COPIA LITERAL

Extracto del libro “GAMBAS, programación visual con Software Libre”, de la editorial EDIT LIN EDITORIAL S.L., cuyos autores son Daniel Campos Fernández y José Luis Redrejo.

LICENCIA DE ESTE DOCUMENTO

Se permite la copia y distribución de la totalidad o parte de esta obra sin ánimo de lucro. Toda copia total o parcial deberá citar expresamente el nombre del autor, nombre de la editorial e incluir esta misma licencia, añadiendo, si es copia literal, la mención “copia literal”. Se autoriza la modificación y traducción de la obra sin ánimo de lucro siempre que se haga constar en la obra resultante de la modificación el nombre de la obra originaria, el autor de la obra originaria y el nombre de la editorial. La obra resultante también será libremente reproducida, distribuida, comunicada al público y transformada en términos similares a los expuestos en esta licencia.

Edit Lin

GAMBAS

Programación visual con Software Libre

Autores
Daniel Campos
José Luis Redrejo

Prólogo
Benoît Minisini

ÍNDICE

CAPÍTULO 1: ¿QUÉ ES GAMBAS? 17

1. 1 El lenguaje BASIC: su historia 18
1. 2 Un entorno libre 21
1. 3 Elementos de Gambas 23
1. 4 Cómo obtenerlo 24
1. 5 Compilación y dependencias 26
1. 6 Familiarizarse con el IDE 27
- El primer ejemplo 29
- Mejor con un ejemplo gráfico 32
- Un poco de magia 35

1. 7 Sistema de componentes 36

CAPÍTULO 2: PROGRAMACIÓN BÁSICA 41

2. 1 Organización de un proyecto de Gambas 42
- Declaración de variables 42
- Subrutinas y funciones 45

2.2 Tipos de datos 49
Conversión de tipos 50
Matrices 52
2.3 Operaciones matemáticas 54
Operaciones lógicas 56
2.4 Manejo de cadenas 56
2.5 Control de flujo 60
IF... THEN... ELSE 60
Select 62
FOR 63
WHILE y REPEAT 64
Depuración en el IDE de Gambas 66
2.6 Entrada y salida de ficheros 68
2.7 Control de errores 72
2.8 Programación orientada a objetos con Gambas 73
2.9 Propiedades, Métodos y Eventos 79

CAPÍTULO 3: LA INTERFAZ GRÁFICA 81

3.1 Concepto 81
Partiendo de la consola 84
El entorno de desarrollo 85
3.2 Manejo básico de los controles 87
Posición y tamaño 87
Visibilidad 89
Textos relacionados 89
Colores 90
Ratón 92
Teclado 95
3.3 Galería de controles 96
Controles básicos 96

Otros controles básicos misceláneos 99
Listas de datos 100
Otros controles avanzados 101
3.4 Diálogos 101
La clase Message 101
La clase Dialog 104
Diálogos personalizados 106
3.5 Menús 111
3.6 Alineación de los controles 114
Propiedades de la alineación 114
Controles con alineación predefinida 117
Diseño de una aplicación que aprovecha
este recurso 117
3.7 Introducción al dibujo de primitivas 120

CAPÍTULO 4: GESTIÓN DE PROCESOS 125

4.1 La ayuda ofrecida por otros programas 125
4.2 Gestión potente de procesos 126
4.3 EXEC 127
Palabra clave WAIT 128
El descriptor del proceso 130
Redirección con TO 133
Matar un proceso 134
Redirección de la salida estándar
de errores 136
Redirección de la salida estándar 139
Evento Kill() y la propiedad Value 142
Redirección de la entrada estándar,
el uso de CLOSE 146
Notas finales sobre el objeto Process 148
4.4 SHELL 148

CAPÍTULO 5: GESTIÓN DE BASES DE DATOS 151

5.1 Sistemas de bases de datos 151
5.2 Bases de datos y Gambas 153
5.3 Gambas-database-manager, el gestor gráfico 154
Crear una base 154
Crear una tabla 158
Gestionar datos de una tabla 164
SQL 165
5.4 Programación 167
Modelo de bases de datos 167
Conectándose por código 168
Consulta de datos 171
Borrar registros 174
Añadir registros 176
Modificar registros 180
5.5 Otras características 190
Estructura de las tablas 190
Más utilidades del Gestor de Bases de Datos 192

CAPÍTULO 6: RED 195

6.1 Conceptos 195
6.2 Creando un servidor TCP 198
6.3 Un cliente TCP 205
6.4 Clientes y servidores locales 211
6.5 UDP 213
6.6 Resolución de nombres 216
6.7 Protocolo HTTP 220
6.8 Protocolo FTP 225

CAPÍTULO 7: XML 227

7.1 Escritura con XmlWriter 230
7.2 Lectura con XmlReader 238
Modelos de lectura 238
Planteamiento inicial 238
Un ejemplo de lectura 240
7.3 XSLT 252
¿Qué es XSLT? 252
Una plantilla de ejemplo 252
Transformando el documento con Gambas 254
7.4 Acerca de XML-RPC 255

CAPÍTULO 8: HERENCIA 257

8.1 Lenguajes orientados a objetos y herencia 257
8.2 Conceptos necesarios 258
La clase padre 258
La clase hija. Palabra clave INHERITS 262
Extendiendo funcionalidades.
Palabra clave SUPER 264
Modificando funcionalidades 265
Reemplazando métodos especiales:
_New y _Free 269
Limitaciones 272
8.3 Crear un nuevo control con Gambas 273
Planteando el código 273
Implementación básica 274
8.4 Nuevos componentes con Gambas 279
Preparación del código. Palabra clave EXPORT 280
Archivos necesarios, ubicaciones 280
Probando el nuevo componente 282

CAPÍTULO 9: ACCESO A LA API 287

9.1 Declarar una función externa 288
Cómo denominar la librería 289
El uso de los alias 290
Tipos de datos 291
9.2 Funciones auxiliares 291
9.3 Un ejemplo con libaspell 293
9.4 Obtener información acerca de la librería 297
9.5 Resumen 300

APÉNDICE: Marcas registradas 301

I. ¿Qué es Gambas?

I. ¿Qué es Gambas?
I. ¿Qué es Gambas?
I. ¿Qué es Gambas?

Es obvio que para los que hablamos el idioma español, la palabra gambas nos sugiere algunas cosas, pero todas ellas bastante alejadas del mundo de los ordenadores en general y del entorno de la programación en particular. El autor original de Gambas, Benoît Minisini, no habla una palabra de nuestro idioma y su inocente ignorancia le condujo a nominar su obra con el acrónimo GAMBAS: *Gambas almost means BASIC*, es decir, 'Gambas casi quiere decir BASIC'. No es la primera vez que un nombre o una marca bautizados en otros idiomas produce estos extraños cambalaches, recordemos que Coca Cola(TM) tuvo que cambiar su pronunciación en China porque la primera versión de su nombre significaba *muerde el renacuajo de cera*, o algunos todavía recordamos nuestra cara de asombro al ver los anuncios de un coche al que Suzuki dio en llamar *Pajero*. Tampoco está mal recordar que ya había antecedentes de otros lenguajes de programación con nombre de animal, como Camel

o Python, aunque en esos casos el nombre estaba en inglés y en español no resultaba tan chocante.

