Desarrollo en un entorno educativo de objetos para el control de una interfaz de domótica

Lic. Gonzalo Zabala
Centro de Altos Estudios en Tecnología Informática
Facultad de Tecnología
Universidad Abierta Interamericana
TE / FAX: (5411) 43015240
Email: gonzalo.zabala@vaneduc.edu.ar

Abstract: En los últimos años han surgido diversas plataformas de hardware para la enseñanza de robótica, tanto en su expresión autónoma como en el control desde una PC. En general, estas interfaces adolecen de un entorno lo suficientemente didáctico para poder acercar a los alumnos de nivel primario y secundario a los conceptos fundamentales de la robótica. Durante el año 2006 se ha desarrollado en nuestro centro de investigaciones una nueva interface controlada por puerto paralelo. En este paper presentamos el desarrollo de un conjunto de objetos gráficos para el manejo de dicha interfaz y de otras controladas por el mismo puerto, realizado sobre un entorno de objetos con fines educativos.

Keywords: robótica, Squeak, domótica, entorno de objetos, morphs.

1. Introducción

Durante el año 2006 se desarrolló en el Centro de Altos Estudios en Tecnología Informática (CAETI) un conjunto de interfaces para control automatizado desde la PC, con el objetivo de aplicar su uso en la enseñanza de robótica y domótica desde los últimos años del nivel primario hasta el universitario. Para su uso fueron diseñadas diversas herramientas de programación, desde un conjunto de clases para .Net como otras de carácter gráfico. En este paper presentaremos un conjunto de clases y objetos gráficos (conocidos como morphs) que se han desarrollado para Squeak. Squeak es un entorno de objetos diseñado por los creadores de SmallTalk, Alan Kay y Dan Ingalls, con el objetivo de crear un espacio de simulación para el ámbito educativo. A continuación presentaremos las características técnicas de la interfaz, Squeak y los desarrollos que hemos hechos para vincularlos.

2. Características de la interfaz

La interfaz utilizada para este proyecto forma parte de un kit de interfaces electrónicas especialmente desarrolladas para ser utilizadas en aplicaciones de control electrónico y automatización eléctrica, de baja potencia, controladas por PC. El Kit consiste en un dispositivo que conectado a una PC, por medio de un programa, permite controlar dispositivos como lamparitas, motores de corriente continua o relés. Su principal característica es el manejo de entradas y salidas digitales controladas por software a través del puerto paralelo. Utiliza elementos de optoelectrónica en sus entradas, más particularmente optoacopladores con salida a transistor, para permitir el uso de un rango variado de tensiones para activar las mimas y proteger la PC. El modelo de dispositivo utilizado en este proyecto **permite controlar 16 salidas digitales, más específicamente de colector abierto, y 8 entradas digitales optoacopladas**.

3. Squeak

En los últimos treinta años, la informática ha realizado avances notables con respecto al hardware, las comunicaciones y las capacidades multimediales de las computadoras. Sin embargo, en el plano educativo –salvo contadas excepciones– las herramientas desarrolladas no usan las capacidades de

simulación y emulación que convierten una computadora en un laboratorio virtual de infinitas posibilidades de creación.

A fines de la década del 60, **Alan Kay** creó la **Dynabook**, convencido de que la simulación es una herramienta notable para la comunicación de ideas y que una computadora debería ser el contenedor de todos los medios de expresión en los que uno pudiera pensar, es decir, un meta-medio. La Dynabook debía de ser un dispositivo portátil, con red inalámbrica y pantalla plana, entre otras cosas. **Este dispositivo funcionaría como un "amplificador de la mente" y como lugar donde un usuario concentraría toda la información que consume y que genera.** En un contexto en el que las computadoras eran grandes máquinas de costo altísimo, de uso privativo de grandes corporaciones, pensar en una computadora personal con estas características era dar un enorme salto hacia el futuro.

Guiados por esta visión, un grupo de referentes de la informática (Alan Kay, Dan Ingalls, Adele Goldberg, etc) crean Smalltalk, un ambiente de objetos de donde surgen "la mayoría de las cosas buenas relacionadas con las computadoras personales (incluido el propio nombre)" [i].

Una lista, no completa, de los aportes de este proyecto al desarrollo de la informática son:

- El concepto de la Computadora Personal
- El paradigma de objetos
- Smalltalk
- Interfaces de usuario gráficas
- Uso del mouse
- Drag & drop (Arrastrar y arrojar)
- Menúes despegables

A mediados de los años noventa, Alan Kay y Dan Ingalls (coautor de Smalltalk) son contratados por Disney para desarrollar un entorno de programación que pudieran utilizar los chicos, basado fundamentalmente en recursos multimediales. **Allí surge Squeak.** Kay tuvo la precaución de solicitar que el desarrollo fuera **Open Source**, es decir, **absolutamente gratuito, modificable y extendible por cualquier persona**. De esta manera, desde 1996 (año de aparición de Squeak) hasta la actualidad, su desarrollo ha sido notable. Muchos programadores, de altísima calidad académica, han dedicado horas de trabajo a este proyecto, logrando el mejor ambiente de objetos en la actualidad.

