Squeak, la progamación del siglo XXI

Mª Dolores Parra Sageras y Carmen Luengo San José
Profesoras de Informática de Enseñanza Secundaria

Síntesis: Descripción de la herramienta Squeak y su aplicación en el aula con alumnos de 1º de Bachillerato.

Descriptores: Informática en educación relacionada con la Programación Orientada a Objetos.

1. INTRODUCCIÓN.

De todas las definiciones que se han dado de **Squeak** la que nos parece más apropiada es aquella que dice que **Squeak** es la *metáfora del mundo*. Realmente es una herramienta que nos permite crear y diseñar cualquier cosa, para ello sólo debemos poner a trabajar nuestra imaginación, creatividad e intuición. Disponemos de un mundo lleno de objetos y nosotros le damos vida, podemos crear objetos nuevos, hacer desaparecer otros y fusionarlos.

Squeak es un entorno de desarrollo integrado para elaborar aplicaciones gráficas interactivas a través de la intuición. Disponemos de un entorno visual con el que interactuamos de la forma deseada.

También podemos decir que **Squeak** es un lenguaje de programación Orientado a Objetos, pero no es sólo eso, que si así fuera ya sería suficiente para exponer sus virtudes.

Es una herramienta que integra los avances tecnológicos de los últimos 30 años puesto que reúne en sí misma imágenes, vídeo, sonido, capacidad de simulación, editor, compilador, sistemas de ventanas, navegador, etc. La gran pregunta sería: ¿se trata de una herramienta de programación o es algo más?, Afirmativamente podemos decir que es todo un sistema operativo, la máquina funciona sólo con **Squeak**.

2. HISTORIA DE SQUEAK

Squeak evoluciona de Smalltalk que es el primer lenguaje Orientado a Objetos. Smalltalk fue creado por Alan Kay y Dan Ingalls en la década de los 70, en él están basados todos los sistemas de ventanas y los lenguajes de programación Orientados a Objetos actuales.

Smalltalk no era sólo un lenguaje de programación sino que se concibió como un sistema operativo ejecutado encima de una máquina virtual lo que hacía que dispusiera de máxima portabilidad.

Hace unos 35 años Alan Kay creó la Dynabook, que era un dispositivo portátil, con red inalámbrica, pantalla plana, táctil y con un sistema operativo visual, debía ser una máquina que no sólo sirviera para concentrar la información del usuario sino que estaba concebida como un *amplificador de la mente.*

Todo esto es muy fácil de imaginar en la actualidad, pero si nos situamos en la informática de los años 70 podríamos decir que Alan Kay, como todos los genios, se adelantó a su tiempo, desarrolló teóricamente el futuro, tanto a nivel de Software como de Hardware. Acertado estuvo cuando dijo que la mejor manera de vivir el futuro es inventarlo.

Smalltalk tenía un objetivo claro, se trataba de que el programador se concentrara al máximo en el sistema que estaba modelando y se olvidara del código.

Los programadores perdían demasiado tiempo aprendiendo un lenguaje de programación, asimilando sus sintaxis y convirtiendo sus ideas en código. Una vez que llegaban a dominar el lenguaje y la sintaxis, podían modelar fácilmente, el problema radicaba en que habían perdido su juventud en ello, y por tanto sus capacidades creativas e intuitivas habían disminuido. De esta forma Smalltalk no esclaviza el programador al código, sino que de forma rápida se pueden empezar a diseñar aplicaciones dejándose llevar por la intuición.

En Smaltalk todo es un objeto al que se le envían mensajes, ejemplos

Ejemplo 1: 2+3

- Objeto: 2
- Mensaje que le enviamos (+), es decir, sumar, con el parámetro 3.

Ejemplo 2: Mostrar el texto "Hola mundo"

- Objeto: *transcript*
- Mensaje: *show* con el parámetro "Hola mundo"

En los 90 Disney contrata a Alan Kay y Dan Ingalls, padres de Smalltalk, para desarrollar un entorno de programación que pudieran utilizar los chicos. De esta forma se ponen en marcha este grupo de desarrolladores para acabar su trabajo en Septiembre de 1996. La única condición que se impuso para realizar este trabajo es que fuera Open-Source, es decir, software libre de código abierto. El resultado fue un Smaltalk enfocado a los chicos y sobretodo a multimedia y 3D.

