publicando juegos libres

Miriam Ruiz <miriam@debian.org>

quién soy

Miriam Ruiz

ingeniera

feminista

desarrolladora de Debian

editora de Barrapunto

fundadora y coordinadora del

Debian Games Team

defensora del Software Libre

de qué voy a hablar

- las licencias de Software Libre
- la base sobre la desarrollar
- las creación del juego
- el desarrollo y los errores habituales
- la publicación y distribución
- la comunidad

de qué NO voy a hablar

- cómo se diseña un juego
- cómo se programa
- ingeniería del software
- paradigmas de desarrollo
- lenguajes de programación
- sistemas operativos
- organización de equipos de trabajo
- juegos privativos

desarrollo de un juego libre

- idea, concepto y diseño inicial
- elección de la arquitectura y las herramientas
- selección de la licencia
- recopilación de componentes
- desarrollo del código y elementos artísticos
- publicación del código fuente
- integración en los diferentes sistemas
- distribución de los binarios
- comunidad de desarrollo
- mantenimiento, bugs y desarrollos posteriores

eligiendo la licencia

la licencia

es imprescindible para poder utilizar, distribuir y modificar el software

todo el mundo odia los legalismos, pero son necesarios

qué es una licencia

conjunto de **términos o condiciones** de uso que otorgan los titulares de los derechos de autor para poder **utilizar, modificar y distribuir** el software en una forma determinada y de conformidad con **unas condiciones convenidas**

libertades esenciales

usar el programa con cualquier propósito

estudiar y modificar el programa

copiar, distribuir y difundir el programa

modificar el programa y publicar los cambios

diferencias entre licencias

atribución

sencillez

copyleft: GPL, LGPL, MPL/CDDL

protección contra **patentes**

protección contra **tivoisación**

cláusula de **no garantía**

limitación de responsabilidad

compatibilidad con otras licencias

selección de la ley a aplicar

integridad del código fuente original

protección por copyright del propio texto de la licencia

señalar las modificaciones

licencias libres

las más sencillas: **BSD**, ISC, MIT/X Windows/Expat, Zlib

la más popular: **GPL**

para bibliotecas de código: **LGPL**

otras licencias: Apache 2.0, MPL, CDDL, EPL, ...

Creative Commons: licencias a la carta

reconocimiento ("by" o attribution)

no comercial ("nc" o non commercial)

sin obra derivada ("nd" o no derivative works)

compartir igual ("sa" o share alike)

6 "sabores": **by**, by-nc, by-nd, by-nc-nd, **by-sa**, by-nc-sa

eligiendo la licencia

- qué permisos y obligaciones quiero para mi juego
  - elegir las librerías y componentes compatibles
- elegir los componentes artísticos que nos sirvan
- dejar muy claro documentada la licencia y versión
  - condicionar los aportes externos a esa licencia

por qué Software Libre

- porque es la mejor forma de **difundirlo** sin invertir muchísimo en publicidad
- porque hay un montón de **elementos libres** sobre los que nos podemos apoyar para crearlo
- porque podemos contar con el apoyo de mucha gente de la **comunidad** para desarrollarlo
- porque podemos extenderlo a **muchos sistemas** para los que no tendríamos capacidad si no
- porque el juego **sobrevivirá en el tiempo**

sobre qué construyo

el sistema operativo

es el elemento que actúa de intermediario entre el hardware y el software, y provee los sistemas y librarías básicas para que podamos ejecutar programas sobre él

hay numerosos sistemas operativos, tanto para ordenadores como para videoconsolas, móviles, etc. algunos de ellos son privativos y otros no

hay librerías, frameworks y entornos de ejecución que nos permiten **abstraernos del sistema operativo** sobre el que se va a ejecutar el programa

la arquitectura

tecnologías de las que vamos a depender: C/C++, Posix, Java, Python, OpenGL, SDL, Flash, Perl, Crystal Space, Ogre, .Net, DirectX, VB, Delphi, XULRunner, ...

entorno de desarrollo: modificación del programa, compiladores soportados, herramientas de transformación, edición de contenidos, ...

componentes: compatibilidad de la licencia, portabilidad de las librerías, estabilidad, mantenimiento, ...

la toolchain

conjunto de **herramientas, binarios y librerías** necesarias para realizar la compilación.