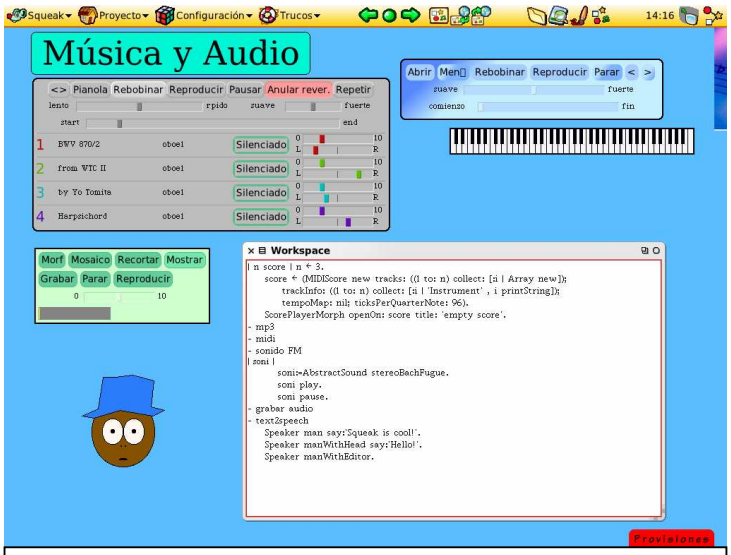


Figura 1 - Una pantalla de Squeak con objetos Morph

Squeak es un Smalltalk, y como tal, es una herramienta de programación. **Un Smalltalk es más que un lenguaje de programación: es un ambiente, un medio donde sus habitantes son objetos**. Estos objetos tienen noción del tiempo, entienden mensajes, y la comunicación que entablamos es

una especie de diálogo primitivo máquina-humano. En este medio hay una gran variedad de objetos con responsabilidades y propósitos distintos: tenemos el objeto que controla la pantalla, el objeto que controla los procesos, los números, el objeto que interpreta y compila nuestras expresiones, etcétera.

Particularmente, en Squeak tenemos un **conjunto de objetos con representación gráfica llamados Morphs.** Cada morph tiene sus características. A ellas se accede mediante los halos, una serie de círculos dispuestos alrededor del morph que hemos seleccionado. Los halos nos proporcionan el mecanismo para realizar diversas acciones (véase Figura 2 - Morph de reloj con el halo activado): cambio de color, cambio de tamaño, rotación, etcétera. Los morphs tienen noción del tiempo que transcurre, por lo cual podemos modificarlos, por ejemplo, para cambien de color cada vez que transcurre cierto intervalo de tiempo.

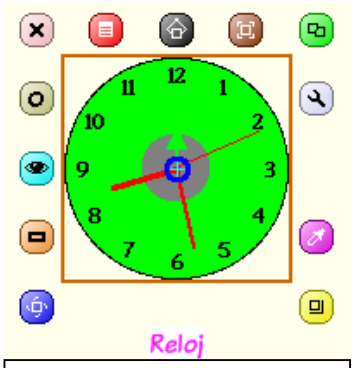


Figura 2 - Morph de reloj con el halo activado

En el proceso de evolución de creación de los morphs, se les ha agregado la capacidad de brindarles comportamiento en forma dinámica. A estos morphs se los conoce como **etoys**. Los etoys son juguetes visuales que nos permiten otorgar comportamiento a los morphs y manipularlos mediante la simple escritura de un guión con instrucciones. Uno de los halos del morph llamado **visualizador** (el ojo color celeste) nos permite, como su nombre lo indica, ver las propiedades del morph. Si arrastramos alguna de ellas al fondo del mundo (o sea, al "escritorio"), se nos activará un **guión**. El guión es un área donde se pueden arrojar propiedades y acciones y que posee botones para arrancar, pausar o detener las acciones que se les haya arrojado. Por ejemplo, supongamos que queremos que una estrella avance y gire en cada ejecución. Seleccionamos una estrella del repositorio de objetos -o simplemente la dibujamos- y luego, mediante los halos, pedimos su visualizador. En el visualizador tenemos los métodos y las propiedades a los cuales podemos acceder del morph. En el caso de la Figura 3 - Estrella con su halo y un guión de ejemplo, se ha creado un guión donde en cada paso, la estrella avanzará 5 pixeles y girará 5 grados en sentido horario. Esa ejecución puede ser ante un evento, o en forma latente, con una frecuencia determinada por el programador.