Tiene un entorno visual muy cercano a los jóvenes y la realización de proyectos se expresa en un lenguaje muy próximo al humano, desprendiéndose del código, ofrece una comunicación sencilla para realizar grandes aplicaciones.

Squeak tiene muchas ventajas entre las que podemos citar que no exige equipamiento de última generación, a menos que nuestras aplicaciones así lo requieran, permite

trabajar en equipo en una red interna o sobre Internet y publicar los proyectos sin necesidad de realizar modificaciones.

Puesto que **Squeak** es un Smalltalk podemos decir que no es sólo un Lenguaje de programación Orientado a Objetos, sino todo un sistema operativo que se monta encima de una máquina virtual. Desde él se pueden realizar presentaciones, ensayos activos, páginas web y aplicaciones multimedia interactivas. Dispone de un potencial enorme, es un laboratorio virtual en el que los chicos pueden experimentar, jugar con sus propios juegos, publicar, editar, en definitiva, realizar todo lo que su imaginación y destreza den de sí.

Squeak, es otra manera de concebir la comunicación con la máquina, tiene bastante que ver con la teoría constructivista de Piaget, donde el alumno construye su conocimiento, a partir, de su propia forma de ser, pensar e interpretar la información. Desde este punto de vista el alumno participa activamente en su proceso de aprendizaje.
Esta teoría mantiene que el conocimiento no se descubre, sino que se construye.

3. PROGRAMACIÓN ORIENTADA A OBJETOS

La evolución de la programación relaciona de forma directa al humano con la máquina.
Los lenguajes de programación son la vía de comunicación de las personas con el computador, mientras que en los primeros tiempos el lenguaje de programación era muy parecido al de la máquina, es decir, los programadores empleaban un modo de comunicación que poco tenía que ver con el lenguaje humano, actualmente los lenguajes se acercan mucho más a la forma de comunicación de las personas y se alejan bastante de la máquina.

La Programación Orientada a Objetos es la última evolución de estos lenguajes y permite resolver problemas complicados de forma más sencilla descomponiendo el problema en subgrupos de partes relacionadas.

La POO- Programación Orientada a Objetos- se basa en el **modelo objeto**, donde el elemento principal es el *objeto*, el cual es una unidad que contiene todas sus características y comportamientos en sí misma, lo que lo hace como un todo independiente, pero que se interrelaciona con objetos de su misma clase o de otras clase, como sucede en el mundo real.

Un *objeto* es una entidad lógica que contiene datos y un código para manipular los datos. Los objetos se obtienen de las clases.
Una *clase* es un tipo de datos que permite crear objetos, podemos decir que el objeto es la concretización de la clase.

Método: Es la implementación de un algoritmo que representa una operación o función que un objeto realiza. El conjunto de los métodos de un objeto determinan el comportamiento del objeto.

La comunicación con un objeto se realiza a través de un *mensaje*, son los que nos permiten activar los *métodos*.

La *herencia* es la propiedad que permite a los objetos construirse a partir de otros objetos. El concepto de herencia está presente en nuestras vidas diarias donde las clases se dividen en subclases.

*Polimorfismo***:** Esta propiedad indica que un elemento puede tomar distintas formas.

4. ESTRUCTURA DE SQUEAK

En **Squeak** todo son objetos, el área de trabajo, los proyectos, los números, los textos, las imágenes, en fin todo lo que nos encontramos frente al monitor es un objeto y por tanto puede ser tratado como tal.

Pero aunque sea un mundo llevado a la máxima simplicidad podemos hablar de una estructura bien organizada. Cuando abrimos **Squeak** nos encontramos con una especie de escritorio que contiene más o menos objetos, este escritorio se llama *mundo*. En realidad el nombre de *mundo* está bien elegido, entramos en nuestro mundo virtual en el que podemos encontrar elementos que pueden ser estáticos o tener movimiento, sonido, lo que deseemos. Podríamos decir que el mundo es un escenario de teatro en el que disponemos de unos personajes, de un decorado, podemos añadir o quitar personajes, podemos cambiar el decorado, o poner diversos decorados. Ese es el mundo, el escenario en el que vamos a desarrollar una obra de teatro, nosotros seremos los autores.

