Toccata
IDE para el trabajo con Lilypond

Javier Salazar Loyola <jsalazar@elo.utfsm.cl>

Resumen

En el contexto actual, los músicos deben hacer muchos malabares para conseguir partituras de calidad con los actuales programas de edición del tipo WYSIWYG. Es por esta razón que Lilypond[1] surge, buscando una opción viable de tipografiar música y obtener resultados profesionales y de calidad superior a los habituales programas disponibles en el mercado.

Sin embargo, Lilypond es complejo en su utilización, por lo que surge la necesidad de tener un ambiente de desarrollo apropiado para el trabajo con éste. **Toccata** surge en este contexto y es el primer escalón en el desarrollo de un ambiente integrado de desarrollo (*aka* IDE) para este programa.

Parte I
Acerca del Programa

En esta parte se detallan las funciones del programa y lo que debe realizar cada una de ellas, además de presentar cómo está construida la ventana.

Prólogo

En las antiguas impresiones de partituras que pueden conseguirse aún en el mercado a través de diferentes editoriales, se ven grabados de excelente calidad, grabados que a un músico le gustaría poder reproducir cuando lo que dispone es la partitura general de una sinfonía y lo que necesita es la parte individual de cada uno de los instrumentos.

Hoy en día pueden conseguirse varios programas del tipo WYSIWYG, tales como *Sibelius*[2] y *Finale*[3], dos de las más populares. Sin embargo, opciones como éstas son bastante costosas (las licencias de ambas no son precisamente económicas), y se ve obligado uno a transcribir las partes a mano, lo que varía entre una persona y otra, y eso puede causar confusión entre diferentes lectores de lo mismo.

Lilypond es la mejor opción que resuelve estos (y otros) problemas, pero tiene el problema de que es un lenguaje interpretado complejo, por lo que es necesario simplificar la vida de las personas que se ven obligadas a trabajar con éste tipo de recursos.

Como punto de partida, el proyecto **Toccata** ofrece resaltado de sintaxis del lenguaje Lilypond (similar en muchos aspectos a LaTeXy completamente integrable con éste y otros), compilación del archivo y visualización de los archivos PDF y PS generados. El presente documento introduce el incipiente trabajo de este proyecto (y algunas de las funcionalidades que se esperan alcanzar a futuro), además de presentar parte del desarrollo y cómo se ha llevado en el proyecto.

[1] ©1999 - 2009 Han-Wen Nienhuys, Jan Nieuwenhuizen y otros
[2] ©Sibelius Software Ltd
[3] ©MakeMusic! Ltd

1. Ejecución del Programa

Debido a que es un conjunto de módulos de Python (*i.e.*, *scripts*), no es necesaria la compilación del programa, sin embargo, tampoco puede llegar y ejecutarse. Lo primero es tener instalado Python[4] y su módulo PyQt4[5].

Una vez que se tenga instalado el intérprete Python y PyQt4, la ejecución es diferente según el sistema.

La instalación del programa se realiza como un archivo `tar.bz`, en Windows, aplicaciones de compresión y descompresión estándares soportan lectura y descompresión de este formato (Ejemplo, Winrar o 7zip).

En Mac no se ha probado, pero dado el hecho de que Python y PyQt4 son multiplataforma, donde se soporten ambos, debiera correr el programa.

1.1. Requisitos del Sistema

Para correr, deben cumplirse un mínimo de requerimientos del sistema, que se listan a continuación.

- Python 2.4 o superior.
- PyQt4 4.2 o superior.
- Doxygen, en caso de querer generar de nuevo la documentación interna del programa.

1.2. Linux

En sistemas Linux, la ejecución puede realizarse como script tradicional: el módulo MainWindow.py contiene la aplicación principal, una vez que tenga los permisos de ejecución, puede ejecutarse como script normal. Desde la consola, debe correrse mediante:

```
$ python directorio/MainWindow.py
```

Donde debe cambiarse "directorio" por el directorio donde se descomprimió el programa.

1.3. Windows

En Windows, si se tiene instalado Python, éste se asocia automáticamente con los archivos `.py` y al ejecutarlos, se ejecuta la consola automáticamente, y, con ella, el programa. El módulo que contiene la aplicación principal es MainWindow.py.

Desde la línea de comandos (Menú Inicio, Ejecutar, `cmd` (`command` en Windows 98)), también se puede, ejecutando:

```
python directorio\MainWindow.py
```

Donde "directorio" se debe reemplazar por el directorio donde se encuentra el archivo.

2. Aplicación Principal y sus Características

La aplicación principal consta de una ventana con un cuadro de edición de texto, un cuadro de mensajes, una barra de herramientras y una barra de menú que incluye todas las funcionalidades hasta ahora implementadas. En la figura 1 se muestra cómo se ve la ventana.

