from sugar.presence import presenceservice
import telepathy
from dbus.service import method, signal
# In build 656 Sugar lacks sugartubeconn
try:
  from sugar.presence.sugartubeconn import SugarTubeConnection
except:
  from sugar.presence.tubeconn import TubeConnection as SugarTubeConnection
from dbus.gobject_service import ExportedGObject

''' In all collaborative Activities in Sugar we are made aware when a player
    enters or leaves. So that everyone knows the state of the Activity we
    use the methods Hello and World. When a participant enters Hello
    sends a signal that reaches all participants and the participants
    respond directly using the method "World", which retrieves
    the current state of the Activity.
    After the updates are given then the signal Play is used by each
    participant to make his move.
    In short this module encapsulates the logic of "collaboration" with the
    following effect:
        - When someone enters the collaboration the Hello signal is sent.
        - Whoever receives the Hello signal responds with World
        - Every time someone makes a move he uses the method Play
        giving a signal which communicates to each participant
        what his move was.
'''

SERVICE = "org.ceibaljam.BatallaNaval"
IFACE = SERVICE
PATH = "/org/ceibaljam/BatallaNaval"

logger = logging.getLogger('BatallaNaval')
logger.setLevel(logging.DEBUG)

class CollaborationWrapper(ExportedGObject):
    ''' A wrapper for the collaboration methods.
        Get the activity and the necessary callbacks.
 '''

    def __init__(self, activity):
        self.activity = activity
        self.presence_service = presenceservice.get_instance()
        self.owner = self.presence_service.get_owner()
```

```
def set_up(self, buddy_joined_cb, buddy_left_cb, World_cb, Play_cb, my_boats):
    self.activity.connect('shared', self._shared_cb)
    if self.activity._shared_activity:
        # We are joining the activity
        self.activity.connect('joined', self._joined_cb)
        if self.activity.get_shared():
            # We've already joined
            self._joined_cb()

    self.buddy_joined = buddy_joined_cb
    self.buddy_left = buddy_left_cb
    self.World_cb = World_cb        # Called when someone passes the board state.
    self.Play_cb = Play_cb          # Called when someone makes a move.

    # Submitted by making World on a new partner
    self.my_boats = [(b.nombre, b.orientacion, b.largo, \
        b.pos[0], b.pos[1]) for b in my_boats]
    self.world = False
    self.entered = False

def _shared_cb(self, activity):
    self._sharing_setup()
    self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES].OfferDBusTube(
        SERVICE, {})
    self.is_initiator = True

def _joined_cb(self, activity):
    self._sharing_setup()
    self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES].ListTubes(
        reply_handler=self._list_tubes_reply_cb,
        error_handler=self._list_tubes_error_cb)
    self.is_initiator = False

def _sharing_setup(self):
    if self.activity._shared_activity is None:
        logger.error('Failed to share or join activity')
        return

    self.conn = self.activity._shared_activity.telepathy_conn
    self.tubes_chan = self.activity._shared_activity.telepathy_tubes_chan
    self.text_chan = self.activity._shared_activity.telepathy_text_chan

    self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES].connect_to_signal(
        'NewTube', self._new_tube_cb)

    self.activity._shared_activity.connect('buddy-joined',\
        self._buddy_joined_cb)
    self.activity._shared_activity.connect('buddy-left', self._buddy_left_cb)

    # Optional - included for example:
    # Find out who's already in the shared activity:
    for buddy in self.activity._shared_activity.get_joined_buddies():
        logger.debug('Buddy %s is already in the activity',
                             buddy.props.nick)

def participant_change_cb(self, added, removed):
    logger.debug('Tube: Added participants: %r', added)
    logger.debug('Tube: Removed participants: %r', removed)
    for handle, bus_name in added:
        buddy = self._get_buddy(handle)
        if buddy is not None:
            logger.debug('Tube: Handle %u (Buddy %s) was added',
                                 handle, buddy.props.nick)
    for handle in removed:
        buddy = self._get_buddy(handle)
        if buddy is not None:
            logger.debug('Buddy %s was removed' % buddy.props.nick)
    if not self.entered:
        if self.is_initiator:
            logger.debug("I'm initiating the tube, will watch for hellos.")
            self.add_hello_handler()
        else:
            logger.debug('Hello, everyone! What did I miss?')
            self.Hello()
    self.entered = True
```

```
    # This is sent to all participants whenever we join an activity
    @signal(dbus_interface=IFACE, signature='')
    def Hello(self):
        """Say Hello to whoever else is in the tube."""
        logger.debug('I said Hello.')

    # This is called by whoever receives our Hello signal
    # This method receives the current game state and puts us in sync
    # with the rest of the participants.
    # The current game state is represented by the game object
    @method(dbus_interface=IFACE, in_signature='a(ssiii)', out_signature='a(ssiii)')
    def World(self, boats):
        """To be called on the incoming XO after they Hello."""
        if not self.world:
            logger.debug('Somebody called World on me')
            self.world = True   # Instead of loading the world,
                                # I am receiving play by play.
            self.World_cb(boats)
            # now I can World others
            self.add_hello_handler()
        else:
            self.world = True
            logger.debug("I've already been welcomed, doing nothing")
        return self.my_boats

    @signal(dbus_interface=IFACE, signature='ii')
    def Play(self, x, y):
        """Say Hello to whoever else is in the tube."""
        logger.debug('Running remote play:%s x %s.', x, y)

    def add_hello_handler(self):
        logger.debug('Adding hello handler.')
        self.tube.add_signal_receiver(self.hello_signal_cb, 'Hello', IFACE,
            path=PATH, sender_keyword='sender')
        self.tube.add_signal_receiver(self.play_signal_cb, 'Play', IFACE,
            path=PATH, sender_keyword='sender')

    def hello_signal_cb(self, sender=None):
        """Somebody Helloed me. World them."""
        if sender == self.tube.get_unique_name():
            # sender is my bus name, so ignore my own signal
            return
        logger.debug('Newcomer %s has joined', sender)
        logger.debug('Welcoming newcomer and sending them the game state')

        self.other = sender

        # I send my ships and I get theirs in return
        enemy_boats = self.tube.get_object(self.other, PATH).World(\
            self.my_boats, dbus_interface=IFACE)

        # I call the callback World, to load the enemy ships
        self.World_cb(enemy_boats)

    def play_signal_cb(self, x, y, sender=None):
        """Somebody placed a stone. """
        if sender == self.tube.get_unique_name():
            return  # sender is my bus name, so ignore my own signal
        logger.debug('Buddy %s placed a stone at %s x %s', sender, x, y)
        # Call our Play callback
        self.Play_cb(x, y)  # In theory, no matter who sent him

    def _list_tubes_error_cb(self, e):
        logger.error('ListTubes() failed: %s', e)

    def _list_tubes_reply_cb(self, tubes):
        for tube_info in tubes:
            self._new_tube_cb(*tube_info)

    def _new_tube_cb(self, id, initiator, type, service, params, state):
        logger.debug('New tube: ID=%d initator=%d type=%d service=%s '
                     'params=%r state=%d', id, initiator, type, service,
                     params, state)
```

```
        if (type == telepathy.TUBE_TYPE_DBUS and
            service == SERVICE):
            if state == telepathy.TUBE_STATE_LOCAL_PENDING:
                self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES].AcceptDBusTube(id)
            self.tube = SugarTubeConnection(self.conn,
                self.tubes_chan[telepathy.CHANNEL_TYPE_TUBES],
                id, group_iface=self.text_chan[telepathy.CHANNEL_INTERFACE_GROUP])
            super(CollaborationWrapper, self).__init__(self.tube, PATH)
            self.tube.watch_participants(self.participant_change_cb)

    def _buddy_joined_cb (self, activity, buddy):
        """Called when a buddy joins the shared activity. """
        logger.debug('Buddy %s joined', buddy.props.nick)
        if self.buddy_joined:
            self.buddy_joined(buddy)

    def _buddy_left_cb (self, activity, buddy):
        """Called when a buddy leaves the shared activity. """
        if self.buddy_left:
            self.buddy_left(buddy)

    def _get_buddy(self, cs_handle):
        """Get a Buddy from a channel specific handle."""
        logger.debug('Trying to find owner of handle %u...', cs_handle)
        group = self.text_chan[telepathy.CHANNEL_INTERFACE_GROUP]
        my_csh = group.GetSelfHandle()
        logger.debug('My handle in that group is %u', my_csh)
        if my_csh == cs_handle:
            handle = self.conn.GetSelfHandle()
            logger.debug('CS handle %u belongs to me, %u', cs_handle, handle)
        elif group.GetGroupFlags() & \
            telepathy.CHANNEL_GROUP_FLAG_CHANNEL_SPECIFIC_HANDLES:
            handle = group.GetHandleOwners([cs_handle])[0]
            logger.debug('CS handle %u belongs to %u', cs_handle, handle)
        else:
            handle = cs_handle
            logger.debug('non-CS handle %u belongs to itself', handle)
            # XXX: deal with failure to get the handle owner
            assert handle != 0
        return self.presence_service.get_buddy_by_telepathy_handle(
            self.conn.service_name, self.conn.object_path, handle)
```

La mayor parte de el código antedicho es similar a qué we' VE vista en los otros ejemplos, y la mayor parte de él pueden ser utilizados como está en cualquier actividad que necesite hacer llamadas del D-Bus. Por esta razón we' foco del ll en el código que es específico a usar el D-Bus. El lugar lógico a comenzar es hola () el método. No hay por supuesto nada mágico sobre el " conocido; Hello". Hola el acoplamiento se significa para ser un " Hola World" programa para usar los tubos del D-Bus, tan por la convención el " de las palabras; Hello" y " World" tuvo que ser utilizado para algo. Hola () el método es difusión a todos los casos de la actividad para informarles que un nuevo caso está listo para recibir la información sobre el estado actual de la actividad compartida. Su propia actividad necesitará probablemente algo similar, pero usted debe sentir libre de nombrarlo algo más, y si you' re escritura el código para una asignación de escuela usted debe definitivamente nombrarla algo más:

```
    # This is sent to all participants whenever we join an activity
    @signal(dbus_interface=IFACE, signature='')
    def Hello(self):
        """Say Hello to whoever else is in the tube."""
        logger.debug('I said Hello.')

    def add_hello_handler(self):
        logger.debug('Adding hello handler.')
        self.tube.add_signal_receiver(self.hello_signal_cb, 'Hello', IFACE,
            path=PATH, sender_keyword='sender')
...

    def hello_signal_cb(self, sender=None):
        """Somebody Helloed me. World them."""
        if sender == self.tube.get_unique_name():
            # sender is my bus name, so ignore my own signal
            return
        logger.debug('Newcomer %s has joined', sender)
```

```
        logger.debug('Welcoming newcomer and sending them the game state')

        self.other = sender

        # I send my ships and I returned theirs
        enemy_boats = self.tube.get_object(self.other, PATH).World(self.my_boats,
dbus_interface=IFACE)

        # I call the callback World, to load the enemy ships
        self.World_cb(enemy_boats)
```

La cosa más interesante sobre este código es esta línea, que el Python llama a **Decorator**:

```
    @signal(dbus_interface=IFACE, signature='')
```

Cuando usted pone @signal delante de un nombre del método que tiene el efecto de agregar los dos parámetros demostrados a la llamada del método siempre que se invoque, en efecto cambiándola de una llamada normal del método a una llamada del D-Bus para una señal. El parámetro de la firma es una secuencia vacía, indicando que la llamada del método no tiene ninguìn parámetro. Hola () el método no hace nada localmente sino cuando es recibido por los otros casos de la actividad que los hace ejecutar el método del mundo (), que devuelve la localización de sus barcos y consigue a nuevos participantes los barcos a cambio.

Batalla Naval se significa al parecer para ser un programa de demostración. El acorazado es un juego para dos jugadores, pero no hay nada en el código prevenir a más jugadores de ensamblar y de ninguna manera para manejarla si lo hacen. Usted quisiera idealmente que el código hiciera solamente el primer carpintero a un jugador real y que hiciera a los espectadores del resto solamente.

We' siguiente; mirada del ll en el método del World():

```
    # This is called by whoever receives our Hello signal
    # This method receives the current game state and puts us in sync
    # with the rest of the participants.
    # The current game state is represented by the game object
    @method(dbus_interface=IFACE, in_signature='a(ssiii)', out_signature='a(ssiii)')
    def World(self, boats):
        """To be called on the incoming XO after they Hello."""
        if not self.world:
            logger.debug('Somebody called World on me')
            self.world = True   # Instead of loading the world, I am receiving play by play.
            self.World_cb(boats)
            # now I can World others
            self.add_hello_handler()
        else:
            self.world = True
            logger.debug("I've already been welcomed, doing nothing")
        return self.my_boats
```

Hay otro decorador aquí, éste que convierte el método del mundo () a una llamada del D-Bus para un método. La firma es más interesante que hola () tenía. Significa un arsenal de los tuples donde cada tuple contiene dos secuencias y tres números enteros. Cada elemento en el arsenal representa una nave y sus cualidades. World_cb se fija para señalar a un método en BatallaNaval.py, (y así que es Play_cb). Si usted estudia el código del init () en BatallaNaval.py you' el ll considera cómo sucede éste. El mundo () se llama en el método del hello_signal_cb () que acabamos de mirar. Se envía al carpintero que nos envió hola ().

Finalmente we' mirada del ll en la señal del Play():

```
    @signal(dbus_interface=IFACE, signature='ii')
    def Play(self, x, y):
        """Say Hello to whoever else is in the tube."""
        logger.debug('Running remote play:%s x %s.', x, y)

    def add_hello_handler(self):
...
        self.tube.add_signal_receiver(self.play_signal_cb, 'Play', IFACE,
            path=PATH, sender_keyword='sender')
```

```
...
    def play_signal_cb(self, x, y, sender=None):
        """Somebody placed a stone. """
        if sender == self.tube.get_unique_name():
            return  # sender is my bus name, so ignore my own signal
        logger.debug('Buddy %s placed a stone at %s x %s', sender, x, y)
        # Call our Play callback
        self.Play_cb(x, y)
```

Esto es una señal tan allí es solamente una secuencia de la firma, ésta que indica que los parámetros de la entrada son dos números enteros.

Hay varias maneras que usted podría mejorar esta actividad. Al jugar contra la computadora en modo de no-distribución el juego apenas hace movimientos al azar. El juego no limita a los jugadores a dos y no hace el resto de los espectadores de los carpinteros. No hace que los jugadores toman vueltas. Cuando un jugador tiene éxito en el hundimiento de el resto de naves de los jugadores nada sucede marcar el acontecimiento. Finalmente, el gettext () no se utiliza para las secuencias de texto exhibidas por la actividad así que él no se puede traducir a idiomas con excepción de español.

En la tradición de libros de textos por todas partes dejaré llevar a cabo estas mejoras como ejercicio para el estudiante.

16. ADICIÓN DEL TEXTO AL DISCURSO

INTRODUCCIÓN

Ciertamente uno de las Actividades más populares disponibles es **Hablar**, que toma las palabras que usted mecanografía adentro y las habla hacia fuera ruidosamente, al mismo tiempo exhibiendo una cara de la historieta que parezca hablar las palabras. Usted puede ser que sea sorprendido aprender cómo poco del código en esa Actividad se utiliza para conseguir las palabras habladas. Si su actividad podría beneficiarse de hacer palabras hablar hacia fuera ruidosamente (las posibilidades de Actividades educativas y de juegos son definitivamente allí) este capítulo le enseñarán a cómo hacer que sucede.

TENEMOS MANERAS DE HACERLE CHARLA

Unas par de maneras, realmente, y ni unas ni otras una son ésa dolorosa. Son:

- Funcionando con el programa del **espeak** directamente
- Usando el **espeak del gstreamer enchufable**

Ambos acercamientos tienen sus ventajas. Primer es el que está usado cerca habla. (Técnico, hable las aplicaciones que el gstreamer enchufable si está disponible, y de otra manera ejecuta el espeak directamente. Para qué hablan está haciendo usando el gstreamer enchufable no es realmente necesario). La ejecución del espeak es definitivamente el método más simple, y puede ser conveniente para su propia actividad. Su ventaja grande es que usted no necesita tener el enchufable del gstreamer instalado. Si su actividad necesita funcionar en algo con excepción de la última versión del azúcar éste será algo considerar.

El gstreamer enchufable es qué es utilizada por **Read Etexts** para hacer el texto al discurso con destacar.

Para este uso necesitamos poder hacer las cosas que no son posibles apenas funcionando el **espeak**. Por ejemplo:

- Necesitamos poder detenerse brevemente y reasumir discurso, porque la actividad necesita hablar un valor entero de la página del texto, no apenas las frases simples.
- Necesitamos destacar las palabras que eran habladas mientras que se hablan.

Usted puede ser que piense que usted podría alcanzar estos objetivos funcionando el espeak en una palabra a la vez. Si usted hace, no se sienta mal porque pensé eso también. En una computadora rápida suena realmente tremendo, como HAL 9000 que desarrolla un tartamudeo hacia el final de la desactivación. En XO ningunos los sonidos salieron en absoluto.

Lea originalmente al speech-dispatcher usado Etexts para hacer lo que lo hace el gstreamer enchufable. Los reveladores de ese programa eran muy provechosos en conseguir destacar en el funcionamiento leído de Etexts, pero el discurso-despachador necesitó ser configurado antes de que usted podría utilizarlo cuál era una edición para nosotros. (Hay más que uno bueno del texto con el software del discurso disponible y el discurso-despachador los apoya la mayor parte de. Esto hace archivos de configuración inevitables). Aleksey Lim de los laboratorios del azúcar subió con la idea de usar un gstreamer enchufable y era la persona que lo escribió. Él también reescribió mucho de **Read Etexts** así que utilizaría al discurso-despachador enchufable si estuviera disponible, del uso si no, y no apoyaría discurso si ni uno ni otro estaba disponible.

ESPEAK CORRIENTE DIRECTAMENTE

Usted puede funcionar con el programa del **espeak del** terminal para probar sus opciones. Para verle qué opciones están disponibles para el espeak puede utilizar el comando del **man**:

```
man espeak
```

Esto le dará una página manual que describe cómo funcionar con el programa y qué opciones están disponibles. Las partes de la página de hombre que nos son las más interesantes son éstas:

```
NAME
       espeak - A multi-lingual software speech synthesizer.

SYNOPSIS
       espeak [options] []

DESCRIPTION
       espeak is a software speech synthesizer for English, and some other languages.

OPTIONS
       -p
              Pitch adjustment, 0 to 99, default is 50

       -s
              Speed in words per minute, default is 160

       -v
              Use voice file of this name from espeak-data/voices

       --voices[=]
               Lists the available voices. If = is present then only those voices which are suitable
for that language are listed.
```

Probemos algunas de estas opciones. Primero déjenos consiguen una lista de **voices**:

```
espeak --voices
Pty Language Age/Gender VoiceName       File        Other Langs
 5  af             M  afrikaans         af
 5  bs             M  bosnian           bs
 5  ca             M  catalan           ca
 5  cs             M  czech             cs
```

```
 5  cy             M  welsh-test        cy
 5  de             M  german            de
 5  el             M  greek             el
 5  en             M  default           default
 5  en-sc          M  en-scottish       en/en-sc    (en 4)
 2  en-uk          M  english           en/en       (en 2)
... and many more ...
```

Ahora que sabemos que cuáles son los nombres de las voces podemos probarlas. ¿Cómo sobre inglés con un acento francés?

