CAZABE: HERRAMIENTA DE AUTOR LIBRE PARA FLASH

CAZABE: FREE AUTHORING TOOL FOR FLASH

Darien Alonso Camacho

Joven Club de Computación y Electrónica, Cuba, darienad030111@vcl.jovenclub.cu, Colón #222A Sagua la Grande, Villa Clara

RESUMEN

El presente documento refleja un estudio realizado acerca de las opciones para la creación de animaciones y software multimedia usando tecnología flash sobre plataformas GNU/Linux. Dada la ausencia de un entorno de desarrollo integrado para flash en este sistema operativo, se comenzó un proyecto práctico e investigativo que permita suplir esta carencia. Se concibió un desarrollo colaborativo, de software libre y código abierto, con resultados satisfactorios en la implementación de la herramienta debido a que se obtuvo una versión preliminar pero funcional de la misma.

Cazabe puede ser útil para diseñadores, animadores y desarrolladores de software en general y además evitará el pago de licencias de programas propietarios de este tipo.

Palabras Clave: Flash, software libre, linux, código abierto

ABSTRACT

This paper outlines a research about the options for animations and multimedia software creation using flash technologies on GNU/Linux platforms. Because there is not an integrated development environment (IDE) for flash on this operating system, a practical and theoretical project was started to provide it. The purpose is a collaborative development of free and open source software, encountering good results with this approach because nowadays exist an immature but functional first version.

Cazabe could be useful to designers, cartoons makers and software developers. It might avoid paying for licence of proprietary software.

KeyWords: Flash, free software, linux, open source

1. INTRODUCCIÓN

Cuando en el año 2009 fueron creados los Grupos de Desarrollo de Software en los Joven Club de Computación y Electrónica (JCCE) de Cuba, se informó por parte de la dirección que el perfil de trabajo de los mismos sería el desarrollo de software multimedia y juegos. La plataforma flash es ampliamente usada en entornos Microsoft Windows para la creación de este tipo de programas, sin embargo, al intentar usar únicamente software de código abierto para este propósito, se llegó a la conclusión de que no existe una herramienta, distribuida bajo alguna licencia de software libre, que sea lo suficientemente usable para cumplir en tiempo con un proyecto de envergadura. El proyecto Cazabe tiene como objetivo lograr un entorno de desarrollo integrado que permita editar gráficos vectoriales y código ActionScript en un mismo ambiente y contribuir a la migración hacia software libre en Cuba y el resto del mundo.

2. METODOLOGÍA

A continuación se describen algunas tecnologías y herramientas libres para el desarrollo flash sobre GNU/Linux así como algunos proyectos anteriores con metas similares

a las de Cazabe. Se explican los principales retos de implementación y las soluciones brindadas.

2.1 Estado del arte

Con un estudio acerca de la tecnología flash se pudo constatar que el formato SWF es público y está bien documentado [1], aunque Adobe controla las modificaciones que se le realizan al mismo; por lo tanto crear o usar software que lea o escriba el formato SWF no constituye un delito.

En entornos GNU/Linux existen varios reproductores capaces de leer el formato SWF, como gnash (http://www.gnu.org/software/gnash/) y swfdec (http://swfdec.freedesktop.org/), hasta el momento soportan solamente ActionScript 2.0 y algunas características de ActionScript 3.0.

Existe, además, un amplio grupo de proyectos libres relacionados con la tecnología flash que se encuentran hospedados en www.osflash.org, muchos pensados para propósitos específicos.

En el transcurso del tiempo ha habido intentos de lograr un clon de Macromedia/Adobe Flash (Flash) para Linux. QFlash fue uno de ellos, lográndolo principalmente en la parte de la interfaz gráfica, que es muy parecida, pero aún así carece de muchas funcionalidades [2]. Por otro lado, F4L [3] fue una aplicación informática que se desarrolló como alternativa a Flash para GNU/Linux, sin embargo esta no fue muy difundida. Más tarde QFlash se unió con F4L para crear un proyecto conjunto llamado UIRA que dejó de funcionar a mediados de 2007 [4].