A continuación se muestran las acciones del menú y lo que realiza cada una de ellas.

[4] www.python.org

[5] Página de Trolltech (encargados de Qt): www.trolltech.com, página de PyQt: www.riverbankcomputing.co.uk/pyqt

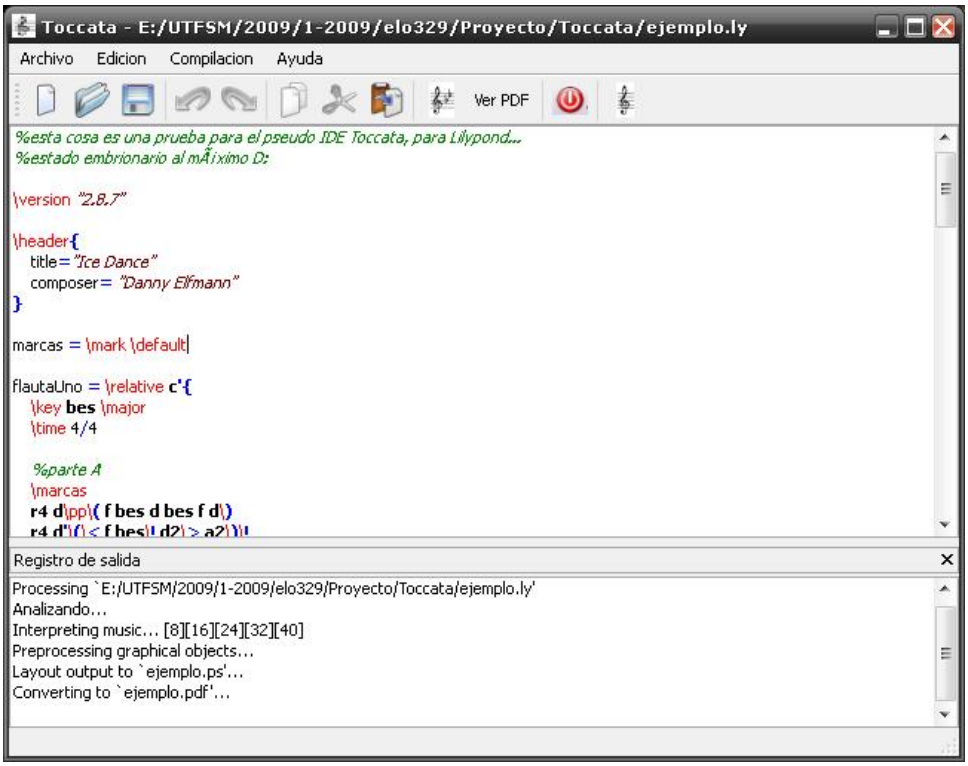

Figura 1: Ventana Principal de la aplicación

2.1. Menú Archivo

Nuevo Crea un archivo en blanco y sin nombre.

Abrir Abre un archivo existente en algún dispositivo de almacenamiento.

Guardar Guarda el archivo; si tiene nombre, con ese nombre; en caso de no tenerlo, abre el cuadro de diálogo guardar como.

Guardar como... Abre un cuadro de diálogo preguntando por un nombre de archivo para guardar el actualmente editado.

Salir Cierra el programa. Si el documento se ha modificado, pide confirmación.

2.2. Menú Edición

Deshacer Deshace la última edición realizada en el documento.

Rehacer Si se ha deshecho algo y no se ha editado encima de ello, permite rehacer lo deshecho.

Copiar Permite copiar el texto seleccionado para poder pegarlo posteriormente.

Cortar Copia el texto y a continuación, lo borra.

Pegar Inserta el texto copiado

Seleccionar Todo Selecciona todo el texto en el editor

2.3. Menú Compilación

Lilypond Compila el archivo actual. Si está guardado con algún nombre, utiliza ese para la compilación, de lo contrario, pide guardar el archivo.

Ver PDF Abre el visor de PDF configurado (por defecto, *xpdf*, no puede editarse sin editar directamente el código, actualmente).

Ver PS Abre el visor de PS configurado (ídem que para el visor de PDF, el por defecto es *evince*)

Ver Registro Permite ver el registro de salida de la compilación (de haberse hecho alguna)

Guardar Registro Permite guardar el registro de salida de la compilación como archivo de texto.

2.4. Menú Ayuda

Acerca de... Muestra información acerca del proyecto.

Acerca de Qt Muestra información acerca de Qt

2.5. Contenido de la ventana

Se detalla a continuación la función de los principales componentes de la ventana.