```
espeak "Your mother was a hamster and your father smelled of elderberries." -v fr
```

Intentemos experimentar con tarifa y echemos:

```
espeak "I'm sorry, Dave. I'm afraid I can't do that." -s 120 -p 30
```

La cosa siguiente a hacer es escribir un cierto código del Python al espeak funcionado. Aquí está un programa corto adaptado del código adentro **Hablar**:

```
import re
import subprocess

PITCH_MAX = 99
RATE_MAX = 99
PITCH_DEFAULT = PITCH_MAX/2
RATE_DEFAULT = RATE_MAX/3

def speak(text,  rate=RATE_DEFAULT,  pitch=PITCH_DEFAULT,  voice="default"):

    # espeak uses 80 to 370
    rate = 80 + (370-80) * int(rate) / 100

    subprocess.call(["espeak", "-p", str(pitch),
            "-s", str(rate), "-v", voice,  text],
            stdout=subprocess.PIPE)

def voices():
    out = []
    result = subprocess.Popen(["espeak", "--voices"], stdout=subprocess.PIPE) \
            .communicate()[0]

    for line in result.split('\n'):
        m = re.match(r'\s*\d+\s+([\w-]+)\s+([MF])\s+([\w_-]+)\s+(.+)', line)
        if not m:
            continue
        language, gender, name, stuff = m.groups()
        if stuff.startswith('mb/') or \
                name in ('en-rhotic','english_rp','english_wmids'):
            # these voices don't produce sound
            continue
        out.append((language, name))

    return out

def main():
    print voices()
    speak("I'm afraid I can't do that, Dave.")
    speak("Your mother was a hamster, and your father smelled of elderberries!",  30,  60,  "fr")

if __name__ == "__main__":
    main()
```

En el depósito de Git en el directorio **Adding_TTS** este archivo se nombra **espeak.py**. Cargue este archivo en **Eric** y **funcione con la escritura del** menú del comienzo para funcionarlo. Además de discurso de la audiencia usted debe ver este texto:

[('af', 'afrikaans'), ('bs', 'bosnian'), ('ca', 'catalan'), ('cs', 'czech'), ('cy', 'welsh-test'), ('de', 'german'), ('el', 'greek'), ('en', 'default'), ('en-sc', 'en-scottish'), ('en-uk', 'english'), ('en-uk-north', 'lancashire'), ('en-us', 'english-us'), ('en-wi', 'en-

westindies'), ('eo', 'esperanto'), ('es', 'spanish'), ('es-la', 'spanish-latin-american'), ('fi', 'finnish'), ('fr', 'french'), ('fr-be', 'french'), ('grc', 'greek-ancient'), ('hi', 'hindi-test'), ('hr', 'croatian'), ('hu', 'hungarian'), ('hy', 'armenian'), ('hy', 'armenian-west'), ('id', 'indonesian-test'), ('is', 'icelandic-test'), ('it', 'italian'), ('ku', 'kurdish'), ('la', 'latin'), ('lv', 'latvian'), ('mk', 'macedonian-test'), ('nl', 'dutch-test'), ('no', 'norwegian-test'), ('pl', 'polish'), ('pt', 'brazil'), ('pt-pt', 'portugal'), ('ro', 'romanian'), ('ru', 'russian_test'), ('sk', 'slovak'), ('sq', 'albanian'), ('sr', 'serbian'), ('sv', 'swedish'), ('sw', 'swahihi-test'), ('ta', 'tamil'), ('tr', 'turkish'), ('vi', 'vietnam-test'), ('zh', 'Mandarin'), ('zh-yue', 'cantonese-test')]

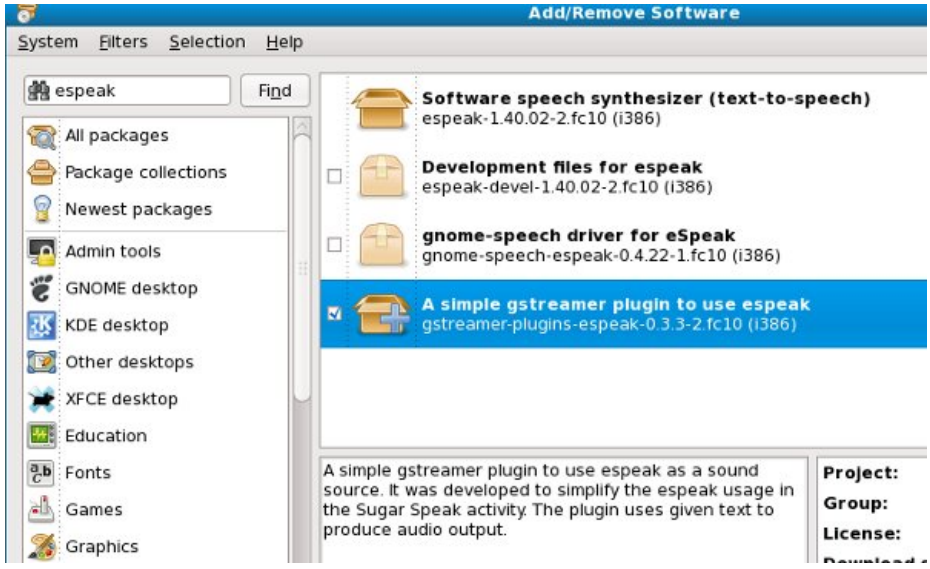
La función de las *voces ()* devuelve una lista de voces como un tuple por voz, y elimina voces de la lista que el espeak no puede presentar en sus la propia. Esta lista de tuples se puede utilizar para poblar un lista de persiana.

La función del *voices()* ajusta el valor de la **tarifa** así que usted puede entrar un valor entre 0 y 99 algo que entre 80 a 370. *hablan ()* es más complejos en la actividad del discurso que qué tenemos aquí porque en esa actividad supervisa el audio hablado y genera los movimientos de la boca basados en la amplitud de la voz. Haciendo la cara muévase es la mayor parte de lo que hace la actividad del discurso, y puesto que no estamos haciendo que necesitamos código muy pequeño hacer que nuestra actividad habla.

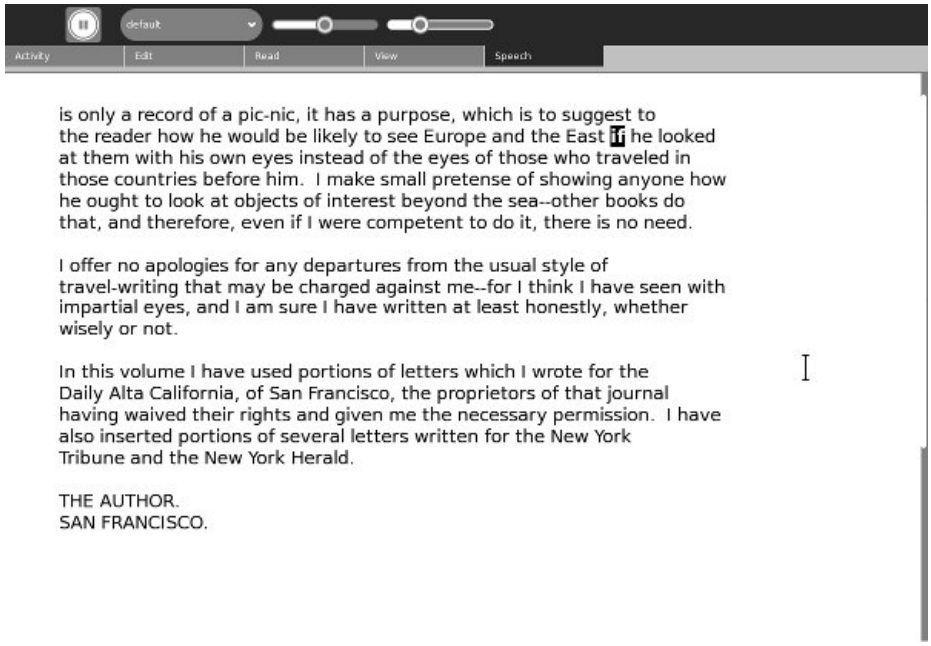
Usted puede utilizar el **import espeak** para incluir este archivo en sus propias Actividades.

USANDO EL ESPEAK DEL GSTREAMER ENCHUFABLE

El espeak del gstreamer enchufable se puede instalar en **Fedora 10** o más adelante usando **Add/Remove Software**.



Cuando usted hace esto hacer usted debe poder transferir la actividad **Read Etexts** (la verdadera, no la versión simplificada que estamos utilizando para el libro) de TAMBIÉN y probar el **Speech** cuadro. Usted debe ahora hacer eso. Mirará algo similar:



El libro usado en los screenshots anteriores de este manual era *orgullo y prejudicar de* Jane Austen. Para balancear cosas hacia fuera el resto de los screenshots utilizará a los *Innocents al exterior* por Mark Twain.

Gstreamer es un marco para las multimedias. Si usted ha mirado los vídeos en la tela usted es familiar con el concepto de fluir medios. En vez de transferir una canción entera o un clip entero de la película y después de jugarlo, el fluir significa que la transferencia y el jugar suceden al mismo tiempo, con la transferencia apenas un pedacito detrás de fluir. Hay muchas diversas clases de archivos de medios: MP3, DivX, WMV, medios verdaderos, y así sucesivamente. Para cada clase de archivo de medios Gstreamer tiene un enchufable.

Gstreamer hace uso de un concepto llamado **pipelining**. La idea es que el resultado de un programa puede convertirse en la entrada a otro. Una forma a dirigir que la situación es poner el resultado del primer programa en un fichero temporal y hacer que el segundo programa lo lea. Esto significaría que el primer programa tendría que acabar de funcionar antes de que segundo podría comenzar. ¿Qué si usted podría hacer ambos programas funcionar con al mismo tiempo y tenga el segundo programa leyeron los datos mientras que primer lo escribió? Es posible, y el mecanismo para conseguir datos a partir de un programa al otro se llama una **pipe**. Una colección de programas unidos juntos de esta manera se llama una **pipeline**. El programa que alimenta datos en una pipa se llama una **source**, y los datos que sacan los datos de la pipa se llama un **sink**.

Las aplicaciones enchufables del espeak del gstreamer una pipa simple: el texto entra espeak en un extremo y el sonido sale el otro y se envía a su carta de sonido. Usted puede ser que piense que no suena mucho diferente antes de lo que hacíamos, pero es. Cuando usted apenas funciona el espeak el programa tiene que cargarse en memoria, habla el texto que usted lo da en la tarjeta de sonidos, después se descarga. Éste es una de las razones que usted no puede apenas utilizar el espeak una palabra a la vez para alcanzar discurso con palabras destacadas. Hay un retraso corto mientras que el programa está cargando. No es que sensible si usted da a espeak una frase o una oración completa para hablar, pero si sucede para cada palabra es *muy* sensible. Usando el gstreamer enchufable podemos tener espeak cargado en memoria todo el tiempo, apenas esperándonos para enviar algunas palabras en su pipa de la entrada. Las hablará y después esperará la hornada siguiente.

Puesto que podemos controlarlo qué entra la pipa somos posibles detenerse brevemente y reasumir

discurso.

El ejemplo que utilizaremos aquí es una versión de **Read Etexts** otra vez, pero en vez de la actividad vamos a modificar la versión independiente. No hay nada especial sobre el gstreamer enchufable que le hace solamente el trabajo con Actividades. Cualquier programa del Python puede utilizarlo. Estoy incluyendo solamente el texto al discurso como asunto en este manual porque cada instalación del azúcar incluye el espeak y muchas Actividades podrían encontrarlo útil.

Hay a en nuestro depósito de Git nombrado **speech.py** que parezca esto:

```
import gst

voice = 'default'
pitch = 0

rate = -20
highlight_cb = None

def _create_pipe():
    pipeline = 'espeak name=source ! autoaudiosink'
    pipe = gst.parse_launch(pipeline)

    def stop_cb(bus, message):
        pipe.set_state(gst.STATE_NULL)

    def mark_cb(bus, message):
        if message.structure.get_name() == 'espeak-mark':
            mark = message.structure['mark']
            highlight_cb(int(mark))

    bus = pipe.get_bus()
    bus.add_signal_watch()
    bus.connect('message::eos', stop_cb)
    bus.connect('message::error', stop_cb)
    bus.connect('message::element', mark_cb)

    return (pipe.get_by_name('source'), pipe)

def _speech(source, pipe, words):
    source.props.pitch = pitch
    source.props.rate = rate
    source.props.voice = voice
    source.props.text = words;
    pipe.set_state(gst.STATE_PLAYING)

info_source, info_pipe = _create_pipe()
play_source, play_pipe = _create_pipe()

# track for marks
play_source.props.track = 2

def voices():
    return info_source.props.voices

def say(words):
    _speech(info_source, info_pipe, words)
    print words

def play(words):
    _speech(play_source, play_pipe, words)

def is_stopped():
    for i in play_pipe.get_state():
        if isinstance(i, gst.State) and i == gst.STATE_NULL:
             return True
    return False

def stop():
    play_pipe.set_state(gst.STATE_NULL)

def is_paused():
    for i in play_pipe.get_state():
```

```
        if isinstance(i, gst.State) and i == gst.STATE_PAUSED:
             return True
    return False

def pause():
    play_pipe.set_state(gst.STATE_PAUSED)

def rate_up():
    global rate
    rate = min(99, rate + 10)

def rate_down():
    global rate
    rate = max(-99, rate - 10)

def pitch_up():
    global pitch
    pitch = min(99, pitch + 10)

def pitch_down():
    global pitch
    pitch = max(-99, pitch - 10)

def prepare_highlighting(label_text):
    i = 0
    j = 0
    word_begin = 0
    word_end = 0
    current_word = 0
    word_tuples = []
    omitted = [' ', '\n', u'\r', '_', '[', '{', ']', '}', '|', '<',\
        '>', '*', '+', '/', '\\' ]
    omitted_chars = set(omitted)
    while i < len(label_text):
        if label_text[i] not in omitted_chars:
            word_begin = i
            j = i
            while j < len(label_text) and label_text[j] not in omitted_chars:
                 j = j + 1
                 word_end = j
                 i = j
            word_t = (word_begin, word_end, label_text[word_begin: word_end].strip())
            if word_t[2] != u'\r':
                 word_tuples.append(word_t)
        i = i + 1
    return word_tuples

def add_word_marks(word_tuples):
    "Adds a mark between each word of text."
    i = 0
    marked_up_text = ' '
    while i < len(word_tuples):
        word_t = word_tuples[i]
        marked_up_text = marked_up_text + '' + word_t[2]
        i = i + 1
    return marked_up_text + ''
```

Hay otro archivo nombrado **ReadEtextsTTS.py** que parezca esto:

```
import sys
import os
import zipfile
import pygtk
import gtk
import getopt
import pango
import gobject
import time
import speech

speech_supported = True

try:
    import gst
    gst.element_factory_make('espeak')
```

```
    print 'speech supported!'
except Exception, e:
    speech_supported = False
    print 'speech not supported!'

page=0
PAGE_SIZE = 45

class ReadEtextsActivity():
    def __init__(self):
        "The entry point to the Activity"
        speech.highlight_cb = self.highlight_next_word
        # print speech.voices()

    def highlight_next_word(self, word_count):
        if word_count < len(self.word_tuples):
            word_tuple = self.word_tuples[word_count]
            textbuffer = self.textview.get_buffer()
            tag = textbuffer.create_tag()
            tag.set_property('weight', pango.WEIGHT_BOLD)
            tag.set_property( 'foreground', "white")
            tag.set_property( 'background', "black")
            iterStart = textbuffer.get_iter_at_offset(word_tuple[0])
            iterEnd = textbuffer.get_iter_at_offset(word_tuple[1])
            bounds = textbuffer.get_bounds()
            textbuffer.remove_all_tags(bounds[0], bounds[1])
            textbuffer.apply_tag(tag, iterStart, iterEnd)
            v_adjustment = self.scrolled_window.get_vadjustment()
            max = v_adjustment.upper - v_adjustment.page_size
            max = max * word_count
            max = max / len(self.word_tuples)
            v_adjustment.value = max
        return True

    def keypress_cb(self, widget, event):
        "Respond when the user presses one of the arrow keys"
        global done
        global speech_supported
        keyname = gtk.gdk.keyval_name(event.keyval)
        if keyname == 'KP_End' and speech_supported:
            if speech.is_paused() or speech.is_stopped():
                speech.play(self.words_on_page)
            else:
                speech.pause()
            return True
        if keyname == 'plus':
            self.font_increase()
            return True
        if keyname == 'minus':
            self.font_decrease()
            return True
        if speech_supported and speech.is_stopped() == False and\
            speech.is_paused == False:
            # If speech is in progress, ignore other keys.
            return True
        if keyname == '7':
            speech.pitch_down()
            speech.say('Pitch Adjusted')
            return True
        if keyname == '8':
            speech.pitch_up()
            speech.say('Pitch Adjusted')
            return True
        if keyname == '9':
            speech.rate_down()
            speech.say('Rate Adjusted')
            return True
        if keyname == '0':
            speech.rate_up()
            speech.say('Rate Adjusted')
            return True
        if keyname == 'KP_Right':
            self.page_next()
            return True
        if keyname == 'Page_Up' or keyname == 'KP_Up':
```

```
            self.page_previous()
            return True
        if keyname == 'KP_Left':
            self.page_previous()
            return True
        if keyname == 'Page_Down' or keyname == 'KP_Down':
            self.page_next()
            return True
        if keyname == 'Up':
            self.scroll_up()
            return True
        if keyname == 'Down':
            self.scroll_down()
            return True
        return False

    def page_previous(self):
        global page
        page=page-1
        if page <⁞ 0: page=0
        self.show_page(page)
        v_adjustment = self.scrolled_window.get_vadjustment()
        v_adjustment.value = v_adjustment.upper - v_adjustment.page_size

    def page_next(self):
        global page
        page=page+1
        if page >= len(self.page_index): page=0
        self.show_page(page)
        v_adjustment = self.scrolled_window.get_vadjustment()
        v_adjustment.value = v_adjustment.lower

    def font_decrease(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size - 1
        if font_size < 1:
            font_size = 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def font_increase(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size + 1
        self.font_desc.set_size(font_size * 1024)
        self.textview.modify_font(self.font_desc)

    def scroll_down(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.upper - v_adjustment.page_size:
            self.page_next()
            return
        if v_adjustment.value < v_adjustment.upper - v_adjustment.page_size:
            new_value = v_adjustment.value + v_adjustment.step_increment
            if new_value > v_adjustment.upper - v_adjustment.page_size:
                new_value = v_adjustment.upper - v_adjustment.page_size
            v_adjustment.value = new_value

    def scroll_up(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.lower:
            self.page_previous()
            return
        if v_adjustment.value > v_adjustment.lower:
            new_value = v_adjustment.value - v_adjustment.step_increment
            if new_value < v_adjustment.lower:
                new_value = v_adjustment.lower
            v_adjustment.value = new_value

    def show_page(self, page_number):
        global PAGE_SIZE, current_word
        position = self.page_index[page_number]
        self.etext_file.seek(position)
        linecount = 0
        label_text = ''
        textbuffer = self.textview.get_buffer()
```

```
        while linecount < PAGE_SIZE:
            line = self.etext_file.readline()
            label_text = label_text + unicode(line, 'iso-8859-1')
            linecount = linecount + 1
        textbuffer.set_text(label_text)
        self.textview.set_buffer(textbuffer)
        self.word_tuples = speech.prepare_highlighting(label_text)
        self.words_on_page = speech.add_word_marks(self.word_tuples)

    def save_extracted_file(self, zipfile, filename):
        "Extract the file to a temp directory for viewing"
        filebytes = zipfile.read(filename)
        f = open("/tmp/" + filename, 'w')
        try:
            f.write(filebytes)
        finally:
            f.close()

    def read_file(self, filename):
        "Read the Etext file"
        global PAGE_SIZE

        if zipfile.is_zipfile(filename):
            self.zf = zipfile.ZipFile(filename, 'r')
            self.book_files = self.zf.namelist()
            self.save_extracted_file(self.zf, self.book_files[0])
            currentFileName = "/tmp/" + self.book_files[0]
        else:
            currentFileName = filename

        self.etext_file = open(currentFileName,"r")
        self.page_index = [ 0 ]
        linecount = 0
        while self.etext_file:
            line = self.etext_file.readline()
            if not line:
                break
            linecount = linecount + 1
            if linecount >= PAGE_SIZE:
                position = self.etext_file.tell()
                self.page_index.append(position)
                linecount = 0
        if filename.endswith(".zip"):
            os.remove(currentFileName)

    def delete_cb(self, widget, event, data=None):
        speech.stop()
        return False

    def destroy_cb(self, widget, data=None):
        speech.stop()
        gtk.main_quit()

    def main(self, file_path):
        self.window = gtk.Window(gtk.WINDOW_TOPLEVEL)
        self.window.connect("delete_event", self.delete_cb)
        self.window.connect("destroy", self.destroy_cb)
        self.window.set_title("Read Etexts Activity")
        self.window.set_size_request(800, 600)
        self.window.set_border_width(0)
        self.read_file(file_path)
        self.scrolled_window = gtk.ScrolledWindow(hadjustment=None, vadjustment=None)
        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_left_margin(50)
        self.textview.set_cursor_visible(False)
        self.textview.connect("key_press_event", self.keypress_cb)
        self.font_desc = pango.FontDescription("sans 12")
        self.textview.modify_font(self.font_desc)
        self.show_page(0)
        self.scrolled_window.add(self.textview)
        self.window.add(self.scrolled_window)
        self.textview.show()
        self.scrolled_window.show()
        self.window.show()
```

```
        gtk.main()

if __name__ == "__main__":
    try:
        opts, args = getopt.getopt(sys.argv[1:], "")
        ReadEtextsActivity().main(args[0])
    except getopt.error, msg:
        print msg
        print "This program has no options"
        sys.exit(2)
```

El programa **ReadEtextsTTS** tiene solamente algunos cambios para hacerlo permitido para el discurso. Primer comprueba para saber si hay la existencia del gstreamer enchufable:

```
speech_supported = True

try:
    import gst
    gst.element_factory_make('espeak')
    print 'speech supported!'
except Exception, e:
    speech_supported = False
    print 'speech not supported!'
```

Este código detecta si el enchufable es instalado intentando importar para el Python que la biblioteca asociada a él nombró el "gst". Si los fall de la importación él lanzan una **Exception** y cogemos esa excepción y la utilizamos para fijar una variable nombrada **speech_supported a False**. Podemos comprobar el valor de esta variable en otros lugares en el programa para hacer un programa que trabaje con el texto al discurso si está disponible y sin él si no es. Haciendo un trabajo del programa en diversos ambientes haciendo estas clases de cheques se llama *degradación agraciado*. La cogida de excepciones en las importaciones es una técnica común en el Python para alcanzar esto. Si usted quisiera que su actividad funcionara en más viejas versiones del azúcar usted puede encontrarse el usar de ella.

El pedacito siguiente del código vamos a mirar puntos culminantes una palabra en el textview y las volutas el textview para mantener la palabra destacada visible.