El caso de Ktoon es otro, el objetivo no es realizar un clon de Flash. Ktoon es una herramienta de diseño y creación de animaciones 2D que no persigue ser el reemplazo de Adobe Flash en Linux ni brindar la posibilidad de crear aplicaciones interactivas, está centrado solamente en la animación [5].

En cuanto a los compiladores de ActionScript, utilidad indispensable si se desean aplicaciones en flash, sobresalen dos: mtasc y haXe. HaXe es capaz de compilar ActionScript 2.0 y 3.0 [6] mientras que mtasc solo ActionScript 2.0 [7].

2.2 Herramientas de desarrollo propuestas

Para el desarrollo de aplicaciones de escritorio sobre GNU/Linux, como es este caso, existen varias opciones. En este epígrafe se detallan algunas de las posibilidades y se justifica la elección final.

2.2.1 Lenguajes de programación

Java

Java es un lenguaje de programación completamente orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Java es un lenguaje multiplataforma, aunque esta portabilidad depende, en gran medida, de la máquina virtual, es decir, sin la máquina virtual de Java instalada tal portabilidad no existe.

Permite usar las mismas funcionalidades de escritorio para hacer programas basados en "WEB". Presenta una arquitectura estándar de desarrollo orientado a Internet, más que un lenguaje, es toda una plataforma.

No tiene punteros y no es necesario destruir los objetos, el sistema posee un recolector automático de basura, de forma tal que el programador determina cuándo se crean los objetos y el entorno en tiempo de ejecución de Java (JRE) es el responsable de gestionar el ciclo de vida de los objetos. Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En tiempo de ejecución, el rendimiento de una aplicación Java depende más de la eficiencia del compilador, o la JVM, que de las propiedades intrínsecas del lenguaje. El bytecode de Java puede ser interpretado en tiempo de ejecución por la máquina virtual, o bien compilado al cargarse el programa, o durante la propia ejecución, para generar código nativo que se ejecuta directamente sobre el hardware. Si es interpretado, será más lento que usando el código máquina intrínseco de la plataforma destino. Si es compilado, durante la carga inicial o la ejecución, la penalización está en el tiempo necesario para llevar a cabo la compilación. El uso de un recolector de basura, añade una sobrecarga que

puede afectar al rendimiento, o ser apenas apreciable, dependiendo de la tecnología del recolector y de la aplicación en concreto.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre [8].

C++

C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup [9]. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto entre las herramientas de desarrollo de aplicaciones.

C/C++ es multipropósito, diseñado para desarrollar los más diversos tipos de aplicaciones. Se le conoce como un lenguaje híbrido, dicha denominación proviene de que soporta los paradigmas de programación estructurada, la programación genérica y la programación orientada a objetos.

Su portabilidad es excelente; una gran cantidad de sistemas operativos están escritos en C. A pesar de tener una sintaxis complicada y de lo engorroso que resulta el manejo directo de memoria a través de los punteros, se considera muy potente porque permite programar tanto a alto como a bajo nivel.

Una particularidad de C++ es la posibilidad de redefinir los operadores (sobrecarga de operadores), y de poder crear nuevos tipos que se comporten como tipos fundamentales, además soporta la herencia múltiple.

El tiempo de desarrollo de aplicaciones en este lenguaje es mayor comparado con otros como Java y Python, debido a que actividades como la recolección de basura y el tipado caen sobre el programador. El hecho de ser compilado íntegramente le confiere mayor velocidad de ejecución.

Python

Python es un lenguaje de programación interpretado, creado por Guido Van Rossum a principios de los años 90. Se desarrolla como un proyecto de código abierto, administrado por la Python Software Foundation.

Es un lenguaje de propósito general, aunque está limitado en el alcance de lo que puede acceder sobre el sistema; no es adecuado para la programación de bajo nivel.