Barra de Herramientas Contiene botones de acceso rápido para algunos comandos: Nuevo, Abrir, Guardar, Deshacer, Rehacer, Copiar, Cortar, Pegar, Lilypond, Ver PDF, Salir, Acerca de...

Editor de Texto Es aquídonde se realiza todo el trabajo con el código fuente de la partitura en Lilypond. Permite resaltar el texto con palabras claves, notas, comentarios, etc.

Registro de Salida Es un editor de texto que muestra los resultados de la compilación (debe mejorarse)

Parte II
Acerca del Desarrollo

En esta parte se detalla la realización del programa, casos de uso, datos relevantes, etc.

3. Elección del Nombre y el Lenguaje de Desarrollo

3.1. El Nombre

Una toccata, según la definición clásica, es una composición musical, generalmente de estilo libre, que según su ubicación era lo que buscaba: a veces, era parte de la *Suite*, generalmente como preludio; mientras que otras veces, en particular en J. S. Bach, era un preludio a las fugas.

En particular, la toccata de la Toccata y Fuga en Re menor BWV565, tenía una función práctica que consistía en temperar el órgano antes de tocar, o probar todos los tubos y que tuvieran lo que se necesitaba para una correcta ejecución. Los *mordentes* y escalas de los primeros compases recorren gran parte del registro del órgano promedio, por lo que cada una de las notas que debían utilizarse eran probadas desde el primer momento de la pieza.

Bajo este contexto, el nombre **Toccata** fue elegido para poder mostrar que la gran intención de este proyecto es la ayuda a los músicos, a crear sus partituras con la ayuda de Lilypond y hacer de éste lenguaje algo un poco más amigable de lo que es en principio para alguien que se dedica a algo completamente diferente de lo que es la programación (ya que Lilypond está mucho más emparentado con la programación que con la música en sí).

3.2. El Lenguaje

PyQt es la adaptación de las librerías Qt (originalmente para C++) para Python, desarrolladas por Trolltech. En este caso, la licencia es Open Source porque no es la intención hacer de éste, un software propietario ya que desde el comienzo está pensado en una orientación Open Source.

Python es un lenguaje orientado a objetos basado en scripts, y que puede orientarse al procedimiento tanto como al objeto. La razón de la elección es porque no es Java, un lenguaje muy poderoso, pero que, a decir verdad, nunca me ha gustado (N. del A.). Por esta razón, y la de querer explorar las posibilidades de Python y su adaptación de Qt es que se eligió como lenguaje.

Además, Lilypond se trabaja sobre 3 lenguajes: C++, Scheme y Python.

4. Casos de Uso

En esta sección se detallan los principales casos de uso que tiene en este momento la aplicación. Incluyen la compilación, el resaltado de sintaxis y la visualización de la partitura generada.

4.1. Resaltado de Sintaxis

Nombre Resaltar Sintaxis

Propósito Implementar colores y distintos tipos de fuentes según sea pertinente

Actores Objeto que resalta (class Highlighter)

Pre-condiciones El objeto editor de texto tiene texto cargado (no es crítico)

Evento Carga de un nuevo archivo ".ly" o ingreso de nuevo texto al archivo cargado

Post-condiciones El objeto editor muestra el texto en formato enriquecido, con la sintaxis resaltada

Tipo Automático: gatillado por eventos

Curso Normal de eventos Se detalla el flujo normal de eventos.

1. Carga el archivo o se modifica el ya cargado
2. El objeto encargado de resaltar el texto lo hace
3. El texto se muestra formateado de acuerdo a ciertas reglas básicas

Curso Alternativo De no poderse generar el resaltado por algún error desconocido (Durante las pruebas se depuraron todos los que fueron posibles), el programa sigue corriendo, pero sin el resaltado de texto

Requerimientos No Funcionales Ninguno explícito

4.2. Compilación

Nombre Compilar Archivo

Propósito Generar el archivo PDF y PS con la partitura para imprimir

Actores Lilypond, Usuario

Pre-condiciones El archivo debe estar guardado y el directorio debe tener permiso deescritura

Evento Usuario lo solicita explícitamente

Post-condiciones Los archivos PDF y PS quedan listos para ver, y el usuario puede ver en la ventana de registro los mensajes del programa (útiles en caso de fallo en la compilación)

Tipo Manual: gatillado por el usuario

Curso Normal de eventos
1. Se crea un proceso hijo con el programa Lilypond
2. Lilypond intenta compilar el archivo
3. El resultado se muestra en el Registro de Salida

Curso Alternativo Si está instalado Lilypond, no debiera haber ningún curso alternativo. El proceso hijo compila o no, lo que debe cumplirse para el programa es que la salida de ese proceso esté accequible.

