


ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación :

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

# SIMULACIÓN DE UN ROBOT PROGRAMABLE CON MACROMEDIA DIRECTOR

David Cordón Blanco

Alfredo Pina Calafi

Pamplona, 28 de Julio de 2010

*Agradezco a mi tutor, Alfredo, por haberme permitido realizar estre proyecto con el cual finalizo mis estudios, por su ayuda para que todo siga adelante.*

*Agradezco a mis compañeros de universidad, especialmente a Daniel, Leticia y María, por todos esos buenos momentos y el apoyo en los malos, por esas tardes de biblioteca, laboratorio, por esas noches de fiesta. Nunca los olvidaré.*

*Agradezco a toda mi familia, en especial a mis padres Ana y Carlos, y mi hermana Maribel, por su constante preocupación en mis estudios, por esos momentos en los que necesitas su apoyo y ellos te lo brindan y por su puesto por estar siempre a mi lado.*

*Agradezco a todos mis amigos que tratan de quitarle importancia a las cosas cuando estoy agobiado, por esos momentos en los que todos nos volvemos a reunir y nos ponemos al día, por esos ratos divertidos, que nunca acaben.*

*Gracias a todos.*

# ÍNDICE

# 1. INTRODUCCIÓN

Actualmente vivimos en una época en la que la tecnología está a la orden del día. Por ello, con la realización de esta aplicación, fomentamos el aprendizaje de los más pequeños desde unas edades muy tempranas.

El objetivo de este proyecto es ofrecer una aplicación para el desarrollo del aprendizaje de los niños, mediante la robótica educativa.

La inspiración para la realización de este proyecto ha sido un robot creado para la educación llamado Bee-Bot. Este robot tiene teclas direccionales que se utilizan para introducir hasta 40 comandos que envían el BeeBot, hacia delante, atrás, izquierda y derecha. Al presionar el botón "GO" el robot comienza el camino que el estudiante ha programado. Los Bee-Bot Mat son una cuadrícula con una hoja de plástico transparente en la parte superior, diseñada para que cada cuadrado represente un paso del Bee-Bot. Hay gran variedad de tableros, con numerosos ejercicios y actividades que fomentan la resolución de problemas, pensamiento crítico y habilidades para tomar decisiones.

Hay muchos lenguajes de programación que nos permiten la realización de éste tipo de aplicaciones. En concreto, son bastantes útiles aquellas programaciones orientadas a multimedia. En nuestro caso Macromedia Director nos ofrece una programación adecuada para la realización de aplicaciones multimedia. Permite integrar con relativa facilidad texto, imágenes, sonidos y video digital, siendo una alternativa a lenguajes más tradicionales, como el C/C++, JAVA, etc. Además el desarrollo de la aplicación es mucho más rápido y flexible. El soporte para publicar suele ser el CD y desde hace algún tiempo, con la ayuda de Adobe Shockwave, también la www. Además, Macromedia Director posee una gran variedad de “xtras” a través de los cuales se puede añadir nuevas funcionalidades como son las bases de datos necesarias para la construcción de una aplicación multiusuario.

El PFC está estructurado en 3 etapas fundamentalmente:

- La primera etapa consiste en estudiar las posibles herramientas y el lenguaje de programación a utilizar. Para programar utilizaremos Lingo, el lenguaje de programación de Macromedia Director. Además de aprender el lenguaje de programación, hay que pensar en la realización de una aplicación multimedia que sea educativa para los niños, y fomente su aprendizaje.

- En la segunda etapa se procedería a la propia programación de la aplicación. Lo cual lleva una nueva investigación del lenguaje de programación para solucionar los diferentes problemas que supone esta etapa.

- En la última etapa, una vez programada la aplicación habría que realizar una interfaz amigable, que sea fácil de usar para un niño y además sea atractiva para ellos.

# 2. LENGUAJES DE PROGRAMACIÓN MULTIMEDIA

Para la realización de este proyecto fin de carrera es necesario sumergirse en los distintos tipos de lenguajes de programación multimedia que existen en el mercado. Para ello vamos a proceder a la descripción de 3 de ellos.

## 2.1 COMIKIT

Comenzaremos con la descripción de ComiKit [11]. ComiKit es un "toolkit" para niños que usa un lenguaje de programación visual basado en tiras de cómics. Con ComiKit se puede crear juegos interactivos e historias con personajes animados. Este es el resultado de una investigación con lenguajes de programación visuales en la Universidad de Linköping. La propuesta de ComiKit es hacer la programación más accesible usando programas que parezcan semejantes al resultado de lo que se ve cuando el programa está funcionando. En definitiva, se trataría de hacer la programación mas fácil para niños convirtiéndola en más visual.

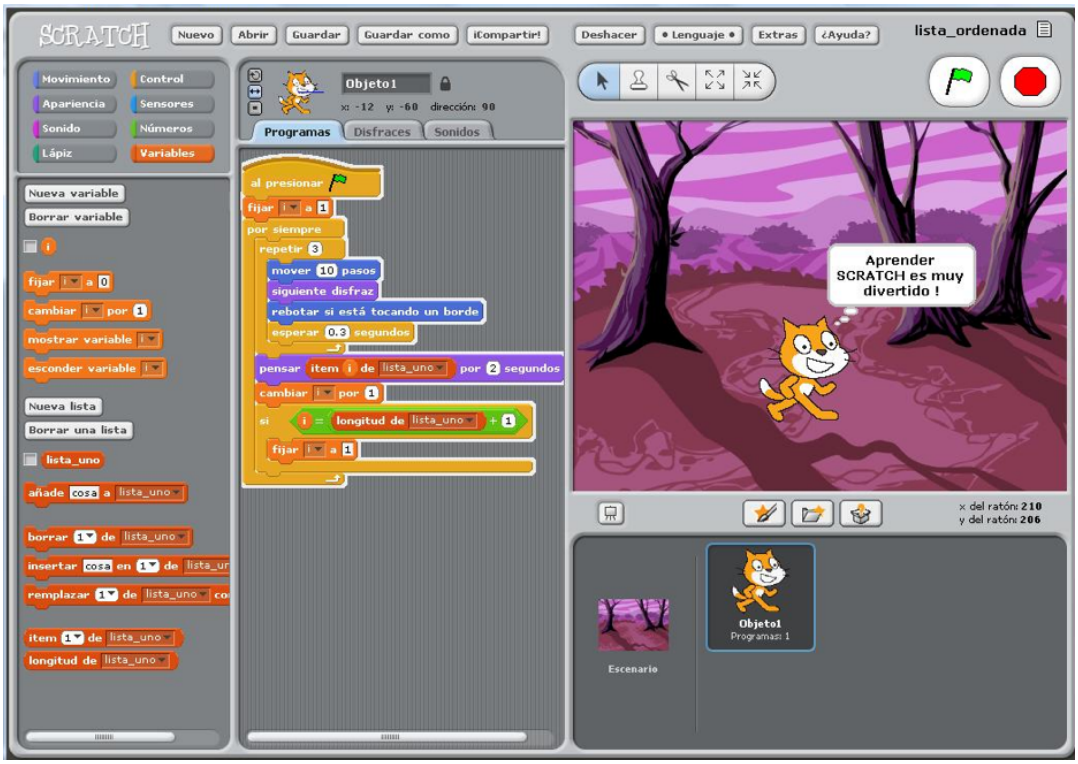
La imagen que se muestra a continuación es el entorno de trabajo de ComiKit.


La forma de trabajar consiste en añadir fondos y personajes y darles un comportamiento a cada uno de ellos. Una vez programado que debe hacer cada personaje se pulsa "play" y todo se mueve según la programación.

## 2.2 SCRATCH

Scratch [10] es una aplicación informática destinada principalmente a los niños que les permite explorar y experimentar con los conceptos de programación de ordenadores mediante el uso de una sencilla interfaz gráfica. En el diseño, la principal prioridad es hacer que el lenguaje y entorno de desarrollo sea muy intuitivo y fácil de aprender para los niños que no tengan experiencia previa en programación. Scratch es una herramienta que utiliza etiquetas para programar. Al usar etiquetas no requiere escribir sentencias, lo que elimina los errores convencionales de sintaxis.

La motivación para programar es la posibilidad que permite a los niños crear juegos y mundos dinámicos e interactivos.




La interfaz de usuario para el entorno de desarrollo de Scratch divide la pantalla en varios paneles: a la izquierda se encuentra la paleta de bloques, en medio de la información sprite actual y el área de scripts, ya la derecha del escenario y la lista de sprites. La programación consiste en ir desplazando los diferentes comportamientos al área de scripts de cada objeto.

Ésta programación multimedia es un poco mas compleja que el anterior lenguaje descrito ComiKit, ya que permite programar condiciones “if”, bucles, y declaración de variables.

## 2.3 MACROMEDIA DIRECTOR

Macromedia Director [1] es una aplicación de Desarrollo de Software Multimedia destinado para la producción de programas ejecutables ricos en contenido multimedia. Es considerada una de las herramientas más poderosas de integración y programación de medios digitales, debido a su versatilidad de poder incorporar imágenes, audio, video digital, películas flash, y un motor 3D, en una sola aplicación, y manipularlas a través de un lenguaje de programación Lingo o JavaScript.