Python es un lenguaje multiparadigma, permite varios estilos: programación orientada a objetos, programación estructurada y programación funcional. Posee una sintaxis sencilla, su tipado es dinámico y soporta herencia múltiple.

El intérprete de Python está disponible en multitud de plataformas (UNIX, Solaris, Linux, DOS, Windows, OS/2, Mac OS, etc.), de esta forma los programas escritos sin usar bibliotecas dependientes de una plataforma específica podrán correr en todos estos sistemas sin grandes cambios [10].

El hecho de ser interpretado hace que ahorre un tiempo considerable en el desarrollo del programa, pues no es necesario compilar ni enlazar. El intérprete se puede utilizar en modo interactivo, lo que facilita experimentar con características del lenguaje, escribir programas desechables o probar funciones durante el desarrollo del programa.

Python no es aconsejable para aplicaciones en las que el rendimiento sea crítico, es un lenguaje de scripts, no genera ejecutables y por lo tanto es más lento que un lenguaje compilado.

2.2.2 Bibliotecas gráficas

GTK+

GTK+ son las siglas de GIMP Toolkit, es una biblioteca que permite crear interfaces gráficas de usuario, se distribuye bajo la licencia pública general (GPL), lo que hace de GTK+ un producto completamente libre.

GTK+ es multiplataforma, se ha extendido hasta Microsoft Windows y muchos derivados de Unix, como Linux y Mac OS. Está escrita en C y tiene extensiones en varios lenguajes como C++, Python, Perl y muchos más.

Se empleó inicialmente en el proyecto Gnu Image Manipulation Program Tool Kit, por eso su nombre:

GTK+, se ha extendido rápidamente por su estabilidad y una de las implementaciones que más se ha usado es la de C++, llamada Gtkmm [11].

Gtkmm es la implementación oficial de GTK+ escrita en C++, le adiciona a GTK+ las potencialidades del paradigma orientado a objetos, la herencia para crear nuevos componentes, polimorfismo, manejo de memoria para construir y destruir objetos, elimina el uso de las macros de C y muchas otras mejoras.

GTK+ depende de otras bibliotecas que hacen de GTK+ un éxito total:

- Glib, una biblioteca de propósito general, no destinada a interfaces gráficas en sí, provee tipos de datos, macros y utilidades de conversión, tratamiento de cadenas y abstracciones muy útiles
- Pango, se encarga de la manipulación de los textos internacionalizados, provee widgets que se encargan de la representación de los textos
- Atk, es el paquete de accesibilidad, provee un conjunto de interfaces que permiten a las interfaces de usuario interactuar con las tecnologías de accesibilidad
- Gdk, es la capa de abstracción que permite a GTK+ ser portable a múltiples plataformas
- GTK+, ella en sí contiene las definiciones de todos los widgets

GTK+ tiene soporte para bases de datos, el proyecto Gnome-DB ofrece una arquitectura basada en CORBA que permite acceso totalmente transparente a distintas fuentes de datos, incluyendo datos que se encuentren en servidores LDAP o en ficheros XML, entre otros. GTK+ provee también mecanismos para comunicar aplicaciones de red mediante los protocolos TCP, UDP y para el trabajo con la tecnología XML.

Qt

Qt es un framework escrito en C++ que permite crear aplicaciones con interfaces gráficas. Qt es totalmente orientado a objetos, fácil de usar, extensible y multiplataforma (soportada en Windows, Unix y derivados) [12].

Qt es un producto creado por la compañía Trolltech. Se comercializa una versión de Qt no libre para los sistemas Windows, además distribuye una versión completamente libre y gratuita para los sistemas Unix, Linux y derivados, que se distribuye bajo licencia GPL y QPL, que está aprobada por la Fundación de Software Libre (FSF).