Requerimientos No Funcionales Ninguno explícito

4.3. Visualizar Archivos Generados

Nombre Ver Partitura Generada

Propósito Invocar a un editor que muestre la partitura generada

Actores Visor de PDF o PS

Pre-condiciones El archivo debe estar compilado y los visores deben existir

Evento Usuario lo solicita explícitamente

Post-condiciones Los archivos son visibles mediante visores de PDF y PS

Tipo Manual: gatillado por el usuario

Curso Normal de eventos
1. Carga el visor con el archivo PDF o PS generado
2. El visor queda disponible para el usuario

Curso Alternativo Dependiendo del visor, puede que éste no abra en caso de que el archivo no exista, lo que puede pasar si es que aún no se compila.

Requerimientos No Funcionales El visor no se cierra al cerrar el programa principal.

5. Algunos datos importantes

Este programa es una colección de scripts en Python, interpretados por éste. Para poder ejecutar correctamente el programa es necesario tener instalados Python, al menos, 2.4 y PyQt4, al menos, 4.2. El programa fue desarrollado con estos requisitos mínimos y probado exitosamente en Windows y Debian Etch (En el caso de Windows, hay que cambiar los visores, puesto que *evince* y *xpdf* son aplicaciones de Linux, en particular, la primera viene incluida con el escritorio Gnome).

No es dependiente el programa de ninguna versión particular de Lilypond, de ésto debe preocuparse el usuario, el programa sólo crea el proceso hijo con el comando "*lilypond*", pero no se han incluido otros parámetros que podrían ser específicos de alguna versión en particular.

Por otra parte, la documentación para el código, con la información de métodos y clases, puede encontrarse en el Manual de Referencia, generado con la herramienta Doxygen 1.5.1.

6. Conclusiones

Desde el desarrollo de este proyecto, puede apreciarse con utilizarlo sólo una vez que está en una etapa muy temprana de su desarrollo y muchas de sus falencias resultan obvias, y ocurren principalmente por un tema del poco tiempo que hubo para desarrollarlo, puesto que había que mezclarlo con muchas cosas, y uno de los principales desafíos fue el de aprender en poco tiempo y de forma eficiente los lenguajes Python y PyQt4 (completamente diferente de PyQt3, la información más sencilla de encontrar es para éste último).

Muchas de las grandes dificultades para partir se resolvieron, como es el manejo de subprocesos, necesario para poder llamar la compilación desde el programa IDE y el resaltado de sintaxis. En éste último hay demasiadas cosas que hacer, puesto que la sintaxis de Lilypond incluye muchos detalles que sería importantes destacar a la hora de ver el código fuente de la partitura.

Un gran problema que hubo durante el desarrollo es que el objetivo principal de la IDE fue sacrificado en aras de obtener un resultado más importante desde el punto de vista del desarrollo: la compilación del códifo fuente a cambio de la visualización de la partitura en la pantalla.

El proceso de mostrar una partitura es uno delicado, a menos que se quiera mostrar algo burdo que pueda hacerse más simple a mano. Incluye el despliege de las líneas del pentagrama, las cabezas de las notas, las plicas, las llaves, las medidas de compaces, las indicaciones de expresión, etc. Son muchos detalles que había muy poco tiempo para cubrir, pero que con el tiempo se esperan alcanzar para poder permitir generar el código para la partitura en Lilypond a través de una interfaz gráfica amigable y natural para el músico, que es la visualización del pentagrama y las notas sobre él.

6.1. Desarrollo por Realizar al Corto Plazo

Si bien la partitura es la razón fundamental de este proyecto, antes de llegar a eso, para poder tener algo con lo que empezar bien, hay que realizar ciertos desarrollos fundamentales por ahora. Entre estos desarrollos destacan por su urgencia en una IDE:

- La configuración del programa, lo que incluye las aplicaciones preferidas (pudiendo estas ser seleccionadas como archivos o como línea de comandos), los colores y estilos preferidos del resaltado.
- La integración entre la IDE y lo que Lilypond puede hacer, e.g., la generación de plantillas básicas, el ingreso de comandos de Lilypond a través de menús, etc.
- Un detalle primordial es la implementación de una interfaz de multiples documentos (MDI), en la que puedan editarse varios archivos en la misma ventana, algo muy importante, ya que Lilypond permite la inclusión de otros archivos fuente en su código.

Un desarrollo completamente ideal es hacer de éste proyecto parte del proyecto Lilypond completo, y hacer que éste venga incluido en futuras instalaciones del programa base, logrando asíla última integración entre el código y su interfaz gráfica o su IDE, según lo que se haya logrado para el momento.