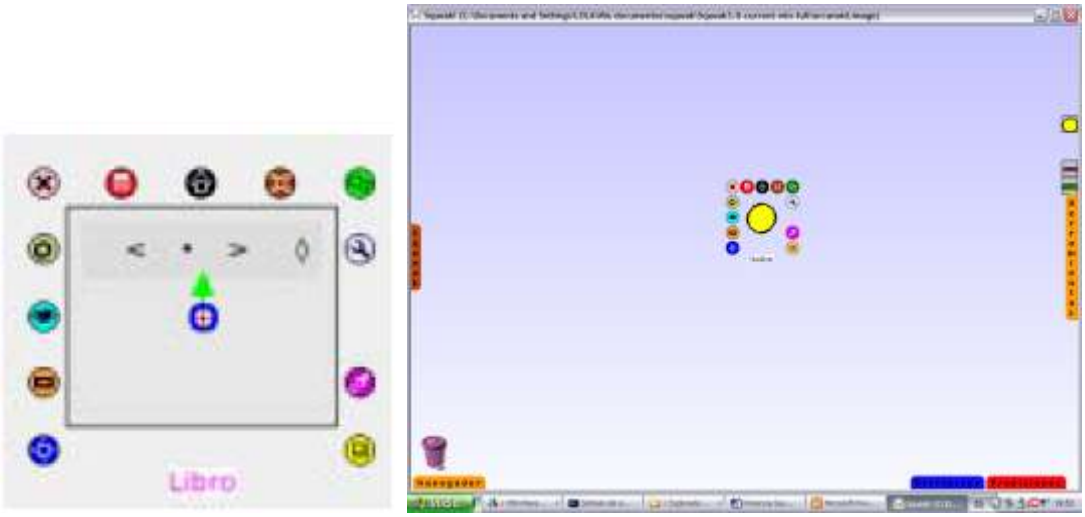
A cada unidad de trabajo la vamos a llamar *proyecto*, un *proyecto* sería como abrir un documento nuevo en un procesador de texto, pero lógicamente en nuestra metáfora deberíamos decir que cada *proyecto* es cada obra de teatro que vamos a representar en el escenario. Podemos representar cuantas obras deseemos, cambiar el decorado, cambiar los personajes y por supuesto, cambiar el texto de la obra. Pero aún es más importante lo que podemos hacer con los *proyectos*, podemos hacer que interactúen, es decir, en una obra de teatro intentar conseguir que entren los personajes y diálogos de otra. ¿Por qué no?.

Una vez que hemos conseguido tener el escenario –*mundo*-, la obra de teatro –*proyecto*- disponer de los personajes o crear nuevos personajes –*objetos*-, debemos darle vida a los personajes, es decir, escribir su *guión*. El *guión* es una de las característica más importantes de los *objetos*.

Como hemos podido exponer anteriormente el *objeto* es el elemento fundamental en el mundo de **Squeak** por ello merece una mención especial en su explicación. Todos los objetos están definidos por unas características comunes, reciben el nombre de *halos*. Mediante los *halos* podemos definir al personaje, cambiarle el color, el tamaño, la posición y obtener las propiedades que lo hacen diferente de otro objeto parecido.

Veamos un par de ejemplos, el primero es el que aparece el *objeto libro* rodeado de sus *halos (véase figura de la izquierda)*, es decir de las características que lo definen o van a definir.

A la derecha nos encontramos con el *mundo* donde hemos colocado el *objeto bolita* con sus *halos*, como podemos apreciar son los mismos que los del *objeto libro.*



Los *objetos* podríamos clasificarlos de dos formas diferentes, los *morfs* y los *eToys*.

Morfs: Es todo lo visible en **Squeak**, cada *morf*, como objeto que es, tiene sus *halos* que nos permiten cambiar el color, el tamaño, su posición, su rotación, etc. Es cada uno de nuestros personajes, lo vestimos, lo peinamos y lo situamos en el escenario. Además un *morf* tiene noción del tiempo, es decir, podemos hacer que cambie de color cada cierto tiempo, por ejemplo.

eToys: Cuando a un *morf* le concedemos un comportamiento lo convertimos en un *eToy*, es decir, cuando escribimos el guión del personaje, le damos vida. En este caso nuestro *morf* se puede mover, se puede ocultar, volver aparecer, girar, danzar, emitir sonido o comportarse de la forma que le indiquemos en el escenario.