Desarrollado originalmente por la empresa Macromedia, es actualmente distribuido por Adobe Systems Incorporated.

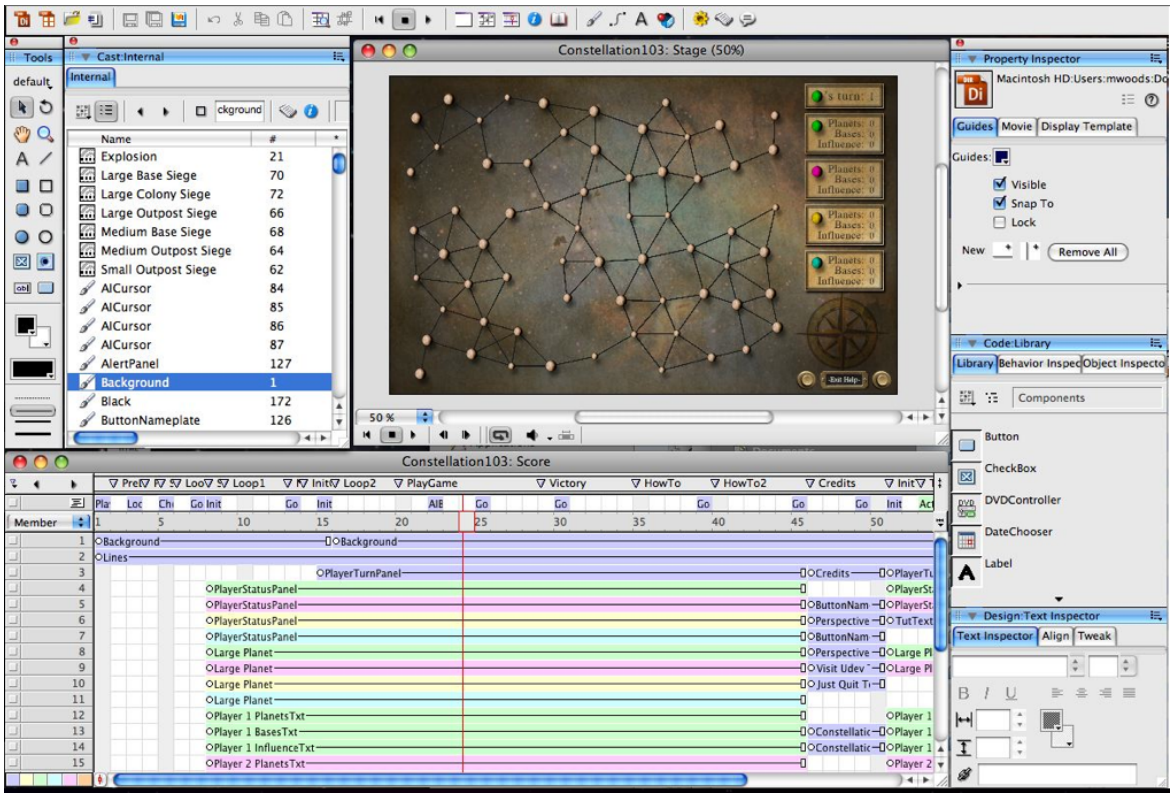
Las presentaciones multimedia generadas por Director pueden ser distribuidas a través de diversos medios, como discos digitales CD, DVD o cualquier otro soporte de información binaria como "pendrives", tarjetas de memoria, discos duros. También permite ser distribuido y ejecutado directamente en plataformas Web gracias al formato Shockwave, creado para esos fines. Con Director también es posible programar una amplia gama de aplicaciones basadas en redes, lo que ha permitido crear innumerables sistemas y juegos multiusuario a través de la red.

Director también permite la manipulación de modelos en 3D, gracias a Shockwave 3D. Es así como diversos programas de modelamiento, como 3D Studio MAX, permiten exportar sus modelos, incluyendo las animaciones, en formato Shockwave 3D, el que puede ser importado a Director, y manipulado a través de instrucciones. A través de variados Xtras, como Havok, Director también puede manipular propiedades físicas de modelos 3D (como por ejemplo, gravedad, coeficientes de roce, restitución, etc.) que permiten lograr simulaciones más realistas, tanto para software de ingeniería avanzada, como para juegos.

Además del potente lenguaje incorporado Lingo, una de sus principales ventajas radica en el uso de los llamados xtras. Se trata de pequeños programas o "plugins" desarrollados en lenguaje C++ por otros usuarios o terceras empresas, que proporcionan al usuario infinidad de utilidades.

Se pueden generar varios tipos de archivos, sin embargo lo más normal es crear un archivo ejecutable para Windows (.exe) o Macintosh (.app). De esta forma puede verse la presentación en cualquier ordenador, sin tener instalado Adobe Director.

Con el lanzamiento de Director 11 y su evolución a la versión 11.5, de la mano de Adobe, se incorporó soporte para DirectX y se extendieron las capacidades en 3D basadas en el "engine" PhysX de NVIDIA, importación de 3D desde Google SketchUp, así como también filtros de "bitmaps", canales de audio 5.1, video en alta definición, soporte para H.264, e integración de Adobe Flash CS3 y Shockwave Player 11.




El entorno de trabajo de Macromedia Director se puede dividir en movie, stage, cast, score y script. El usuario es como el director de la película, que controla todos sus aspectos. La creación o "armado" de una película *(movie)* se realiza sobre un escenario (Stage). Para ello, existen ventanas como el reparto de "actores" *(cast)*, otra para el montaje *(score)*, otra para los guiones *(scripts)*.

Una vez analizados los diferentes lenguajes de programación que hay, elegimos Macromedia Director para la realización de nuestro proyecto. Su gran potencia de programación gracias a Lingo, y su facilidad para manejar imágenes, audio, y video, hace que sea el idóneo para este proyecto.

# 3. BEEBOT

## 3.1 BEEBOT ORIGINAL

La inspiración de este proyecto ha sido BeeBot [9]. Éste ha sido creado por una empresa inglesa llamada TTS Group Limited.

Bee-Bot es un robot visualmente atractivo y programable. Es útil para apoyar el desarrollo de habilidades en una amplia gama de áreas. Se permite a los estudiantes dar instrucciones con diferentes grados de dificultad. También puede ser utilizado por alumnos con necesidades educativas especiales. Sus principales características son el desarrollo de habilidades de motricidad mediante el uso de los botones de dirección, el juego imaginativo a través del uso de diferentes tableros, y demostrar las habilidades del estudiante de una manera que un enfoque tradicional no podrían demostrarse.

El robot consta de teclas direccionales que se utilizan para introducir hasta 40 comandos que envían el BeeBot la dirección que debe tomar. Una vez programado el camino a realizar se presiona el botón "GO" y el robot comienza a realizar el camino.

Para jugar con el robot se han creado los BeeBot Mat. Son una cuadrícula con una hoja de plástico transparente en la parte superior, diseñada para que cada cuadrado representa un paso BeeBot. Hay gran variedad de tableros, con numerosos ejercicios y actividades que fomentan la resolución de problemas, pensamiento crítico y habilidades para tomar decisiones.

El precio del BeeBot es de unos 65€. Y los tableros BeeBot Mats varían entre 2.5€ y 25€. Además si quieres disponer del software del ordenador también hay que comprarlo. Una licencia de dicho software vale 35€. En total un pack básico puede costar unos 135€.


El software se compone de 4 partes fundamentalmente. La parte central, donde se ejecuta los comandos que se han programado previamente. En la parte de la izquierda se encuentran los comandos de dirección, con los cuales programaremos el robot. Arriba se encuentra una barra de herramientas con las cuales podemos guardar o abrir un camino, o cambiar a los diferentes tipos de vista que BeeBot nos ofrece. Y por último a la derecha se encuentran los diferentes juegos de este software.

## 3.2 PLANTEAMIENTOS DEL PROYECTO

Una vez descrito el BeeBot, el cual ha sido nuestra inspiración, nuestro objetivo para este proyecto va a ser la creación de un software libre que se asemeje al BeeBot y permita desarrollar las mismas habilidades de aprendizaje.

Esta herramienta consistirá en una aplicación multimedia y multiusuario. Por lo tanto, además de aprender y mejorar día a día, cada usuario podrá tener guardados los caminos que haya ido programando a lo largo de su aprendizaje.