```
class ReadEtextsActivity():
    def __init__(self):
        "The entry point to the Activity"
        speech.highlight_cb = self.highlight_next_word
        # print speech.voices()

    def highlight_next_word(self, word_count):
        if word_count < len(self.word_tuples):
            word_tuple = self.word_tuples[word_count]
            textbuffer = self.textview.get_buffer()
            tag = textbuffer.create_tag()
            tag.set_property('weight', pango.WEIGHT_BOLD)
            tag.set_property( 'foreground', "white")
            tag.set_property( 'background', "black")
            iterStart = textbuffer.get_iter_at_offset(word_tuple[0])
            iterEnd = textbuffer.get_iter_at_offset(word_tuple[1])
            bounds = textbuffer.get_bounds()
            textbuffer.remove_all_tags(bounds[0], bounds[1])
            textbuffer.apply_tag(tag, iterStart, iterEnd)
            v_adjustment = self.scrolled_window.get_vadjustment()
            max = v_adjustment.upper - v_adjustment.page_size
            max = max * word_count
            max = max / len(self.word_tuples)
            v_adjustment.value = max
        return True
```

En el método del *__init ()* asignamos un *highlight_cb* llamado variable en **speech.py** con un método llamado *highlight_next_word ()*. Esto da a **speech.py una** manera de llamar ese método cada vez que una nueva palabra en el textview necesita ser destacada.

La línea siguiente imprimirá la lista de tuples que contienen nombres de la voz al terminal si usted

uncomment él. No estamos dejando al usuario cambiar voces en este uso pero no sería difícil agregar esa característica.

El código para el método que destaca las palabras sigue. Qué lo hace es mirada en una lista de tuples que contengan las compensaciones que comienzan y de terminaciones de cada palabra en el almacenador intermediario del texto de los textarea. El llamador de este método pasa en un número de la palabra (por ejemplo la primera palabra en el almacenador intermediario es la palabra 0, la segunda es la palabra 1, y así sucesivamente). El método mira para arriba esa entrada en la lista, consigue su comenzar y la terminación de compensaciones, quita destacar anterior, después destaca el nuevo texto. Además de eso imagina qué fracción del número total de palabras debe y enrolla la palabra actual el textviewer bastante cerciorarse de que la palabra es visible.

Por supuesto este método trabaja mejor en las páginas sin muchas líneas en blanco, que es afortunadamente la mayoría de las páginas. No trabaja tan bien en las páginas de título. Un programador experimentado podía subir probablemente con una manera más elegante y más confiable de hacer este movimiento en sentido vertical. Déjeme saber con lo que usted sube.

Foméntenos abajo ven que el código que consigue los golpes de teclado el usuario incorpora y que hace cosas discurso-relacionadas con ellas:

```
def keypress_cb(self, widget, event):
    "Respond when the user presses one of the arrow keys"
    global done
    global speech_supported
    keyname = gtk.gdk.keyval_name(event.keyval)
    if keyname == 'KP_End' and speech_supported:
        if speech.is_paused() or speech.is_stopped():
            speech.play(self.words_on_page)
        else:
            speech.pause()
        return True
    if speech_supported and speech.is_stopped() == False and \
        speech.is_paused == False:
        # If speech is in progress, ignore other keys.
        return True
    if keyname == '7':
        speech.pitch_down()
        speech.say('Pitch Adjusted')
        return True
    if keyname == '8':
        speech.pitch_up()
        speech.say('Pitch Adjusted')
        return True
    if keyname == '9':
        speech.rate_down()
        speech.say('Rate Adjusted')
        return True
    if keyname == '0':
        speech.rate_up()
        speech.say('Rate Adjusted')
        return True
```

Como usted puede ver, las funciones que estamos llamando están todas en el archivo **speech.py** que importamos. Usted no tiene que completamente entender cómo estas funciones trabajan para hacer uso de ellas en sus propias Actividades. Note que el código según lo escrito previene al usuario de echada o de tarifa cambiante mientras que el discurso está en curso. Note también que hay dos diversos métodos en speech.py para hacer discurso. **el juego ()** es el método para hacer el texto al discurso con destacar de la palabra. **diga que ()** está para decir las frases cortas producidas por el interfaz utilizador, en este caso "eche ajustado" y "tarifa ajustado". Por supuesto si usted pusiera código como esto en su actividad usted utilizaría la función del _ () así que estas frases se podrían traducir a otras idiomas.

Hay un más pedacito de código que necesitamos hacer el texto al discurso con destacar: necesitamos preparar las palabras para ser hablado para ser destacado en el textviewer.

```
    def show_page(self, page_number):
        global PAGE_SIZE, current_word
        position = self.page_index[page_number]
        self.etext_file.seek(position)
        linecount = 0
        label_text = ''
        textbuffer = self.textview.get_buffer()
        while linecount < PAGE_SIZE:
            line = self.etext_file.readline()
            label_text = label_text + unicode(line, 'iso-8859-1')
            linecount = linecount + 1
        textbuffer.set_text(label_text)
        self.textview.set_buffer(textbuffer)
        self.word_tuples = speech.prepare_highlighting(label_text)
        self.words_on_page = speech.add_word_marks(self.word_tuples)
```

El principio de este método lee el valor de una página del texto en una secuencia llamada label_text y lo pone en el almacenador intermediario de los textview. Los dos pasados alinea fracturas el texto en palabras, yéndose en la puntuación, y pone cada palabra y su principio y conclusión compensa en un tuple. Los tuples se agregan a una lista.

speech.add_word_marks () convierte las palabras en la lista a un documento en formato de *SSML* (*lengua de margen de beneficio de la síntesis de discurso*). SSML es un estándar para agregar las etiquetas (clase como de las etiquetas usadas para hacer Web pages) al texto para decir a software del discurso cuál hacer con el texto. Apenas estamos utilizando una parte muy pequeña de este estándar para producir marcado encima del documento con una marca entre cada palabra, como esto:

```
Thequickbrown
    foxjumps
```

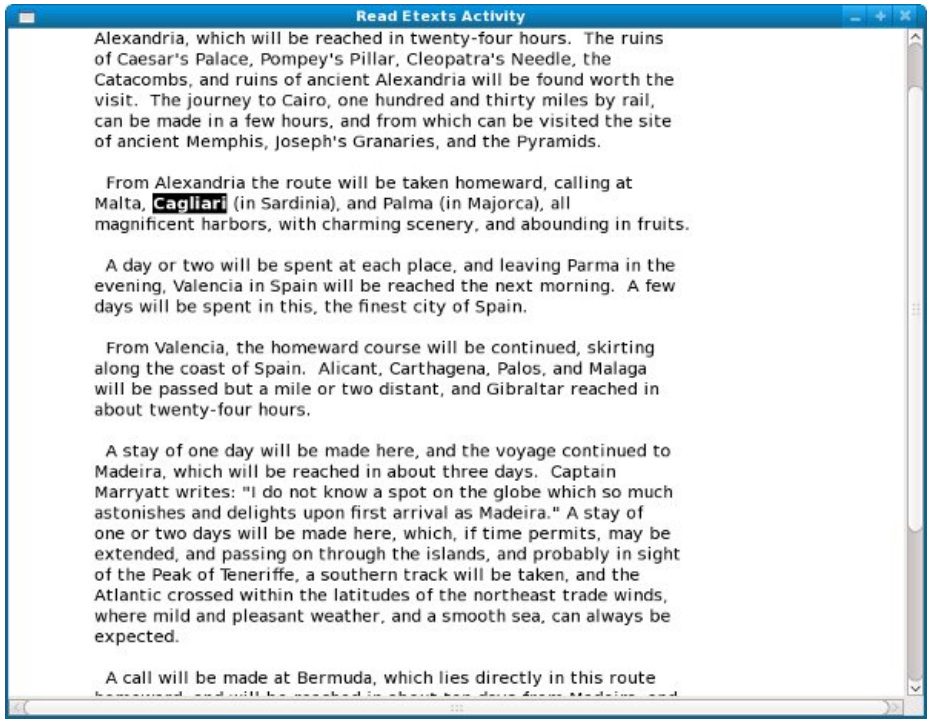
Cuando el espeak lee este archivo hará un *servicio repetido* en nuestro programa que lee cada vez una de las etiquetas de la marca. El servicio repetido contendrá el número de la palabra en la lista de los **word_tuples** que conseguirá de la cualidad **conocida de la** etiqueta de la **marca**. De esta manera el método que es llamado sabrá qué palabra a destacar. La ventaja de usar el nombre de la marca algo que apenas destacando la palabra siguiente en el textviewer es que si el espeak no puede hacer uno de los servicios repetidos el destacar no será lanzado de la sinc. Esto era un problema con el discurso-despachador.

Un servicio repetido es apenas como lo que suena. Cuando un programa llama otro programa puede pasar en una función o un método sus los propios que quisiera que el segundo programa llamara cuando sucede algo.

Para probar el nuevo funcionamiento del programa

```
./ReadEtextsTTS.py bookfile
```

del terminal. Usted puede ajustar la echada y la tarifa arriba y abajo de usar las llaves **7, 8, 9**, y **0** en la fila superior del teclado. Debe decir la "echada ajustada" o la "tarifa ajustada" cuando usted hace eso. Usted puede comenzar, detenerse brevemente, y reasumir discurso con destacar usando la llave de **final** en el telclado numérico. (En el ordenador portátil de XO las llaves del "juego" se trazan a cuál es el telclado numérico numérico en un teclado normal. Esto hace estas llaves prácticas para el uso cuando el XO se dobla en modo de la tableta y el teclado no está disponible). Usted no puede cambiar la echada o clasificar mientras que el discurso está en curso. Tentativas de hacer que serán no hechas caso. El programa en la acción parece esto:



Eso nos trae al final del asunto del texto al discurso. Si usted debe como ver más, el depósito de Git para este libro tiene algunos más programas de muestra que utilicen el espeak del gstreamer enchufable. Estos ejemplos fueron creados por el autor del enchufable y demuestran algunas otras maneras que usted puede utilizarlo. Hay incluso un “choir” programa que demuestra las voces múltiples que hablan al mismo tiempo.

17. DIVERSIÓN CON EL DIARIO

INTRODUCCIÓN

Por abandono cada actividad crea y lee una entrada de diario. La mayoría de las Actividades no necesitan hacer más con el diario que eso, y si su actividad es como ésa usted no necesitará la información en este capítulo. Las ocasiones son que usted querrá algún día hacer más que eso, así que si usted guarda el leer.

Primero repasemos cuáles es el diario. El diario es una colección de archivos que cada uno tenga **meta data** (datos sobre datos) asociados a ellos. Los meta datos se almacenan como secuencias de texto e incluyen las cosas tales como el **Title**, **Description**, **Tags**, **MIME Type**, y un tiro de pantalla de la actividad cuando era último usado.

Su actividad no puede leer y escribir estos archivos directamente. En lugar azucare proporciona un API (interfaz de programación de uso) que le da una manera indirecta de agregar, suprimir y modificar entradas en el diario, tan bien como una manera de buscar entradas de diario y de hacer una lista de las entradas que cumplen los criterios de búsqueda.

El API que utilizaremos está en el paquete del **datastore**. Después de la versión .82 del azúcar este API fue reescrito, así que necesitaremos aprender cómo apoyar ambas versiones en la misma actividad.

Si usted ha leído esto lejos usted ha visto varios ejemplos donde azúcar comenzado hacia fuera a hacer una cosa y después cambiado para hacer la misma cosa una mejor manera pero todavía proporcionado una manera de crear las Actividades que trabajarían con el viejo o la nueva manera. Usted puede preguntarse si es normal que un proyecto haga esto. Pues un programador profesional yo puede decirle que hacer trucos como esto para mantener compatibilidad hacia atrás es extremadamente común, y azúcar no hace no más de esto que ninguìn otro proyecto. Hay decisiones tomadas por Herman Hollerith cuando él tabuló el censo 1890 usando tarjetas perforadas que los informáticos deben vivir con a este día.

PRESENTAR AL SUGAR COMMANDER

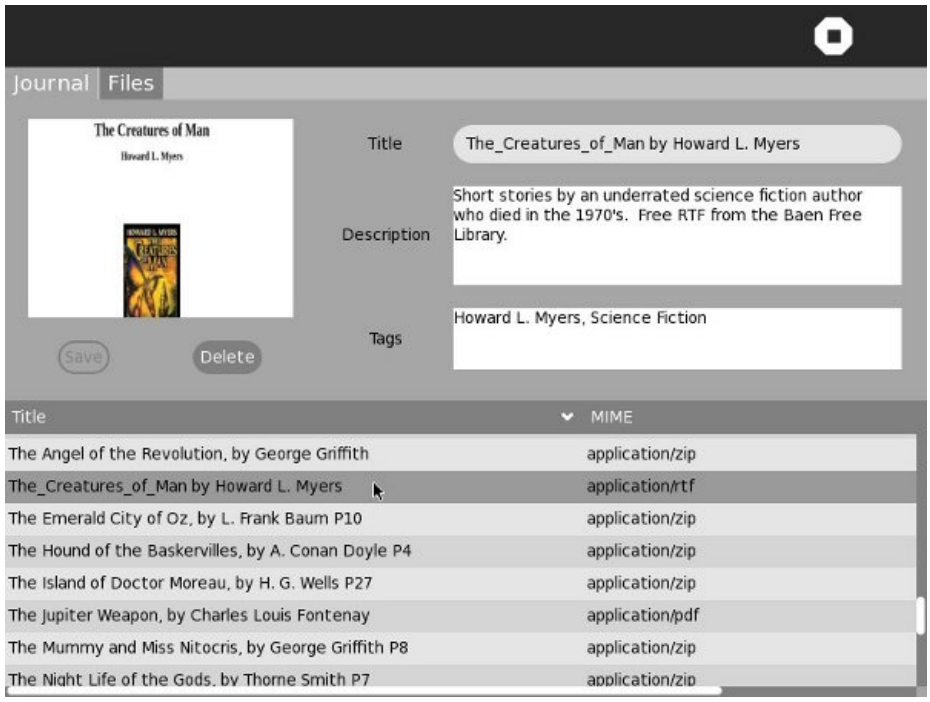
Soy un ventilador grande del concepto del diario pero no tanto de la **actividad del diario** que el azúcar utiliza para navegar a través de él y para mantenerlo. Mi queja más grande contra ella es que ella representa el contenido de las impulsiones del pulgar y las tarjetas del SD como si los archivos en éstos fueran también entradas de diario. Mi sensación es que los archivos y los directorios son una cosa y el diario es otro, y el interfaz utilizador debe reconocer eso.

En realidad la actividad del diario es y no es una actividad. Hereda código de la clase de la actividad apenas como cualquier otra actividad, y se escribe en Python y utiliza el mismo datastore API que otras Actividades utilizan. Sin embargo, se funciona de una manera especial que lo dé accione y las capacidades mucho más alla de las de una actividad ordinaria. Particularmente puede hacer dos cosas:

- Puede escribir a los archivos en medios externos como impulsiones del pulgar y tarjetas del SD.
- Solamente puede ser utilizado para reasumir entradas de diario usando otras Actividades.

Mientras que quisiera escribir una actividad del diario que hace todo la original hace pero tiene un interfaz utilizador más a mi propio gusto que el modelo de seguridad del azúcar no permitirá eso. Llegué recientemente a la conclusión que una versión más apacible de la actividad del diario pudo ser útil. Apenas pues Kal-EL a veces encuentra más útil para ser Clark Kent que superhombre, mi propia actividad pudo ser una alternativa digna a la actividad incorporada del diario cuando las energías estupendas no son necesarias.

Mi actividad, que llamo **Sugar Commander**, tiene dos lengüetas. Uno representa el diario y parece esto:



Esta lengüeta le deja hojear a través del diario clasificado por Title o tipo de MIME, entradas selectas y ver sus detalles, título de la actualización, descripción o etiquetas, y entradas de la cancelación que usted quiere no más. La otra lengüeta demuestra archivos y carpetas y parece esto:

Esta lengüeta le deja hojear a través de los archivos y las carpetas o el sistema de ficheros regular, incluyendo impulsiones del pulgar y tarjetas del SD. Usted puede seleccionar un archivo y hacer una entrada de diario fuera de ella empujando el botón en la parte inferior de la pantalla.

Esta actividad tiene código muy pequeño y todavía lo maneja hacer todo que una actividad ordinaria puede hacer con el diario. Usted puede transferir el depósito de Git usando este comando:

```
git clone git://git.sugarlabs.org/sugar-commander/mainline.git
```

Hay solamente un archivo de fuente, **sugarcommander.py**:

```
import logging
import os
import gtk
import pango
import zipfile
from sugar import mime
from sugar.activity import activity
from sugar.datastore import datastore
from sugar.graphics.alert import NotifyAlert
from sugar.graphics import style
from gettext import gettext as _
import gobject
import dbus

COLUMN_TITLE = 0
COLUMN_MIME = 1
COLUMN_JOBJECT = 2

DS_DBUS_SERVICE = 'org.laptop.sugar.DataStore'
DS_DBUS_INTERFACE = 'org.laptop.sugar.DataStore'
DS_DBUS_PATH = '/org/laptop/sugar/DataStore'

_logger = logging.getLogger('sugar-commander')

class SugarCommander(activity.Activity):
    def __init__(self, handle, create_jobject=True):
        "The entry point to the Activity"
        activity.Activity.__init__(self, handle,  False)
        self.selected_journal_entry = None
        self.selected_path = None

        canvas = gtk.Notebook()
        canvas.props.show_border = True
        canvas.props.show_tabs = True
        canvas.show()

        self.ls_journal = gtk.ListStore(gobject.TYPE_STRING,
                gobject.TYPE_STRING,
                gobject.TYPE_PYOBJECT)
        self.tv_journal = gtk.TreeView(self.ls_journal)
        self.tv_journal.set_rules_hint(True)
        self.tv_journal.set_search_column(COLUMN_TITLE)
        self.selection_journal = self.tv_journal.get_selection()
        self.selection_journal.set_mode(gtk.SELECTION_SINGLE)
        self.selection_journal.connect("changed", self.selection_journal_cb)
        renderer = gtk.CellRendererText()
        renderer.set_property('wrap-mode', gtk.WRAP_WORD)
        renderer.set_property('wrap-width', 500)
        renderer.set_property('width', 500)
        self.col_journal = gtk.TreeViewColumn(_('Title'), renderer,
                                              text=COLUMN_TITLE)
        self.col_journal.set_sort_column_id(COLUMN_TITLE)
        self.tv_journal.append_column(self.col_journal)

        mime_renderer = gtk.CellRendererText()
        mime_renderer.set_property('width', 500)
        self.col_mime = gtk.TreeViewColumn(_('MIME'), mime_renderer,
                                           text=COLUMN_MIME)
        self.col_mime.set_sort_column_id(COLUMN_MIME)
        self.tv_journal.append_column(self.col_mime)
```

```
self.list_scroller_journal = gtk.ScrolledWindow(
               hadjustment=None, vadjustment=None)
self.list_scroller_journal.set_policy(
           gtk.POLICY_AUTOMATIC, gtk.POLICY_AUTOMATIC)
self.list_scroller_journal.add(self.tv_journal)

label_attributes = pango.AttrList()
label_attributes.insert(pango.AttrSize(14000, 0, -1))
label_attributes.insert(pango.AttrForeground(65535, 65535, 65535, 0, -1))

tab1_label = gtk.Label(_("Journal"))
tab1_label.set_attributes(label_attributes)
tab1_label.show()
self.tv_journal.show()
self.list_scroller_journal.show()

column_table = gtk.Table(rows=1,  columns=2,  homogeneous = False)

image_table = gtk.Table(rows=2,  columns=2,  homogeneous=False)
self.image = gtk.Image()
image_table.attach(self.image, 0, 2, 0, 1, xoptions=gtk.FILL|gtk.SHRINK,
                   yoptions=gtk.FILL|gtk.SHRINK, xpadding=10, ypadding=10)

self.btn_save = gtk.Button(_("Save"))
self.btn_save.connect('button_press_event',
                      self.save_button_press_event_cb)
image_table.attach(self.btn_save,  0, 1, 1, 2,  xoptions=gtk.SHRINK,
                    yoptions=gtk.SHRINK,  xpadding=10,  ypadding=10)
self.btn_save.props.sensitive = False
self.btn_save.show()

self.btn_delete = gtk.Button(_("Delete"))
self.btn_delete.connect('button_press_event',
                        self.delete_button_press_event_cb)
image_table.attach(self.btn_delete,  1, 2, 1, 2,  xoptions=gtk.SHRINK,
                   yoptions=gtk.SHRINK,  xpadding=10,  ypadding=10)
self.btn_delete.props.sensitive = False
self.btn_delete.show()

column_table.attach(image_table,  0, 1, 0, 1,
                    xoptions=gtk.FILL|gtk.SHRINK,
                      yoptions=gtk.SHRINK,  xpadding=10,  ypadding=10)

entry_table = gtk.Table(rows=3, columns=2,
                        homogeneous=False)

title_label = gtk.Label(_("Title"))
entry_table.attach(title_label, 0, 1, 0, 1,
                   xoptions=gtk.SHRINK,
                   yoptions=gtk.SHRINK,
                   xpadding=10, ypadding=10)
title_label.show()

self.title_entry = gtk.Entry(max=0)
entry_table.attach(self.title_entry, 1, 2, 0, 1,
                   xoptions=gtk.FILL|gtk.SHRINK,
                   yoptions=gtk.SHRINK, xpadding=10, ypadding=10)
self.title_entry.connect('key_press_event',
                         self.key_press_event_cb)
self.title_entry.show()

description_label = gtk.Label(_("Description"))
entry_table.attach(description_label, 0, 1, 1, 2,
                   xoptions=gtk.SHRINK,
                   yoptions=gtk.SHRINK,
                   xpadding=10, ypadding=10)
description_label.show()

self.description_textview = gtk.TextView()
self.description_textview.set_wrap_mode(gtk.WRAP_WORD)
entry_table.attach(self.description_textview, 1, 2, 1, 2,
                   xoptions=gtk.EXPAND|gtk.FILL|gtk.SHRINK,
                   yoptions=gtk.EXPAND|gtk.FILL|gtk.SHRINK,
                   xpadding=10, ypadding=10)
self.description_textview.props.accepts_tab = False
```