Escribir un *guión* no es una tarea fácil, pero tampoco difícil, aquí es donde los alumnos pueden construir su propio conocimiento. El éxito de un buen *proyecto* va a tener mucho que ver con el *guión* que escribamos para sus *objetos*. Es aquí donde va a intervenir la imaginación, la intuición, la participación más activa del aprendizaje. Es aquí donde los iniciados en este mundo van a poder escribir las aplicaciones interactivas sin conocer código ni sintaxis.

Veamos un ejemplo:

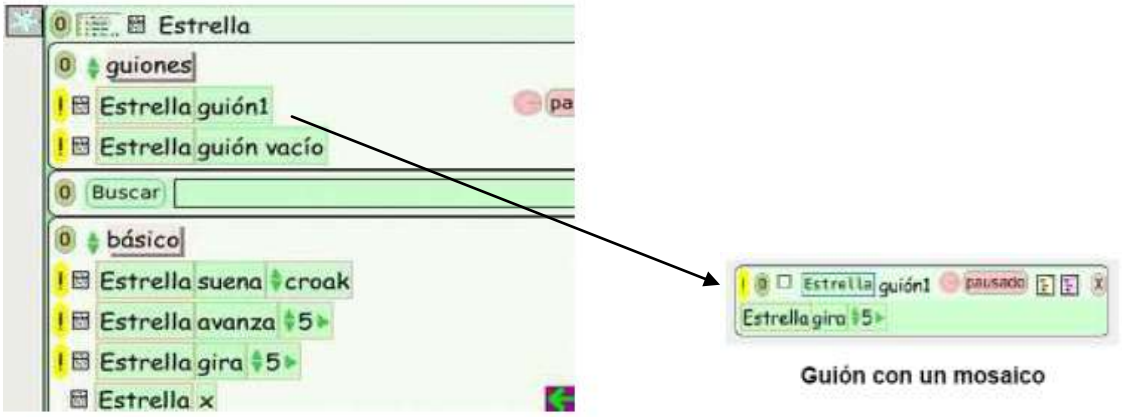
Estrellita con halos

La estrella es un *objeto* con sus *halos* mediante los cuales podemos *vestirla*, cambiar su tamaño, color, posición. Por tanto, es un *morf*.

A continuación y presionando el visualizador, representado con un ojo en los *halos* obtenemos más propiedades, métodos y guiones.

Estrellita con halos

Si deseamos que la estrella gire debemos escribir su *guión*:



Visualizador de nuestra estrella

Guión con un mosaico

De esta forma hemos conseguido convertir nuestra estrella *morf* en un *eToy* escribiendo el *guión1* que consiste únicamente en ver cómo la estrella gira.

Podemos hablar de forma muy extendida de **Squeak** e incluso no acabar, ya que una vez que entras en su mundo es una fuente inagotable de descubrimientos, de estímulo para la creación, pero nos gustaría comentar otra de las posibilidades que ofrece **Squeak**, los *ensayos activos*.

Un ensayo es un escrito, generalmente breve, constituido por pensamientos del autor sobre un tema, sin el aparato ni la extensión que requiere un tratado completo sobre la misma materia.

En **Squeak** podemos realizar *ensayos activos*, lo que implica que podemos desarrollar un tema donde los ejemplos aparezcan de forma activa. Supongamos que estamos realizando un ensayo sobre la balanza, en él vamos explicando el concepto de peso de

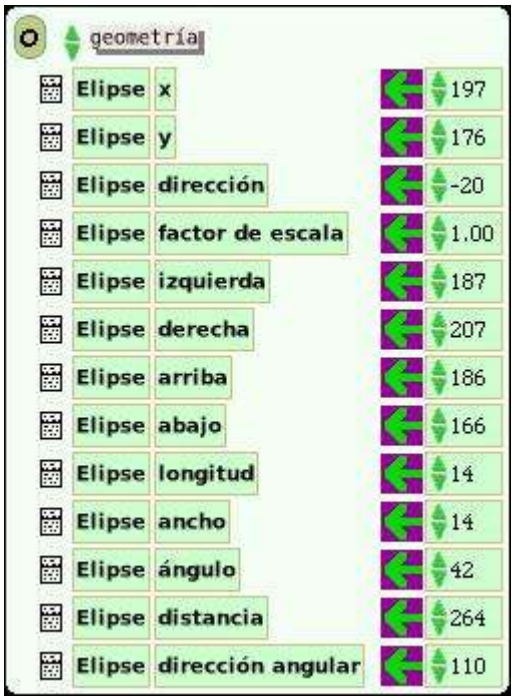
los objetos en el mundo real. Durante nuestra explicación iremos indicando que la balanza sube o baja dependiendo del peso de los elementos que ponemos a cada lado.