Qt provee mecanismos para la visualización de imágenes vectoriales y raster, utilizando widgets para hacer el render y el procesamiento de las operaciones principales: zoom, escala, rotación, traslación y pan. Provee también mecanismos para la edición de las imágenes y los objetos gráficos en general, entre ellos cambio de brocha, pincel, estilo de los textos, terminaciones de brocha, estilos de línea, entre otros.

Este toolkit incluye un poderoso módulo que se encarga del manejo de archivos en formato XML, proporciona un mecanismo de análisis de la estructura de un documento XML (parser) usando SAX2 y el modelo de objetos (DOM).

El framework de Qt, al igual que ha hecho GTK+, se ha extendido a otros lenguajes además de C++, existen implementaciones como PyQt, Qt Jambi, PerlQt, QtRyby para Python, Java, Perl y Ruby respectivamente.

Qt incluye módulos para la representación gráfica 3D utilizando OpenGl, otros para integrarse con otras aplicaciones utilizando los protocolos de red TCP, UDP y HTTP, para desarrollar aplicaciones relacionadas con bases de datos con soporte para MySQl, PostgreSQL, MSSQL, Oracle, SQLite e Interbase.

2.2.3 Justificación de la elección

Como resultado del análisis de los lenguajes de programación se decidió usar C++ porque posee una serie de propiedades que no se encuentran disponibles en otros lenguajes de alto nivel como la sobrecarga de operadores y la programación de bajo nivel; por ser compilado su velocidad de ejecución es mayor que la de aplicaciones escritas en Java o Python y a la hora de elegir un lenguaje, la necesidad de velocidad

de la aplicación inclinaría la balanza hacia C++.

Con relación a la biblioteca gráfica a emplear se optó por Qt teniendo en cuenta la experiencia personal de su desarrollador en primer plano y porque brinda mecanismos para la edición de imágenes y objetos gráficos en general; cambio de brocha, pincel, estilo de los textos, terminaciones de brocha, estilos de línea, entre otros, además de incluir un poderoso módulo que se encarga del manejo de archivos en formato XML, proporciona un de análisis de la estructura de un documento XML (parser) usando SAX2 y el modelo de objetos (DOM). Es completamente orientado a objetos y está escrito en lenguaje C++, heredando las bondades del mismo.

2.3 Descripción del problema y solución

En esencia, para crear una película flash íntegramente libre, se necesita una herramienta libre que auxilie en el diseño de la escena, dígase inclusión de texto, imágenes, sonido, etc. Por otra parte, se hace indispensable un compilador libre de código ActionScript. La filosofía de desarrollo con flash sobre Linux es un tanto diferente al desarrollo sobre los ambientes de Microsoft. Lo habitual es crear una biblioteca de componentes que contenga los sonidos, dibujos y textos que posteriormente serán incluidos en la película swf definitiva, en las coordenadas deseadas, usando el lenguaje ActionScript.

Para satisfacer la necesidad de automatizar el diseño gráfico existe una herramienta llamada swfmill (http://swfmill.org/) que genera una biblioteca de componentes a partir de un archivo XML que contiene las rutas de los sonidos e imágenes que se desean desplegar en la escena.

En cuanto a los compiladores de ActionScript se conoce que haXe es capaz de compilar ActionScript 2.0 y 3.0 mientras que mtasc solo ActionScript 2.0, haciendo que la balanza se incline por haXe. Tanto swfmill como haXe se ejecutan en modo consola, haciendo tedioso el trabajo con los mismos. A continuación se presenta un ejemplo práctico de lo descrito:

```
<? xml version="1.0" encoding="ISO-8859-1"?>
<movie width="298 "height="298" framerate="12">
 <background color="#ffffff"/>
 <frame>
 <library>
 <clip id="main">
 <frame name="1">
 <clip id="id0.png" import="temp_images/id0.png"/>
 <place id="id0.png" name="nombre"  x="0"  y="0"/>
 </frame>
 </clip>
 </library>
 </frame>
 </movie>
```