```
self.description_textview.connect('key_press_event',
                                  self.key_press_event_cb)
self.description_textview.show()

tags_label = gtk.Label(_("Tags"))
entry_table.attach(tags_label, 0, 1, 2, 3,
                   xoptions=gtk.SHRINK,
                   yoptions=gtk.SHRINK,
                   xpadding=10, ypadding=10)
tags_label.show()

self.tags_textview = gtk.TextView()
self.tags_textview.set_wrap_mode(gtk.WRAP_WORD)
entry_table.attach(self.tags_textview, 1, 2, 2, 3,
                   xoptions=gtk.FILL,
                   yoptions=gtk.EXPAND|gtk.FILL,
                   xpadding=10, ypadding=10)
self.tags_textview.props.accepts_tab = False
self.tags_textview.connect('key_press_event',
                           self.key_press_event_cb)
self.tags_textview.show()

entry_table.show()

self.scroller_entry = gtk.ScrolledWindow(
                      hadjustment=None, vadjustment=None)
self.scroller_entry.set_policy(gtk.POLICY_NEVER, gtk.POLICY_AUTOMATIC)
self.scroller_entry.add_with_viewport(entry_table)
self.scroller_entry.show()

column_table.attach(self.scroller_entry,  1, 2, 0, 1,
                    xoptions=gtk.FILL|gtk.EXPAND|gtk.SHRINK,
                    yoptions=gtk.FILL|gtk.EXPAND|gtk.SHRINK,
                    xpadding=10,  ypadding=10)
image_table.show()
column_table.show()

vbox = gtk.VBox(homogeneous=True,  spacing=5)
vbox.pack_start(column_table)
vbox.pack_end(self.list_scroller_journal)

canvas.append_page(vbox,  tab1_label)

self._filechooser = gtk.FileChooserWidget(
    action=gtk.FILE_CHOOSER_ACTION_OPEN, backend=None)
self._filechooser.set_current_folder("/media")
self.copy_button = gtk.Button(_("Copy File To The Journal"))
self.copy_button.connect('clicked',  self.create_journal_entry)
self.copy_button.show()
self._filechooser.set_extra_widget(self.copy_button)
preview = gtk.Image()
self._filechooser.set_preview_widget(preview)
self._filechooser.connect("update-preview",
                          self.update_preview_cb, preview)
tab2_label = gtk.Label(_("Files"))
tab2_label.set_attributes(label_attributes)
tab2_label.show()
canvas.append_page(self._filechooser,  tab2_label)

self.set_canvas(canvas)
self.show_all()

toolbox = activity.ActivityToolbox(self)
activity_toolbar = toolbox.get_activity_toolbar()
activity_toolbar.keep.props.visible = False
activity_toolbar.share.props.visible = False
self.set_toolbox(toolbox)
toolbox.show()

self.load_journal_table()

bus = dbus.SessionBus()
remote_object = bus.get_object(DS_DBUS_SERVICE, DS_DBUS_PATH)
_datastore = dbus.Interface(remote_object, DS_DBUS_INTERFACE)
_datastore.connect_to_signal('Created', self.datastore_created_cb)
```

```
    _datastore.connect_to_signal('Updated', self.datastore_updated_cb)
    _datastore.connect_to_signal('Deleted', self.datastore_deleted_cb)

    self.selected_journal_entry = None

def update_preview_cb(self, file_chooser, preview):
    filename = file_chooser.get_preview_filename()
    try:
        file_mimetype = mime.get_for_file(filename)
        if file_mimetype.startswith('image/'):
            pixbuf = gtk.gdk.pixbuf_new_from_file_at_size(filename,
                                style.zoom(320), style.zoom(240))
            preview.set_from_pixbuf(pixbuf)
            have_preview = True
        elif file_mimetype  == 'application/x-cbz':
            fname = self.extract_image(filename)
            pixbuf = gtk.gdk.pixbuf_new_from_file_at_size(fname,
                                style.zoom(320), style.zoom(240))
            preview.set_from_pixbuf(pixbuf)
            have_preview = True
            os.remove(fname)
        else:
            have_preview = False
    except:
        have_preview = False
    file_chooser.set_preview_widget_active(have_preview)
    return

def key_press_event_cb(self, entry, event):
    self.btn_save.props.sensitive = True

def save_button_press_event_cb(self, entry, event):
    self.update_entry()

def delete_button_press_event_cb(self, entry, event):
    datastore.delete(self.selected_journal_entry.object_id)

def datastore_created_cb(self, uid):
    new_jobject = datastore.get(uid)
    iter = self.ls_journal.append()
    title = new_jobject.metadata['title']
    self.ls_journal.set(iter, COLUMN_TITLE, title)
    mime = new_jobject.metadata['mime_type']
    self.ls_journal.set(iter, COLUMN_MIME, mime)
    self.ls_journal.set(iter, COLUMN_JOBJECT, new_jobject)

def datastore_updated_cb(self,  uid):
    new_jobject = datastore.get(uid)
    iter = self.ls_journal.get_iter_first()
    for row in self.ls_journal:
        jobject = row[COLUMN_JOBJECT]
        if jobject.object_id == uid:
            title = new_jobject.metadata['title']
            self.ls_journal.set_value(iter, COLUMN_TITLE, title)
            break
        iter = self.ls_journal.iter_next(iter)
    object_id = self.selected_journal_entry.object_id
    if object_id == uid:
        self.set_form_fields(new_jobject)

def datastore_deleted_cb(self,  uid):
    save_path = self.selected_path
    iter = self.ls_journal.get_iter_first()
    for row in self.ls_journal:
        jobject = row[COLUMN_JOBJECT]
        if jobject.object_id == uid:
            self.ls_journal.remove(iter)
            break
        iter = self.ls_journal.iter_next(iter)

    try:
        self.selection_journal.select_path(save_path)
        self.tv_journal.grab_focus()
    except:
        self.title_entry.set_text('')
```

```
        description_textbuffer = self.description_textview.get_buffer()
        description_textbuffer.set_text('')
        tags_textbuffer = self.tags_textview.get_buffer()
        tags_textbuffer.set_text('')
        self.btn_save.props.sensitive = False
        self.btn_delete.props.sensitive = False
        self.image.clear()
        self.image.show()

    def update_entry(self):
        needs_update = False

        if self.selected_journal_entry is None:
            return

        object_id = self.selected_journal_entry.object_id
        jobject = datastore.get(object_id)

        old_title = jobject.metadata.get('title', None)
        if old_title != self.title_entry.props.text:
            jobject.metadata['title'] = self.title_entry.props.text
            jobject.metadata['title_set_by_user'] = '1'
            needs_update = True

        old_tags = jobject.metadata.get('tags', None)
        new_tags = self.tags_textview.props.buffer.props.text
        if old_tags != new_tags:
            jobject.metadata['tags'] = new_tags
            needs_update = True

        old_description = jobject.metadata.get('description', None)
        new_description = self.description_textview.props.buffer.props.text
        if old_description != new_description:
            jobject.metadata['description'] = new_description
            needs_update = True

        if needs_update:
            datastore.write(jobject, update_mtime=False,
                            reply_handler=self.datastore_write_cb,
                            error_handler=self.datastore_write_error_cb)
        self.btn_save.props.sensitive = False

    def datastore_write_cb(self):
        pass

    def datastore_write_error_cb(self, error):
        logging.error('sugarcommander.datastore_write_error_cb: %r' % error)

    def close(self,  skip_save=False):
        "Override the close method so we don't try to create a Journal entry."
        activity.Activity.close(self,  True)

    def selection_journal_cb(self, selection):
        self.btn_delete.props.sensitive = True
        tv = selection.get_tree_view()
        model = tv.get_model()
        sel = selection.get_selected()
        if sel:
            model, iter = sel
            jobject = model.get_value(iter,COLUMN_JOBJECT)
            jobject = datastore.get(jobject.object_id)
            self.selected_journal_entry = jobject
            self.set_form_fields(jobject)
            self.selected_path = model.get_path(iter)

    def set_form_fields(self, jobject):
        self.title_entry.set_text(jobject.metadata['title'])
        description_textbuffer = self.description_textview.get_buffer()
        if jobject.metadata.has_key('description'):
            description_textbuffer.set_text(jobject.metadata['description'])
        else:
            description_textbuffer.set_text('')
        tags_textbuffer = self.tags_textview.get_buffer()
        if jobject.metadata.has_key('tags'):
            tags_textbuffer.set_text(jobject.metadata['tags'])
```

```
        else:
            tags_textbuffer.set_text('')
        self.create_preview(jobject.object_id)

    def create_preview(self,  object_id):
        jobject = datastore.get(object_id)

        if jobject.metadata.has_key('preview'):
            preview = jobject.metadata['preview']
            if preview is None or preview == '' or preview == 'None':
                if jobject.metadata['mime_type'] .startswith('image/'):
                    filename = jobject.get_file_path()
                    self.show_image(filename)
                    return
                if jobject.metadata['mime_type']  == 'application/x-cbz':
                    filename = jobject.get_file_path()
                    fname = self.extract_image(filename)
                    self.show_image(fname)
                    os.remove(fname)
                    return

        if jobject.metadata.has_key('preview') and \
                len(jobject.metadata['preview']) > 4:

            if jobject.metadata['preview'][1:4] == 'PNG':
                preview_data = jobject.metadata['preview']
            else:
                import base64
                preview_data = base64.b64decode(jobject.metadata['preview'])

            loader = gtk.gdk.PixbufLoader()
            loader.write(preview_data)
            scaled_buf = loader.get_pixbuf()
            loader.close()
            self.image.set_from_pixbuf(scaled_buf)
            self.image.show()
        else:
            self.image.clear()
            self.image.show()

    def load_journal_table(self):
        self.btn_save.props.sensitive = False
        self.btn_delete.props.sensitive = False
        ds_mounts = datastore.mounts()
        mountpoint_id = None
        if len(ds_mounts) == 1 and ds_mounts[0]['id'] == 1:
               pass
        else:
            for mountpoint in ds_mounts:
                id = mountpoint['id']
                uri = mountpoint['uri']
                if uri.startswith('/home'):
                    mountpoint_id = id

        query = {}
        if mountpoint_id is not None:
            query['mountpoints'] = [ mountpoint_id ]
        ds_objects, num_objects = datastore.find(query, properties=['uid',
            'title',  'mime_type'])

        self.ls_journal.clear()
        for i in xrange (0, num_objects, 1):
            iter = self.ls_journal.append()
            title = ds_objects[i].metadata['title']
            self.ls_journal.set(iter, COLUMN_TITLE, title)
            mime = ds_objects[i].metadata['mime_type']
            self.ls_journal.set(iter, COLUMN_MIME, mime)
            self.ls_journal.set(iter, COLUMN_JOBJECT, ds_objects[i])
            if not self.selected_journal_entry is None and \
                self.selected_journal_entry.object_id == ds_objects[i].object_id:
                self.selection_journal.select_iter(iter)

        self.ls_journal.set_sort_column_id(COLUMN_TITLE,  gtk.SORT_ASCENDING)
        v_adjustment = self.list_scroller_journal.get_vadjustment()
        v_adjustment.value = 0
```

```
        return ds_objects[0]

    def create_journal_entry(self,  widget,  data=None):
        filename = self._filechooser.get_filename()
        journal_entry = datastore.create()
        journal_entry.metadata['title'] = self.make_new_filename(filename)
        journal_entry.metadata['title_set_by_user'] = '1'
        journal_entry.metadata['keep'] = '0'
        file_mimetype = mime.get_for_file(filename)
        if not file_mimetype is None:
            journal_entry.metadata['mime_type'] = file_mimetype
        journal_entry.metadata['buddies'] = ''
        if file_mimetype.startswith('image/'):
            preview = self.create_preview_metadata(filename)
        elif file_mimetype  == 'application/x-cbz':
            fname = self.extract_image(filename)
            preview = self.create_preview_metadata(fname)
            os.remove(fname)
        else:
            preview = ''
        if not preview  == '':
            journal_entry.metadata['preview'] =  dbus.ByteArray(preview)
        else:
            journal_entry.metadata['preview'] =  ''

        journal_entry.file_path = filename
        datastore.write(journal_entry)
        self.alert(_('Success'),  _('%s added to Journal.')
                   % self.make_new_filename(filename))

    def alert(self, title, text=None):
        alert = NotifyAlert(timeout=20)
        alert.props.title = title
        alert.props.msg = text
        self.add_alert(alert)
        alert.connect('response', self.alert_cancel_cb)
        alert.show()

    def alert_cancel_cb(self, alert, response_id):
        self.remove_alert(alert)

    def show_image(self, filename):
        "display a resized image in a preview"
        scaled_buf = gtk.gdk.pixbuf_new_from_file_at_size(filename,
                                  style.zoom(320), style.zoom(240))
        self.image.set_from_pixbuf(scaled_buf)
        self.image.show()

    def extract_image(self,  filename):
        zf = zipfile.ZipFile(filename, 'r')
        image_files = zf.namelist()
        image_files.sort()
        file_to_extract = image_files[0]
        extract_new_filename = self.make_new_filename(file_to_extract)
        if extract_new_filename is None or extract_new_filename == '':
            # skip over directory name if the images are in a subdirectory.
            file_to_extract = image_files[1]
            extract_new_filename = self.make_new_filename(file_to_extract)

        if len(image_files) > 0:
            if self.save_extracted_file(zf, file_to_extract):
                fname = os.path.join(self.get_activity_root(), 'instance',
                                     extract_new_filename)
                return fname

    def save_extracted_file(self, zipfile, filename):
        "Extract the file to a temp directory for viewing"
        try:
            filebytes = zipfile.read(filename)
        except zipfile.BadZipfile, err:
            print 'Error opening the zip file: %s' % (err)
            return False
        except KeyError,  err:
            self.alert('Key Error', 'Zipfile key not found: '
                       + str(filename))
```

```
            return
        outfn = self.make_new_filename(filename)
        if (outfn == ''):
            return False
        fname = os.path.join(self.get_activity_root(), 'instance',  outfn)
        f = open(fname, 'w')
        try:
            f.write(filebytes)
        finally:
            f.close()
        return True

    def make_new_filename(self, filename):
        partition_tuple = filename.rpartition('/')
        return partition_tuple[2]

    def create_preview_metadata(self,  filename):

        file_mimetype = mime.get_for_file(filename)
        if not file_mimetype.startswith('image/'):
            return ''

        scaled_pixbuf = gtk.gdk.pixbuf_new_from_file_at_size(filename,
                                    style.zoom(320), style.zoom(240))
        preview_data = []

        def save_func(buf, data):
            data.append(buf)

        scaled_pixbuf.save_to_callback(save_func, 'png',
                                       user_data=preview_data)
        preview_data = ''.join(preview_data)

        return preview_data
```

Miremos este método del código uno a la vez.

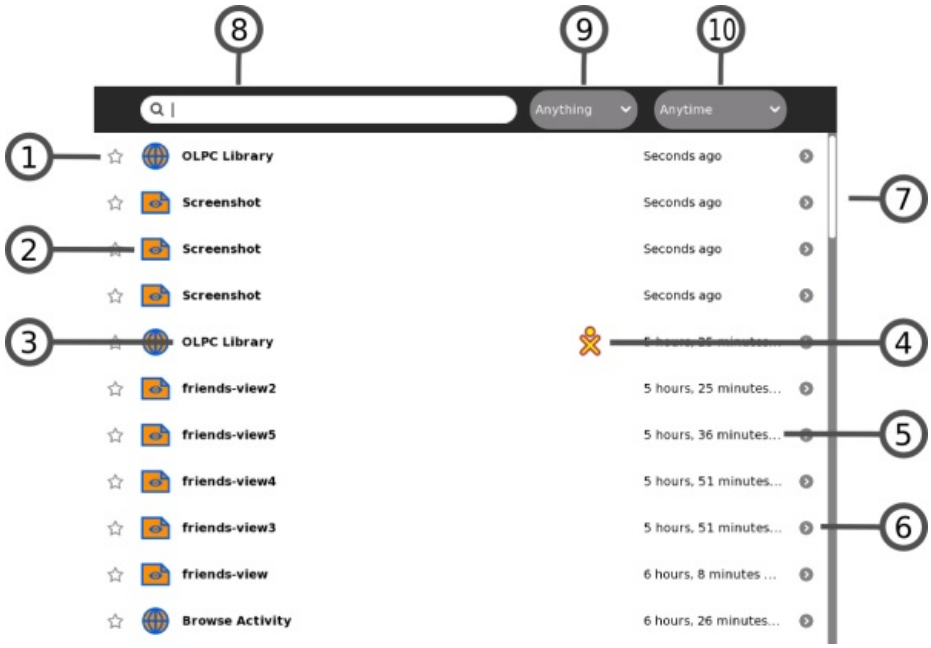
ADICIÓN DE UNA ENTRADA DE DIARIO

Agregamos una entrada de diario cuando alguien empuja un botón en el gtk.FileChooser. Éste es el código que consigue funcionamiento:

```
    def create_journal_entry(self,  widget,  data=None):
        filename = self._filechooser.get_filename()
        journal_entry = datastore.create()
        journal_entry.metadata['title'] = self.make_new_filename(filename)
        journal_entry.metadata['title_set_by_user'] = '1'
        journal_entry.metadata['keep'] = '0'
        file_mimetype = mime.get_for_file(filename)
        if not file_mimetype is None:
            journal_entry.metadata['mime_type'] = file_mimetype
        journal_entry.metadata['buddies'] = ''
        if file_mimetype.startswith('image/'):
            preview = self.create_preview_metadata(filename)
        elif file_mimetype  == 'application/x-cbz':
            fname = self.extract_image(filename)
            preview = self.create_preview_metadata(fname)
            os.remove(fname)
        else:
            preview = ''
        if not preview  == '':
            journal_entry.metadata['preview'] =  dbus.ByteArray(preview)
        else:
            journal_entry.metadata['preview'] =  ''
        journal_entry.file_path = filename
        datastore.write(journal_entry)
```

La única cosa digno de el comentario encendido aquí es los meta datos. **title** es qué aparece como #3 en el cuadro abajo. **el title_set_by_user** se fija a 1 de modo que la actividad no incite al usuario cambiar el título cuando la actividad se cierra. **keep** refiere a la pequeña estrella que aparece al principio de la entrada de

diario (véase #1 en el cuadro abajo). Destaqúela fijando esto a 1, si no fije a 0. **buddies** son una lista de usuarios que colaboraron en la entrada de diario, y en este caso no hay cualquiera (este aparezca como #4 en el cuadro abajo).



preview es un archivo de imagen en el formato del png que es un screenshot de la actividad en la acción. Esto es creada por la actividad sí mismo cuando se funciona tan allí no es ninguna necesidad hacer uno cuando usted agrega una entrada de diario. Usted puede utilizar simplemente una secuencia vacía (") para esta característica.

Porque las inspecciones previoes son mucho más visibles en comandante del azúcar que están en la actividad regular del diario que decidía que el comandante del azúcar debe hacer una imagen de la inspección previo para los archivos y los cómic de imagen tan pronto como se agreguen al diario. Para hacer esto hice un pixbuf de la imagen que cabría dentro de las dimensiones escaladas de los pixeles 320x240 e hizo un **dbus.ByteArray** fuera de él, que es el formato que las aplicaciones del diario de almacenar imágenes de la inspección previo.

el mime_type describe el formato del archivo y generalmente se asigna basado en el sufijo del nombre de fichero. Por ejemplo, los archivos que terminan en .html tienen un tipo del MIME del "texto/HTML". El Python tiene un paquete llamado los **mimetypes** que tome un nombre de fichero e imagine lo que debe ser su tipo del MIME, solamente el azúcar proporciona su propio paquete para hacer la misma cosa. Para la mayoría de los archivos cualquiera uno daría la respuesta correcta, pero el azúcar tiene sus propios tipos del MIME para las cosas como paquetes de la actividad, el etc. así que para los mejores resultados usted debe utilizar realmente el paquete del mime del azúcar. Usted puede importarlo tiene gusto de esto:

```
from sugar import mime
```

El resto de los meta datos (icono, modificado tiempo) se crea automáticamente.

ADICIÓN DE UNA ENTRADA DE DIARIO

Las Actividades del azúcar por abandono crean una entrada de diario usando *el* método *write_file*(). Habrá las Actividades que no necesitan hacer esto. Por ejemplo, **consiga los e**-libros de las transferencias

directas de los **libros del archivo del Internet al** diario, pero no tiene ninguna necesidad de una entrada de diario sus los propios. La misma cosa es verdad de **Sugar Commander**. Usted puede ser que haga un juego que no pierde de vista altas cuentas. Usted podría mantener esas cuentas una entrada de diario, pero ésa requeriría a jugadores reasumir el juego del diario algo que apenas comenzándolo para arriba del anillo de la actividad. Por esa razón usted puede ser que prefiera almacenar las altas cuentas en un archivo en el directorio de **data** algo que el diario, y no dejar una entrada de diario detrás en absoluto.

El azúcar le da una manera de hacer eso. Primero usted necesita especificar una discusión adicional en el método del *__init de* su actividad *()* como esto:

```
class SugarCommander(activity.Activity):
    def __init__(self, handle, create_jobject=True):
        "The entry point to the Activity"
        activity.Activity.__init__(self, handle, False)
```

En segundo lugar, usted necesita eliminar *() el* método *cercano* como esto:

```
    def close(self,  skip_save=False):
        "Override the close method so we don't try to create a Journal entry."
        activity.Activity.close(self,  True)
```

Ése es todo allí está a él.

ENUMERAR HACIA FUERA ENTRADAS DE DIARIO

Si usted necesita enumerar hacia fuera entradas de diario usted puede utilizar el método del *find() de* **datastore**. El método del hallazgo toma una discusión que contiene criterios de búsqueda. Si usted quiere buscar para los archivos de imagen usted puede buscar por el mime-tipo usando una declaración como esto:

```
        ds_objects, num_objects = datastore.find({'mime_type':['image/jpeg',
            'image/gif', 'image/tiff',  'image/png']}, properties=['uid',
            'title', 'mime_type']))
```

Usted puede utilizar cualquier cualidad de los meta datos para buscar encendido. Si usted quiere enumerar hacia fuera todo en el diario usted puede utilizar criterios de una búsqueda vacíos como esto:

```
        ds_objects, num_objects = datastore.find({}, properties=['uid',
            'title', 'mime_type']))
```

La discusión de las características especifica qué meta datos a volver para cada objeto en la lista. Usted debe limitar éstos a lo que usted planea utilizar, pero incluye siempre el **uid**. Una cosa que usted debe *nunca* incluir en una lista es **preview**. Esto es una demostración del archivo de imagen qué la actividad para el objeto del diario parecía cuando era última usada. Si por alguna razón usted necesita esto hay una manera simple de conseguirla para un objeto individual del diario, pero usted nunca quiere incluirlo en una lista porque retrasará su actividad enormemente.

El enumerar hacia fuera cuál está en el diario es complicado debido a la reescritura del datastore hecha para el azúcar .84. Antes de que .84 el método de *datastore.find ()* enumeraran hacia fuera entradas de diario y los archivos en medios externos como impulsiones del pulgar y las tarjetas del SD y usted necesite imaginar que sea cuál. En .84 y más adelante enumera solamente hacia fuera entradas de diario. Es afortunadamente posible escribir el código que apoya cualquier comportamiento. Aquí está el código en **Sugar Commander** que solamente las entradas de diario de las listas:

```
    def load_journal_table(self):
        self.btn_save.props.sensitive = False
        self.btn_delete.props.sensitive = False
        ds_mounts = datastore.mounts()
        mountpoint_id = None
        if len(ds_mounts) == 1 and ds_mounts[0]['id'] == 1:
               pass
```

```
        else:
            for mountpoint in ds_mounts:
                id = mountpoint['id']
                uri = mountpoint['uri']
                if uri.startswith('/home'):
                    mountpoint_id = id

        query = {}
        if mountpoint_id is not None:
            query['mountpoints'] = [ mountpoint_id ]
        ds_objects, num_objects = datastore.find(query, properties=['uid',
            'title',  'mime_type'])

        self.ls_journal.clear()
        for i in xrange (0, num_objects, 1):
            iter = self.ls_journal.append()
            title = ds_objects[i].metadata['title']
            self.ls_journal.set(iter, COLUMN_TITLE, title)
            mime = ds_objects[i].metadata['mime_type']
            self.ls_journal.set(iter, COLUMN_MIME, mime)
            self.ls_journal.set(iter, COLUMN_JOBJECT, ds_objects[i])
            if not self.selected_journal_entry is None and \
                self.selected_journal_entry.object_id == ds_objects[i].object_id:
                self.selection_journal.select_iter(iter)

        self.ls_journal.set_sort_column_id(COLUMN_TITLE,  gtk.SORT_ASCENDING)
        v_adjustment = self.list_scroller_journal.get_vadjustment()
        v_adjustment.value = 0
        return ds_objects[0]
```

Necesitamos utilizar el método de *datastore.mounts()* para dos propósitos:

- En el azúcar .82 y debajo de él enumerará hacia fuera todos los puntos de montaje, incluyendo el lugar el diario se monta encendido y los medios externos de los lugares se montan encendido. El mountpoint es un diccionario del Python que contiene una característica del **uri** (que sea la trayectoria al punto de montaje) y una característica de la **id** (que es un nombre dado al punto de montaje). Cada entrada de diario tiene una cualidad de los meta datos nombrada **mountpoint**. **El uri del** diario será el único que comienza con **/home**, así que si limitamos la búsqueda para meter los objetos en diario donde la **id de** ese mountpoint iguala los meta datos del **mountpoint** en los objetos del diario que podemos enumerar fácilmente solamente objetos del diario.
- En el azúcar .84 y más adelante el método de *datastore.mounts()* todavía existe pero no le dice cualquier cosa sobre mountpoints. Sin embargo, usted puede utilizar el código arriba para ver si hay solamente un mountpoint y si su identificación es 1. Si es usted sabe que usted se está ocupando del datastore reescrito de .84 y más adelante. La otra diferencia es que los objetos del diario tienen no más meta datos con una llave del **mountpoint**. Si usted utiliza el código sobre él explicará esta diferencia y trabajo con cualquier versión del azúcar.

¿Qué si usted quiere el comportamiento del azúcar .82, enumerar entradas de diario y archivos del USB como objetos del diario, en .82 y .84 y para arriba? Quise hacer eso para las **View Slides** y terminé encima de usar este código:

```
    def load_journal_table(self):
        ds_objects, num_objects = datastore.find({'mime_type':['image/jpeg',
            'image/gif', 'image/tiff',  'image/png']},
            properties=['uid', 'title', 'mime_type'])
        self.ls_right.clear()
        for i in xrange (0, num_objects, 1):
            iter = self.ls_right.append()
            title = ds_objects[i].metadata['title']
            mime_type = ds_objects[i].metadata['mime_type']
            if mime_type == 'image/jpeg' and not title.endswith('.jpg')\
                and not title.endswith('.jpeg')  \
                and not title.endswith('.JPG')  and not title.endswith('.JPEG') :
                title = title + '.jpg'
            if mime_type == 'image/png' and not title.endswith('.png') \
                and not title.endswith('.PNG'):
                title = title + '.png'
```

```
            if mime_type == 'image/gif' and not title.endswith('.gif')\
                and not title.endswith('.GIF'):
                title = title + '.gif'
            if mime_type == 'image/tiff' and not title.endswith('.tiff')\
                and not title.endswith('.TIFF'):
                title = title + '.tiff'
            self.ls_right.set(iter, COLUMN_IMAGE, title)
            jobject_wrapper = JobjectWrapper()
            jobject_wrapper.set_jobject(ds_objects[i])
            self.ls_right.set(iter,  COLUMN_PATH,  jobject_wrapper)

        valid_endings = ('.jpg',  '.jpeg', '.JPEG',  '.JPG', '.gif',
            '.GIF', '.tiff', '.TIFF', '.png', '.PNG')
        ds_mounts = datastore.mounts()
        if len(ds_mounts) == 1 and ds_mounts[0]['id'] == 1:
            # datastore.mounts() is stubbed out, we're running .84 or better
            for dirname,  dirnames,  filenames in os.walk('/media'):
                if '.olpc.store' in dirnames:
                    dirnames.remove('.olpc.store')
                    # don't visit .olpc.store directories
                for filename in filenames:
                    if filename.endswith(valid_endings):
                        iter = self.ls_right.append()
                        jobject_wrapper = JobjectWrapper()
                        jobject_wrapper.set_file_path(os.path.join(dirname,
                            filename))
                        self.ls_right.set(iter,  COLUMN_IMAGE,  filename)
                        self.ls_right.set(iter,  COLUMN_PATH,  jobject_wrapper)

        self.ls_right.set_sort_column_id(COLUMN_IMAGE,  gtk.SORT_ASCENDING)
```

En este caso utilizo el método de *datastore.mounts()* para imaginar qué versión del datastore tengo y entonces si estoy funcionando .84 y más adelante utilizo *os.walk()* para crear una lista plana de todos los archivos en todos los directorios encontrados bajo directorio **/media** (que es adonde las tarjetas del USB y del SD se montan siempre). No puedo hacer estos archivos en los directorios, pero qué puedo hacer es hacer una clase de la envoltura que pueda contener un objeto del diario o un archivo y utilizar esos objetos donde utilizaría normalmente objetos del diario. La clase de la envoltura parece esto:

```
class JobjectWrapper():
    def __init__(self):
        self.__jobject = None
        self.__file_path = None

    def set_jobject(self,  jobject):
        self.__jobject = jobject

    def set_file_path(self,  file_path):
        self.__file_path = file_path

    def get_file_path(self):
        if  self.__jobject != None:
            return self.__jobject.get_file_path()
        else:
            return self.__file_path
```

USANDO ENTRADAS DE DIARIO

Cuando usted está listo para leer un archivo almacenado en un objeto del diario que usted puede utilizar el método del *get_file_path() del* objeto del diario para conseguir una trayectoria del archivo y para abrirla para la lectura, como esto:

```
        fname = jobject.get_file_path ()
```

Una palabra de la precaución: sea consciente que esta trayectoria no existe hasta que usted llame el *get_file_path ()* y no existirá de largo después. Con el diario usted trabaja con las copias de los archivos en el diario, no las originales. Por esa razón usted no quiere almacenar el valor de vuelta del *get_file_path ()* para el uso posterior porque más adelante puede no ser válido. En lugar, almacene el objeto sí mismo del diario

y llame la derecha del método antes de que usted necesite la trayectoria.

Las entradas de los meta datos para los objetos del diario contienen secuencias y trabajan generalmente la manera que usted esperaría, con una excepción, cuál es la **preview**.

```
    def create_preview(self,  object_id):
        jobject = datastore.get(object_id)

        if jobject.metadata.has_key('preview'):
            preview = jobject.metadata['preview']
            if preview is None or preview == '' or preview == 'None':
                if jobject.metadata['mime_type'] .startswith('image/'):
                    filename = jobject.get_file_path()
                    self.show_image(filename)
                    return
                if jobject.metadata['mime_type']  == 'application/x-cbz':
                    filename = jobject.get_file_path()
                    fname = self.extract_image(filename)
                    self.show_image(fname)
                    os.remove(fname)
                    return

        if jobject.metadata.has_key('preview') and \
                len(jobject.metadata['preview']) > 4:

            if jobject.metadata['preview'][1:4] == 'PNG':
                preview_data = jobject.metadata['preview']
            else:
                import base64
                preview_data = base64.b64decode(jobject.metadata['preview'])

            loader = gtk.gdk.PixbufLoader()
            loader.write(preview_data)
            scaled_buf = loader.get_pixbuf()
            loader.close()
            self.image.set_from_pixbuf(scaled_buf)
            self.image.show()
        else:
            self.image.clear()
            self.image.show()
```

La cualidad de los meta datos de la **preview** es diferente de dos maneras:

- Debemos nunca pedir la **preview** mientras que los meta datos que se volverán en nuestra lista de diario se oponen. Necesitaremos conseguir una copia completa del objeto del diario para conseguirlo. Puesto que tenemos ya un objeto del diario podemos conseguir el completo metemos el objeto en diario consiguiendo su **object id** entonces que piden un nuevo copiamos del datastore usando la identificación
- La imagen de la inspección previo es un objeto **binario** (**dbus.ByteArray**) pero en las versiones del azúcar más viejas de .82 él serán almacenados como secuencia de texto. Para lograr esto es la **base 64 codificada**.

El código que usted utilizaría para conseguir una copia completa de los parecer de un objeto del diario esto:

```
        object_id = jobject.object_id
        jobject = datastore.get(object_id)
```

Ahora para una explicación de la codificación de la base 64. Usted ha oído probablemente que las computadoras utilizan el sistema de numeración bajo dos, en el cual los únicos dígitos usados son 1 y 0. Una unidad de almacenaje de datos que puede llevar a cabo un cero o el se llama un **bit**. Las computadoras necesitan almacenar la información además de números, así que acomodar esto agrupamos pedacitos en grupos de 8 (generalmente) y llaman estos grupos los **bytes**. Si usted utiliza solamente 7 de los 8 pedacitos en un octeto usted puede almacenar una letra del alfabeto romano, un signo de puntuación, o un solo dígito, cosas más como lengüetas y caracteres de avance de línea. Cualquier archivo que se pueda crear usando solamente 7 pedacitos fuera de los 8 se llama un **archivo de texto**. Todo que necesita los 8 pedacitos de

cada octeto hacer, incluyendo programas de computadora, películas, la música, y los cuadros de Jessica Alba es un **binary**. En versiones del azúcar antes de que .82 meta dato del objeto del diario pueda almacenar solamente secuencias de texto. Necesitamos de alguna manera representar bytes de 8 bits en 7 pedacitos. Hacemos esto agrupando los octetos juntos en una colección más grande de pedacitos y entonces partiendo se retiran en los grupos de 7 pedacitos. El Python tiene el paquete **base64** para hacer esto para nosotros.

La codificación de la base 64 es realmente una técnica bastante común. Si usted ha enviado nunca un email con un archivo atado el archivo era la base 64 codificada.

El código antedicho tiene unas par de maneras de crear una imagen de la inspección previo. Si los meta datos de la inspección previo contienen una imagen del png se carga en un pixbuf y se exhibe. Si no hay meta datos de la inspección previo sino el tipo del MIME está para un archivo de imagen o un archivo de cierre relámpago del cómic que creamos la inspección previo de la entrada de diario sí mismo.

Los cheques de código los primeros tres caracteres de los meta datos de la inspección previo para ver si son "png". Si es así el archivo es una imagen de los **Portable Network Graphics** almacenada como binario y no necesita ser convertido de la codificación de la base 64, si no hace.

PUESTA AL DÍA DE UN OBJETO DEL DIARIO

El código para poner al día un objeto del diario parece esto:

```
    def update_entry(self):
        needs_update = False

        if self.selected_journal_entry is None:
            return

        object_id = self.selected_journal_entry.object_id
        jobject = datastore.get(object_id)

        old_title = jobject.metadata.get('title', None)
        if old_title != self.title_entry.props.text:
            jobject.metadata['title'] = self.title_entry.props.text
            jobject.metadata['title_set_by_user'] = '1'
            needs_update = True

        old_tags = jobject.metadata.get('tags', None)
        new_tags = self.tags_textview.props.buffer.props.text
        if old_tags != new_tags:
            jobject.metadata['tags'] = new_tags
            needs_update = True

        old_description = jobject.metadata.get('description', None)
        new_description = self.description_textview.props.buffer.props.text
        if old_description != new_description:
            jobject.metadata['description'] = new_description
            needs_update = True

        if needs_update:
            datastore.write(jobject, update_mtime=False,
                            reply_handler=self.datastore_write_cb,
                            error_handler=self.datastore_write_error_cb)
        self.btn_save.props.sensitive = False

    def datastore_write_cb(self):
        pass

    def datastore_write_error_cb(self, error):
        logging.error('sugarcommander.datastore_write_error_cb: %r' % error)
```

SUPRESIÓN DE UNA ENTRADA DE DIARIO

El código para suprimir una entrada de diario es éste:

```
    def delete_button_press_event_cb(self, entry, event):
        datastore.delete(self.selected_journal_entry.object_id)
```

CONSEGUIR SERVICIOS REPETIDOS DEL DIARIO USANDO EL D-BUS

En el capítulo en la **Fabricación las Actividades compartidas** vimos cómo las llamadas del D-Bus enviadas sobre los tubos de la telepatía se podrían utilizar para enviar mensajes de una actividad que funcionaba en una computadora a la misma actividad que funcionaba en una diversa computadora. El D-Bus no se utiliza normalmente que manera; se utiliza típicamente para enviar mensajes entre los programas que funcionan en la misma computadora.

Por ejemplo, si usted está trabajando con el diario usted puede conseguir servicios repetidos siempre que el diario sea actualizado. Usted consigue los servicios repetidos si la actualización fuera hecha por su actividad o a otra parte. Si es importante que su actividad sepa cuándo se ha puesto al día el diario usted querrá conseguir estos servicios repetidos.

La primera cosa que usted necesita hacer es definir algunos constantes e importar el paquete del D-Bus:

```
DS_DBUS_SERVICE = “org.laptop.sugar.DataStore”
DS_DBUS_INTERFACE = “org.laptop.sugar.DataStore”
DS_DBUS_PATH = “/org/laptop/sugar/DataStore”
import dbus
```

Después, en su código puesto método del __init__() a conectar con las señales y para hacer los servicios repetidos:

```
        bus = dbus.SessionBus()
        remote_object = bus.get_object(DS_DBUS_SERVICE, DS_DBUS_PATH)
        _datastore = dbus.Interface(remote_object, DS_DBUS_INTERFACE)
        _datastore.connect_to_signal('Created', self._datastore_created_cb)
        _datastore.connect_to_signal('Updated', self._datastore_updated_cb)
        _datastore.connect_to_signal('Deleted', self._datastore_deleted_cb)
```

Los métodos que eran funcionados por los servicios repetidos pudieron mirar algo similar:

```
    def datastore_created_cb(self, uid):
        new_jobject = datastore.get(uid)
        iter = self.ls_journal.append()
        title = new_jobject.metadata['title']
        self.ls_journal.set(iter, COLUMN_TITLE, title)
        mime = new_jobject.metadata['mime_type']
        self.ls_journal.set(iter, COLUMN_MIME, mime)
        self.ls_journal.set(iter, COLUMN_JOBJECT, new_jobject)

    def datastore_updated_cb(self,  uid):
        new_jobject = datastore.get(uid)
        iter = self.ls_journal.get_iter_first()
        for row in self.ls_journal:
            jobject = row[COLUMN_JOBJECT]
            if jobject.object_id == uid:
                title = new_jobject.metadata['title']
                self.ls_journal.set_value(iter, COLUMN_TITLE, title)
                break
            iter = self.ls_journal.iter_next(iter)
        object_id = self.selected_journal_entry.object_id
        if object_id == uid:
            self.set_form_fields(new_jobject)

    def datastore_deleted_cb(self,  uid):
        save_path = self.selected_path
        iter = self.ls_journal.get_iter_first()
        for row in self.ls_journal:
```

```
            jobject = row[COLUMN_JOBJECT]
            if jobject.object_id == uid:
                self.ls_journal.remove(iter)
                break
            iter = self.ls_journal.iter_next(iter)

        try:
            self.selection_journal.select_path(save_path)
            self.tv_journal.grab_focus()
        except:
            self.title_entry.set_text('')
            description_textbuffer = self.description_textview.get_buffer()
            description_textbuffer.set_text('')
            tags_textbuffer = self.tags_textview.get_buffer()
            tags_textbuffer.set_text('')
            self.btn_save.props.sensitive = False
            self.btn_delete.props.sensitive = False
            self.image.clear()
            self.image.show()
```

El uid pasó a cada método de servicio repetido es la **object id** objeto del diario se ha agregado, se ha puesto al día, o se ha suprimido que. Si una entrada se agrega al diario que consigo el objeto del diario del datastore por su uid, después agregúelo al gtk.ListStore para el gtk.TreeModel estoy utilizando para enumerar hacia fuera entradas de diario. Si una entrada es actualizada o suprimido necesito explicar la posibilidad que la entrada de diario yo es visión o el corregir pudo haber sido puesto al día o haber sido quitado. Utilizo el uid para imaginar que que reman en el gtk.ListStore necesita ser quitado o ser modificado colocando a través de las entradas en el gtk.ListStore que busca un fósforo.

Ahora usted sabe que todo que usted necesitará nunca saber para trabajar con el diario.

18. CREACIÓN DE ACTIVIDADES USANDO PYGAME

INTRODUCCIÓN

PyGame y **PyGTK** son dos maneras diferentes de hacer programas en Python con un interfaz gráfico. Normalmente usted no utilizaría ambos en el mismo programa. Cada uno de ellos tiene su propia manera de crear una ventana y cada uno tiene su propia manera de manejar eventos.

La clase base Actividad que hemos estado usando es una extensión de la clase Windows (Ventana) de PyGTK y utiliza el manejador de eventos de PyGTK. Las barras de herramientas que todas las Actividades usan son componentes de PyGTK. En fin, cualquier Actividad escrita en Python debe utilizar PyGTK. Poner un programa de PyGame en el medio de un programa de PyGTK es un poco como poner un barco a escala en una botella. Afortunadamente hay cierto código de Python llamado **SugarGame** que permita hacer eso.

Antes de que imaginemos cómo lograremos entrar en la botella, vamos a echar una una mirada a nuestro barco.

CREACIÓN DE UN JUEGO INDEPENDIENTE USANDO PYGAME

Como usted puede esperar, es una buena idea hacer un juego de Python independiente usando PyGame antes de que usted haga una Actividad con él. No soy un desarrollador experto en Python, pero usando el tutorial *Desarrollo Rápido de Juegos usando Pytohn* de Richard Jones en este URL:

http://richard.cgpublisher.com/product/pub.84/prod.11

pude armar un juego modesto en alrededor de un día. Habría sido más pronto pero los ejemplos del tutorial tenían bugs y tuve que invertir un tiempo considerable usando **The GIMP** para crear los archivos de imagen para los personajes del juego.

Los Sprites son las pequeñas imágenes, animadas a menudo, que representan objetos en un juego. Tienen generalmente un fondo transparente así que pueden ser dibujados encima de una imagen de fondo. Utilicé el formato del **png** para mis archivos del sprite porque soportan un **canal alfa** (otro término que indica que la parte de la imagen es transparente).

PyGame tiene código para mostrar las imágenes de fondo, para crear sprites y para moverlos en frente del fondo, y para detectar cuando chocan los sprites el uno con el otro y hacer algo cuando esto sucede. Ésta es la base para hacer muchos juegos en 2D. Hay muchos juegos escritos con PyGame que se podrían adaptar fácilmente para ser Actividades de Sugar.

Mi juego es similar al juego del coche del tutorial, pero en vez de un coche tengo un aeroplano. El aeroplano es el *Demoiselle* creado por Alberto Santos-Dumont en 1909. En vez de tener "cojines" para chocar, tengo cuatro estudiantes de Otto Lilienthal sobrevolando inmóviles en sus planeadores. Los planeadores caen hacia abajo cuando Santos-Dumont choca con ellos. Los controles usados para el juego también han sido modificados. Utilizo las teclas Más y Menos en el teclado principal y el teclado numérico, más las teclas 9 y 3 del teclado numérico, para abrir y para cerrar la válvula reguladora y las teclas hacia arriba y hacia abajo en el teclado principal y el teclado numérico para mover la palanca de mando hacia adelante y hacia atrás. Usar las teclas del teclado numérico es útil por un par de razones. Primero, algunas versiones del **sugar-emulator** (emulador de Sugar) no reconocen las teclas de flecha en el teclado principal. En segundo lugar,

las teclas de flecha en el teclado numérico corresponden al controlador de juegos en el ordenador portátil XO, y las teclas que no son flechas en el telclado numérico corresponden con los otros botones en la pantalla del ordenador portátil XO. Estos botones se pueden utilizar para jugar al juego cuando el XO está en modo de la tableta.

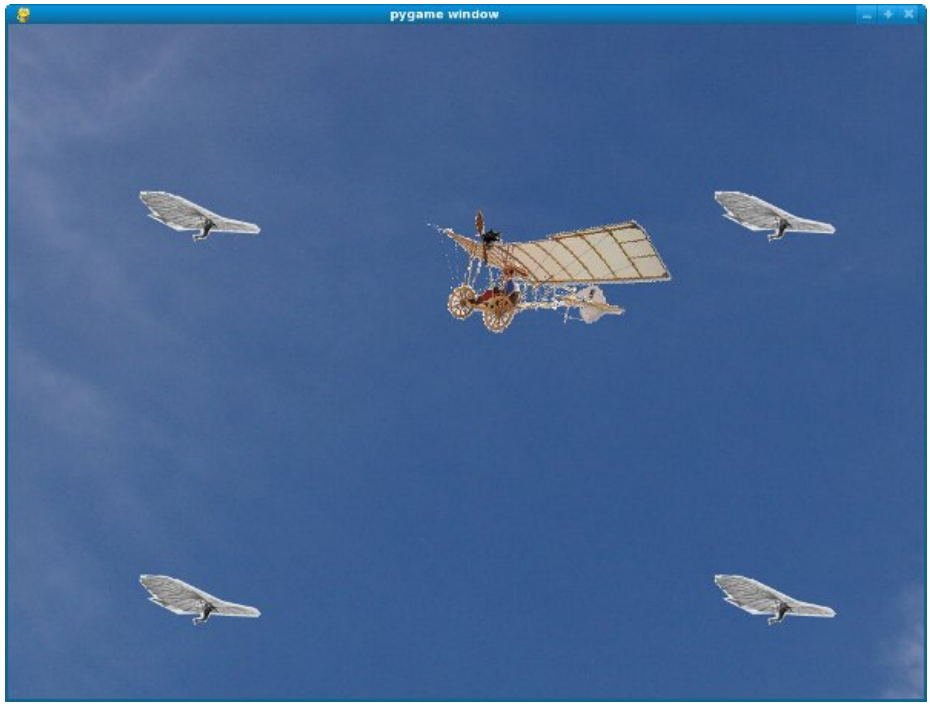
Como simulador de vuelo no es mucho, pero si puede demostrar por lo menos algunas de las cosas que PyGame puede hacer. Aquí está el código para el juego, que estoy llamando **Demoiselle**:

```
#! /usr/bin/env python
import pygame
import math
import sys

class Demoiselle:
    "This is a simple demonstration of using PyGame \
    sprites and collision detection."
    def __init__(self):
        self.background = pygame.image.load('sky.jpg')
        self.screen = pygame.display.get_surface()
        self.screen.blit(self.background, (0, 0))
        self.clock = pygame.time.Clock()
        self.running = True

        gliders = [
            GliderSprite((200, 200)),
            GliderSprite((800, 200)),
            GliderSprite((200, 600)),
            GliderSprite((800, 600)),
        ]
        self. glider_group = pygame.sprite.RenderPlain(gliders)

    def run(self):
        "This method processes PyGame messages"
        rect = self.screen.get_rect()
        airplane = AirplaneSprite('demoiselle.png', rect.center)
        airplane_sprite = pygame.sprite.RenderPlain(airplane)

        while self.running:
            self.clock.tick(30)

            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.running = False
                    return
                elif event.type == pygame.VIDEORESIZE:
                    pygame.display.set_mode(event.size,  pygame.RESIZABLE)
                    self.screen.blit(self.background, (0, 0))

                if not hasattr(event, 'key'):
                    continue
                down = event.type == pygame.KEYDOWN
                if event.key == pygame.K_DOWN or \
                    event.key == pygame.K_KP2:
                    airplane.joystick_back = down * 5
                elif event.key == pygame.K_UP or \
                    event.key == pygame.K_KP8:
                    airplane.joystick_forward = down * -5
                elif event.key == pygame.K_EQUALS or \
                    event.key == pygame.K_KP_PLUS or \
                    event.key == pygame.K_KP9:
                    airplane.throttle_up = down * 2
                elif event.key == pygame.K_MINUS or \
                    event.key == pygame.K_KP_MINUS or \
                    event.key == pygame.K_KP3:
                    airplane.throttle_down = down * -2

            self.glider_group.clear(self.screen, self.background)
            airplane_sprite.clear(self.screen, self.background)
            collisions = pygame.sprite.spritecollide(airplane, \
                self.glider_group,  False)
            self.glider_group.update(collisions)
            self.glider_group.draw(self.screen)
            airplane_sprite.update()
```

```
            airplane_sprite.draw(self.screen)
            pygame.display.flip()

class AirplaneSprite(pygame.sprite.Sprite):
    "This class represents an airplane, the Demoiselle \
    created by Alberto Santos-Dumont"
    MAX_FORWARD_SPEED = 10
    MIN_FORWARD_SPEED = 1
    ACCELERATION = 2
    TURN_SPEED = 5
    def __init__(self, image, position):
        pygame.sprite.Sprite.__init__(self)
        self.src_image = pygame.image.load(image)
        self.rect = pygame.Rect(self.src_image.get_rect())
        self.position = position
        self.rect.center = self.position
        self.speed = 1
        self.direction = 0
        self.joystick_back = self.joystick_forward = \
            self.throttle_down = self.throttle_up = 0

    def update(self):
        "This method redraws the airplane in response\
        to events."
        self.speed += (self.throttle_up + self.throttle_down)
        if self.speed > self.MAX_FORWARD_SPEED:
            self.speed = self.MAX_FORWARD_SPEED
        if self.speed < self.MIN_FORWARD_SPEED:
            self.speed = self.MIN_FORWARD_SPEED
        self.direction += (self.joystick_forward + self.joystick_back)
        x_coord, y_coord = self.position
        rad = self.direction * math.pi / 180
        x_coord += -self.speed * math.cos(rad)
        y_coord += -self.speed * math.sin(rad)
        screen = pygame.display.get_surface()
        if y_coord < 0:
            y_coord = screen.get_height()

        if x_coord < 0:
            x_coord = screen.get_width()

        if x_coord > screen.get_width():
            x_coord = 0

        if y_coord > screen.get_height():
            y_coord = 0
        self.position = (x_coord, y_coord)
        self.image = pygame.transform.rotate(self.src_image, -self.direction)
        self.rect = self.image.get_rect()
        self.rect.center = self.position

class GliderSprite(pygame.sprite.Sprite):
    "This class represents an individual hang glider as developed\
    by Otto Lilienthal."
    def __init__(self, position):
        pygame.sprite.Sprite.__init__(self)
        self.normal = pygame.image.load('glider_normal.png')
        self.rect = pygame.Rect(self.normal.get_rect())
        self.rect.center = position
        self.image = self.normal
        self.hit = pygame.image.load('glider_hit.png')
    def update(self, hit_list):
        "This method redraws the glider when it collides\
        with the airplane and when it is no longer \
        colliding with the airplane."
        if self in hit_list:
            self.image = self.hit
        else:
            self.image = self.normal

def main():
    "This function is called when the game is run from the command line"
    pygame.init()
    pygame.display.set_mode((0, 0), pygame.RESIZABLE)
    game = Demoiselle()
```

```
    game.run()
    sys.exit(0)

if __name__ == '__main__':
    main()
```

Y aquí está el juego en acción:



Usted encontrará el código para este juego en el archivo **demoiselle.py** en el proyecto de los ejemplos del libro en Git.

INTRODUCCIÓN DE SUGARGAME

SugarGame no es parte de azúcar apropiada. Si usted quiere utilizarlo usted necesitará incluir el código del Python para SugarGame dentro de su paquete de la actividad. He incluido la versión de SugarGame que estoy utilizando en el proyecto de los ejemplos del libro en el directorio del **sugargame**, pero cuando usted hace sus propios juegos usted querré estar seguro y conseguir el último código para incluir. Usted puede hacer eso transfiriendo el proyecto de Gitorious usando estos comandos:

```
mkdir sugargame
cd sugargame
git clone git://git.sugarlabs.org/sugargame/mainline.git
```

Usted verá dos sub-directórios en este proyecto: **sugargame** y **test**, más un archivo de **README.txt** que contiene la información sobre usar el sugargame en sus propias Actividades. El directorio de la prueba contiene un programa simple de PyGame que pueda ser independiente funcionado o como actividad. El programa independiente está en el archivo nombrado **TestGame.py**. La actividad, que es una clase de envoltura alrededor de la versión independiente, está en el archivo **TestActivity.py**.

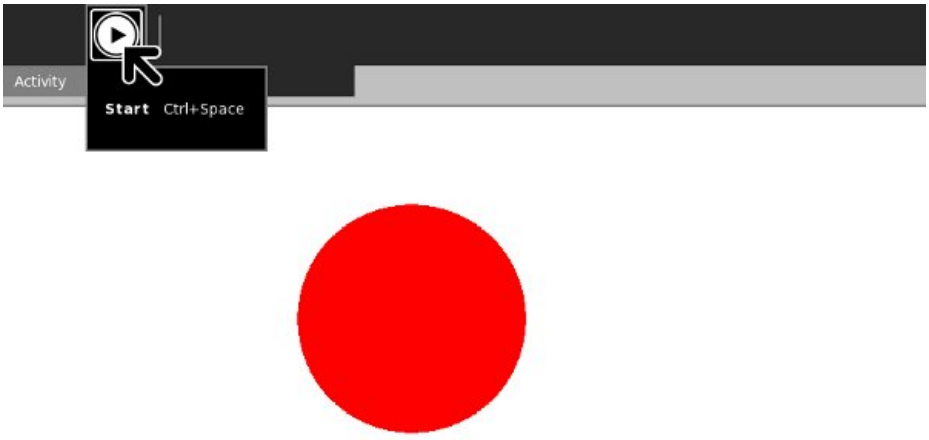
Si usted funciona **TestGame.py de la** línea de comando usted verá que exhibe una bola que despide en un fondo blanco. Para intentar funcionar con la versión de la actividad que usted necesitará funcionar con

```
./setup.py dev
```

de la línea de comando primero. No podía conseguir la actividad para trabajar debajo de azúcar-emulador hasta que realizara dos cambios a él:

- Hice una copia del directorio del **sugargame** dentro del directorio de la **test**.
- Quité la línea lectura "**sys.path.append(". .") # Import sugargame package from top directory**. "de **TestActivity.py**. Esta línea se supone obviamente para ayudar al programa a encontrar el directorio del **sugargame** en el proyecto pero no trabajó en Fedora 10. Su propia experiencia puede ser diferente.

La actividad parece esto:



La barra de herramientas de **PyGame** tiene un solo botón que le deje hacer que la bola que despide se detiene brevemente y reasume el despedir.

FABRICACIÓN DE UNA ACTIVIDAD DEL SUGAR FUERA DE UN PROGRAMA DE PYGAME

Ahora es hora de poner nuestra nave en esa botella. La primera cosa que necesitamos hacer es hacer una copia del directorio del **sugargame del** proyecto de SugarGame en el directorio del mainline de nuestro propio proyecto.

El archivo de **README.txt** en el proyecto de SugarGame vale el leer. Nos dice hacer una actividad basada en el ejemplo de **TestActivity.py** en el proyecto de SugarGame. Ésta será nuestra botella. Aquí está el código para el mío, que se nombra **DemoiselleActivity.py**:

```
# DemoiselleActivity.py

# Copyright (C) 2010  James D. Simmons
#
# This program is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
```

```
# the Free Software Foundation; either version 2 of the License, or
# (at your option) any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License along
# with this program; if not, write to the Free Software Foundation, Inc.,
# 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA
#

from gettext import gettext as _

import gtk
import pygame
from sugar.activity import activity
from sugar.graphics.toolbutton import ToolButton
import gobject
import sugargame.canvas
import demoiselle2

class DemoiselleActivity(activity.Activity):
    def __init__(self, handle):
        super(DemoiselleActivity, self).__init__(handle)

        # Build the activity toolbar.
        self.build_toolbar()

        # Create the game instance.
        self.game = demoiselle2.Demoiselle()

        # Build the Pygame canvas.
        self._pygamecanvas = sugargame.canvas.PygameCanvas(self)
        # Note that set_canvas implicitly calls read_file when
        # resuming from the Journal.
        self.set_canvas(self._pygamecanvas)
        self.score = ''

        # Start the game running.
        self._pygamecanvas.run_pygame(self.game.run)

    def build_toolbar(self):
        toolbox = activity.ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False
        activity_toolbar.share.props.visible = False

        self.view_toolbar = ViewToolbar()
        toolbox.add_toolbar(_('View'), self.view_toolbar)
        self.view_toolbar.connect('go-fullscreen',
                self.view_toolbar_go_fullscreen_cb)
        self.view_toolbar.show()

        toolbox.show()
        self.set_toolbox(toolbox)

    def view_toolbar_go_fullscreen_cb(self, view_toolbar):
        self.fullscreen()

    def read_file(self, file_path):
        score_file = open(file_path, "r")
        while score_file:
            self.score = score_file.readline()
            self.game.set_score(int(self.score))
        score_file.close()

    def write_file(self, file_path):
        score = self.game.get_score()
        f = open(file_path, 'wb')
        try:
            f.write(str(score))
        finally:
            f.close
```

```
class ViewToolbar(gtk.Toolbar):
    __gtype_name__ = 'ViewToolbar'

    __gsignals__ = {
        'needs-update-size': (gobject.SIGNAL_RUN_FIRST,
                              gobject.TYPE_NONE,
                              ([])),
        'go-fullscreen': (gobject.SIGNAL_RUN_FIRST,
                          gobject.TYPE_NONE,
                          ([]))
    }

    def __init__(self):
        gtk.Toolbar.__init__(self)
        self.fullscreen = ToolButton('view-fullscreen')
        self.fullscreen.set_tooltip(_('Fullscreen'))
        self.fullscreen.connect('clicked', self.fullscreen_cb)
        self.insert(self.fullscreen, -1)
        self.fullscreen.show()

    def fullscreen_cb(self, button):
        self.emit('go-fullscreen')
```

Esto es un fancier del pedacito que **TestActivity.py**. Decidía que mi juego no necesitó realmente ser detenido brevemente y ser reasumido, así que substituí la barra de herramientas de **PyGame** por una barra de herramientas de la **visión** que deja al usuario ocultar la barra de herramientas cuando no es necesaria. Utilizo *() los* métodos *read_file ()* y *write_file* para ahorrar y para restaurar la cuenta de juego. (Esto se falsifica realmente, porque nunca pongo en cualquier lógica que anota en el juego). También oculto los controles de la **subsistencia** y de la **parte** en la barra de herramientas principal.

Pues usted esperaría, conseguir una nave en una botella requiere la nave ser modificada. Aquí está **demoiselle2.py**, que tiene las modificaciones:

```
#! /usr/bin/env python
import pygame
import gtk
import math
import sys

class Demoiselle:
    "This is a simple demonstration of using PyGame \
    sprites and collision detection."
    def __init__(self):
        self.clock = pygame.time.Clock()
        self.running = True
        self.background = pygame.image.load('sky.jpg')

    def get_score(self):
        return '99'

    def run(self):
        "This method processes PyGame messages"

        screen = pygame.display.get_surface()
        screen.blit(self.background, (0, 0))

        gliders = [
            GliderSprite((200, 200)),
            GliderSprite((800, 200)),
            GliderSprite((200, 600)),
            GliderSprite((800, 600)),
        ]
        glider_group = pygame.sprite.RenderPlain(gliders)

        rect = screen.get_rect()
        airplane = AirplaneSprite('demoiselle.png', rect.center)
        airplane_sprite = pygame.sprite.RenderPlain(airplane)

        while self.running:
            self.clock.tick(30)
```

```
            # Pump GTK messages.
            while gtk.events_pending():
                gtk.main_iteration()

            # Pump PyGame messages.
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.running = False
                    return
                elif event.type == pygame.VIDEORESIZE:
                    pygame.display.set_mode(event.size,  pygame.RESIZABLE)
                    screen.blit(self.background, (0, 0))

                if not hasattr(event, 'key'):
                    continue
                down = event.type == pygame.KEYDOWN
                if event.key == pygame.K_DOWN or \
                    event.key == pygame.K_KP2:
                    airplane.joystick_back = down * 5
                elif event.key == pygame.K_UP or \
                    event.key == pygame.K_KP8:
                    airplane.joystick_forward = down * -5
                elif event.key == pygame.K_EQUALS or \
                    event.key == pygame.K_KP_PLUS or \
                    event.key == pygame.K_KP9:
                    airplane.throttle_up = down * 2
                elif event.key == pygame.K_MINUS or \
                    event.key == pygame.K_KP_MINUS or \
                    event.key == pygame.K_KP3:
                    airplane.throttle_down = down * -2

            glider_group.clear(screen, self.background)
            airplane_sprite.clear(screen, self.background)
            collisions = pygame.sprite.spritecollide(airplane, \
                                                     glider_group,  False)
            glider_group.update(collisions)
            glider_group.draw(screen)
            airplane_sprite.update()
            airplane_sprite.draw(screen)
            pygame.display.flip()

class AirplaneSprite(pygame.sprite.Sprite):
    "This class represents an airplane, the Demoiselle \
    created by Alberto Santos-Dumont"
    MAX_FORWARD_SPEED = 10
    MIN_FORWARD_SPEED = 1
    ACCELERATION = 2
    TURN_SPEED = 5
    def __init__(self, image, position):
        pygame.sprite.Sprite.__init__(self)
        self.src_image = pygame.image.load(image)
        self.rect = pygame.Rect(self.src_image.get_rect())
        self.position = position
        self.rect.center = self.position
        self.speed = 1
        self.direction = 0
        self.joystick_back = self.joystick_forward = \
            self.throttle_down = self.throttle_up = 0

    def update(self):
        "This method redraws the airplane in response\
        to events."
        self.speed += (self.throttle_up + self.throttle_down)
        if self.speed > self.MAX_FORWARD_SPEED:
            self.speed = self.MAX_FORWARD_SPEED
        if self.speed < self.MIN_FORWARD_SPEED:
            self.speed = self.MIN_FORWARD_SPEED
        self.direction += (self.joystick_forward + self.joystick_back)
        x_coord, y_coord = self.position
        rad = self.direction * math.pi / 180
        x_coord += -self.speed * math.cos(rad)
        y_coord += -self.speed * math.sin(rad)
        screen = pygame.display.get_surface()
        if y_coord < 0:
            y_coord = screen.get_height()
```

```
        if x_coord < 0:
            x_coord = screen.get_width()

        if x_coord > screen.get_width():
            x_coord = 0

        if y_coord > screen.get_height():
            y_coord = 0
        self.position = (x_coord, y_coord)
        self.image = pygame.transform.rotate(self.src_image, -self.direction)
        self.rect = self.image.get_rect()
        self.rect.center = self.position

class GliderSprite(pygame.sprite.Sprite):
    "This class represents an individual hang glider as developed\
    by Otto Lilienthal."
    def __init__(self, position):
        pygame.sprite.Sprite.__init__(self)
        self.normal = pygame.image.load('glider_normal.png')
        self.rect = pygame.Rect(self.normal.get_rect())
        self.rect.center = position
        self.image = self.normal
        self.hit = pygame.image.load('glider_hit.png')
    def update(self, hit_list):
        "This method redraws the glider when it collides\
        with the airplane and when it is no longer \
        colliding with the airplane."
        if self in hit_list:
            self.image = self.hit
        else:
            self.image = self.normal

def main():
    "This function is called when the game is run from the command line"
    pygame.init()
    pygame.display.set_mode((0, 0), pygame.RESIZABLE)
    game = Demoiselle()
    game.run()
    sys.exit(0)

if __name__ == '__main__':
    main()
```

¿Por qué no cargue **demoiselle.py** y **demoiselle2.py** en Eric y tarde algunos minutos para ver si usted puede imaginar qué cambió entre las dos versiones?

Asombrosamente poco es diferente. Agregué un cierto código al lazo principal de PyGame para comprobar para saber si hay acontecimientos de PyGTK y para ocuparme de ellos:

```
        while self.running:
            self.clock.tick(30)

            # Pump GTK messages.
              while gtk.events_pending():
                   gtk.main_iteration()

            # Pump PyGame messages.
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    self.running = False
                    return
                elif event.type == pygame.VIDEORESIZE:
                    pygame.display.set_mode(event.size,  pygame.RESIZABLE)
                    screen.blit(self.background, (0, 0))

                if not hasattr(event, 'key'):
                    continue
                down = event.type == pygame.KEYDOWN
                if event.key == pygame.K_DOWN or \
```

… continúe ocupándose de los acontecimientos de PyGame…

Esto tiene el efecto de hacer PyGame y la toma de PyGTK da vuelta a manejar acontecimientos. Si este código no fuera actuales acontecimientos de GTK fuera no hecho caso y usted no tendría ninguna manera de cerrar la actividad, oculta la barra de herramientas, el etc. Usted necesita agregar el **gtk de la importación** en la tapa del archivo así que estos métodos pueden ser encontrados.

Por supuesto también agregué los métodos para fijar y para volver cuentas:

```
def get_score(self):
   return self.score

def set_score(self, score):
   self.score = score
```

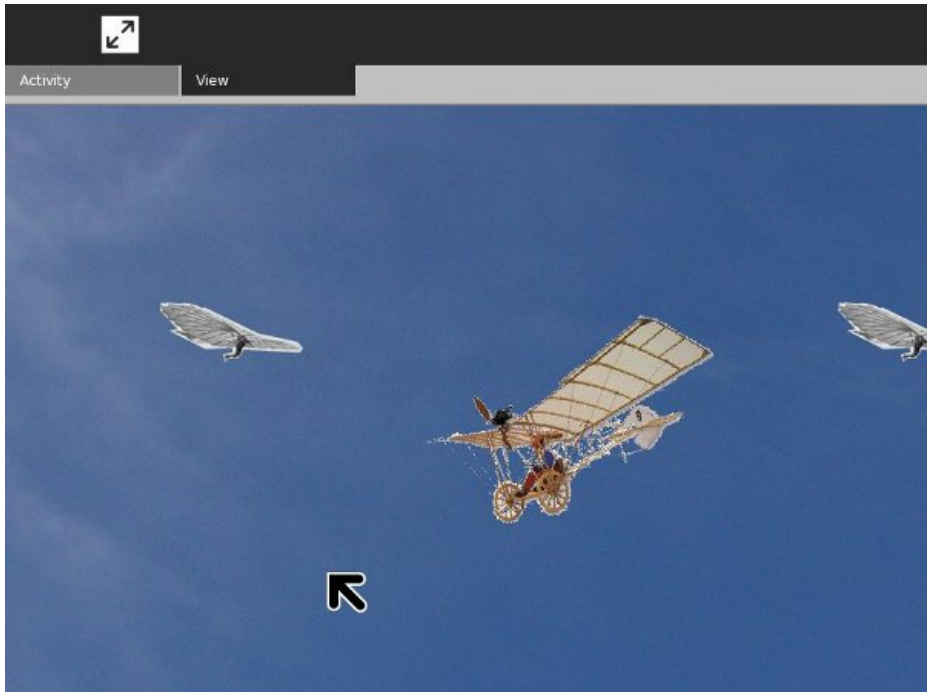
El cambio más grande está en el método del *__init () de la* clase del **Demoiselle**. Tenía originalmente código para exhibir la imagen de fondo en la pantalla:

```
def __init__(self):
    self.background = pygame.image.load('sky.jpg')
    self.screen = pygame.display.get_surface()
    self.screen.blit(self.background, (0, 0))
```

El problema con esto es que el sugargame va a crear un objeto especial de la lona de PyGTK para substituir la exhibición de PyGame y el código de DemoiselleActivity no ha hecho eso todavía, así que **self.screen** tendrá un valor de ningunos. La única manera de conseguir alrededor de eso es mover cualquier código que refiera a la **exhibición** fuera del método del *__init__() de la* clase y en el principio del método que contiene el lazo del acontecimiento. Esto puede dejarle con un método del *__init__()* que no haga poco o nada. Sobre la única cosa que usted querrá es código para crear variables de caso.

Nada que hemos hecho a **demoiselle2.py** evitará que sea funcionado como programa independiente del Python.

Para probar el **revelador de ./setup.py del** funcionamiento del juego dentro del directorio de **Making_Activities_Using_PyGame**. Cuando usted prueba la actividad debe parecer esto:

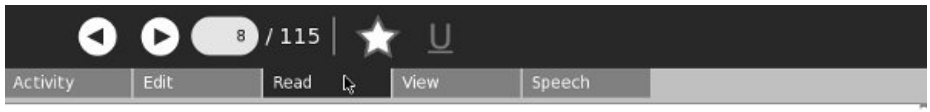


19. FABRICACIÓN DE NUEVAS BARRAS DE HERRAMIENTAS DEL ESTILO

INTRODUCCIÓN

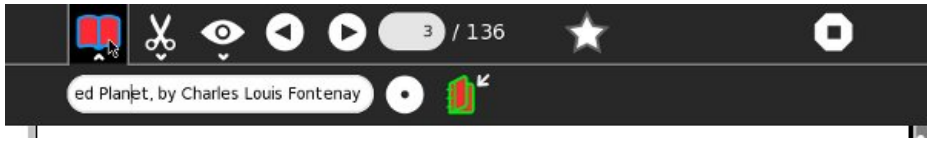
Dicen que "no hay barra de herramientas como una barra de herramientas vieja" y si sus usuarios no están usanando la versión muy última de Sugar ellos tienen razón. Las Actividades necesitarán usar las barras de herramientas originales del estilo durante un tiempo. Sin embargo, es posible hacer una actividad que apoye ambos y que sea lo que vamos a hacer en este capítulo.

Las nuevas barras de herramientas ocurrieron porque estaban problemas con las barras de herramientas viejas. Los usuarios de una Actividad tenían dificultades porque no sabían cómo parar una Actividad porque el botón **Close** está solamente en la barra de herramientas de la Actividad. Si la Actividad comienza en una barra de herramientas diferente, como muchas hacen, no es obvia que usted necesita cambiar a la barra de herramientas de la Actividad para parar la Actividad. El otro tema traído encima de era que las lengüetas para las barras de herramientas tomaron espacio de la pantalla que se podrían utilizar mejor a otra parte. Comparemos las barras de herramientas para las Actividades similares. Primero, la barra de herramientas del viejo estilo para **Read Etexts**::



Ahora compárela con la nueva barra de herramientas del estilo para la actividad **leída**:

Esto es más fino que la más vieja versión y el botón **cercano** es siempre visibles. Algunas funciones están en la barra de herramientas principal y otras se atan a las barras de herramientas que caen abajo cuando usted chasca encendido su icono. Primero, la nueva actividad cae abajo la barra de herramientas:



Después la barra de herramientas del **corregir**:

Finalmente, la barra de herramientas de la **visión**:

ADICIÓN DE NUEVAS BARRAS DE HERRAMIENTAS DEL ESTILO PARA READ ETEXTS II

Al trabajar en la original **lea la** actividad de **Etexts que** pedí prestado mucho código del interfaz utilizador de la original **leo** actividad y no veo ninguna razón para parar el hacer de eso ahora. Una complicación a hacer esto es que **leído** tiene algunas dependencias que eviten que la última versión de **leído** trabaje con más viejas versiones del azúcar, y que no es ninguna necesidad el ser el caso allí en absoluto de leído para apoyar barras de herramientas viejas y nuevas. **Read Etexts IV** no será tan afortunado; necesitará imaginar en el tiempo de pasada lo que un poco se apoya la barra de herramientas y uso eso.

Puedo probar la actividad con las barras de herramientas viejas y nuevas del estilo en la misma caja porque estoy funcionando Fedora 11, que tiene un ambiente instalado del azúcar que apoye las barras de herramientas viejas, más que he transferido y que funciono con el **sugar-jhbuild**, que apoya las nuevas barras de herramientas en su versión del Sugar.

Aquí está el código para **ReadEtextsActivity4.py**:

```
import os
import re
import logging
import time
import zipfile
import gtk
import pango
import dbus
import gobject
import telepathy
from sugar.activity import activity

from sugar.graphics.toolbutton import ToolButton

_NEW_TOOLBAR_SUPPORT = True
try:
    from sugar.graphics.toolbarbox import ToolbarBox
    from sugar.graphics.toolbarbox import ToolbarButton
    from sugar.activity.widgets import StopButton
    from toolbar import ViewToolbar
    from mybutton import MyActivityToolbarButton
except:
    _NEW_TOOLBAR_SUPPORT = False
    from toolbar import ReadToolbar,  ViewToolbar

from sugar.graphics.toggletoolbutton import ToggleToolButton
from sugar.graphics.menuitem import MenuItem

from sugar.graphics import style
from sugar import network
from sugar.datastore import datastore
from sugar.graphics.alert import NotifyAlert
from gettext import gettext as _

page=0
PAGE_SIZE = 45
TOOLBAR_READ = 2

logger = logging.getLogger('read-etexts2-activity')

class ReadHTTPRequestHandler(network.ChunkedGlibHTTPRequestHandler):
    """HTTP Request Handler for transferring document while collaborating.
```

```
    RequestHandler class that integrates with Glib mainloop. It writes
    the specified file to the client in chunks, returning control to the
    mainloop between chunks.

    """
    def translate_path(self, path):
        """Return the filepath to the shared document."""
        return self.server.filepath


class ReadHTTPServer(network.GlibTCPServer):
    """HTTP Server for transferring document while collaborating."""
    def __init__(self, server_address, filepath):
        """Set up the GlibTCPServer with the ReadHTTPRequestHandler.

        filepath -- path to shared document to be served.
        """
        self.filepath = filepath
        network.GlibTCPServer.__init__(self, server_address,
                                       ReadHTTPRequestHandler)


class ReadURLDownloader(network.GlibURLDownloader):
    """URLDownloader that provides content-length and content-type."""

    def get_content_length(self):
        """Return the content-length of the download."""
        if self._info is not None:
            return int(self._info.headers.get('Content-Length'))

    def get_content_type(self):
        """Return the content-type of the download."""
        if self._info is not None:
            return self._info.headers.get('Content-type')
        return None

READ_STREAM_SERVICE = 'read-etexts-activity-http'

class ReadEtextsActivity(activity.Activity):
    def __init__(self, handle):
        "The entry point to the Activity"
        global page
        activity.Activity.__init__(self, handle)

        self.fileserver = None
        self.object_id = handle.object_id

        if _NEW_TOOLBAR_SUPPORT:
            self.create_new_toolbar()
        else:
            self.create_old_toolbar()

        self.scrolled_window = gtk.ScrolledWindow()
        self.scrolled_window.set_policy(gtk.POLICY_NEVER, gtk.POLICY_AUTOMATIC)
        self.scrolled_window.props.shadow_type = gtk.SHADOW_NONE

        self.textview = gtk.TextView()
        self.textview.set_editable(False)
        self.textview.set_cursor_visible(False)
        self.textview.set_left_margin(50)
        self.textview.connect("key_press_event", self.keypress_cb)

        self.progressbar = gtk.ProgressBar()
        self.progressbar.set_orientation(gtk.PROGRESS_LEFT_TO_RIGHT)
        self.progressbar.set_fraction(0.0)

        self.scrolled_window.add(self.textview)
        self.textview.show()
        self.scrolled_window.show()

        vbox = gtk.VBox()
        vbox.pack_start(self.progressbar,  False,  False,  10)
        vbox.pack_start(self.scrolled_window)
        self.set_canvas(vbox)
```

```
        vbox.show()

        page = 0
        self.clipboard = gtk.Clipboard(display=gtk.gdk.display_get_default(), \
                                       selection="CLIPBOARD")
        self.textview.grab_focus()
        self.font_desc = pango.FontDescription("sans %d" % style.zoom(10))
        self.textview.modify_font(self.font_desc)

        buffer = self.textview.get_buffer()
        self.markset_id = buffer.connect("mark-set", self.mark_set_cb)

        self.unused_download_tubes = set()
        self.want_document = True
        self.download_content_length = 0
        self.download_content_type = None
        # Status of temp file used for write_file:
        self.tempfile = None
        self.close_requested = False
        self.connect("shared", self.shared_cb)

        self.is_received_document = False

        if self._shared_activity and handle.object_id == None:
            # We're joining, and we don't already have the document.
            if self.get_shared():
                # Already joined for some reason, just get the document
                self.joined_cb(self)
            else:
                # Wait for a successful join before trying to get the document
                self.connect("joined", self.joined_cb)

    def create_old_toolbar(self):
        toolbox = activity.ActivityToolbox(self)
        activity_toolbar = toolbox.get_activity_toolbar()
        activity_toolbar.keep.props.visible = False

        self.edit_toolbar = activity.EditToolbar()
        self.edit_toolbar.undo.props.visible = False
        self.edit_toolbar.redo.props.visible = False
        self.edit_toolbar.separator.props.visible = False
        self.edit_toolbar.copy.set_sensitive(False)
        self.edit_toolbar.copy.connect('clicked', self.edit_toolbar_copy_cb)
        self.edit_toolbar.paste.props.visible = False
        toolbox.add_toolbar(_('Edit'), self.edit_toolbar)
        self.edit_toolbar.show()

        self.read_toolbar = ReadToolbar()
        toolbox.add_toolbar(_('Read'), self.read_toolbar)
        self.read_toolbar.back.connect('clicked', self.go_back_cb)
        self.read_toolbar.forward.connect('clicked', self.go_forward_cb)
        self.read_toolbar.num_page_entry.connect('activate',  \
            self.num_page_entry_activate_cb)
        self.read_toolbar.show()

        self.view_toolbar = ViewToolbar()
        toolbox.add_toolbar(_('View'), self.view_toolbar)
        self.view_toolbar.connect('go-fullscreen', \
            self.view_toolbar_go_fullscreen_cb)
        self.view_toolbar.zoom_in.connect('clicked', self.zoom_in_cb)
        self.view_toolbar.zoom_out.connect('clicked', self.zoom_out_cb)
        self.view_toolbar.show()

        self.set_toolbox(toolbox)
        toolbox.show()
        self.toolbox.set_current_toolbar(TOOLBAR_READ)

    def create_new_toolbar(self):
        toolbar_box = ToolbarBox()

        activity_button = MyActivityToolbarButton(self)
        toolbar_box.toolbar.insert(activity_button, 0)
        activity_button.show()

        self.edit_toolbar = activity.EditToolbar()
```

```
self.edit_toolbar.undo.props.visible = False
self.edit_toolbar.redo.props.visible = False
self.edit_toolbar.separator.props.visible = False
self.edit_toolbar.copy.set_sensitive(False)
self.edit_toolbar.copy.connect('clicked', self.edit_toolbar_copy_cb)
self.edit_toolbar.paste.props.visible = False

edit_toolbar_button = ToolbarButton(
    page=self.edit_toolbar,
    icon_name='toolbar-edit')
self.edit_toolbar.show()
toolbar_box.toolbar.insert(edit_toolbar_button, -1)
edit_toolbar_button.show()

self.view_toolbar = ViewToolbar()
self.view_toolbar.connect('go-fullscreen', \
    self.view_toolbar_go_fullscreen_cb)
self.view_toolbar.zoom_in.connect('clicked', self.zoom_in_cb)
self.view_toolbar.zoom_out.connect('clicked', self.zoom_out_cb)
self.view_toolbar.show()
view_toolbar_button = ToolbarButton(
    page=self.view_toolbar,
    icon_name='toolbar-view')
toolbar_box.toolbar.insert(view_toolbar_button, -1)
view_toolbar_button.show()

self.back = ToolButton('go-previous')
self.back.set_tooltip(_('Back'))
self.back.props.sensitive = False
self.back.connect('clicked', self.go_back_cb)
toolbar_box.toolbar.insert(self.back, -1)
self.back.show()

self.forward = ToolButton('go-next')
self.forward.set_tooltip(_('Forward'))
self.forward.props.sensitive = False
self.forward.connect('clicked', self.go_forward_cb)
toolbar_box.toolbar.insert(self.forward, -1)
self.forward.show()

num_page_item = gtk.ToolItem()
self.num_page_entry = gtk.Entry()
self.num_page_entry.set_text('0')
self.num_page_entry.set_alignment(1)
self.num_page_entry.connect('insert-text',
                            self.__new_num_page_entry_insert_text_cb)
self.num_page_entry.connect('activate',
                            self.__new_num_page_entry_activate_cb)
self.num_page_entry.set_width_chars(4)
num_page_item.add(self.num_page_entry)
self.num_page_entry.show()
toolbar_box.toolbar.insert(num_page_item, -1)
num_page_item.show()

total_page_item = gtk.ToolItem()
self.total_page_label = gtk.Label()

label_attributes = pango.AttrList()
label_attributes.insert(pango.AttrSize(14000, 0, -1))
label_attributes.insert(pango.AttrForeground(65535, 65535,
                                             65535, 0, -1))
self.total_page_label.set_attributes(label_attributes)

self.total_page_label.set_text(' / 0')
total_page_item.add(self.total_page_label)
self.total_page_label.show()
toolbar_box.toolbar.insert(total_page_item, -1)
total_page_item.show()

separator = gtk.SeparatorToolItem()
separator.props.draw = False
separator.set_expand(True)
toolbar_box.toolbar.insert(separator, -1)
separator.show()
```

```
        stop_button = StopButton(self)
        stop_button.props.accelerator = 'Q'
        toolbar_box.toolbar.insert(stop_button, -1)
        stop_button.show()

        self.set_toolbar_box(toolbar_box)
        toolbar_box.show()

    def __new_num_page_entry_insert_text_cb(self, entry, text, length, position):
        if not re.match('[0-9]', text):
            entry.emit_stop_by_name('insert-text')
            return True
        return False

    def __new_num_page_entry_activate_cb(self, entry):
        global page
        if entry.props.text:
            new_page = int(entry.props.text) - 1
        else:
            new_page = 0

        if new_page >= self.total_pages:
            new_page = self.total_pages - 1
        elif new_page  0
        self.forward.props.sensitive = \
            current_page = self.read_toolbar.total_pages:
            new_page = self.read_toolbar.total_pages - 1
        elif new_page = len(self.page_index): page=0
        if _NEW_TOOLBAR_SUPPORT:
            self.set_current_page(page)
        else:
            self.read_toolbar.set_current_page(page)
        self.show_page(page)
        v_adjustment = self.scrolled_window.get_vadjustment()
        v_adjustment.value = v_adjustment.lower

    def zoom_in_cb(self,  button):
        self.font_increase()

    def zoom_out_cb(self,  button):
        self.font_decrease()

    def font_decrease(self):
        font_size = self.font_desc.get_size() / 1024
        font_size = font_size - 1
        if font_size  v_adjustment.upper - v_adjustment.page_size:
                new_value = v_adjustment.upper - v_adjustment.page_size
            v_adjustment.value = new_value

    def scroll_up(self):
        v_adjustment = self.scrolled_window.get_vadjustment()
        if v_adjustment.value == v_adjustment.lower:
            self.page_previous()
            return
        if v_adjustment.value > v_adjustment.lower:
            new_value = v_adjustment.value - \
                v_adjustment.step_increment
            if new_value  0):
                newPage = title[i] + newPage
                i = i - 1
            if title[i] == 'P':
                page = int(newPage) - 1
            else:
                # not a page number; maybe a volume number.
                page = 0

    def save_page_number(self):
        global page
        title = self.metadata.get('title', '')
        if title == ''  or not title[len(title)- 1].isdigit():
            title = title + ' P' +  str(page + 1)
        else:
            i = len(title) - 1
            while (title[i].isdigit() and i > 0):
                i = i - 1
```

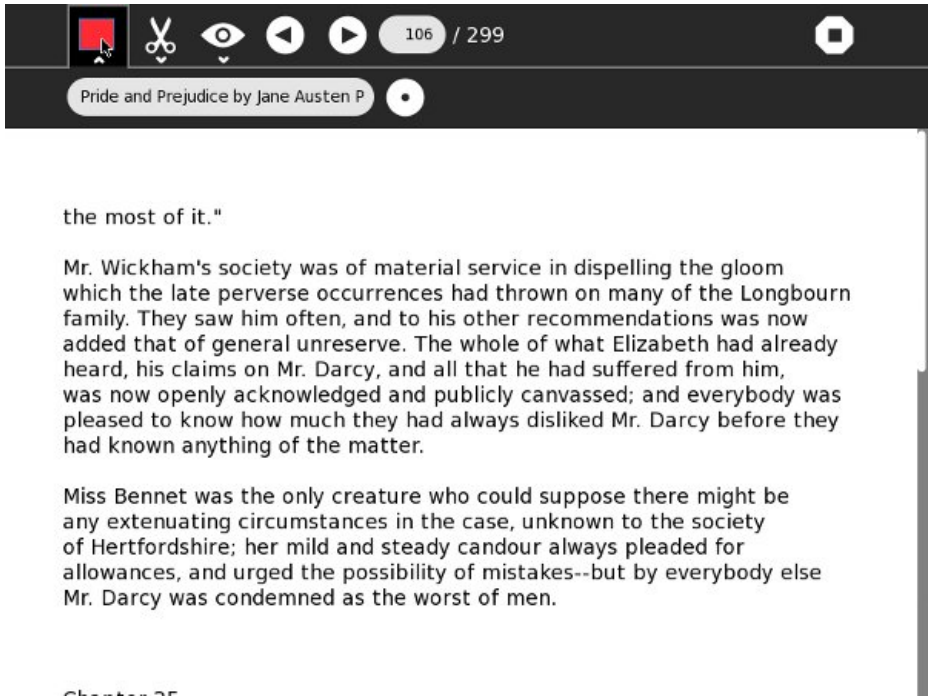
```
        if title[i] == 'P':
            title = title[0:i] + 'P' + str(page + 1)
        else:
            title = title + ' P' + str(page + 1)
    self.metadata['title'] = title

def read_file(self, filename):
    "Read the Etext file"
    global PAGE_SIZE,  page

    tempfile = os.path.join(self.get_activity_root(),  'instance', \
            'tmp%i' % time.time())
    os.link(filename,  tempfile)
    self.tempfile = tempfile

    if zipfile.is_zipfile(filename):
        self.zf = zipfile.ZipFile(filename, 'r')
        self.book_files = self.zf.namelist()
        self.save_extracted_file(self.zf, self.book_files[0])
        currentFileName = os.path.join(self.get_activity_root(), \
                'tmp',  self.book_files[0])
    else:
        currentFileName = filename

    self.etext_file = open(currentFileName,"r")
    self.page_index = [ 0 ]
    pagecount = 0
    linecount = 0
    while self.etext_file:
        line = self.etext_file.readline()
        if not line:
            break
        linecount = linecount + 1
        if linecount >= PAGE_SIZE:
            position = self.etext_file.tell()
            self.page_index.append(position)
            linecount = 0
            pagecount = pagecount + 1
    if filename.endswith(".zip"):
        os.remove(currentFileName)
    self.get_saved_page_number()
    self.show_page(page)
    if _NEW_TOOLBAR_SUPPORT:
        self.set_total_pages(pagecount + 1)
        self.set_current_page(page)
    else:
        self.read_toolbar.set_total_pages(pagecount + 1)
        self.read_toolbar.set_current_page(page)

    # We've got the document, so if we're a shared activity, offer it
    if self.get_shared():
        self.watch_for_tubes()
        self.share_document()

def make_new_filename(self, filename):
    partition_tuple = filename.rpartition('/')
    return partition_tuple[2]

def write_file(self, filename):
    "Save meta data for the file."
    if self.is_received_document:
        # This document was given to us by someone, so we have
        # to save it to the Journal.
        self.etext_file.seek(0)
        filebytes = self.etext_file.read()
        print 'saving shared document'
        f = open(filename, 'wb')
        try:
            f.write(filebytes)
        finally:
            f.close()
    elif self.tempfile:
        if self.close_requested:
            os.link(self.tempfile,  filename)
            logger.debug("Removing temp file %s because we will close", \
```

```
                        self.tempfile)
            os.unlink(self.tempfile)
            self.tempfile = None
        else:
            # skip saving empty file
            raise NotImplementedError

        self.metadata['activity'] = self.get_bundle_id()
        self.save_page_number()

    def can_close(self):
        self.close_requested = True
        return True

    def joined_cb(self, also_self):
        """Callback for when a shared activity is joined.