Con el *ensayo activo* de **Squeak** no sólo daremos una explicación detallada de la cuestión sino que además mostraremos de forma gráfica la balanza y los objetos a pesar. La novedad reside en que permitiremos al lector del ensayo que interactúe con la balanza, es decir que coloque los objetos en ambos lados de ella y compruebe hacia donde se dirige dependiendo de los pesos de los objetos.

5. **COMPARATIVA DE SQUEAK Y LA PROGRAMACIÓN ORIENTADA A OBJETOS.**

Durante toda la exposición se comenta que **Squeak** es una herramienta de programación Orientada a Objetos, es en lo que más hemos hecho hincapié, a pesar de reiterar que no puede ser considerada sólo eso. Sin embargo, es la parte en la que más nos centramos en esta narración.

El concepto de *Objeto* es un hecho evidente porque en el entorno en que nos movemos todo lo que nuestros sentidos visuales perciben son *objetos*. Un *objeto* lo constituyen propiedades y métodos. Vamos a tomar uno de los objetos de **Squeak** , la elipse.



Todos ellos son propiedades de la elipse, el color, las coordenadas donde se sitúa, el ancho, el largo, … todas ellas hacen la definición del *objeto elipse*.

Los métodos de los objetos son los que definen su comportamiento, vemos algunos de los métodos que acompañan a nuestro *objeto elipse*:

Por *método* entendemos todo aquello que define el comportamiento de un *objeto*, en el ejemplo que acompañamos apreciamos que copiar, borrar, mostrar, ocultar, son *métodos* que definen o pueden definir el comportamiento del *objeto elipse*, activando estos métodos podemos hacer que nuestro objeto se muestre o se oculte según el diseño de la aplicación que estemos modelizando. Para activar un *método* es necesario enviar un *mensaje* al *método* indicando que este comportamiento se ponga en ejecución. Este *mensaje* puede no llevar parámetros, caso de activar el *método mostrar*, o bien puede activarse con un determinado parámetro, caso del *método realizar la opción del menú* donde el parámetro es *enviar para atrás.*

Otra característica destacada en la programación Orientada a Objetos es la *herencia*, donde unos objetos pueden heredar métodos de otros, en este caso podemos hablar del *método borrar* perteneciente a clase *morf Kernel* y que heredan todos los *eToys*.

El *polimorfismo* es otra de las características que definen la programación de la que hablamos, mediante esta propiedad un *objeto* puede tomar formas diferentes. Imaginemos que disponemos de un campo de juego en el que existen *objetos* diversos, bolas, cuadrados, rectángulos…siempre podremos enviar un mensaje para activar el *método borrar* de todos ellos, como por herencia de la *clase morf* todos los *objetos* disponen de ese *método*, por *polimorfismo* conseguimos que todos los *objetos* se comporten como un *morf* y por tanto se borren.

En **Squeak** quedan expresamente reflejados las *clases, objetos, métodos, herencia* y *polimorfismo*. No se podía esperar menos tratándose de un Smalltalk, padre de toda la programación Orientada a Objetos.

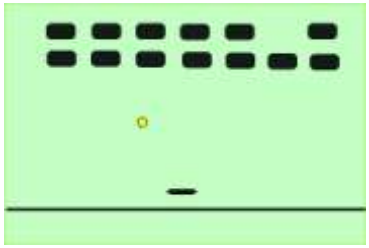
6. EXPERIENCIA EN EL AULA.

Los alumnos que trabajaron con Squeak fueron alumnos de 1º de Bachillerato de las modalidades de Ciencias Sociales y de Ciencias de la Naturaleza y la Salud. Todos ellos desconocían cualquier metodología de la programación, nunca habían diseñado código aunque tenían conocimientos de ofimática, de diseño de páginas web y de manipulación de fotografía. Es decir, estaban muy relacionados con el ordenador pero nunca a nivel de programación. Desconocían plenamente el concepto de *objeto, clase, método, herencia, polimorfismo* e incluso programa.