dependiendo de las herramientas de desarrollo elegidas se podrá usar en unos sistemas operativos u otros

si existe una **toolchain libre** en la que se pueda compilar, se puede compilar para **cualquier sistema**

si se necesita una **toolchain privativa**, solamente se podrá portar para algunos sistemas, y estará **condicionada a la empresa** que la fabrique

la parte artística

no se puede coger cualquier cosa que encontremos por ahí: copyright, licencia, condiciones

será tan libre, distribuible, portable, etc. como lo sea la **parte más restrictiva** de las piezas que lo componen

elementos habitualmente problemáticos: texturas, mapas, modelos, fotos, tipos de letra, sonidos, música, iconos, etc.

la creación

iniciando el proyecto

el proyecto no es la idea: hay que **desarrollarla**

otra opción: unirse a un **proyecto ya existente**, o construir a partir de su código

no vamos a conseguir colaboración externa antes de tener **algo jugable**. los grandes planes se quedan en vaporware

hay numerosas forjas donde poder iniciar un proyecto libre: sourceforge, savannah, ...

buscando colaboración

hacer que **colaborar sea fácil**: curva de entrada aceptable, mantenibilidad del código, documentación

la mayoría de las **colaboraciones** van a ser **puntuales**, aportando parches, gráficos, mapas, etc. pero sin un compromiso de continuidad

herramientas colaborativas y de comunicación: repositorio de código (svn, git, bzr, hg, cvs, ...), wiki, listas de correo o foros, etc.

y si me lo estropean

la colaboración de la comunidad mejora la calidad de los proyectos (cathedral vs. bazaar)

no puedes evitar que alguien lo modifique por su cuenta (el derecho al fork)

la propia comunidad selecciona el proyecto mejor

¡¡cuidadín: errores!!

desarrollo incorrecto

- idea, concepto y diseño inicial
- empezar a programar sin plantearse las cosas
- se cogen muchas cosas de "por ahí"
- publicación de los binarios
- alguien escribe solicitando el código fuente
- selección o redacción de una licencia propia
- publicación del código fuente
- me olvido del juego
- integración en los diferentes sistemas
- comunidad de desarrollo ajena

endianess

unidad mínima de almacenamiento: byte,

cómo almacenar un número multibyte: 1234:

big-endian ("de mayor a final"): del + a - significativo

1-2-3-4 (ej: Motorola)

little-endian ("de pequeño a final"): de - a + significativo

4-3-2-1 (ej: Intel)

algunos sistemas pueden trabajar con ambos formatos: ARM, PowerPC, DEC Alpha, PA-RISC, MIPS

middle-endian (ni big-e ni little-e): ordenaciones extrañas como 3-4-1-2 ó 2-1-4-3, o incluso decimal empaquetado

tamaño de palabra

es el **número de bits** que son manejados como un conjunto por la máquina

el **valor numérico típico** manipulado por el ordenador es habitualmente el tamaño de palabra
ordenadores modernos: 16, 32 ó 64 bits

define el máximo valor numérico que la CPU puede usar directamente (overflow)

lo que NO se debe hacer

ejemplo 1:

```
int i = 0;
myfile.read(&i, sizeof(int));
```

ejemplo 2:

```
void *pointer;
int value;
value = (int)pointer;
```

empaquetamiento de datos

cuando se almacena información en la memoria del ordenador no se hace linealmente, sino que a menudo se desperdicia parte de la misma para **optimizar** el acceso a la misma

la **estructura de los datos en memoria** depende de la arquitectura, del compilador, de las opciones de compilación y, en general, salvo que el lenguaje elegido garantice una estructura determinada, está **fuera del control** de quien programa.

lo que NO se debe hacer

```
typedef struct {
   uint32_t moment;
   Player player[ cMaxPlayers ];
   uint8_t grid[ cGridX ][ cGridY ];
} Game;

FILE   *file;
file = fopen( "game.dat", "w" );
if ( file == (FILE *) NULL )
   return ERROR;
fwrite( game, 1, sizeof( Game ), file );
fclose( file );
```

internacionalización (i18n)

para quienes lo usan, es muy importante que el software esté en su idioma. en muchos casos incluso imprescindible.

ha de ser posible **realizar traducciones** del programa

respetar la forma de trabajar de traductores: **formatos y herramientas estándar**: GNU gettext PO, XLIFF (XML)

no inventar formatos nuevos

en sistemas multiusuario, **diferentes usuarios** puedan usar el mismo programa en **diferente idioma**