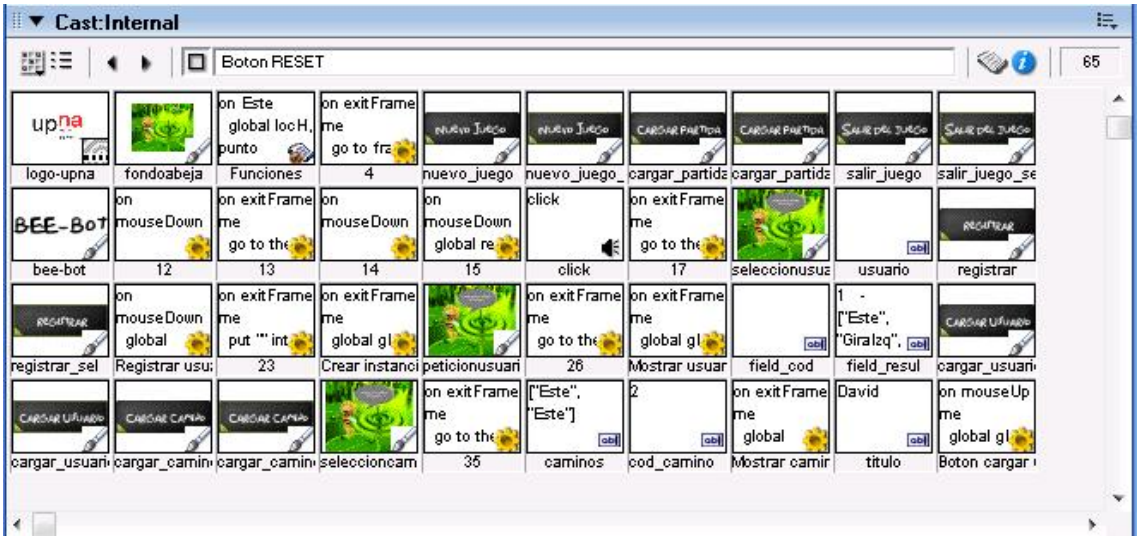
Además con la realización de este proyecto se pretende crear un punto de inicio para futuros proyectos, como puede ser la modelización y creación de un robot. Mejora de este software añadiéndole nuevas funcionalidades. O incluso pruebas con niños y profesores una vez creado el robot y el software.

# 4. PROGRAMACIÓN DE BEEBOT CON MACROMEDIA DIRECTOR MX 2004

## 4.1 INTRODUCCIÓN A MACROMEDIA DIRECTOR

- La ventana Cast (Menu -> Windows -> Cast).

  En esta ventana es donde almacenamos los elementos importados o creados en el mismo Director, que formarán parte de la película. Este es el "almacén" de nuestro proyecto.




  Los diferentes elementos que podemos encontrar en esta ventana son:

  - Navegación por el Cast: Nos permite avanzar y retroceder a través de los miembros del reparto.

  - Utilidad para mover los miembros del reparto: Esta es una herramienta muy útil cuando tenemos un gran número de elementos en el cast y queremos mover alguno a otra posición no visible. Para hacerlo, seleccionamos el miembro del cast que queremos mover y luego arrastramos el cuadrado hacia el lugar del Cast a donde queremos mover este miembro seleccionado. Al soltar sobre la casilla que queramos, veremos que el miembro que teníamos seleccionado se coloca en esa casilla.

  - Script e Info del miembro de reparto: Estos son dos botones fundamentales para trabajar en el Cast.

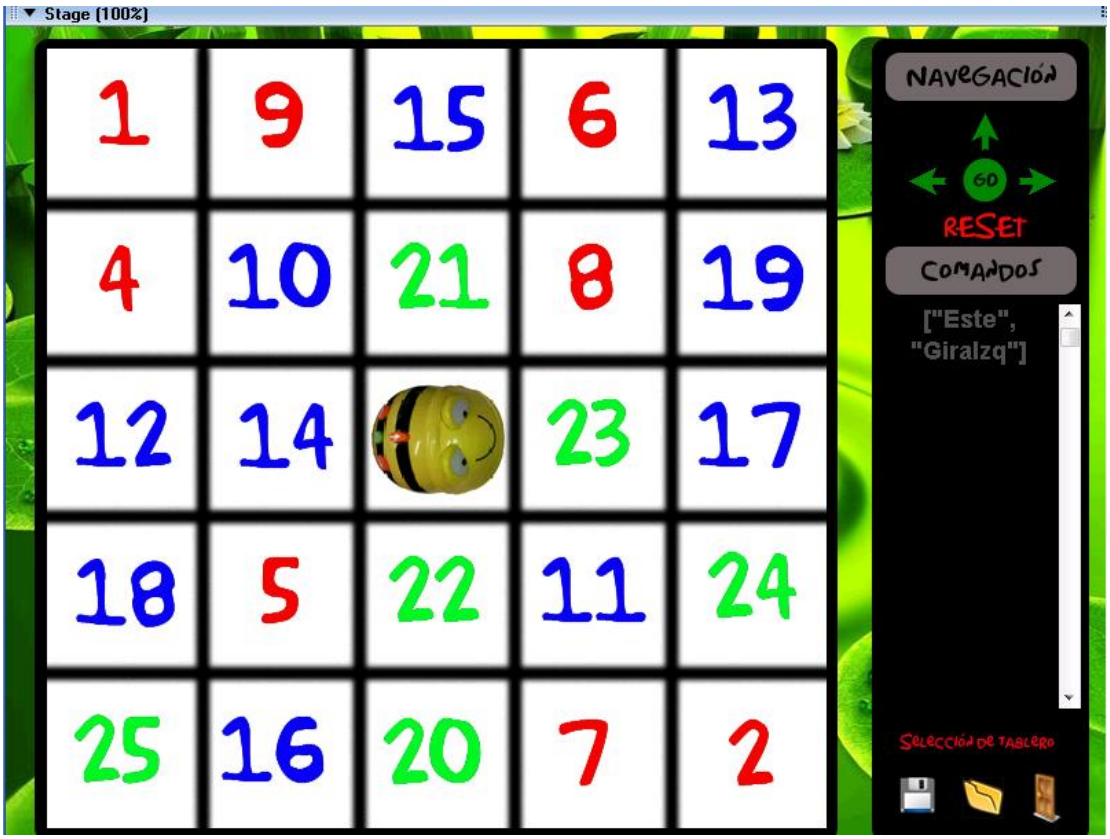
    En el primero (script) podemos asignar un Script al miembro seleccionado. De esta forma este miembro, al ser colocado en el escenario se comportará tal y como se le haya asignado en el script de Cast, independientemente de donde este colocado. Por ejemplo, si a un miembro le asignamos el script:

```
on mouseDown
    beep
end
```

este personaje siempre reproducirá un sonido de alerta del sistema cada vez que se haga clic sobre él.

El segundo botón (info) nos sirve para asignar propiedades particulares al miembro del reparto. Más adelante veremos ejemplos en donde utilizaremos estas dos herramientas.

- La ventana Stage (Menu Window -> Stage).




El stage es la pantalla principal donde se ejecutara la película. Un miembro de reparto se convierte en *sprite* cuando se incorpora en el *stage*. Un *sprite* no es el gráfico que aparece dentro de la ventana *cast,* sino un clon o instancia del miembro de reparto cuando se ubica en el *stage.* Pueden colocarse sobre el *stage* múltiples copias del miembro de reparto. Los cambios que se realicen sobre el miembro de reparto sí afectarán a su *sprite* asociado, pero los cambios que se hagan a un *sprite* no afectarán a su miembro de reparto. Un *sprite* se puede desplazar por el *stage* y posee atributos independientes de su miembro de reparto asociado.

- La ventana Tool Palette (Menu Window -> Tool Palette).


Esta es la paleta de herramientas de Director. Las diferentes herramientas que nos encontramos en ella son:

- La primera herramienta es el puntero. Sirve para seleccionar y mover objetos por el Stage (escenario).

- La segunda es la herramienta de giro. Sirve para girar de forma libre los sprites que hemos colocado en el escenario.

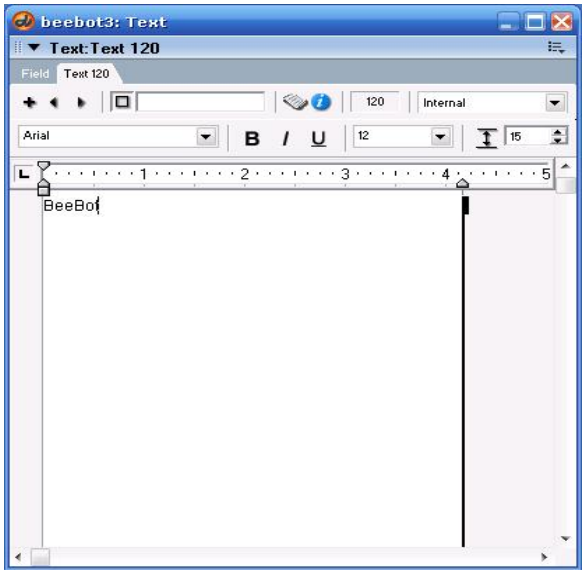
- Luego tenemos las figuras con relleno y sin relleno. Estas figuras tiene la particularidad de ocupar muy poca memoria, lo que agilizará nuestra película si se reproduce en Internet, por ejemplo, en donde la velocidad de ejecución es un punto crítico.

- Los botones de selección y radio, son los típicos botones que encontramos en aplicaciones y páginas Web.

- Las herramienta siguiente nos sirve para asignar un color de fondo y frontal a nuestras figuras, campos de texto, etc.

- Las últimas herramientas son las de color y grosor de línea.

- Inspector de texto (Menu Window -> Inspector -> Text).