La edad de los alumnos se encontraba entre 16 y 18 años. Squeak se presentó ante ellos como una herramienta con un potencial de diseño enorme. En un primer momento se

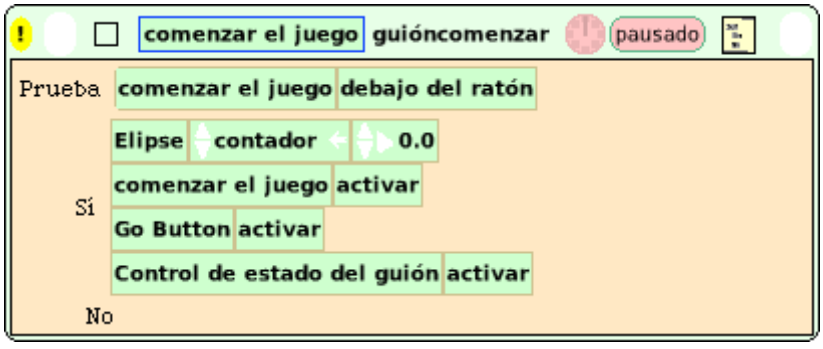
enfrentaron a la herramienta con escepticismo, éste fue disminuyendo con rapidez. Buscaron artefactos, provisiones, herramientas y proyectos realizados por otros estudiantes, la curiosidad les condujo a las preguntas y las ganas de hacer.

La primera propuesta fue hacer mover una bola, comenzamos a mover la bola en un campo de juego y la hicimos rebotar con obstáculos, a los pocos días surgió la idea de implementar el juego arcanoid. Este juego consiste en ir derribando ladrillos de una pared mediante una bola que rebota en un soporte –barrita- en la parte inferior de la pantalla. La barrita se mueve a derecha e izquierda impidiendo perder la bola que derribaba los ladrillos.



Escribieron el guión de los ladrillos, estos debían ocultarse cuando la bola los tocase. Posteriormente se escribió el guión de la bola que debía rebotar en la barrita y desaparecer cuando el jugador la perdiera.

Lo cierto es que se convirtió en un auténtico desafío para ellos, cada vez deseaban complicar más el juego, y fueron diseñando complejos guiones donde usaban variables, contadores, activaban mensajes, enviaban parámetros, expresaban condiciones dentro del código del guión.



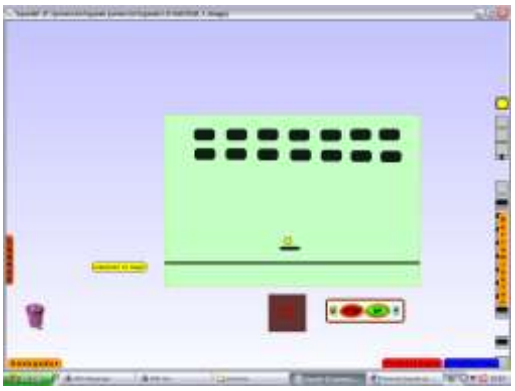
La figura del profesor fue importante para guiar sus ideas, fue el activador de su proceso de aprendizaje, los alumnos comprendían el entorno con facilidad y se hacían dueños de sus herramientas sin necesidad de ayuda, establecían con naturalidad los pasos que

debía seguir el modelo para obtener el objetivo, diseñar el juego, pero era indudable que tenían que asimilar los conceptos de variables, mensajes y métodos.

Momento crítico fue intentar mover la barrita a través del *joystick*, dos objetos diferentes debían estar plenamente relacionados, la intervención del profesor se hizo necesaria para la implementación del guión.

Como conclusión nos gustaría apuntar que en el aula se debe inducir al conocimiento, a la curiosidad, se debe mostrar todos los elementos y herramientas que tienen a su disposición, explicar de forma clara su funcionamiento, pero todo ello ajustado a las necesidades que les van surgiendo a los chicos a medida que modelizan.

El juego arcanoid quedó diseñado, con evidentes deficiencias pendientes de mejora, pero el objetivo se había cumplido, los alumnos habían construido su propio conocimiento, lo demandaban al profesor y deseaban perfeccionar su modelo a medida que transcurrían las clases, el éxito se debió a la magia de Squeak y sobretodo a esa sensación que sentimos todos cuando somos los autores de una obra.



Bibliografía:

http://cprmerida.juntaextremadura.net/squeak/squeak.htm
www.small-land.org
www.squeak.org
www.squeakland.org