Figura 3 - Estrella con su halo y un guión de ejemplo

Dado que este paper no está orientado a profundizar en este aspecto, en la bibliografía indicamos material y links donde se desarrollan los etoys en forma más detallada. Lo que queremos mostrar aquí es la simpleza con la cual se pueden generar programas en un ambiente absolutamente gráfico, con un lenguaje muy potente (que no se muestra aquí) y que puede ser enriquecido por los desarrolladores para los etoys que crean. En nuestro caso, justamente lo que presentamos es un etoy para poder controlar la interfaz de robótica, enriqueciendo el lenguaje de los morphs con un conjunto de mensajes vinculados al encendido y apagado de bits, al testeo de condiciones, etc.

4. Vinculación con el puerto paralelo

Dado que el control de la interfaz desde la PC se realiza con el puerto paralelo, se buscó algun desarrollo previo que permitiera la comunicación de Squeak con dicho puerto. Nuestro sistema operativo de trabajo es Windows, y fue necesario utilizar una dll que permitiera el acceso a bajo nivel de ese puerto. Para ello, tomamos el trabajo realizado por **Germán Viscuso** en el grupo SmallTalking[ii]. En las clases desarrolladas por Germán, se utiliza la librería **io.dll**, creada por **Geek Hideout**[iii]. Esta librería es de uso gratuito, y ofrece un conjunto de funciones vinculadas al acceso a nivel de bit y de palabra del puerto paralelo. **El trabajo de Germán fue crear una capa sobre este acceso a bajo nivel, ofreciendo la clase ParallelPort que brinda objetos transparentes para el acceso al puerto.**

5. Clases y etoys desarrollados

Utilizando la clase mencionada en el punto anterior, se realizaron un conjunto de morphs que representan a las funcionalidades brindadas por la interfaz y objetos que podemos conectar a la misma. En primer lugar, desarrollamos la clase **ParallelPortExtendedMorph**. Esta clase nos presenta un morph que representa al puerto paralelo, con **la posibilidad de activar o desactivar en forma independiente cada bit**. Como vemos en la Figura 4 - Morph de la interfaz del puerto paralelo, haciendo click en las elipses numeradas coloreadas de verde y rojo, podemos encender o apagar cada bit en forma independiente. **Por otra parte, en las elipses azules podemos ver el estado de los bits de entrada.**

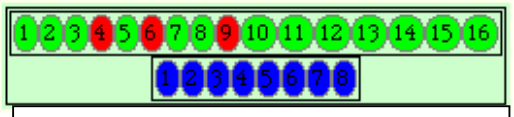
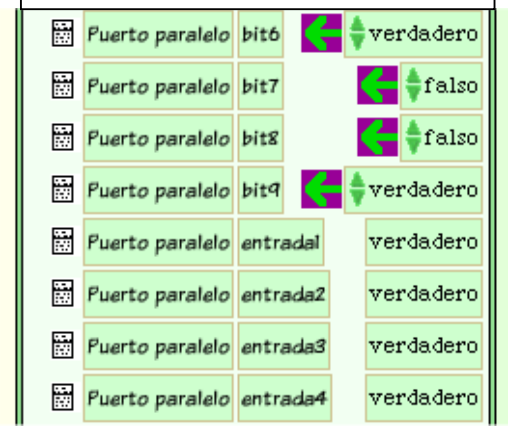


Figura 4 - Morph de la interfaz del puerto paralelo

Como etoy, se han agregado mensajes a su protocolo que nos permiten programar el comportamiento de los bits en forma independiente y testear tanto los bits de entrada como los de salida. En la Figura 5 - Algunos de las propiedades del etoy desarrollado, podemos ver las propiedades vinculadas a los 16 bits de salida, y las vinculadas a los 8 bits de entrada. De esta manera, podemos realizar en forma sencilla un programa como el que se muestra en Figura 6 - Ejemplo de guión usando las propiedades del etoy, donde el guión se ejecuta cada una determinada cantidad de segundos (se define en el reloj que está a la derecha de la palabra ejemplo), y según el estado del bit de entrada 1, enciende o apaga el bit de salida 5.