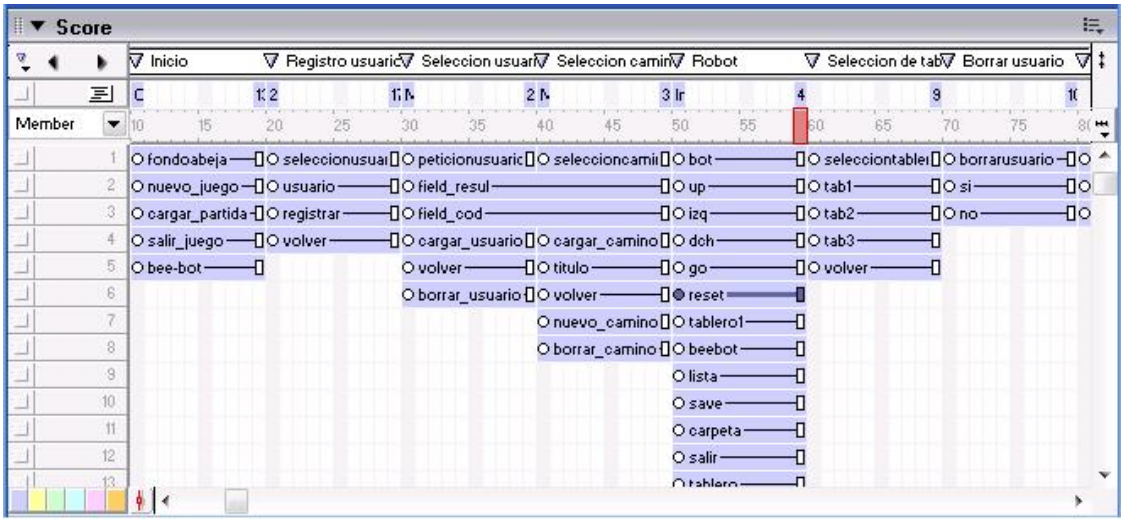

Aquí asignamos tipos de letra, alineación, espacio entre líneas, etc, para los textos y campos de texto.

• La ventana Behaviour (Menu Window -> Inspector -> Behaviour).


Es un pequeño asistente que nos permite asignar Scripts de Lingo, sin necesidad de escribir el código, sino seleccionando los handlers y las acciones que queramos ejecutar. Utilizar la ventana Behavior no es la forma más limpia de programar en Director, y siempre que podamos, debemos escribir el código nosotros mismos. Esto no es una simple cuestión de estilo, sino que el abuso de los scripts generados en la ventana behavior puede llegar a saturar proyectos medianamente grandes, provocando comportamientos inesperados. Las indicaciones del gráfico, de momento, son lo suficientemente explicativas. Cuando entremos en temas de Lingo, aprenderemos a utilizar los scripts de forma extensa y pormenorizada.

• La ventana Score (Menu Window -> Score).

El Score, es sin duda, la ventana que más intimida a los principiantes que se acercan a Director. Sin embargo, una vez que comprendemos sus funciones, y empezamos a hacer nuestros primeros juegos en ella, se va transformando simplemente en una herramienta poderosa y versátil.



Debemos entender el Score, como un secuenciador, o una partitura, en donde colocamos todos los elementos de nuestra película. Cuando la cabeza de reproducción comienza, va leyendo todo lo que encuentra, a la velocidad que le hayamos asignado, interpretando y obedeciendo a los scripts que hayamos puesto en su camino y moviendo o mostrando los sprites que hayamos colocado en los canales.

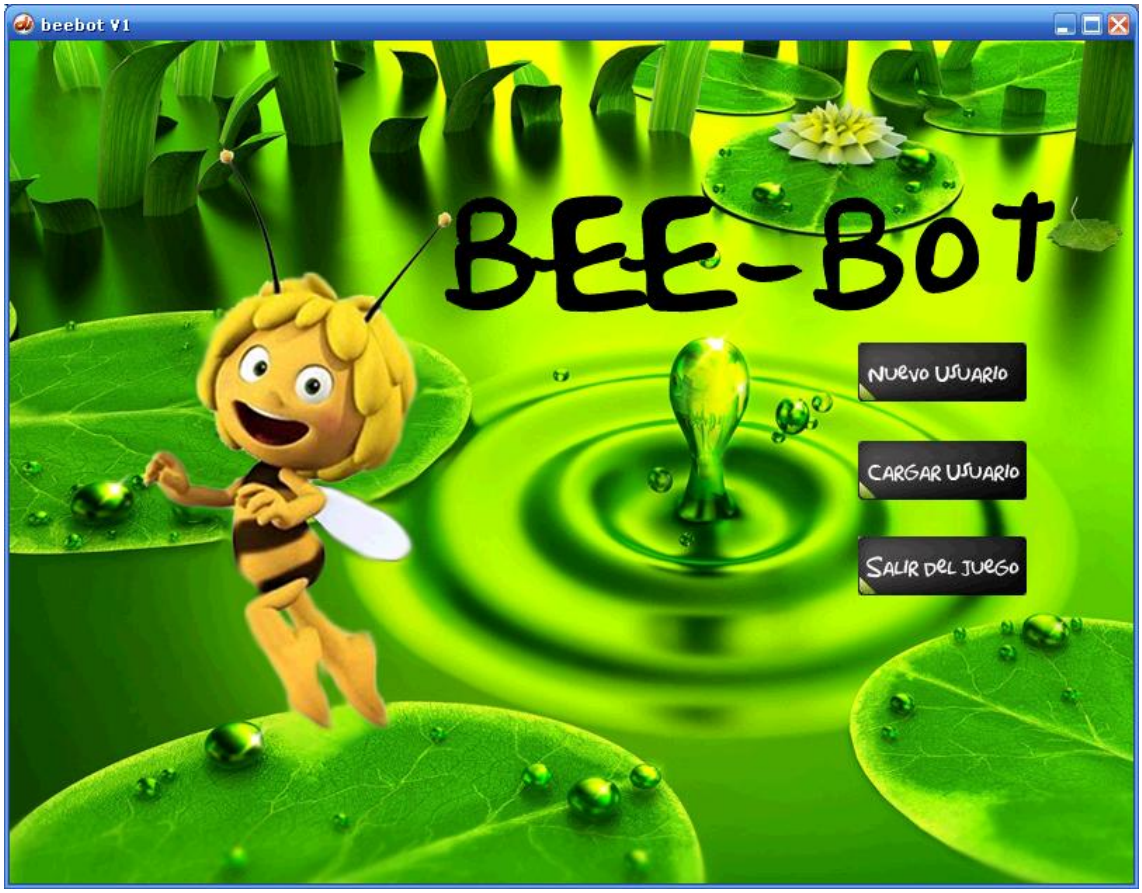
## 4.2 DISEÑO DE BEEBOT

El BeeBot es una herramienta dirigida a niños. Tanto los colores como los gráficos han sido escogidos para que sean llamativos. Los niños pequeños necesitan materiales que se adapten a sus edades. Los juguetes que tienen mucho color y son atractivos son buenos, ya que consiguen captar su atención.

Los niños a su corta edad son capaces de desarrollar sus sentidos y sus destrezas motoras. Cuando empiezan a manipular juguetes, aprenden pre-matemáticas, pre-alfabetización y destrezas de escritura. Estas son todas precursoras de actividades complejas en las cuales se involucrarán después durante su desarrollo.

Es muy importante escoger los juguetes que permiten que los niños usen su imaginación. Los juguetes de múltiples posibilidades pueden crecer y adaptarse al niño. También pueden desarrollar las destrezas de lenguaje de un niño, porque pueden describir actividades que pueden hacer con esos juguetes. Les ayudan a desarrollar conceptos y racionalizaciones.

Las imágenes y los botones han sido diseñados de acuerdo a las características anteriores con Adobe PhotoShop para lograr que los estudiantes se involucren más con la aplicación.



## 4.3 BOTONES ROLLOVER

Un rollover o imagen de sustitución es un efecto muy conocido en donde una imagen existente es sustituida por otro cuando el puntero del ratón se posa sobre ella, volviendo a la imagen anterior cuando quitamos el puntero de ella.

Para este proyecto se han utilizado rollovers para todos los botones ya que este método aporta una apariencia muy dinámica.

En el caso del menú se ha optado por un realzamiento de los botones. Este se consigue aplicando una línea alrededor del botón.



Sin embargo en los botones para programar el robot, cuando pasas el ratón por encima se produce una ampliación de la imagen produciendo un efecto muy dinámico.