En fin, como Benoît, que tiene los derechos de autor, no desea cambiar el nombre, nos tendremos que ir acostumbrando a que Gambas empiece a sonarnos a algo más que a buen marisco. De hecho, Gambas abre el entorno de la programación visual en Linux a todo el mundo, como lo hizo en su día Visual Basic(TM) en Windows. Pero como el tiempo no pasa en vano, Gambas intenta no reproducir los errores que se cometieron entonces. La ampliación del lenguaje BASIC alcanza con Gambas amplias cotas de potencia, profesionalidad y modernidad, sin abandonar nunca la sencillez y claridad de este lenguaje de programación de alto nivel. Ya nunca más se podrá decir que construir aplicaciones visuales para Linux es un proceso largo y complejo que lleva años de trabajo a gurús y maniáticos de la informática.

Gambas no es sólo un lenguaje de programación, es también un entorno de *programación visual* para desarrollar aplicaciones gráficas o de consola. Hace posible el desarrollo de aplicaciones complicadas muy rápidamente. El programador diseña las ventanas de forma gráfica, arrastra objetos desde la **Caja de Herramientas** y escribe código en BASIC para cada objeto. Gambas está *orientado a eventos*, lo que significa que llama automáticamente a los procedimientos cuando el usuario de la aplicación elige un menú, hace clic con el ratón, mueve objetos en la pantalla, etc.

1.1 El lenguaje BASIC: su historia

El nombre BASIC corresponde a las siglas *Beginner's All Purpose Symbolic Instruction Code* (Código para principiantes de instrucciones simbólicas con cualquier propósito). El lenguaje fue desarrollado en 1964 en el Dartmouth College por los matemáticos John George Kemeny y Tom Kurtzas. Intentaban construir un lenguaje de programación fácil de aprender para sus estudiantes de licenciatura. Debía ser un paso intermedio antes de aprender otros más potentes de aquella

época, como FORTRAN o ALGOL. Este último era el lenguaje más utilizado en aplicaciones de procesos de datos, mientras que FORTRAN era empleado en las aplicaciones científicas. Sin embargo, ambos eran difíciles de aprender, tenían gran cantidad de reglas en las estructuras de los programas y su sintaxis. El primer programa hecho en BASIC se ejecutó a las 4 de la madrugada del 1 de mayo de 1964. Debido a su sencillez, BASIC se hizo inmediatamente muy popular y se empezó a usar tanto en aplicaciones científicas como comerciales. Tuvo el mismo impacto en los lenguajes de programación que la aparición del PC sobre los grandes ordenadores.

Cuando se desarrolló BASIC eran los tiempos en los que la informática estaba recluida en universidades y grandes empresas, con ordenadores del tamaño de una habitación. Pero pronto las cosas empezaron a cambiar. En 1971 Intel fabricaba el primer microprocesador. En 1975, la empresa MITS lanzó al mercado un kit de ordenador llamado Altair 8800 a un precio de 397 dólares. Era un ordenador barato, pero no para gente inexperta, había que saber electrónica para montarlo. Además tenía sólo 256 bytes (no es una errata, solo bytes, nada de Kbytes, megas o gigas) y se programaba en código máquina a base de 0 y 1, moviendo unos interruptores que tenía en el frontal. Dos jovencitos vieron un modelo en una revista de electrónica y decidieron montarlo. Le ofrecieron al dueño de MITS, además, hacer un intérprete de BASIC para los nuevos modelos de Altair. Eran William Gates y Paul Allen, y aquel BASIC, con un tamaño de 4 Kbytes, fue el primer producto que entregó una nueva empresa llamada Microsoft. Fue sólo el principio. A finales de los 70, Allen y Gates habían portado BASIC ya a un buen número de plataformas: Atari, Apple, Commodore... Y en 1981, cuando desarrollaron DOS para IBM y su nuevo PC, añadieron también su propio intérprete de BASIC al sistema. En posteriores años siguieron otras versiones hechas por otras compañías como Borland, pero el declive de BASIC había empezado. Las interfaces gráficas de ventanas que Apple popularizó y Microsoft adoptó con sucesivas versiones de

Figura 1. Lanzamiento del Altair 8800.

Windows(TM), se convirtieron en un estándar y BASIC no era un lenguaje preparado para estos entornos.

Sin embargo, en marzo de 1988, un desarrollador de software llamado Alan Cooper[1] intentaba vender una aplicación que permitía personalizar fácilmente el entorno de ventanas usando el ratón. El programa se llamaba Tripod y en aquellas fechas consiguió que William Gates lo viera y le encargara el desarrollo de una nueva versión a la que llamaron Ruby, y a la que añadieron un pequeño lenguaje de programación. Microsoft reemplazó ese lenguaje por su propia versión de BASIC, Quickbasic y el 20 de marzo de 1991 se lanzó al mercado con el nombre de Visual Basic. Al principio fue un verdadero fracaso de ventas, pero la versión 3 publicada en el otoño de 1996 fue un éxito total, tanto que actualmente es el lenguaje de programación más usado. Visual Basic siguió evolucionando hasta la versión 6.0. En 2002 fue integrado en la plataforma .NET de desarrollo, en lo que para muchos de los seguidores ha supuesto el abandono de Microsoft, ya que ha cambiado buena parte de la sintaxis añadiéndole complejidad en contradicción con el espíritu y el nombre del lenguaje. En cualquier caso, a día de hoy se calcula que entre el 70% y 80% de todas las aplicaciones desarrolladas en Windows se han hecho con alguna de las versiones de Visual Basic.