El texto que antecede es una muestra de un archivo XML de nombre "ejemplo.xml", que especifica que la película resultante tendrá una anchura y una altura igual a 298, color de fondo blanco y clip de película principal de un solo fotograma conteniendo una imagen (id0.png) que será proyectada en las coordenadas 0,0. Para obtener la biblioteca de componentes a partir de este xml es necesario, desde la línea de comandos, ejecutar [13]:

```
swfmill  simple ejemplo.xml componentes.swf
```

Una vez obtenida la biblioteca de componentes será necesario adjuntar dichos componentes desde código ActionScript como sigue:

```
//Fichero película.hx
var a:MovieClip = lib.current.attachMovie("main","asd", 0);
a.gotoAndPlay("1");
```

Donde "main" es el nombre (id) del clip principal que se especificó en ejemplo.xml. Este código ActionScript se compila así [6]:

```
haxe -main Main -swf-lib componentes.swf –swf movie.swf
```

Finalmente la película será "movie.swf".

De esta manera surge la idea de crear Cazabe; una aplicación con una interfaz gráfica de usuario agradable y fácil de operar, que use swfmill y haXe como herramientas auxiliares para la generación de la biblioteca de componentes visuales y la compilación de ActionScript respectivamente. La meta inicial era clara y poco ambiciosa: automatizar el proceso de creación del XML para obtener la biblioteca de componentes y la película final a partir de esta.

En este punto existían tres retos primarios:

- Generar un xml a partir de una escena para pasarlo como argumento al ejecutable de swfmill para que este genere la biblioteca de componentes.
- Comunicarse desde la aplicación con los ejecutables de swfmill y el compilador haXe.
- Lograr proyectar la película creada en un reproductor de flash.

Con el objetivo de dar solución a los tres retos descritos, se implementó, usando el marco de trabajo Qt4, una interfaz gráfica de usuario multiplataforma que incluye una escena de dibujo y un pequeño conjunto de herramientas de diseño necesarias para demostrar la viabilidad del proyecto.

El proceso que lleva a cabo Cazabe para conseguir un SWF es como sigue: los dibujos y textos incluidos en la escena se convierten a un formato de imagen, generalmente jpg, y se crea un fichero XML que contiene las rutas de dichas imágenes. Usando un objeto de la clase QProcess del framework Qt4 desde código C++, es posible ejecutar la herramienta swfmill y pasarle como argumento el XML construido previamente, como resultado swfmill produce un fichero en formato SWF que se puede utilizar como biblioteca de componentes.

Una vez que la biblioteca posee las imágenes y la información del tamaño de escena, cantidad de fotogramas y color de fondo deseado, solo queda adjuntar el clip de película y reproducirlo con código ActionScript que será compilado con haXe pasando los argumentos indicados, ejecutándolo de manera similar a swfmill e igualmente usando utilidades del framework Qt4. El fichero SWF resultante de la compilación se llama desde algún reproductor de flash y se visualiza la película obtenida.

El engorroso proceso de crear la biblioteca de componentes y compilar código ActionScript descrito anteriormente, que era necesario hacer a mano desde la línea de comandos es, precisamente, lo que Cazabe es capaz de automatizar. De esta forma quedaron solucionados los tres retos iniciales y se obtuvo un programa que sentó las bases de un posterior desarrollo.

Lo descrito hasta el momento corresponde a una fase de Cazabe eminentemente investigativa y de búsqueda de las mejores alternativas. Luego de implementar la primera solución se detectaron un conjunto de deficiencias y desventajas:

a. En el proceso de creación del archivo SWF visto anteriormente, se menciona que los dibujos y textos incluidos en la escena se convierten a un formato de imagen y se crea un fichero XML que contiene las rutas de dichas imágenes. Este enfoque posee el problema de que los dibujos contenidos en el SWF final no son imágenes vectoriales, sino rasterizadas y trae como consecuencia distorsión en los contornos de las figuras (Fig. 1), por lo tanto la calidad de la película resultante disminuye en gran medida.
b. No brindaba la opción de insertar código ActionScript en la línea de tiempo y existía gran incertidumbre acerca de la posibilidad real de implementarla debido al conjunto de programas reutilizados.
c. Lograr que Cazabe soportara las funcionalidades de creación de símbolos (Gráficos, Botones y Clip de películas) iba a ser engorroso y en el peor de los casos imposible.