Para que este efecto se produzca hay que introducir un script para el botón. El código es el siguiente:

```
on mouseEnter
        sprite(6).member = "reset_sel"
end


on mouseLeave
        sprite(6).member= "reset"
end
```

## 4.4 DIRECCIÓN Y POSICIONAMIENTO

Cuando programamos el robot, éste tiene que moverse por el tablero. Para determinar en que posición se encuentra el robot y que sepa en que dirección tiene que ir, se ha pensado en construir una matriz 5x5, donde en cada punto puede ir en 4 direcciones (Norte, Sur, Este, Oeste). Exceptuando los bordes, ya que el BeeBot no podrá salir del tablero.

| | | | | |
|---|---|---|---|---|
| (1,1) | (1,2) | (1,3) | (1,4) | (1,5) |
| (2,1) | (2,2) | (2,3) | (2,4) | (2,5) |
| (3,1) | (3,2) | (3,3) | (3,4) | (3,5) |
| (4,1) | (4,2) | (4,3) | (4,4) | (4,5) |
| (5,1) | (5,2) | (5,3) | (5,4) | (5,5) |

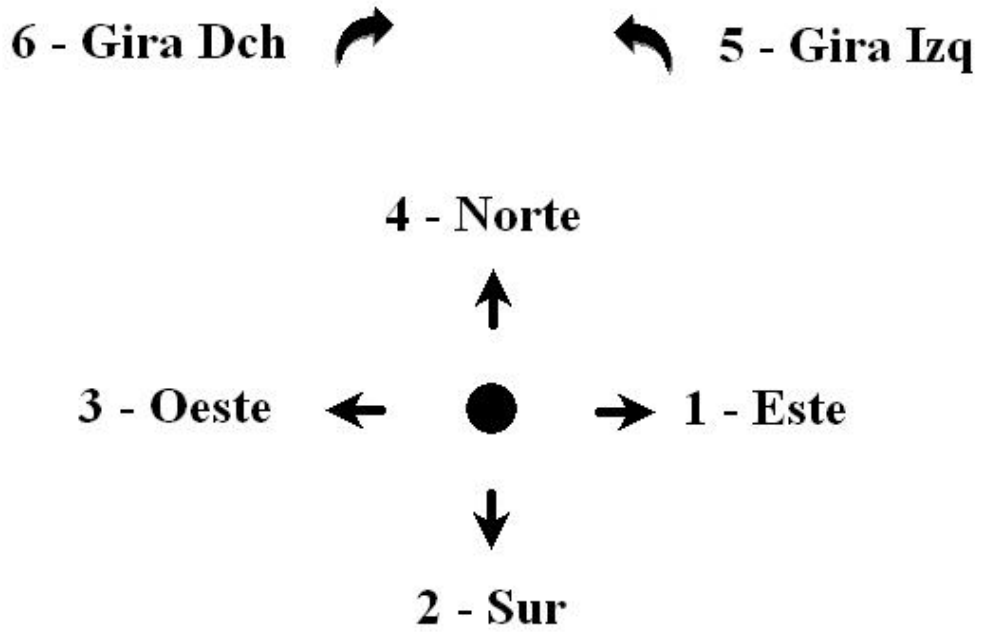
Para programar el BeeBot hay que pulsar las flechas de dirección situadas en la derecha de la pantalla de programación del robot.


Cada vez que pulsamos en una flecha la aplicación guarda en una lista todos los movimientos programados. El robot siempre parte desde la posición (3,3) del tablero y apuntando hacia la dirección Este, que equivale al número 1.

La flecha de navegación “arriba” es la que hace que el robot avance según en la dirección que apunte, y las flechas “izquierda” y “derecha” hacen que la dirección del robot cambie.

Para dejar esto claro propongo un ejemplo:

La lista con las direcciones [1,1,5,4,5,3] indica que el robot hará los siguientes movimientos [Este, Este, Gira_Izq, Norte, Gira_Izq, Oeste]. Esta segunda lista con los nombres del camino programado quedará guardado en otra lista con el fin de mostrarlos al usuario de forma mas clara.




Como he explicado anteriormente, el robot no se puede salir del tablero. Para solucionar este problema se ha construido una función que se encarga de comprobar si puede realizar el siguiente paso antes de que el robot avance una casilla. El código de esta función se detalla en el anexo.

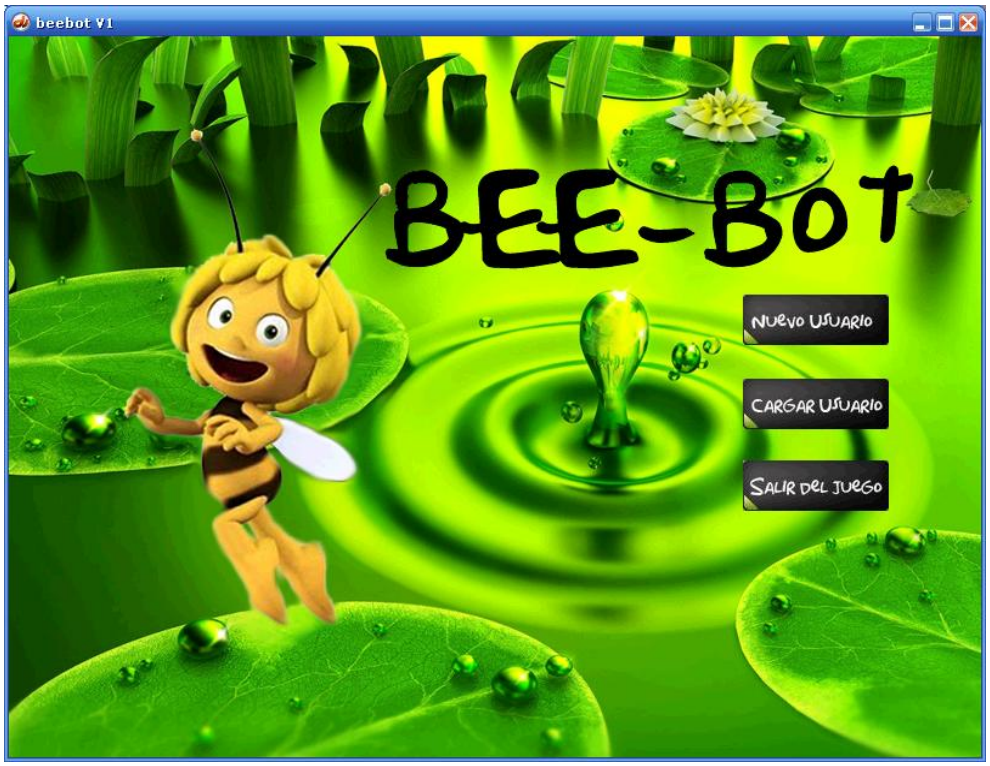
## 4.5 NAVEGACIÓN

En este capítulo analizaremos todas las pantallas de navegación del BeeBot. Para cada una de ellas se hace una explicación de los botones, inputs, etc, que tiene y para que sirven cada uno de ellos.

Nada mas ejecutar la aplicación aparece un alert que nos indica que la versión del "xtra" utilizado está sin registrar. Esto no es ningún problema ya que no afecta al funcionamiento de la aplicación.


- Pantalla principal, contiene:

  - Nuevo Usuario: Botón para crear un nuevo usuario.
  - Cargar Usuario: Botón para cargar un usuario existente.
  - Salir del juego: Botón para salir del juego.



- Pantalla para salir del juego, contiene:

  - Si: Botón para salir del juego.
  - No: Botón para volver al menú principal.

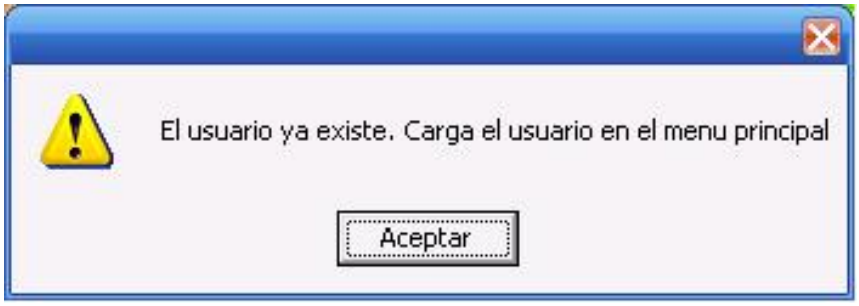

- Pantalla de nuevo usuario, contiene:

  - Botón volver: Vuelve al menú principal.
  - Registrar: Botón para registrar el usuario.
  - Input usuario: Input para introducir el nombre de usuario a registrar.

- Alert de creación de usuario: Mensaje para avisar que el usuario ha sido creado. Después de este mensaje la aplicación te lleva a la pantalla del robot para empezar a programar.