Las causas del éxito de Visual Basic son numerosas, pero entre otras se puede señalar como obvia el uso del lenguaje BASIC que fue pensado para un aprendizaje fácil. Otro de los motivos es disponer de un entorno de desarrollo cómodo, que hace un juego de niños el diseño de la interfaz gráfica de cualquier aplicación, apartando al programador de perder tiempo en escribir el código necesario para crear ventanas, botones, etc., y dejándole centrarse únicamente en la solución al problema que cualquier programa intenta resolver. Con la popularización de sistemas operativos libres como GNU/Linux, éstas y otras razones hacían prever que la aparición de un entorno equivalente libre sería un éxito y contribuiría a la presentación de muchos nuevos desarrollos que lo utilizarían. Ha habido varios intentos que no han cuajado, bien por la lentitud de su evolución, bien por su dificultad de uso o por no ser totalmente libres y no haber arrastrado a una comunidad de usuarios detrás. Finalmente, Benoît Minisini, un programador con experiencia en la escritura de compiladores

que estaba harto de luchar contra los fallos de diseño de Visual Basic, y deseaba poder usar un entorno de GNU/Linux fácil en su distribución, comenzó a desarrollar su propio entorno para Linux basado en BASIC. El 28 de febrero de 2002 puso en Internet la primera versión pública de Gambas: gambas 0.20. Benoît eliminó del diseño del lenguaje bastantes de los problemas que Visual Basic tenía, como la gestión de errores, y le añadió características comunes en los lenguajes actuales más modernos, como la orientación a objetos y la propia estructura de los programas[2]. Como prueba de fuego, el propio entorno de desarrollo fue programado en Gambas desde la primera versión, sirviendo a un tiempo de demostración de la potencia del lenguaje y de detección de necesidades y corrección de errores que se fueron incorporando a las distintas versiones.

En enero de 2005, Benoît publicó la versión 1.0, en la que ya se incorporaba un puñado de componentes desarrollados por otros programadores que colaboraron con él: Daniel Campos, Nigel Gerrard, Laurent Carlier, Rob Kudla y Ahmad Kahmal. Esta versión se consideró suficientemente estable y cerró un ciclo. A partir de esa fecha empezó la programación de la versión 2.0. Ésta ya incluye algunas mejoras en el lenguaje, muchos más componentes y un nuevo modelo de objetos que permitirán usar Gambas en un futuro para el desarrollo de aplicaciones web con la misma filosofía y facilidad que actualmente se usa para aplicaciones de escritorio.

1.2 Un entorno libre

Gambas es un entorno de desarrollo que se distribuye con la licencia GPL GNU (*General Public Licence*[3]). Esto significa que se distribuye siempre con el código fuente y respeta las cuatro libertades que define la Free Software Foundation:

- La libertad de usar el programa con cualquier propósito (libertad 0).
- La libertad de estudiar cómo funciona el programa y adaptarlo a las propias necesidades (libertad 1). El acceso al código fuente es una condición previa para esto.

- La libertad de distribuir copias, con las que se puede ayudar al vecino (libertad 2).

- La libertad de mejorar el programa y hacer públicas las mejoras a los demás, de modo que toda la comunidad se beneficie (libertad 3). El acceso al código fuente es un requisito previo para esto.

Una de los engaños más comunes en el uso de Software Libre es la creencia de que este modelo de desarrollo obliga a que el trabajo se publique gratis, lo que es del todo incierto. Estas cuatro libertades permiten que, quien lo desee, venda copias de Gambas (entregando siempre el código fuente y respetando esas cuatro libertades) y, por supuesto, de cualquier aplicación desarrollada con este programa. Las aplicaciones desarrolladas con Gambas pueden o no acogerse a la licencia GPL.

También cualquier programador es libre de alterar el propio lenguaje y modificarlo a su gusto, siempre y cuando entregue el código correspondiente a esas modificaciones y respete los derechos de autor de los desarrolladores originales.

Aparte de estas libertades propias de la naturaleza de un proyecto de Software Libre sobre GNU/Linux, Gambas añade más facilidades para el programador:

- Una ayuda muy completa del lenguaje y cada uno de los componentes, algo que es muy de agradecer para los que empiezan a programar en Gambas, y que no es habitual en los proyectos de Software Libre. La ayuda que se publica está en inglés, pero existe un grupo de personas trabajando en la traducción a español[4]. Si todos nos animamos a colaborar[5] en la traducción, pronto estará completa y disponible para el resto de usuarios.

- Una API (Interfaz para programar la aplicación) sencilla y bien documentada, lo que facilita a los programadores crear nuevos componentes para Gambas. La API no es de utilidad inmediata para quien desarrolle con este lenguaje, pero permite a los programadores avanzados que lo deseen añadir funcionalidades al entorno de desarrollo y crear nuevas herramientas para Gambas.

El lenguaje está preparado para ser independiente del gestor de ventanas que use. Esto significa que, sin cambiar una sola línea de código, una aplicación puede ser compilada para ser ejecutada en un escritorio Gnome o KDE, usando las librerías propias de ese escritorio y siendo una aplicación nativa de ese entorno. En el futuro se pueden desarrollar componentes para Windows(TM), Fluxbox y otros gestores de ventanas, y los programas no tendrán que modificar su código para que sean aplicaciones nativas de esos entornos. Marcando, antes de compilar, una opción en el entorno de desarrollo para elegir el componente a usar (actualmente se puede elegir entre *gtk* y *qt*, para Gnome o KDE), se generan distintas aplicaciones para distintos entornos con el mismo código fuente. Esta característica no se encuentra disponible en ningún otro lenguaje existente, lo que convierte a Gambas en un entorno único.

1. 3 Elementos de Gambas

Para poder desarrollar y ejecutar programas hechos con Gambas, son necesarios distintos elementos:

- Un **compilador**, que se encargará de transformar todo el código fuente y archivos que formen parte de un proyecto hecho en Gambas, en un programa ejecutable.

- Un **intérprete** capaz de hacer que los programas hechos en Gambas sean ejecutados por el sistema operativo.

- Un **entorno de desarrollo** que facilite la programación y diseño de las interfaces gráficas de los programas.

- **Componentes** que añaden funcionalidades al lenguaje. La palabra *componente* en Gambas tiene un significado específico, ya que no alude a partes genéricas, sino a librerías específicas que le dotan de más posibilidades. En la actualidad existen componentes para usar xml, conexiones de red, opengl, sdl, ODBC, distintas bases de datos, expresiones regulares, escritorios basados en *qt*, en *gtk*,

etc. Estos componentes son desarrollados por distintos programadores, siguiendo las directrices de la API de Gambas y la documentación publicada al efecto por Benoît Minisini.

1.4 Cómo obtenerlo

Las nuevas versiones de Gambas se publican a través de la página web oficial del proyecto: **http://gambas.sourceforge.net**. En la actualidad existen dos ramas de Gambas: la rama estable o 1.0 y la rama de desarrollo o 1.9 que desembocará en la versión 2.0. De la rama estable sólo se publican nuevas versiones cuando es para corregir algún fallo que se haya descubierto.