Figura 5 - Algunos de las propiedades del etoy desarrollado



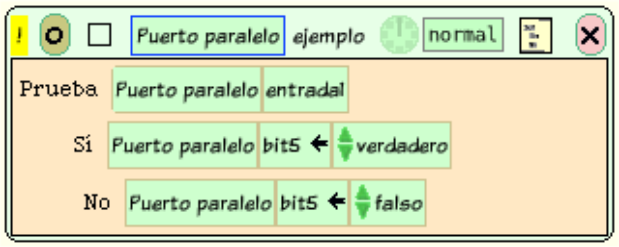


Figura 6 - Ejemplo de guión usando las propiedades del etoy

Por ejemplo, si quisiéramos conectar un motor paso a paso en los bits 1 a 4, simplemente el guión consiste en un test que determine el estado de cada bit de salida, y cuando encuentra uno encendido, enciende el siguiente y apaga el actual, salvo en el caso del bit 4, donde encendería el bit 1.

Para simplificar aún más el uso de la interfaz, se crearon tres etoys más. El primero de ellos es un etoy que representa a una lámpara (**LamparaMorph**), que al ser creada desde el menú del etoy de la interfaz (como se puede ver en la Figura 7 - Menú del etoy de la interfaz) con la opción "Dame una lámpara", nos pregunta a qué bit se conecta. Dentro del protocolo de la lámpara se han agregado los mensajes **encender, apagar e invertir** y la propiedad del bit al cual se conecta. Al encender o apagar la lámpara, automáticamente se enciende o apaga el bit correspondiente de la interfaz que brindó la lámpara. De la misma manera tenemos un etoy **MotorMerloMorph**, representando un motor paso a paso, que al ser solicitado nos pregunta cuál es el bit inicial de los 4 que lo controlarán. El protocolo de este etoy contiene los mensajes **encender, apagar e invertir** y las propiedades **bitInicial y tiempo**, que representa el tiempo que debe pasar entre cambio de fases del motor. Por último, para poder realizar controles mediante tiempo, creamos un etoy **EtoyLedTimer** que representa un timer. Al mismo podemos resetearlo, y automáticamente realiza cuenta arriba por segundo hasta que se lo resetee nuevamente, comenzando otra vez. En todo momento le puedo solicitar cuántos segundos van desde el último reset. **Este etoy nos es útil para generar eventos relacionados con el paso del tiempo.**

Figura 7 - Menú del etoy de la interfaz

7. Conclusiones y futuros trabajos

Este proyecto ha sido el primer paso de un proyecto más general vinculado al desarrollo de plataformas de software para la robótica educativa. Una de las razones de utilizar Squeak como plataforma, es su simpleza en el desarrollo para espacios de simulación, y porque es uno de los softwares fundamentales del proyecto OLPC (One laptop per child). Por lo tanto, queda como trabajo futuro la construcción de nuevos etoys que puedan ser vinculados a otros kits de robótica, como el Lego Mindstorms y el Nxt, el Lego Dacta, las interfaces que utiliza el Gobierno de la Ciudad de Buenos Aires en sus colegios, y otras interfaces, generando finalmente un framework que unifique el modelo de trabajo de la robótica educativa en este y otros ambientes.

Particularmente en este trabajo, queda como tarea la generación de nuevos etoys que representen motores de continua con inversión, relés, etc. **Además, es necesario portar la capa de bajo nivel del io.dll para que funcione también en el sistema operativo Linux.**

8. Referencias

1. Squeak: Open Personal Computing and Multimedia, Prentice Hall, 2002.
2. B. J. Allen Conn y Kim Rose, "Powerful ideas in the Classroom", Viewpoints Research Institute, 2003..
3. Diego Gómez Deck y José L. Redrejo Rodríguez, "Squeak en España como parte del Proyecto LinEx", 2003.

4. Stephan Ducasse, “Learning Programming in Squeak”, Morgan Kaufman Publisher, 2003.
5. Jorge Fueyo Díaz y otros, “Manual de Squeak”, versión digital en la red, 2004.
6. Fernando Fraga y Adriana Gewerc, “Una experiencia interdisciplinar en Ed. Primaria mediante el uso de Squeak”, Universidad de Santiago de Compostela, 2004.
7. Ana Pizarro Galán y otros, “Un mundo para aprender”, Editorial Editlin, 2005.
8. Diego Gómez Deck, “Programando con Smalltalk: un ambiente de objetos vivos”, Editorial Editlin, 2006.
9. Gonzalo Zabala, “Squeak, el laboratorio infinito”, Revista Novedades Educativas Edición 185, Editorial Novedades Educativas, Mayo 2006.

[i] 1997. Seymour Papert, del Instituto Tecnológico de Massachusetts (MIT). Conferencia: La Familia Conectada: educación y familia en la era de la Internet.
[ii] http://www.smalltalking.net/
[iii] http://www.geekhideout.com/iodll.shtml