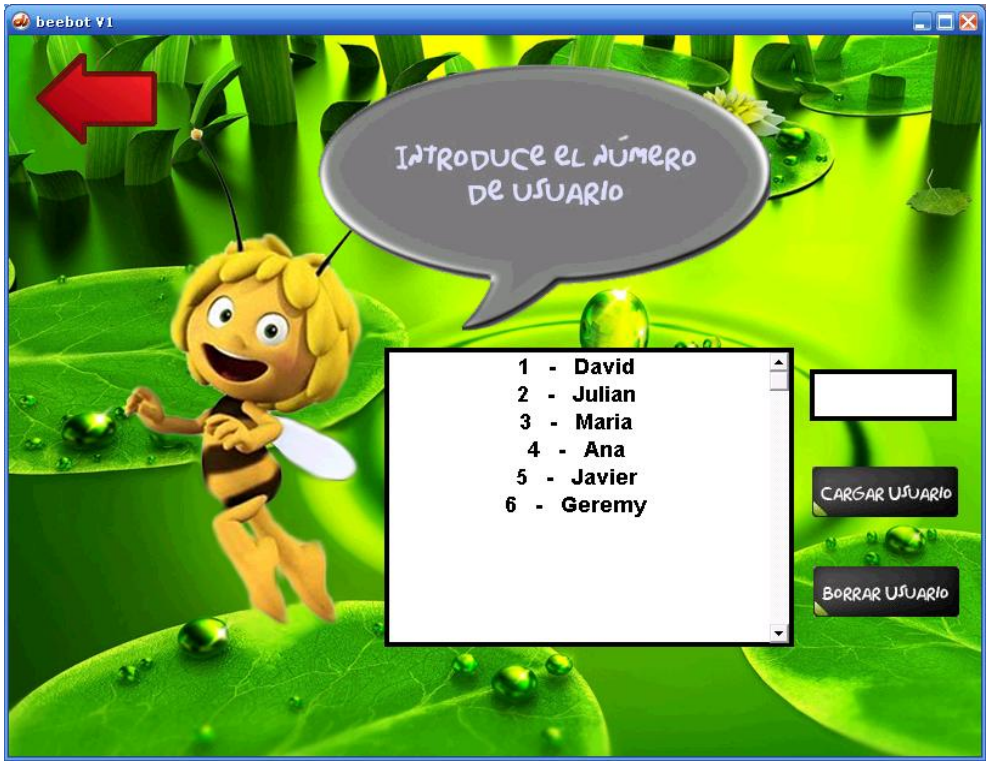

- Alert de error usuario en blanco: este error aparece si no se ha introducido ningún nombre en el input de usuario.


- Alert de error de usuario existente: el error aparece si se introduce un usuario que ya existe en la base de datos.



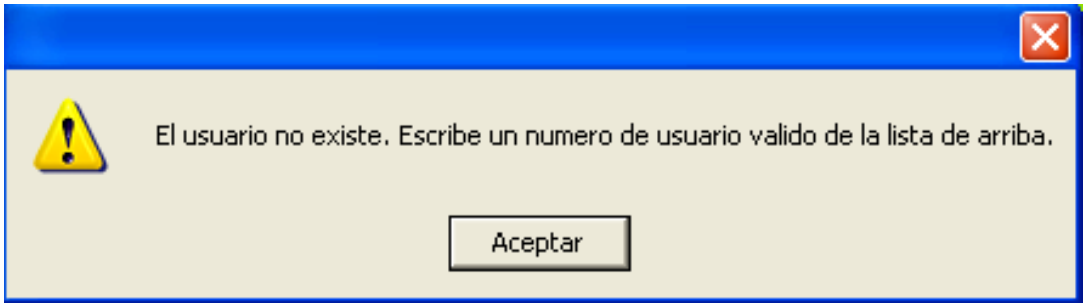

- Pantalla de carga de usuario, contiene:

- Cargar usuario: Botón para cargar el usuario seleccionado.
- Borrar usuario: Botón para borrar el usuario seleccionado.
- Botón volver: Botón para volver a la pantalla principal.
- Input de usuario: Input para introducir el número de usuario a cargar.
- Output de usuario: Output que muestra el código y el nombre de los usuarios existentes en la base de datos.




- Alert de error de input de usuario en blanco: el error aparece si no se introduce un código de usuario en el input.


- Alert de error de input de usuario inexistente: el error aparece si el código de usuario introducido en el input no existe.



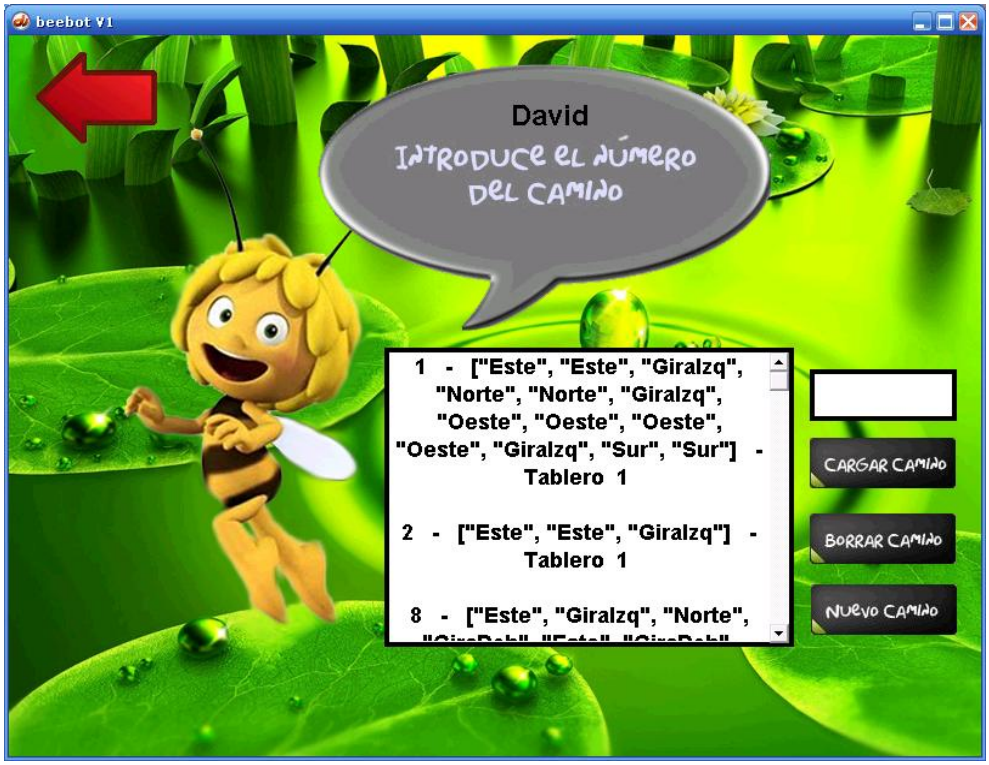
- Pantalla de borrado de usuario, contiene:

  - Si: Botón para borrar el usuario.
  - No: Botón para cancelar el borrado de usuario.


- Alert de usuario borrado: el alert aparece para mostrar que el usuario a sido borrado.

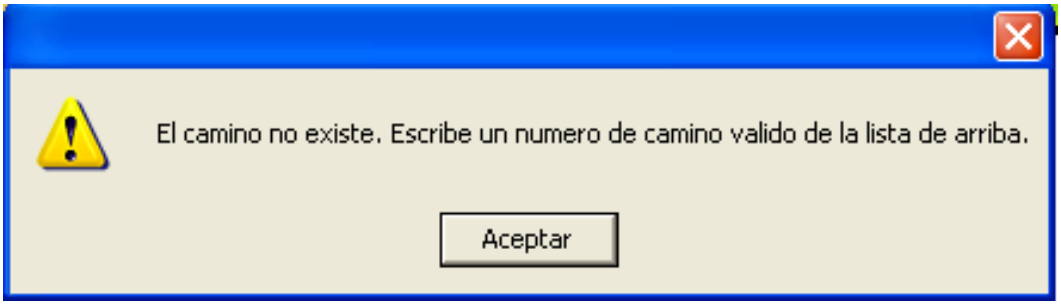

- Pantalla de carga de caminos, contiene:

- Nuevo camino: Botón para crear un nuevo camino.
- Cargar camino: Botón para cargar el camino seleccionado.
- Borrar camino: Botón para borrar el camino seleccionado.
- Botón volver: Botón para volver a la pantalla de selección de usuario.
- Input de camino: Input para introducir el número de camino a cargar.
- Output de caminos: Output que muestra el código, el camino y el tablero utilizado la vez que el usuario guardó ese camino.




- Alert de error de input de camino en blanco: el error aparece si no se introduce un código de camino en el input.


- Alert de error de input de camino inexistente: el error aparece si el código de camino introducido en el input no existe.