En el momento de escribir estas líneas, la versión estable era la 1.0.11 y no se prevé que haya cambios en el futuro. La versión de desarrollo está en continuo cambio, mejora y adición de nuevos componentes. Esto la hace muy atractiva, debido a la existencia de importantes funcionalidades que no están en la versión estable (como los componentes gtk y ODBC). Sin embargo, al ser una rama en desarrollo, es muy probable que tenga fallos no descubiertos y es seguro que sufrirá cambios en un futuro próximo. En este texto trataremos todas las novedades que la versión de desarrollo contiene para que sea el usuario el que escoja con qué rama trabajar. Buena parte de las diferencias se encuentran en los nuevos componentes. Algunos de estos serán tratados en este texto, por lo que si el lector quiere trabajar con ellos deberá usar la rama de desarrollo.

Las nuevas versiones se publican siempre en forma de código fuente, para que los usuarios que lo deseen compilen el código y obtengan todas las partes que Gambas tiene. Los autores de algunos de los componentes que se han hecho para Gambas, publican de forma separada en distintos sitios web las versiones nuevas de estos, pero todas se envían a Benoît Minisini y pasan a formar parte de la publicación completa de este lenguaje de programación en la siguiente versión. De este modo, se puede decir que cuando Benoît hace pública una nueva, el paquete del código fuente contiene las últimas versiones de todo el conjunto en ese momento.

Como la compilación de Gambas y todos los componentes asociados puede ser una tarea difícil para usuarios no expertos, es común que se creen paquetes binarios con la compilación ya hecha y listos para ser instalados en distintas distribuciones de gnu/Linux. En la misma página web donde se puede bajar el código fuente se encuentran los enlaces para la descarga de los paquetes compilados para estas distribuciones. Existen actualmente paquetes disponibles para Debian, Fedora, Mandriva, Gentoo, Slackware, QiLinux y Suse. En algunos casos, como para Fedora y Debian, están disponibles tanto los paquetes de la versión estable como la de desarrollo.

En el caso de Debian, los paquetes son realizados en gnuLinEx y, posteriormente, subidos a Debian para que estén disponibles y usables en esta distribución y en todas sus derivadas, como Knoppix, Guadalinex, Progeny, Xandros, Linspire, Skolelinux, etc. Por este motivo, las últimas versiones están siempre disponibles antes en los repositorios de gnuLinEx hasta que son subidos y aprobados en Debian. Las líneas del archivo */etc/apt/sources.list* de un sistema Debian para instalar la versión más actualizada de los paquetes de Gambas son:

Para la versión estable:

```
deb http://apt.linex.org/linex  gambas/
```

Para la versión de desarrollo[6]:

```
deb http://www.linex.org/sources/linex/debian/ cl gambas
```

A continuación, en cualquiera de los dos casos, para instalar todos los paquetes de Gambas, hay que ejecutar como usuario root:

```
apt-get update
apt-get install gambas
```

Aunque el código fuente de Gambas se distribuye en un único archivo comprimido, la instalación desde paquetes compilados se hace con un buen puñado de archivos.

La razón es que no todos son necesarios para ejecutar aplicaciones hechas en Gambas. En las distribuciones de Linux se ha seguido el criterio de separar en distintos paquetes el entorno de desarrollo (paquete *gambas-ide*), el intérprete (paquete *gambas-runtime*), y se ha hecho un paquete separado para cada uno de los componentes. Si se quiere programar en Gambas son necesarios la mayoría de ellos, al menos los que el entorno de desarrollo necesita. Si se quiere ejecutar un programa hecho con este lenguaje, sólo es necesario *gambas-runtime* y un paquete por cada uno de los componentes que el programa use. Por ejemplo, si es un programa que está hecho para el escritorio Gnome y no usa ningún otro componente, sólo sería necesario instalar en el sistema los paquetes *gambas-runtime* y *gambas-gb-gtk*.

1.5 Compilación y dependencias

Si en lugar de instalar paquetes ya compilados para la distribución de gnu/Linux deseamos compilar Gambas desde el código fuente, deberemos seguir los pasos habituales en los sistemas GNU. Es decir, descomprimir el archivo con las fuentes y, desde el directorio que se crea al descomprimir y usando un terminal, ejecutar las siguientes instrucciones:

```
./configure
make
make install
```

La última de ellas debemos hacerla como root, si queremos que el programa esté disponible para todos los usuarios del ordenador. Si estamos habituados a compilar aplicaciones en sistemas GNU, disponemos ya de un compilador instalado y de bastantes librerías de desarrollo. Las instrucciones anteriores tratarán de compilar e instalar todos los componentes de Gambas, que son muchos. Si no tenemos las librerías correspondientes a alguno de ellos, simplemente no se compilarán y la instrucción **./configure** nos informará de ello. Es importante saber que el entorno de desarrollo está hecho sobre las librerías gráficas *qt*, por tanto, para poder usar el entorno necesitaremos tener instalado, al menos, estas librerías de desarrollo con una versión igual

o superior a la 3.2. La versión del compilador *gcc* ha de ser también ésta, como mínimo. Cada uno de los componentes tiene dependencias de sus propias librerías y dependerá de la distribución de Linux que usemos, para saber el nombre del paquete que deberemos instalar antes de poder realizar la compilación.

1.6 Familiarizarse con el IDE

Aunque un programa en Gambas se podría hacer perfectamente usando un editor de texto plano cualquiera, sería un desperdicio no aprovechar uno de los mayores atractivos que el lenguaje tiene: su IDE o entorno de desarrollo. El IDE de Gambas ahorra al programador buena parte del trabajo más tedioso, le proporciona herramientas que hacen mucho más fácil su tarea, con utilidades de ayuda, de diseño de interfaces, autocompletado de instrucciones, traducción de programas, etc. En la imagen siguiente podemos ver algunas de las ventanas más importantes del entorno, que se usan durante el desarrollo de una aplicación:

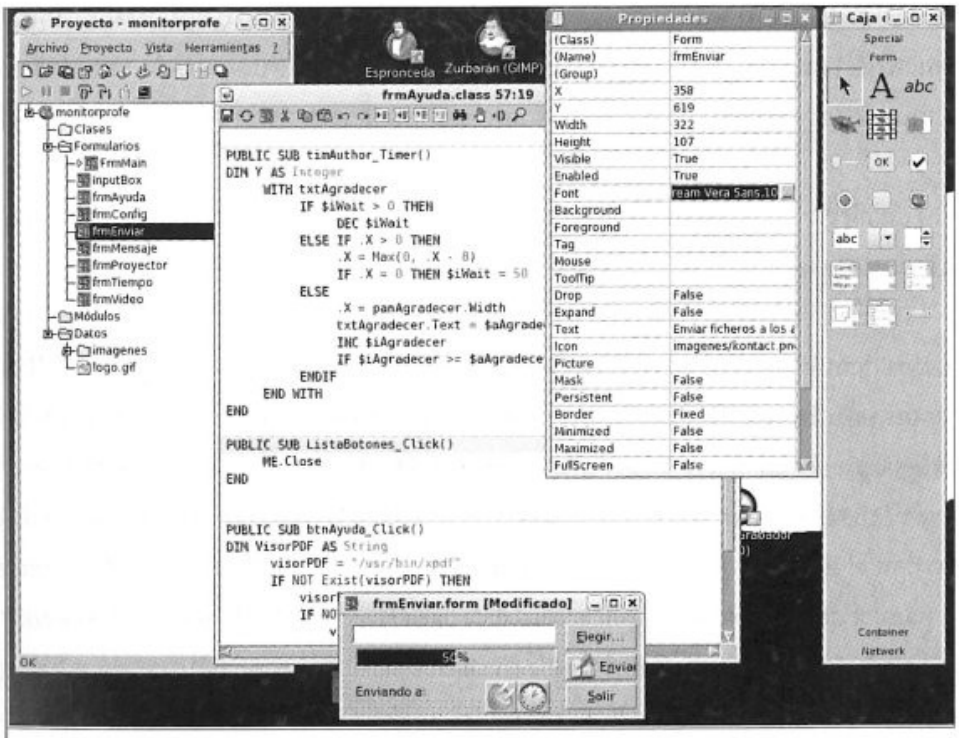


Figura 2. Entorno de desarrollo de Gambas.

Cuando se arranca Gambas, lo primero que nos aparece es la ventana de bienvenida.

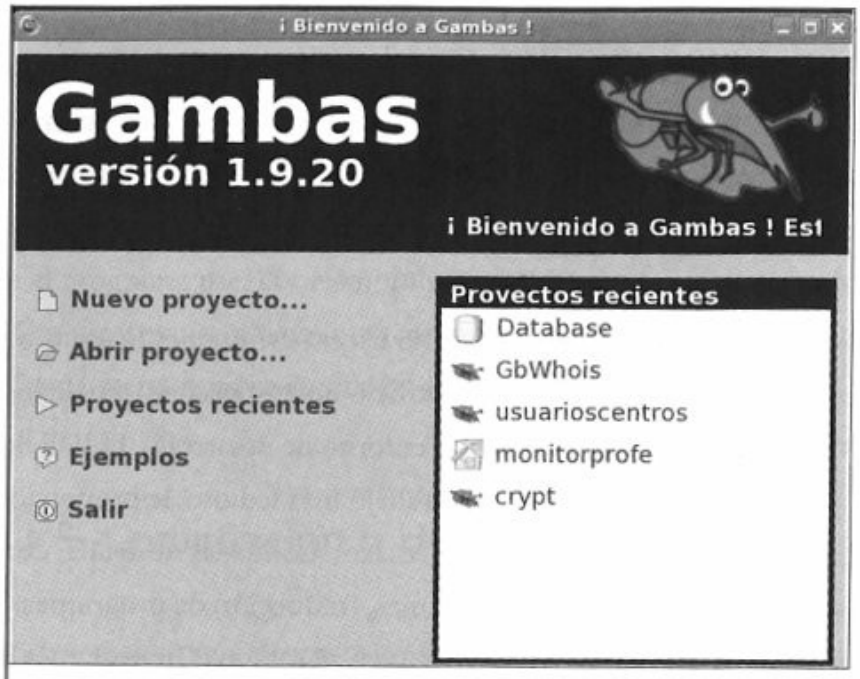


Figura 3. Ventana de bienvenida.

Aquí se nos ofrece la opción de comenzar un nuevo proyecto o aplicación, abrir un proyecto del que tengamos sus archivos disponibles, abrir uno usado recientemente o uno de los numerosos ejemplos que están incluidos en la ayuda de Gambas.

Antes de elegir cualquiera de estas opciones es necesario saber que todos los códigos fuente de una aplicación hecha en Gambas es lo que se denomina *proyecto*. El proyecto está formado por una serie de archivos que en Gambas están SIEMPRE situados dentro de un único directorio. En él puede haber, a gusto del desarrollador, distintos subdirectorios y organizar todo como se desee, pero cualquier gráfico, texto y código que forme parte de la aplicación estará dentro de él. Por ello, si elegimos en esta ventana la opción **Nuevo proyecto...**, el asistente siempre creará un nuevo directorio con el nombre del proyecto y ahí irá introduciendo todos los archivos necesarios para el desarrollo de la aplicación. Así, para enviar a alguien el código fuente de una aplicación hecha en Gambas o cambiarla de ordenador o disco, sólo hay que transportar el directorio con el nombre del proyecto, sin tener que preocuparse de otros archivos. Del mismo modo, si desde el entorno de desarrollo escogemos un

archivo o un gráfico para integrarlo en nuestro trabajo, el archivo será copiado automáticamente al directorio del proyecto.

El primer ejemplo

Una de las formas más habituales de empezar a trabajar con un lenguaje de programación es haciendo un pequeño programa que muestre el mensaje *Hola Mundo*. Por tanto, empezaremos a conocer el entorno de desarrollo y el lenguaje de programación con este típico ejemplo. Comenzaremos haciendo un *hola mundo* que sea puro BASIC, es decir, que sea igual al que hubieran hecho los autores de BASIC allá por el año 1964.

En aquellos tiempos las interfaces gráficas no existían, por lo que este primer programa será un programa feo, de terminal. En el BASIC original, hacer que aparezca un mensaje en el terminal es tan simple como escribir la línea:

```
PRINT "Hola Mundo"
```

No es necesario ningún encabezado previo, la instrucción **PRINT** sirve para mostrar cualquier cadena de texto en el terminal, que en BASIC se presentan entre comillas dobles. De ahí que el programa sea tan simple como ese. Vamos a ver cómo hacerlo con el entorno de desarrollo de GAMBAS:

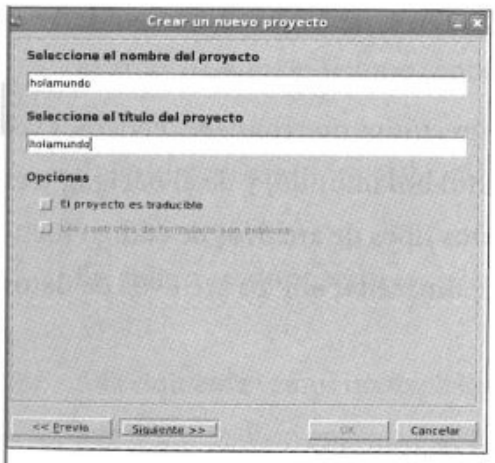


Figura 4. Asistente de Gambas.