Fig. 1: Irregularidad en el contorno

Con el objetivo de resolver esta situación se investigó acerca de bibliotecas que permitan crear SWFs directamente, encontrando en el proyecto Ming (http://www.libming.org) la solución ideal.

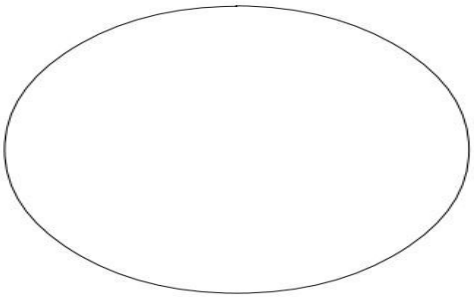

Fig. 2: Contorno sin irregularidad

La biblioteca libming permite crear una película flash que contenga dibujos, sonidos, scripts de código y símbolos usando lenguajes como C, C++, php y python.

Esta segunda implementación de Cazabe brinda una interfaz gráfica de usuario multiplataforma que incluye una escena de dibujo, un pequeño conjunto de herramientas de diseño básicas y un editor de texto que facilita la edición de código ActionScript. Los dibujos, textos, símbolos y animaciones son exportados directamente a un fichero SWF usando funciones y objetos de la biblioteca libming desde el lenguaje de programación C++. El SWF resultante se visualiza con el reproductor libre de flash gnash.

Esta solución deja de usar swfmill y haXe como base fundamental para construirse sobre libming. Los elementos gráficos que están en cada fotograma de la línea de tiempo de Cazabe son dibujados directamente en un archivo .swf con la ayuda de libming, por lo que no persiste el problema de la irregularidad en el contorno de las figuras (Fig. 2). Las porciones de código ActionScript son insertadas en su sitio correspondiente de la línea de tiempo de la película o en los símbolos, permitiendo un mayor dinamismo. La creación de símbolos (Gráficos, Botones y Clip de películas) es posible mediante la interfaz de usuario de Cazabe. Las próximas versiones darán soporte para haXe en proyectos de solo código.

3. RESULTADOS Y DISCUSIÓN

A raíz de este trabajo se logró un software base que a mediano plazo mejorará la experiencia de los desarrolladores de aplicaciones interactivas con flash sobre sistemas operativos GNU/Linux.

La posibilidad de insertar textos en una escena, de dibujar algunas figuras básicas usando los colores deseados y visualizar inmediatamente el resultado con un reproductor de flash es un hecho (Fig. 3 y Fig. 4). Así mismo incrustar código ActionScript en un determinado fotograma de la línea de tiempo es posible gracias a un editor de texto que brinda algunas funcionalidades de completamiento (Fig. 5).

A pesar de que los resultados obtenidos son alentadores, Cazabe no deja de tener limitaciones y errores de programación ya conocidos, que a medida en que avance su desarrollo se irán puliendo y mitigando. Lo cierto es que los parciales que aquí se presentan demuestran que lograr una herramienta de este tipo no es un mito.

Una de las características más importantes en un programa de creación es la posibilidad de guardar lo que se ha hecho hasta un determinado momento, funcionalidad que aún no está presente en Cazabe. Existen dos formas de lograr esto: crear un formato de proyecto propio para Cazabe o dar soporte a los formatos de Macromedia y Adobe. La primera opción, aunque no es un completo error, tiene el inconveniente de no ser compatible con el resto de los formatos existentes.