- Pantalla de borrado de camino, contiene:

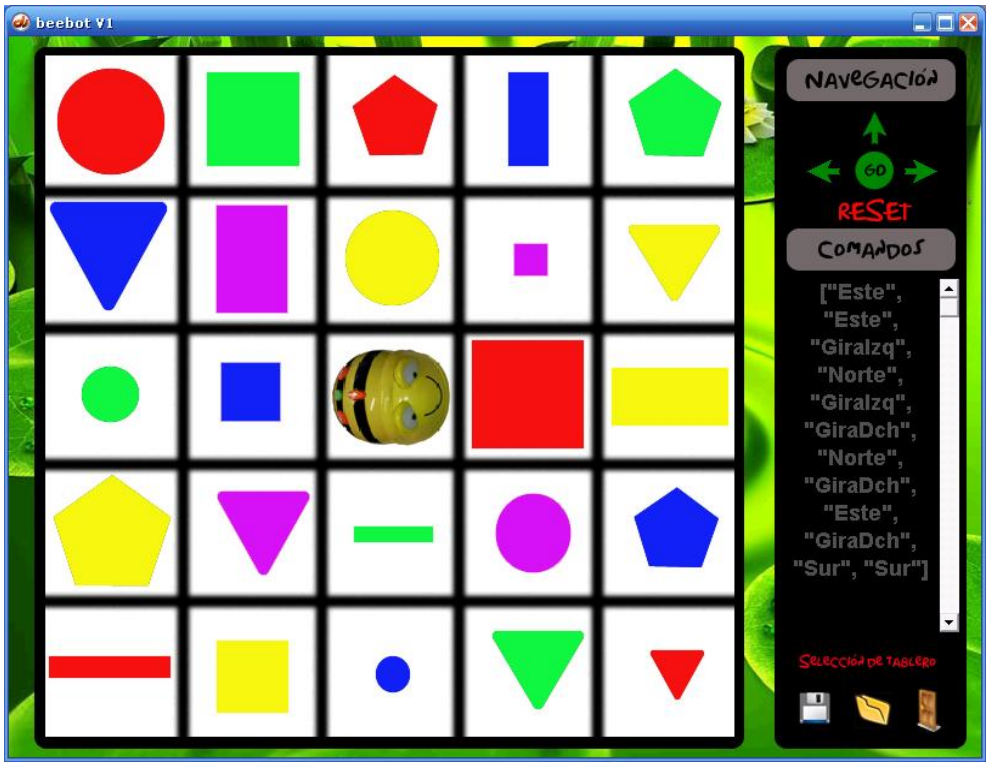
  - Si: Botón para borrar el camino.
  - No: Botón para cancelar el borrado del camino.


- Alert de camino borrado: el alert aparece para mostrar que el camino a sido borrado.


- Pantalla de carga de programación del robot, contiene:

  - Flechas de navegación: Flechas para ir arriba, girar a la izquierda y girar a la derecha. Se utilizan en la programación del robot.
  - Go: Ejecuta la lista de movimientos programada por el usuario.
  - Reset: Borra todos los comandos y devuelve el robot a su posición inicial.
  - Field comandos: Muestra los comandos que ha programado el usuario.
  - BeeBot: Robot que simula los movimientos programados por el estudiante.

- Tablero: Escenario por donde se mueve el BeeBot. El robot no puede salir del tablero.
- Selección de tablero: Botón que te lleva al menú de selección de tablero.
- Guardar camino: Botón que sirve para guardar el camino programado.
- Abrir camino guardado: Botón que te lleva al menú de selección de caminos guardados.
- Salir: Botón para salir al menú principal.




- Alert de juego guardado: el alert aparece para mostrar que el camino a sido guardado en la base de datos.

- Pantalla de selección de tablero, contiene:

- Formas y colores: Botón para seleccionar el tablero formas y colores.
- Números: Botón para seleccionar el tablero números.
- Animales: Botón para seleccionar el tablero animales.
- Botón volver: Botón para volver a la pantalla de programación del robot.

## 4.6 JUEGOS Y DISCIPLINAS DE APRENDIZAJE

Los juegos son el lenguaje principal de los niños, éstos se comunican con el mundo a través del juego.

El juego estimula todos los sentidos, además muestra la ruta a la vida interior de los niños, expresan sus deseos, fantasías, temores y conflictos.

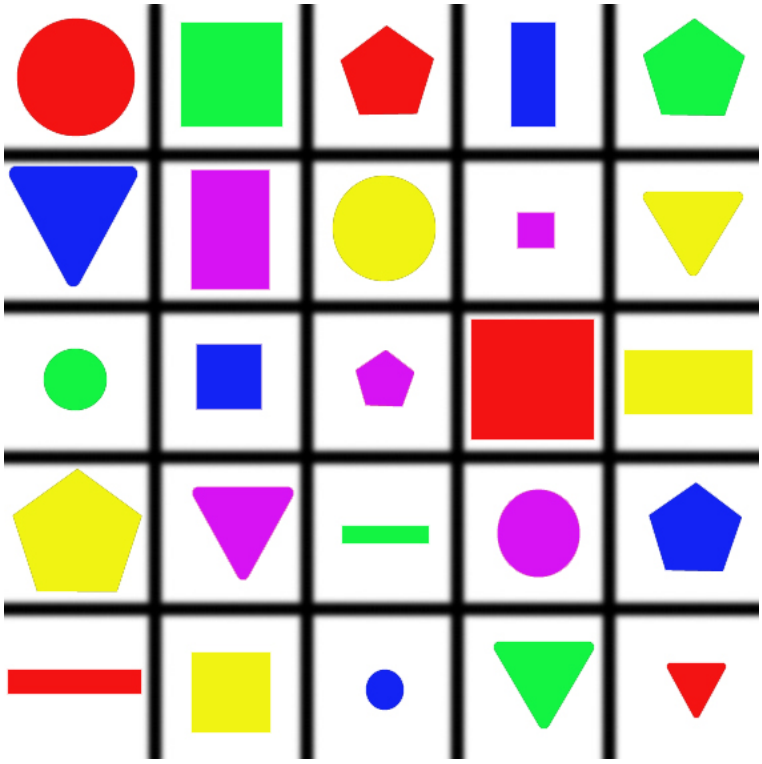
El juego enriquece la creatividad, la imaginación y ayuda a utilizar energía física y mental de maneras productivas y entretenidas.

Los juegos divertidos son muy útiles ya que los niños tienden a recordar las lecciones aprendidas cuando se están divirtiendo.

A continuación se dan algunas ideas de cómo usar los diferentes tableros de BeeBot para fomentar las habilidades de los niños.

- Formas y colores

Tablero para explorar las formas. Sirve para conocer conceptos matemáticos además de habilidades de reconocimiento de formas y colores.



- Describir las diferencias de formas, colores y tamaños.

- ¿Pueden los niños describir las similitudes y diferencias de las formas?

- Elige una forma determinada y da pistas sobre la forma elegida ¿Pueden los niños adivinar cual es la forma seleccionada?

- ¿Cuántos triángulos pueden ver?

- ¿Cuántos colores pueden reconocer?

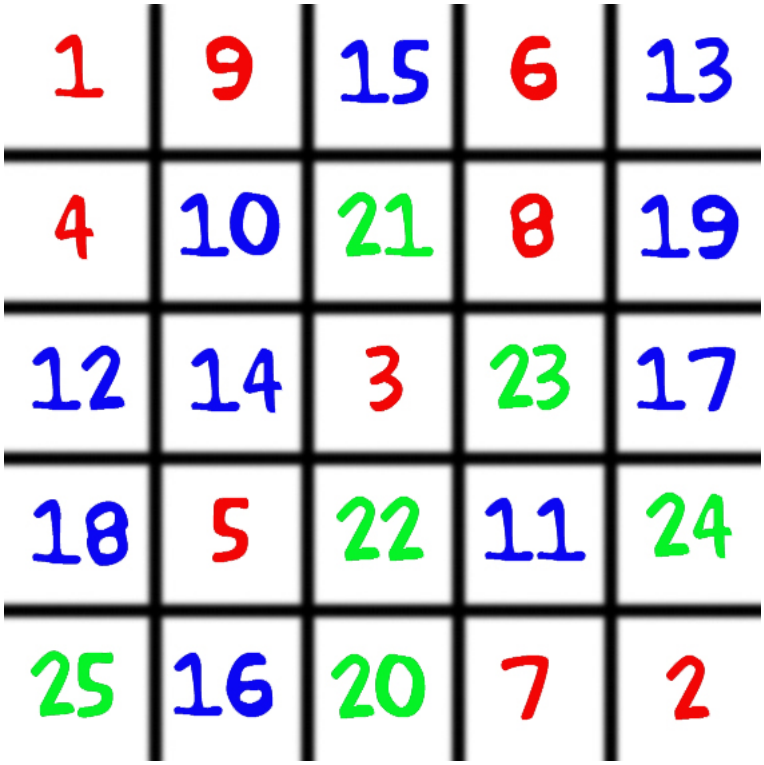
- Elegir una forma determinada. Observar para cada niño cual ha sido el camino elegido. Contar cuantos pasos han sido necesarios.

- Recorrer todos rectángulos del tablero.

- Programar el BeeBot para que vaya desde el pentágono morado, al pentágono amarillo.

- Números

Tablero para trabajar con números. Fomenta el desarrollo de las habilidades matemáticas.




- Proponer sencillos problemas matemáticos y programar el BeeBot para que vaya a la solución.

- Programar el robot para resolver cuentas, sumas, restas, multiplicaciones y divisiones.

- Recorrer todos los números del 1 al 25.

- Seleccionar los múltiplos de 3.

- Recorrer todos los números de color azul de forma creciente.

- Animales

Tablero para aprender sobre los animales. Fomenta las capacidades cognitivas del estudiante.

- Reconocer todos los animales del tablero y aprender los nombres.

- Programar el BeeBot para que recorra los animales que puedan volar.

- Elegir los animales mamíferos del tablero.

- Seleccionar los animales que vivan en el agua.

- Recorrer todos los animales carnívoros.

- Recorrer todos los animales herbívoros.

- Programar el robot para que recorra los animales omnívoros.

- Elegir el camino para seleccionar tu mascota favorita.