1. Escogemos la opción **Nuevo proyecto...** en la ventana anterior (**Figura 3**). Aparecerá un asistente, en el que pulsamos **Siguiente**. Surgirá una nueva ventana para elegir el tipo de aplicación. Escogemos la tercera opción: **Crear un proyecto de texto**[7].

2. Le damos un nombre al proyecto, por ejemplo *holamundo* y un título. Pulsamos el botón **Siguiente** (**Figura 4**).

Elegimos el directorio del disco en el que queremos crearlo; pulsamos de nuevo el botón **Siguiente** y aparecerá un resumen con los datos del proyecto (**Figura 5**).

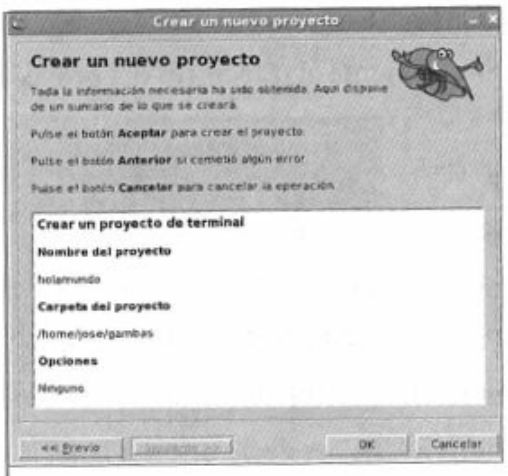


Figura 5. Datos del proyecto.

3. A continuación pulsamos el botón **OK**. Se abrirá el entorno de desarrollo de Gambas listo para empezar a programar. Al ser una aplicación de terminal, que no lleva interfaz gráfica, de momento, sólo necesitamos fijarnos en la ventana de la izquierda, que en nuestro caso tendrá como título: *Proyecto – holamundo* (**Figura 6**).

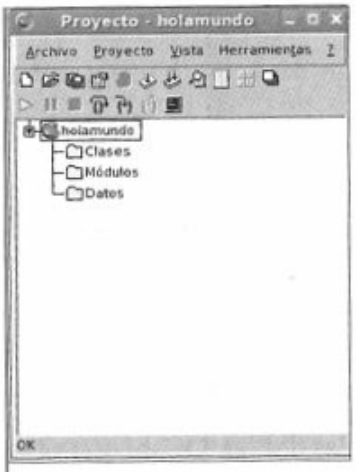


Figura 6. **Proyecto – holamundo**.

En realidad ésta es, probablemente, la ventana más importante para el manejo del entorno de desarrollo. A simple vista se puede ver el menú superior, que contiene las entradas necesarias para guardar y cargar proyectos, activar las distintas ventanas del IDE, manejar la ejecución de los programas, personalizar el entorno (en **Herramientas | Preferencias | Otros | Mostrar mascota**, podemos ocultar la animación de la gamba, que nos está mirando al abrir el proyecto), etc.

De momento nos fijamos sólo en el árbol de directorios que contiene. Podemos ver que la raíz del árbol es el nombre del proyecto: **holamundo**, y de él cuelgan tres ramas: **Clases** y **Módulos**, que son para distintos tipos de archivos de código fuente; y **Datos**, cuyo nombre indica su finalidad, almacenar ahí los archivos de datos que la aplicación requiera.

Desde el principio, Gambas nos da dos formas de realizar los programas, incluso si son tan simples como el que hemos hecho. Podemos elegir entre una programación

orientada a objetos, paradigma típico de los lenguajes de programación más potentes o una programación estructurada simple. Según ello, el archivo que contenga el código de nuestro programa será una *Clase* o un *Módulo*. Por simplicidad, de momento vamos a usar un *Módulo*. Haciendo clic con el botón derecho del ratón sobre el árbol de carpetas aparecerá un menú contextual. Elegimos las opciones **Nuevo | Módulo**. Surgirá una ventana en la que escribimos el nombre del módulo, por ejemplo *miprograma*, y pulsamos el botón **OK**, con lo que aparecerá nuestra primera ventana, con el título *miprograma.module*, donde poder escribir código.

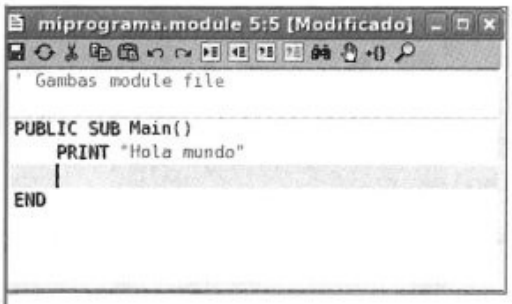


Figura 7. Ventana donde escribir el código.

Por fin podemos escribir nuestro código en BASIC. Lo haremos justo antes de la línea donde pone **END**, tal y como queda reflejado en la figura de la izquierda.

Cuando después de escribir el código que ya sabíamos, pulsamos la tecla **INTRO**, vemos que el entorno colorea el texto de una forma particular. Podemos pararnos un instante a ver los distintos colores que se muestran (**Figura 7**):

- En **gris** aparece una línea que comienza por una comilla simple ('). Esto indica que la línea es un comentario, es decir, no se trata de ningún código de programación y el texto que sigue a la comilla no se ejecuta nunca, son comentarios que el programador puede/debería poner para facilitar que otros (o él mismo, pasado un tiempo) entiendan lo que el programa hace en ese punto.

- En **azul** podemos ver palabras clave del lenguaje BASIC.

- En color **rosado** aparece la cadena de texto.

- A la izquierda vemos un resalte **amarillo** al comienzo de las líneas que han sido modificadas. Esto aparecerá siempre en las líneas que contengan modificaciones que no hayan sido compiladas.

Bien, ya está listo el programa. Para comprobarlo se pulsa en el botón verde, con el símbolo del **Play**, que está en la pantalla del proyecto. Al hacerlo aparecerá una nueva ventana llamada *Consola* en la que se verá la salida de nuestro programa. En este caso será el simple texto *Hola Mundo*. Éste es todo el código necesario, ya se puede compilar el programa para generar un archivo ejecutable que funciona sin necesidad del entorno de desarrollo. Para ello, en el menú de la ventana del proyecto hay que escoger: **Proyecto | Crear ejecutable**. Saldrá un cuadro de diálogo para elegir el directorio en el que queremos crear el ejecutable. Pulsando **OK** lo generará. Al cerrar ahora el entorno de desarrollo de Gambas y abrir un terminal o pasar a una consola de Linux, podemos probar su funcionamiento. Para ello, en el directorio donde se haya creado el ejecutable, hacemos:

```
jose@a00-o04:~$ cd gambas/holamundo/
jose@a00-o04:~/gambas/holamundo$ ./holamundo.gambas
Hola mundo
```


Mejor con un ejemplo gráfico

El ejemplo anterior mostraba una aplicación de consola, que nos recuerda a los viejos tiempos de otros sistemas operativos o a la forma de trabajar de los hackers informáticos. En realidad, hacer ese tipo de programas no demuestra el potencial de Gambas, puesto que son realmente simples e igualmente fáciles de realizar en otros lenguajes como Python o cualquier vieja versión de BASIC.