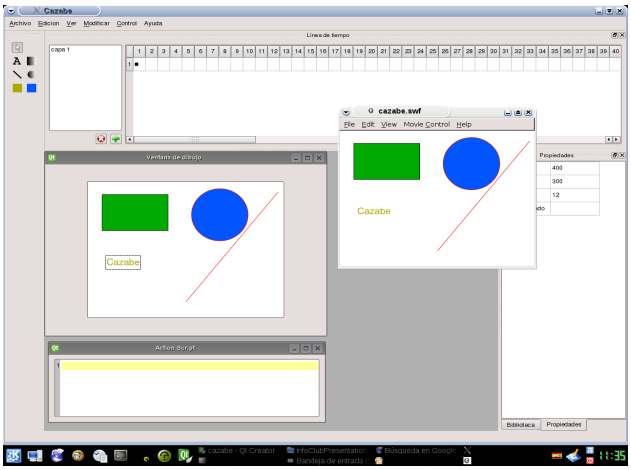

Fig. 3: Interfaz principal de Cazabe

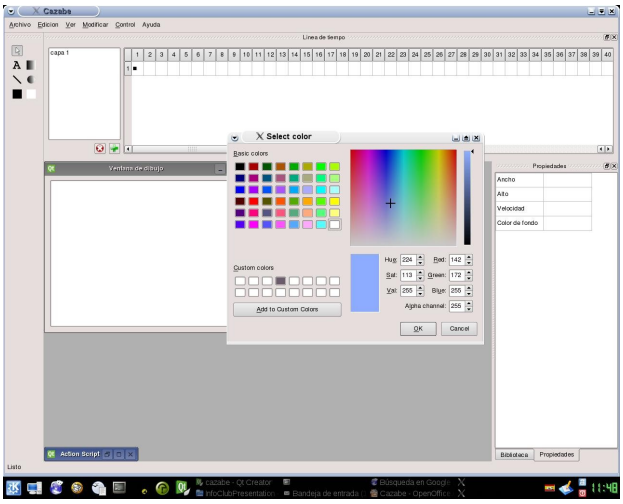

Fig. 4: Selección de colores en Cazabe

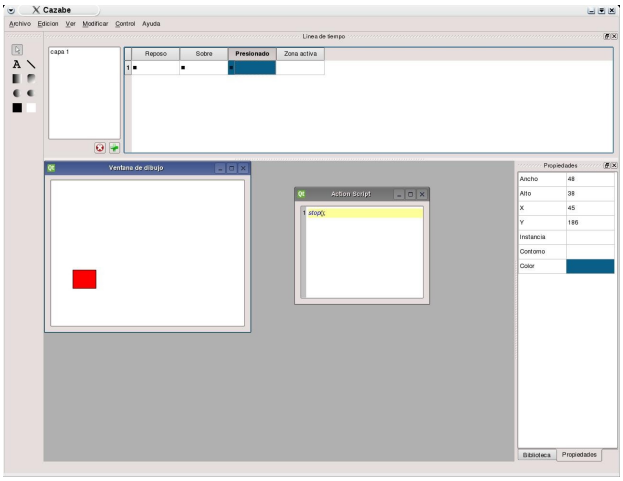

Fig. 5: Inclusión de código en un botón

La segunda significa exportar e importar los formatos FLA y XFL. El formato FLA es un formato binario cuya especificación oficial Adobe nunca ha publicado. Lo poco que se conoce acerca de él ha sido a través de ingeniería inversa. Dar soporte sin errores a este formato es una tarea de gran envergadura si es que se llega a lograr. Por otro lado, XFL es un formato de Adobe basado en XML cuya especificación oficial no se ha publicado aún, sin embargo, al ser un formato abierto basado en XML es menos complicado lograr exportar e importar este formato. El desarrollo inmediato pretende enfocarse en soportar el formato XFL. En primer lugar es necesario que Cazabe sea capaz de crear/leer un archivo con las especificaciones de este formato y posteriormente es de vital importancia implementar un compilador que transforme un XFL a un SWF.