Codificación del texto

NO EXISTE EL TEXTO PLANO o texto sin formato

las codificaciones **ASCII, ANSI, etc. no son soluciones válidas** para idiomas diferentes del inglés

UNICODE: proporciona un número único para cada carácter o grafema, sin importar la plataforma o el idioma. **sin UNICODE no es posible soportar muchos idiomas**

diferentes codificaciones: **UTF-8**; UTF-7; UCS-2, UTF-16, UCS-4, UTF-32 (cuidado: endianess) y alguna otra

sistema de archivos

no es posible grabar cualquier archivo en cualquier directorio para cualquier usuario/a

separación de archivos: programa, datos de juego, configuración, cachés, logs

en muchos juegos es necesario que quien juega pueda **grabar datos**: estado de la partida, records, configuraciones, edición de mapas

no es válido considerar que todos los archivos del juego van a estar en el **mismo directorio**

la distribución

compilación

documentar las dependencias tanto de compilación como para la ejecución

permitir el **enlazado con las bibliotecas dinámicas** que vienen incluídas en las principales distribuciones ("duplicated libraries considered harmful")

elegir el **sistema de compilación** preferido de los ya existentes. no inventar uno nuevo.

cualquiera debe **poder compilar el juego** en su sistema

publicación

tarball que contenga de forma ordenada el **código fuente**, especificando de forma clara los datos respecto a la propiedad intelectual (**copyright**), la **licencia**, las **dependencias** y el **proceso de compilación**. los gráficos, mapas, documentación, etc. también pueden tener "código fuente": forma preferida de modificación

precompilado para ciertas plataformas: Windows, OS X

trabajo conjunto con quienes desarrollan las diferentes distribuciones para ayudarles a mantenerlo en ellas

empaquetamiento

existen numerosos **sistemas operativos diferentes**, y versiones y distribuciones de los mismos. Cada uno de ellos tiene **sus propias "reglas de juego"**

es imposible hacerlo perfecto en todas las arquitecturas posibles. a menudo desarrolladores de las diferentes distribuciones pueden ayudarnos, así como la propia comunidad

el mantenimiento de paquetes de distribuciones de Linux es **tarea de los mantenedores**, no de los creadores

difusión

anunciar el proyecto en páginas web especializadas (ej: http://happypenguin.org)

facilitar a usuarios y usuarias la tarea de descargárselo y probarlo

no le va a gustar a todo el mundo: críticas y quejas

"release early, release often"

la comunidad

un nuevo paradigma

el esquema de desarrollo **estilo "catedral"**, en el que se establece una jerarquía de decisiones, se está viendo cada vez más desfasada por un modelo que aporta mejores resultados en muchos casos: el **modelo "bazar"**

los nuevos medios tecnológicos están transformando la sociedad, de tal forma que cada vez van desapareciendo más los conceptos de producer y consumer, y va surgiendo uno nuevo: "prosumer"

herramientas comunitarias

- correo electrónico
- repositorio de código
  - sistema de control de versiones -
  (svn, git, bzr, hg, cvs, ...)
- wiki
- sistemas automatizados (buildds, qa, ...)
- página web o portal
- blog
- encuentros en persona

qué buscar en la comunidad

- **infraestructura** para el desarrollo (repositorios, sistemas de gestión de proyectos, espacios web,...)
- **elementos** que podemos utilizar **para construir**
- **críticas, muchas críticas**
- **aportaciones puntuales**: traducciones, diseño de fases, portabilidad a otras plataformas, extensiones de la funcionalidad
- **soporte técnico**: listas de correo, foros, irc
- ideas
- más **colaboradores/as** para el proyecto

el resultado

jugando libremente

a pesar de todos los problemas y dificultades, el hecho es que tenemos **motores libres multiplataforma excelentes**, librerías y **entornos de programación muy buenos**, y bastantes **juegos que merecen la pena**

las carencias que pueden tener los y las desarrolladoras originales de los juegos, quedan solucionadas por el apoyo y la asistencia de miembros y miembras de la comunidad

pero **siempre se puede mejorar**,
aún nos queda mucho camino por recorrer

preguntar es gratis

:)

Miriam Ruiz <miriam@debian.org>