Es mejor hacer el programa *Hola Mundo* para el entorno gráfico que inunda los escritorios de los ordenadores actuales. Para ello empezaremos igual que antes, arrancando Gambas y creando un **Nuevo proyecto** siguiendo exactamente los mismos pasos, excepto que en lugar de escoger **Crear un proyecto de texto**, cuando el asistente nos presente las distintas opciones, elegiremos **Crear un proyecto gráfico**.

Para no repetir nombre, podemos denominar al proyecto *holamundo2*. Al acabar el proceso aparecerá de nuevo la ventana de proyecto, pero en esta ocasión tendrá una rama más en el árbol: **Formularios**. Un formulario es el área donde se diseña

la interfaz gráfica de la aplicación, es decir, donde se insertan objetos como botones, cuadros de texto, listas, casillas de verificación, etc. Los formularios se corresponderán con las ventanas que la aplicación mostrará.

Haciendo clic con el botón derecho del ratón sobre el árbol del proyecto, elegimos ahora en el menú contextual que aparece: **Nuevo | Formulario**. Por simplicidad, en este caso ni siquiera hace falta cambiar el nombre, sólo pulsando en el botón **OK** aparecerá en pantalla la ventana del formulario y una ventana para escribir código BASIC casi idéntica a la del ejemplo anterior, sin nada escrito. El resultado será algo parecido a lo siguiente:

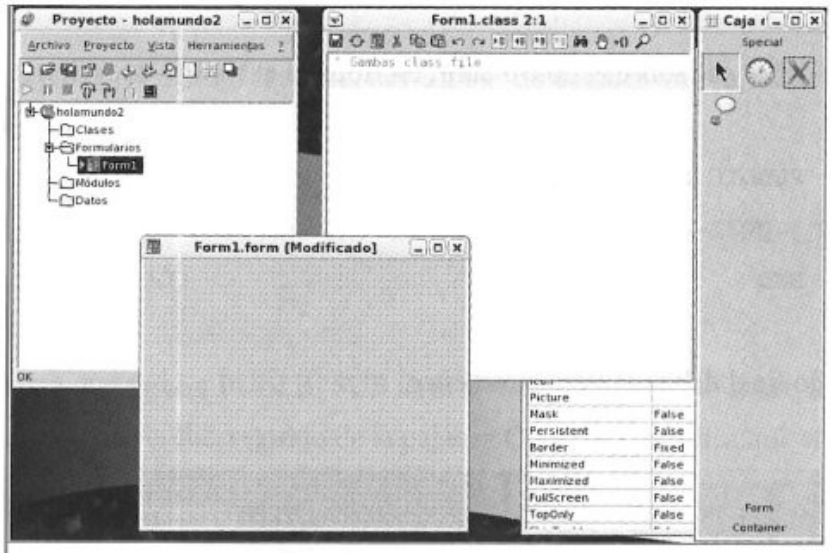


Figura 8. Pantalla del formulario.

Haciendo clic sobre la palabra **Form** en la ventana de la derecha, correspondiente a la **Caja de Herramientas**, aparecerán distintos objetos que podemos colocar en los formularios. Entre estos está el icono del botón (se distingue rápidamente por tener la palabra **OK** escrito dentro). Haciendo un clic de ratón en él, se puede dibujar en el formulario un botón, tan sólo hay que arrastrarlo con el ratón y el botón izquierdo pulsado, para dar la forma que queramos. Gambas escribe el texto *Button1* sobre el ratón, pero como ese texto no es demasiado intuitivo lo mejor es cambiarlo. Para ello hay que pinchar en el botón y después, en la ventana de propiedades, hacer un clic en la línea donde pone *Text*. Ahora se puede escribir el nuevo texto, por ejemplo **Púlsame**.

una aplicación nativa para el escritorio KDE. Sin embargo, el componente *gtk* disponible en la versión de desarrollo de Gambas, permite hacer programas enlazados con las librerías de *Gtk* para realizar aplicaciones del escritorio Gnome.

Si hemos instalado los paquetes de Gambas correspondientes a la rama de desarrollo, incluyendo el paquete *gambas-gb-gtk*, podemos recuperar la aplicación *holamundo2* del ejemplo anterior, ir a la ventana de *Proyecto* y hacer clic en el menú **Proyecto | Propiedades**, pestaña **Componentes**, y seleccionar **gb.gtk**. El resultado al ejecutar la aplicación es que se trata de una nueva, pero antes de KDE y ahora de Gnome. ¡Y sin tocar una sola línea de código!

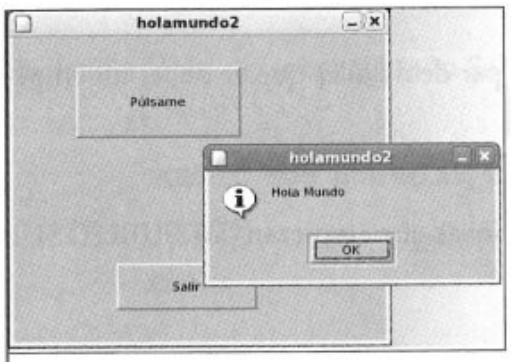


Figura 10. Aplicación *holamundo2* en KDE.

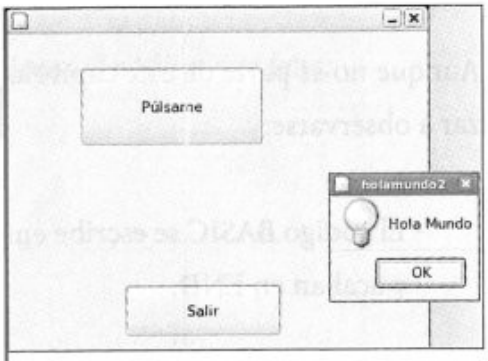


Figura 11. Aplicación *holamundo2* en Gnome.

Hasta el día de hoy no existe ningún otro lenguaje de programación con el que se pueda hacer esto.

I. 7 Sistema de componentes

A lo largo de los apartados anteriores han aparecido distintas referencias a los componentes de Gambas, incluyendo su descripción en el tercer apartado de este capítulo. Estos permiten extender este lenguaje de programación. La interfaz desarrollada por Benoît para su programación hace que hayan sido varios los programadores que han colaborado con él, desarrollando nuevos componentes que se van añadiendo a las distintas versiones de Gambas en cada nueva publicación. En la versión 1.0 estable de

Gambas sólo se podían desarrollar en C y C++, pero desde la versión 1.9.4 de la rama de desarrollo también se pueden escribir componentes en Gambas, lo que abre numerosas posibilidades futuras ya que es mucho más sencillo hacerlo que en C.