        Get the shared document from another participant.
        """
        self.watch_for_tubes()
        gobject.idle_add(self.get_document)

    def get_document(self):
        if not self.want_document:
            return False

        # Assign a file path to download if one doesn't exist yet
        if not self._jobject.file_path:
            path = os.path.join(self.get_activity_root(), 'instance',
                                'tmp%i' % time.time())
        else:
            path = self._jobject.file_path

        # Pick an arbitrary tube we can try to download the document from
        try:
            tube_id = self.unused_download_tubes.pop()
        except (ValueError, KeyError), e:
            logger.debug('No tubes to get the document from right now: %s',
                         e)
            return False

        # Avoid trying to download the document multiple times at once
        self.want_document = False
        gobject.idle_add(self.download_document, tube_id, path)
        return False

    def download_document(self, tube_id, path):
        chan = self._shared_activity.telepathy_tubes_chan
        iface = chan[telepathy.CHANNEL_TYPE_TUBES]
        addr = iface.AcceptStreamTube(tube_id,
                telepathy.SOCKET_ADDRESS_TYPE_IPV4,
                telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0,
                utf8_strings=True)
        logger.debug('Accepted stream tube: listening address is %r', \
                     addr)
        assert isinstance(addr, dbus.Struct)
        assert len(addr) == 2
        assert isinstance(addr[0], str)
        assert isinstance(addr[1], (int, long))
        assert addr[1] > 0 and addr[1]  0:
            logger.debug("Downloaded %u of %u bytes from tube %u...",
                         bytes_downloaded, self.download_content_length,
                         tube_id)
        else:
            logger.debug("Downloaded %u bytes from tube %u...",
                         bytes_downloaded, tube_id)
        total = self.download_content_length
        self.set_downloaded_bytes(bytes_downloaded,  total)
        gtk.gdk.threads_enter()
        while gtk.events_pending():
            gtk.main_iteration()
        gtk.gdk.threads_leave()

    def set_downloaded_bytes(self, bytes,  total):
        fraction = float(bytes) / float(total)
```

```
        self.progressbar.set_fraction(fraction)
        logger.debug("Downloaded percent",  fraction)

    def clear_downloaded_bytes(self):
        self.progressbar.set_fraction(0.0)
        logger.debug("Cleared download bytes")

    def download_error_cb(self, getter, err, tube_id):
        self.progressbar.hide()
        logger.debug("Error getting document from tube %u: %s",
                     tube_id, err)
        self.alert(_('Failure'), _('Error getting document from tube'))
        self.want_document = True
        self.download_content_length = 0
        self.download_content_type = None
        gobject.idle_add(self.get_document)

    def download_result_cb(self, getter, tempfile, suggested_name, tube_id):
        if self.download_content_type.startswith('text/html'):
            # got an error page instead
            self.download_error_cb(getter, 'HTTP Error', tube_id)
            return

        del self.unused_download_tubes

        self.tempfile = tempfile
        file_path = os.path.join(self.get_activity_root(), 'instance',
                                 '%i' % time.time())
        logger.debug("Saving file %s to datastore...", file_path)
        os.link(tempfile, file_path)
        self._jobject.file_path = file_path
        datastore.write(self._jobject, transfer_ownership=True)

        logger.debug("Got document %s (%s) from tube %u",
                     tempfile, suggested_name, tube_id)
        self.is_received_document = True
        self.read_file(tempfile)
        self.save()
        self.progressbar.hide()

    def shared_cb(self, activityid):
        """Callback when activity shared.

        Set up to share the document.

        """
        # We initiated this activity and have now shared it, so by
        # definition we have the file.
        logger.debug('Activity became shared')
        self.watch_for_tubes()
        self.share_document()

    def share_document(self):
        """Share the document."""
        h = hash(self._activity_id)
        port = 1024 + (h % 64511)
        logger.debug('Starting HTTP server on port %d', port)
        self.fileserver = ReadHTTPServer(("", port),
            self.tempfile)

        # Make a tube for it
        chan = self._shared_activity.telepathy_tubes_chan
        iface = chan[telepathy.CHANNEL_TYPE_TUBES]
        self.fileserver_tube_id = iface.OfferStreamTube(READ_STREAM_SERVICE,
                {},
                telepathy.SOCKET_ADDRESS_TYPE_IPV4,
                ('127.0.0.1', dbus.UInt16(port)),
                telepathy.SOCKET_ACCESS_CONTROL_LOCALHOST, 0)

    def watch_for_tubes(self):
        """Watch for new tubes."""
        tubes_chan = self._shared_activity.telepathy_tubes_chan

        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].connect_to_signal('NewTube',
            self.new_tube_cb)
```

```
        tubes_chan[telepathy.CHANNEL_TYPE_TUBES].ListTubes(
            reply_handler=self.list_tubes_reply_cb,
            error_handler=self.list_tubes_error_cb)

    def new_tube_cb(self, tube_id, initiator, tube_type, service, params,
                    state):
        """Callback when a new tube becomes available."""
        logger.debug('New tube: ID=%d initator=%d type=%d service=%s '
                     'params=%r state=%d', tube_id, initiator, tube_type,
                     service, params, state)
        if service == READ_STREAM_SERVICE:
            logger.debug('I could download from that tube')
            self.unused_download_tubes.add(tube_id)
            # if no download is in progress, let's fetch the document
            if self.want_document:
                gobject.idle_add(self.get_document)

    def list_tubes_reply_cb(self, tubes):
        """Callback when new tubes are available."""
        for tube_info in tubes:
            self.new_tube_cb(*tube_info)

    def list_tubes_error_cb(self, e):
        """Handle ListTubes error by logging."""
        logger.error('ListTubes() failed: %s', e)

    def alert(self, title, text=None):
        alert = NotifyAlert(timeout=20)
        alert.props.title = title
        alert.props.msg = text
        self.add_alert(alert)
        alert.connect('response', self.alert_cancel_cb)
        alert.show()

    def alert_cancel_cb(self, alert, response_id):
        self.remove_alert(alert)
        self.textview.grab_focus()
```

Aquí es lo que parece el funcionamiento debajo del **sugar-jhbuild**:



Tengamos una mirada en cómo trabaja. Si usted ha prestado la atención a otros capítulos cuando he

hablado de la idea de la "degradación agraciado" las importaciones en este código estarán sobre lo que usted esperaría:

```
_NEW_TOOLBAR_SUPPORT = True
try:
    from sugar.graphics.toolbarbox import ToolbarBox
    from sugar.graphics.toolbarbox import ToolbarButton
    from sugar.activity.widgets import StopButton
    from toolbar import ViewToolbar
    from mybutton import MyActivityToolbarButton
except:
    _NEW_TOOLBAR_SUPPORT = False
    from toolbar import ReadToolbar,  ViewToolbar
```

Aquí intentamos importar un manojo de materia que exista solamente en las versiones del azúcar que apoyan las nuevas barras de herramientas. Si tenemos éxito, después el _NEW_TOOLBAR_SUPPORT seguirá siendo sistema a verdad. Si las importaciones unas de los fallan entonces la variable se fija a falso. Observe que unas par de importaciones que deben tener éxito siempre están puestas después de que los tres que pudieron fallar. Si es un de los primeros tres me fallan no quisieran que estas importaciones fueran hechas.

Este pedacito siguiente del código en el método del *__init ()* no debe ser asombrosamente:

```
        if _NEW_TOOLBAR_SUPPORT:
            self.create_new_toolbar()
        else:
            self.create_old_toolbar()
```

Me moví creando las barras de herramientas en sus propios métodos para hacerlo más fácil comparar cómo se crean las dos diversas barras de herramientas. El viejo código de la barra de herramientas es sin cambios. Aquí está el nuevo código de la barra de herramientas:

```
    def create_new_toolbar(self):
        toolbar_box = ToolbarBox()

        activity_button = MyActivityToolbarButton(self)
        toolbar_box.toolbar.insert(activity_button, 0)
        activity_button.show()

        self.edit_toolbar = activity.EditToolbar()
        self.edit_toolbar.undo.props.visible = False
        self.edit_toolbar.redo.props.visible = False
        self.edit_toolbar.separator.props.visible = False
        self.edit_toolbar.copy.set_sensitive(False)
        self.edit_toolbar.copy.connect('clicked', self.edit_toolbar_copy_cb)
        self.edit_toolbar.paste.props.visible = False

        edit_toolbar_button = ToolbarButton(
            page=self.edit_toolbar,
            icon_name='toolbar-edit')
        self.edit_toolbar.show()
        toolbar_box.toolbar.insert(edit_toolbar_button, -1)
        edit_toolbar_button.show()

        self.view_toolbar = ViewToolbar()
        self.view_toolbar.connect('go-fullscreen', \
            self.view_toolbar_go_fullscreen_cb)
        self.view_toolbar.zoom_in.connect('clicked', self.zoom_in_cb)
        self.view_toolbar.zoom_out.connect('clicked', self.zoom_out_cb)
        self.view_toolbar.show()
        view_toolbar_button = ToolbarButton(
            page=self.view_toolbar,
            icon_name='toolbar-view')
        toolbar_box.toolbar.insert(view_toolbar_button, -1)
        view_toolbar_button.show()

        self.back = ToolButton('go-previous')
        self.back.set_tooltip(_('Back'))
        self.back.props.sensitive = False
```

```
        self.back.connect('clicked', self.go_back_cb)
        toolbar_box.toolbar.insert(self.back, -1)
        self.back.show()

        self.forward = ToolButton('go-next')
        self.forward.set_tooltip(_('Forward'))
        self.forward.props.sensitive = False
        self.forward.connect('clicked', self.go_forward_cb)
        toolbar_box.toolbar.insert(self.forward, -1)
        self.forward.show()

        num_page_item = gtk.ToolItem()
        self.num_page_entry = gtk.Entry()
        self.num_page_entry.set_text('0')
        self.num_page_entry.set_alignment(1)
        self.num_page_entry.connect('insert-text',
                                    self.__new_num_page_entry_insert_text_cb)
        self.num_page_entry.connect('activate',
                                    self.__new_num_page_entry_activate_cb)
        self.num_page_entry.set_width_chars(4)
        num_page_item.add(self.num_page_entry)
        self.num_page_entry.show()
        toolbar_box.toolbar.insert(num_page_item, -1)
        num_page_item.show()

        total_page_item = gtk.ToolItem()
        self.total_page_label = gtk.Label()

        label_attributes = pango.AttrList()
        label_attributes.insert(pango.AttrSize(14000, 0, -1))
        label_attributes.insert(pango.AttrForeground(65535, 65535,
                                                     65535, 0, -1))
        self.total_page_label.set_attributes(label_attributes)

        self.total_page_label.set_text(' / 0')
        total_page_item.add(self.total_page_label)
        self.total_page_label.show()
        toolbar_box.toolbar.insert(total_page_item, -1)
        total_page_item.show()

        separator = gtk.SeparatorToolItem()
        separator.props.draw = False
        separator.set_expand(True)
        toolbar_box.toolbar.insert(separator, -1)
        separator.show()

        stop_button = StopButton(self)
        stop_button.props.accelerator = 'Q'
        toolbar_box.toolbar.insert(stop_button, -1)
        stop_button.show()

        self.set_toolbar_box(toolbar_box)
        toolbar_box.show()

    def __new_num_page_entry_insert_text_cb(self, entry, text, length, position):
        if not re.match('[0-9]', text):
            entry.emit_stop_by_name('insert-text')
            return True
        return False

    def __new_num_page_entry_activate_cb(self, entry):
        global page
        if entry.props.text:
            new_page = int(entry.props.text) - 1
        else:
            new_page = 0

        if new_page >= self.total_pages:
            new_page = self.total_pages - 1
        elif new_page  0
        self.forward.props.sensitive = \
            current_page
```

Mucho del código en los dos métodos es igual. Particularmente, la barra de herramientas de la **visión** y la

barra de herramientas del **corregir** están exactamente igual en ambos. En vez de convertirse en la barra de herramientas activa caen abajo de la barra de herramientas para convertirse en barras de herramientas secundarias. Si habíamos hecho la barra de herramientas leída la misma manera habríamos podido ejecutar viejo y las nuevas barras de herramientas con muy pequeño cifran. Sin embargo, la barra de herramientas **leída** contiene los controles que son bastante importantes para la actividad que deben estar disponibles siempre, así que los ponemos en la barra de herramientas principal en lugar de otro. Debido a este cada lugar en donde el código refiere a la barra de herramientas **leída** tiene que tener dos maneras que puede ser realizada, como esto:

```
if _NEW_TOOLBAR_SUPPORT:
    self.set_total_pages(pagecount + 1)
    self.set_current_page(page)
else:
    self.read_toolbar.set_total_pages(pagecount + 1)
    self.read_toolbar.set_current_page(page)
```

Hay un más punto del interés cuando viene a la barra de herramientas principal. Cuando usted tiene una barra de herramientas del viejo estilo usted consigue el botón de paro como parte de la barra de herramientas de la **actividad**. Con la nueva barra de herramientas del estilo usted necesita agregarla al extremo de la barra de herramientas principal usted mismo:

```
separator = gtk.SeparatorToolItem()
separator.props.draw = False
separator.set_expand(True)
toolbar_box.toolbar.insert(separator, -1)
separator.show()

stop_button = StopButton(self)
stop_button.props.accelerator = 'Q'
toolbar_box.toolbar.insert(stop_button, -1)
stop_button.show()
```

Observe que usted debe poner un **gtk.SeparatorToolItem** con el igual del *set_expand ()* para verdad antes del **StopButton**. Esto empujará el botón hasta el final a la derecha de la barra de herramientas, donde pertenece.

Eso apenas sale de la barra de herramientas de la **actividad** para discutir:

```
toolbar_box = ToolbarBox()

activity_button = MyActivityToolbarButton(self)
toolbar_box.toolbar.insert(activity_button, 0)
activity_button.show()
```

Usted utilizaría normalmente la clase **ActivityToolbarButton** para crear el defecto cae abajo la barra de herramientas de la actividad. El problema que tengo con ése es si hago que no hay manera de ocultar el botón de la **subsistencia** o el control de la **parte**. Esta versión de la actividad necesita el control de la **parte**, pero no tiene ninguìn uso en absoluto para el botón de la **subsistencia**.

Ha habido algunas discusiones enérgicas sobre el botón de la **subsistencia** en las listas de personas a quienes se mandan propaganda. Los nuevos usuarios de la computadora no saben para cuáles es, y los usuarios experimentados de la computadora esperan que sea como un botón **de ahorro del juego** o una **reserva como...** opción del menú en un uso regular. No es absolutamente como tampoco uno, y ése puede llevar a la confusión. Por estas razones he decidido que ninguna actividad la mía saldrá del botón de la **subsistencia** unhidden. Para ocultar el botón copié un pedacito del código para el **ActivityToolbarButton** original en un archivo nombrado **mybutton.py**:

```
import gtk
import gconf

from sugar.graphics.toolbarbox import ToolbarButton
from sugar.activity.widgets import ActivityToolbar
from sugar.graphics.xocolor import XoColor
```

```
from sugar.graphics.icon import Icon
from sugar.bundle.activitybundle import ActivityBundle

def _create_activity_icon(metadata):
    if metadata.get('icon-color', ''):
        color = XoColor(metadata['icon-color'])
    else:
        client = gconf.client_get_default()
        color = XoColor(client.get_string('/desktop/sugar/user/color'))

    from sugar.activity.activity import get_bundle_path
    bundle = ActivityBundle(get_bundle_path())
    icon = Icon(file=bundle.get_icon(), xo_color=color)

    return icon

class MyActivityToolbarButton(ToolbarButton):

    def __init__(self, activity, **kwargs):
        toolbar = ActivityToolbar(activity, orientation_left=True)
        toolbar.stop.hide()
        toolbar.keep.hide()

        ToolbarButton.__init__(self, page=toolbar, **kwargs)

        icon = _create_activity_icon(activity.metadata)
        self.set_icon_widget(icon)
        icon.show()
```

La línea en **en negrilla** es la una diferencia entre los míos y la original. Si la **barra de herramientas** había sido hecha una variable de caso (**self.toolbar**) habría podido utilizar el original clasifico.

APPENDIX

20. ¿ADÓNDE IR DE AQUÍ?

Este libro intenta dar a un programador principiante la información que ella necesita para desarrollar y publicar sus propias Actividades para Sugar. Contiene ya muchos URL de los sitios de la Web que contienen la información no cubierta en el libro. Este capítulo contendrá los URL y los indicadores a muchos más recursos que sean útiles a cualquier desarrollador Sugar.

LIBRO DE PYGTK DE PETER GILL

Mucho del trabajo que usted hará escribiendo Actividades implica PyGTK. Peter Gill está trabajando en un libro de PyGTK que cubre el tema con gran detalle. Usted puede descargar el libro aquí:

http://www.majorsilence.com/PyGTK_Book

MANUAL DE ACTIVIDADES DE OLPC AUSTRIA

Ésta es la primera tentativa de escribir un manual para crear Actividades en Sugar. Está dirigida a programadores experimentados y cubre tópicos no cubiertos en este libro, tal como cómo escribir Actividades usando lenguajes distintos de Python. El libro fue escrito en el 2008 y consecuentemente algunos de los consejos son un tanto anticuados. Sin embargo, sigue siendo una excelente fuente de información. Los autores son Cristóbal Derndorfer y Daniel Jahre.

http://wiki.sugarlabs.org/images/5/51/Activity_Handbook_200805_online.pdf

http://www.olpcaustria.org

EL ALMANAQUE DEL AZÚCAR

Ésta es una serie de artículos de Wiki que cubren el **API** de Sugar (**interfaz de programación de aplicaciones**). Es una buena fuente de información que he usado muchas veces.

http://wiki.sugarlabs.org/go/Development_Team/Almanac

LISTAS DE CORREOS DE SUGAR LABS

Los Laboratorios Sugar (Sugar Labs) tienen varias listas de email a las cuales vale suscribirse. Las que sigo mayormente son la lista **IAEP (It's An Education Project - Es Un Proyecto Educativo)** y **Sugar-Devel**. El Sugar-Devel es un buen lugar para hacer preguntas acerca el desarrollo de Actividades de Sugar y para conocer sobre lo más reciente de Sugar mismo. IAEP es un buen lugar para conseguir ideas sobre qué clase de Actividades quieren los profesores y estudiantes, y conseguir retroalimentación sobre sus propias Actividades. Cualquier persona puede registrarse a estas listas de correo aquí:

http://lists.sugarlabs.org/

PYDOC

PyDoc es una utilidad para ver la documentación generada de las librerías de Python en su computadora, incluyendo las librerías de Sugar. Para ejecutarlo utilice este comando desde una terminal:

```
pydoc -p 1234
```

Este comando no acabará. Funciona como una clase de web server en su sistema donde *1234* es un puerto. Usted puede tener acceso al Web site que sirve en **http://localhost:1234**. No hay nada mágico sobre el número 1234. Usted puede utilizar cualquier número que prefiera.

El Web site le deja seguir enlaces en la documentación en todas las librerías del Python que usted tenga instaladas. Cuando haya terminado de navegar por la documentación, puede parar el comando pydoc a través de la terminal, presionando Ctrl-C (mantenga presionada la Ctrl y luego presiones la tecla "c").

21. GLOSSARIO

bug
: defecto, error o fallo en un programa de computador

debug
: depurador hacer un debug es depurar un programa, no significa exactamente hallar errores.

22. ACERCA DE LOS AUTORES

JAMES SIMMONS

James Simmons ha programado profesionalmente desde el año 1978. En esa época, los programas de computación eran elaborados por medio de una máquina especial que perforaba tarjetas, los carretes de cinta magnética eran el medio más común de almacenamiento de data, y los discos duros eran tan caros y exóticos que el inventario de disco duro de una compañía Fortune 500 hoy sólo podría contener una foto de Jessica Alba.

La industria ha progresado mucho desde entonces y, a un menor grado, James ha progresado también.

James aprendió a programar en el Instituto de Enseñanza Superior de Oakton en Morton Grove, Illinois y la Universidad Illinois Occidental en Macomb, Illinois. Los tiempos eran duros en ese entonces y la mejor oportunidad de un hombre joven ser empleado al graduarse era convertirse en Contador o en Programador de Computadoras. Fue mientras asistía a OCC que James vió un bosquejo de Monty Python sobre un Contador que deseaba convertirse en un domador de leones. Esto convenció a James de que él debería convertirse en Programador de Computadoras.

El año anterior a su entrada en WIU, ésta fue nominada como la escuela más parrandera por la revista Playboy. James estaba demasiado ocupado siendo un programador de computadoras para constatar si todavía siendo así. Sus estudios tuvieron un comienzo difícil cuando se inscribió en Lenguaje Ensamblador Básico, como su primera materia real sobre computadoras, pensando erradamente que "Básico" significaba que era para "principiantes". Este curso lo aprobó apenas con una "D", pero en el proceso se dió cuenta que disfrutaba programar computadoras. Decidió así continuar con sus estudios de computación y se graduó con una Licenciatura en Cientas de la Información.

James nació en 1956, el año antes de que se enviara el Sputnik al espacio. Él era un niño nerd. En varias ocasiones perdía el tiempo con juegos Erector, juegos de química, microscopios, juegos de disección, autos modelo, aviones modelo, cohetes modelo, radio amateur, rodaje de películas, y escribiendo historias de ciencia ficción. Él no alcanzó ningún éxito verdadero en ninguna de estas actividades.

James participó en la primera promoción *Da una Consigue una* (*Give One Get One*) promoción del proyecto Un laptop para cada Niño, e inició el desarrollo de Actividades para la plataforma Sugar, inmediatamente después. Ha escrito las Actividades **Read Etexts** (*Lectura de Libros*), **View Slides** (*Ver Diapositivas*), y **Get Internet Archive Books** (*Obtener Libros de los Archivos de Internet*).

JAMES CAMERON

James Cameron ha programado como niño desde 1978, y profesionalmente desde 1982. Él aprendió en las calculadoras programables, Apple II, TRS-80, el Commodore 64, y luego en la DEC VAX.

James terminó una Licenciatura en Negocios en 1991, con especialización en Gerencia de Sistemas de Información. Ha trabajado para empresas de ingenería eléctrica y manufactura de computadoras. Se interesó por el proyecto Un laptop para cada Niño como voluntario y ha provisto pruebas de alcance de radio en el interior de Australia, y ahora está trabajando para la OLPC como Coordinador de Pruebas de Sistemas.

James repasó los ejemplos de código en este libro e hizo muchas sugerencias para mejorarlo.

Thanks for reading!

Visit http://flossmanuals.net to make corrections or to find more manuals.