Es muy importante tener en cuenta los asuntos legales de la aplicación debido a que su alcance puede sobrepasar los límites nacionales. Como parte de su Open Screen Project Adobe hizo público [1] el formato .swf, por lo que no constituye un delito usar o crear herramientas que lean o publiquen este formato. Cazabe se ha implementado usando las siguientes herramientas y componentes:

- Qt4, Edición de Código Abierto, para asistir en la creación de la interfaz gráfica de usuario. Qt es distribuido bajo tres licencias, en este caso es usado bajo GPL.
- La codificación se llevó a cabo usando C++ y para la generación de los binarios el compilador g++.
- Se usó la biblioteca libming para la generación del archivo en formato SWF. Esta biblioteca es distribuida bajo la licencia LGPL.
- El desarrollo de la herramienta ha sido sobre un sistema operativo Debian GNU/Linux 5.0 (Lenny).
- El reproductor gnash para visualizar la película en formato SWF. Gnash es distribuido bajo GPL v3.

Por todo lo expuesto se concluye que Cazabe es software libre íntegramente y no incurre en delitos de violación de patentes o licencias de software. Cazabe será distribuido según los términos de la licencia GNU GPL versión 2.0.

Se implementó un entorno de desarrollo integrado que permite editar gráficos vectoriales y código ActionScript en un mismo ambiente que contribuye a la migración hacia software libre en Cuba y el resto del mundo.

4. CONCLUSIONES

Después de un estudio, se seleccionaron las herramientas y tecnologías a emplear para desarrollar la herramienta de autor. Ante determinados fallos e ineficiencias se brindaron mejores soluciones. Se ha obtenido una primera versión ejecutable del programa que funciona tanto sobre plataformas Windows como sobre distribuciones GNU/Linux. La película resultante del uso de Cazabe se proyecta en un reproductor

de flash, lo que permite ver de forma instantánea los cambios realizados a la misma.

5. AGRADECIMIENTOS

Agradezco a la Instructora Viviana Rodríguez Martínez del Joven Club de Computación y Electrónica Sagua # 1 por la ayuda prestada para el aprendizaje de la herramienta Macromedia Flash 8 y por su apoyo e interés constantes.

6. REFERENCIAS BIBLIOGRÁFICAS

[1]Adobe. (2007).*SWF File Format Specification*. [En línea]. Disponible en:
http://download.macromedia.com/pub/flash/flash_file_format_specification.pdf

[2] http://qflash.sourceforge.net/webpage/

[3] http://f4l.sourceforge.net/

[4] 2007; http://www.unfreeze.net/?page_id=52

[5] "Sobre KTooN." 2010; http://www.ktoon.net/portal/es/acerca

[6] "Comenzando con haXe/Flash." 2007; http://haxe.org/doc/start/flash

[7] http://www.mtasc.org/

[8] Stallman, R. "Libre pero encadenado. La trampa del java." 2004; http://www.gnu.org/philosophy/java-trap.es.html

[9] Stroustrup,B. (1985). *"A tour of C++."* [En línea]. Disponible en: http://www.research.att.com/~bs/3rd_tour.pdf

[10] Python Software Foundation. "About Python." 1990-2010; http://python.org/about/

[11] Fundación GTK. "Documentación Oficial de Gtkmm." 2005; http://www.gtkmm.org/gtkmm2/docs/

[12] Trolltech. "Elementos técnicos del producto Qt." 2006; http://www.trolltech.com/products/qt/features/index

[13] Winterhalder, M. "Using swfmill to create SWFs without Flash." 2005; http://swfmill.org/doc/using-swfmill.html

6.1 Otras bibliografías consultadas

Ayuda de Qt. Qt4 Assistant. Disponible con la instalación del framework de Qt.

Prechelt, L. (2000). "An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl." [En Línea]. Disponible en: http://page.mi.fu-berlin.de/prechelt/Biblio/jccpprt_computer2000.pdf