El listado de componentes disponibles es amplio y se aumenta continuamente en la versión de desarrollo. Para la rama estable están fijados los siguientes:

- *gb.compress*: para compresión de archivos con protocolos *zip* y *bzip2*.
- *gb.qt*: para objetos visuales de las librerías gráficas *qt*.
- *gb.qt.ext*: para objetos visuales de las librerías gráficas *qt* que no son estándar.
- *gb.qt.editor*: un editor de texto hecho usando las librerías gráficas *qt*.
- *gb.qt.kde*: objetos visuales propios del escritorio KDE.
- *gb.qt.kde.html*: un navegador web del escritorio KDE.
- *gb.net*: objetos para conectar a servidores de red y a puertos serie de comunicaciones.
- *gb.net.curl*: objetos para construir servidores de red.
- *gb.db*: objetos de conexión a bases de datos.
- *gb.db.mysql*: driver para conectar al servidor de bases de datos MySQL.
- *gb.db.postgresql*: driver para conectar al servidor de bases de datos PostgreSQL.
- *gb.db.sqlite*: driver para usar bases de datos Sqlite 2.x.
- *gb.xml*: objetos para parsear archivos XML.

- *gb.vb*: colección de funciones para facilitar la migración desde Visual Basic.

- *gb.sdl*: objetos para reproducir, mezclar y grabar archivos de sonido.

- *gb.pcre*: objetos para usar expresiones regulares en el lenguaje.

En la rama de desarrollo existen estos componentes (algunos de los cuales han sido muy mejorados y aumentados, como el *gb.sdl*) y otros más, con una lista en continuo crecimiento. En el momento de escribir estas líneas podemos contar con:

- *gb.crypt*: objetos para encriptación DES y MD5.

- *gb.form*: objetos gráficos para formularios independientes de las librerías gráficas usadas.

- *gb.gtk*: objetos gráficos para formularios de las librerías *gtk*. Tiene los mismos objetos que el componente *gb.qt*, pero enlazan con esta otra librería de desarrollo.

- *gb.db.odbc*: driver para conectar a bases de datos a través de ODBC.

- *gb.info*: objetos que proporcionan distinta información acerca de los componentes y el sistema donde la aplicación se ejecuta.

- *gb.opengl*: objetos para dibujos tridimensionales con aceleración OpenGL.

- *gb.xml.rpc*: objetos para el uso del protocolo *rpc-xml*.

- *gb.sdl.image*: objetos para dibujos en dos dimensiones con aceleración gráfica.

- *gb.v4l*: objetos para capturar imágenes de cámaras de vídeo en Linux.

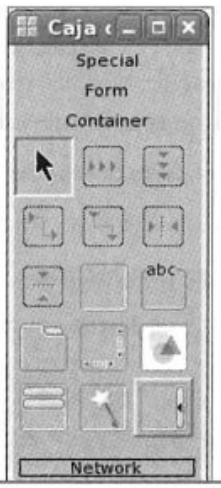
- *gb.gtk.pdf*: se utiliza para renderizar documentos *pdf* desde aplicaciones hechas en Gambas.

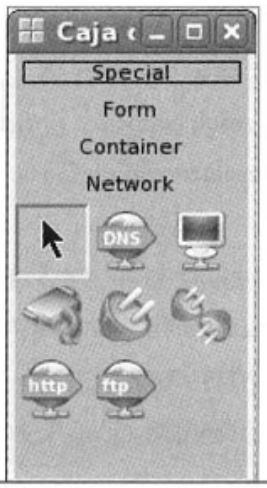
El listado de componentes disponibles para el programador puede verse en la pestaña **Componentes**, accesible a través del menú **Proyecto | Propiedades**. Cada uno de los componentes se corresponde a un paquete compilado en la distribución, de forma que si se añade al proyecto, por ejemplo, el componente *gb.sdl*, ha de instalarse el paquete *gambas-gb-sdl* en los ordenadores donde se quiera ejecutar la aplicación compilada.

Figura 12. Componentes.

Al hacer clic sobre cada uno de los componentes aparece una pequeña descripción de su función, el nombre del autor o autores del componente, y un listado de los controles que están disponibles para el desarrollador si selecciona ese componente (**Figura 12**).

Los controles son clases para crear objetos útiles en la programación. Los objetos creados pueden ser visuales (como pestañas, editores de texto, etc.) u objetos de código (como servidores de red o conexiones a bases de datos). Si el componente tiene objetos visuales, estos se incorporan a alguna de las pestañas de la **Caja de Herramientas** del entorno de desarrollo.





Figuras 13, 14 y 15. **Caja de Herramientas** con componentes con objetos gráficos.

En las imágenes anteriores podemos ver algunos de los objetos gráficos que están disponibles al seleccionar los distintos componentes.

Cada componente tiene su propia documentación que se encuentra incluida en la ayuda de Gambas. En la rama estable esta documentación está siempre available; en la rama de desarrollo sólo está disponible si ha sido seleccionado para su uso en el proyecto.

Todas las cosas que se pueden hacer con Gambas, y no son parte del propio lenguaje BASIC, se programan mediante el uso de componentes. Esto significa que, por ejemplo, para hacer una aplicación de bases de datos es necesario seleccionar el componente *gb.db* o no estarán disponibles los objetos de conexión a bases de datos. Lo mismo ocurre con las conexiones de red, captura de vídeo, etc. Estos objetos no son parte del lenguaje BASIC.

Notas

1 **http://www.cooper.com/alan/father_of_vb.html**

2 Diferencias entre Gambas y Visual Basic: **http://wiki.gnulinex.org/gambas/210**

3 **http://www.fsf.org/licensing/licenses/gpl.html**. Hay una traducción (no oficial) al español en **http://gugs.sindominio.net/licencias/**

4 Documentación de Gambas en español: **http://wiki.gnulinex.org/gambas/**

5 Instrucciones para colaborar en la documentación en español: **http://wiki.gnulinex.org/gambas/6**

6 Si la instalación se hace desde un sistema gnuLinEx, no es necesario añadir esta línea puesto que la versión de desarrollo de Gambas es parte del repositorio oficial de gnuLinEx.

7 Debido a la mayor extensión de la cadena **Crear un proyecto de texto** en español que en otros idiomas, es posible que ese mensaje aparezca cortado y no se vea completo. No hay que preocuparse por ello, no afecta para nada a su elección y suponemos que será corregido en posteriores versiones del IDE.