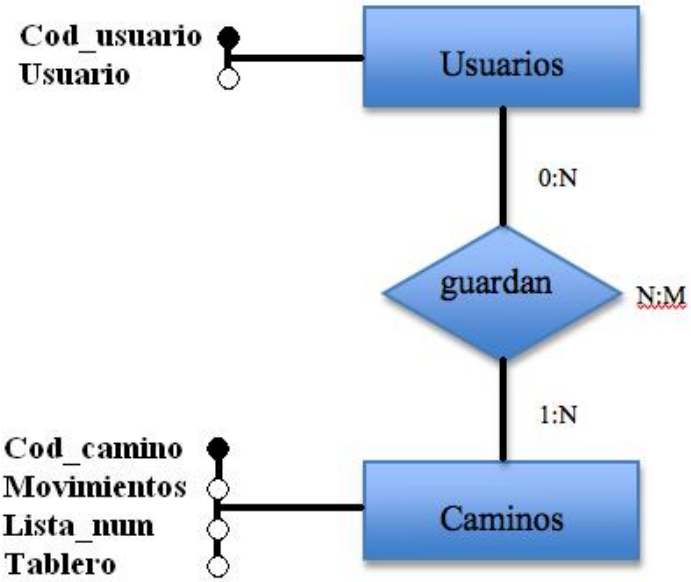
## 4.7 BASE DE DATOS

Para la realización de nuestra aplicación multiusuario necesitamos crear una base de datos en la que poder almacenar los usuarios que vayan a utilizar el BeeBot. Además para cada usuario quedarán guardados los caminos que haya ido programando.

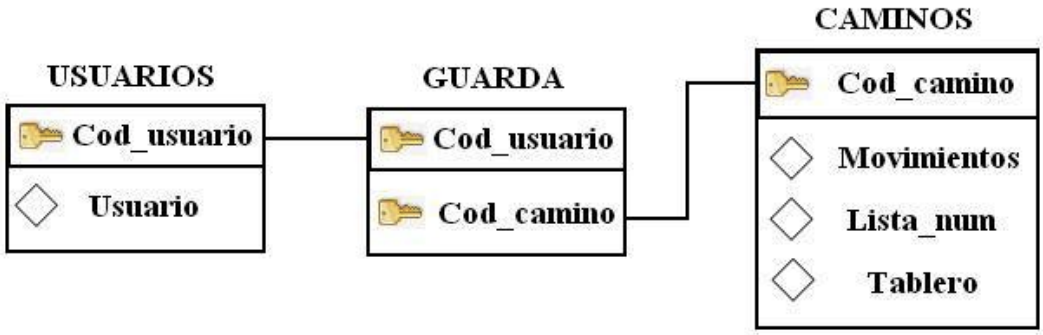
Esta base de datos será dinámica ya que el contenido irá cambiando constantemente según los usuarios que usen la aplicación y los caminos que se deseen guardar.

La base de datos es muy sencilla ya que no requiere mucha complejidad, consta de tres tablas relacionadas entre ellas.

- El esquema entidad/relación es el siguiente:




- Tablas de la base de datos:



- Esquema relacional detallado:

USUARIOS

Cod_usuario "Código del usuario, identifica al usuario de manera única"

Usuario "Nombre del usuario"

GUARDA

Cod_usuario "Código del usuario, identifica al usuario de manera única"

Cod_camino "Código del camino, identifica al camino de manera única"

CAMINOS

Cod_camino "Código del camino, identifica al camino de manera única"

Movimientos "Lista de movimientos en forma de nombres. ([Este, Gira_Izq ,Norte…])"

Lista_num "Lista de movimientos en forma de números. ([2,5,3…])"

Tablero "Número de tablero"

- Paso a tablas:

**Usuarios** (Cod_usuario, usuario)

**Guarda** (Cod_usuario, cod_camino)

**Caminos** (Cod_camino, movimientos, lista_num, tablero)

### 4.7.1 ARCA DATABASE XTRA

Como he comentado anteriormente un "xtra" sirve para añadir nuevas funcionalidades a Director, ya que éste no ofrece ninguna herramienta para la creación y mantenimiento de una base de datos. Por ello añadiremos un xtra llamado Arca Database Xtra [8], el cual nos ofrece todo lo necesario para crear y utilizar archivos de base de datos. Es una solución de base de datos diseñada específicamente para desarrolladores de Director. Es rápido, potente y muy sencillo de aprender.

Este "Xtra" acepta consultas SQL para insertar, actualizar y recuperar datos directamente en el proyecto. Y la mayoría de casos también es posible la reutilización de las consultas SQL desarrollado para otros motores de base de datos, especialmente MySQL, PostgreSQL, SQLite, Oracle y MS SQL Server.

Es a la vez un cliente de base de datos y la base de datos de servidor. El proyecto se puede ejecutar desde un CD-ROM sin instalaciones necesarias en las máquinas cliente.

Arca se construye encima de una versión personalizada de SQLite. SQLite está presente en varios proyectos de software y herramientas, incluyendo PHP5 y Firefox. En la mayoría de los casos base de datos de archivos de Arca puede ser abierta y utilizada por otras herramientas que también utilizan el motor de SQLite, para facilitar el intercambio de datos.

Arca Database Xtra viene con una aplicación complementaria para crear, revisar y modificar bases de datos, el Arca Database Browser. Es una aplicación libre, creada en C++ por los desarrolladores Tabuleiro para ayudar a la creación de bases de datos y edición de Arca de los datos mediante una interfaz gráfica de usuario y una interfaz familiar de hoja de cálculo.

Algunas de las características del navegador de Arca de base de datos son:

- Registros de la búsqueda, navegar y editar usando una interfaz de hoja de cálculo.

- Importar y exportar el texto a partir de registros.

- Visualización, importación y exportación JPEG, PNG y BMP directamente a los registros. Importado imágenes JPEG se guardan con sus datos intactos JPEG sin recompresión y PNG y BMP se convierten directamente a los objetos de imagen Lingo, listo para ser utilizado en las rutinas de imágenes Lingo.

- Crear, borrar y modificar tablas.

- Crear, borrar y modificar los campos.

- Crear y eliminar índices.

- Importación y Exportación de tablas a archivos de texto. Estos archivos pueden ser creados o utilizados por ejemplo en MS Excel o MS Access para el intercambio de datos.

- Importación y exportación de bases de datos a SQL, archivos de volcado. Estos archivos se pueden cargar y crear, por ejemplo, para facilitar el intercambio de datos con motor de base de datos, o para cargar copias de seguridad y archivos de bases de datos.

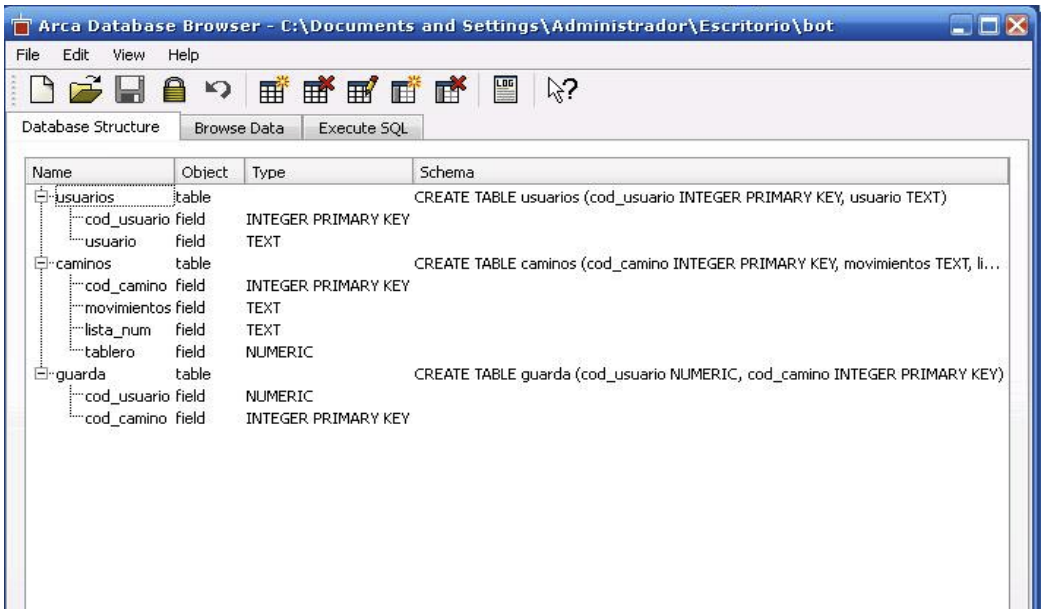
- Ejecuta las consultas SQL directamente a través de una interfaz visual. Esta es una forma rápida de poner a prueba tus consultas SQL y mejorar su conocimiento de la sintaxis SQL. Los códigos de error y los resultados pueden ser evaluados de inmediato en un diseño de columnas conveniente.

- Capacidad para guardar los cambios en la base de datos o volver a la versión guardada anteriormente. Esto es útil para experimentar con las consultas SQL, sin riesgo de pérdida de datos.

Arca DataBase Browser permite probar y desarrollar sus bases de datos más fácil y con mayor rapidez, y puede ser redistribuido de forma gratuita a los clientes.

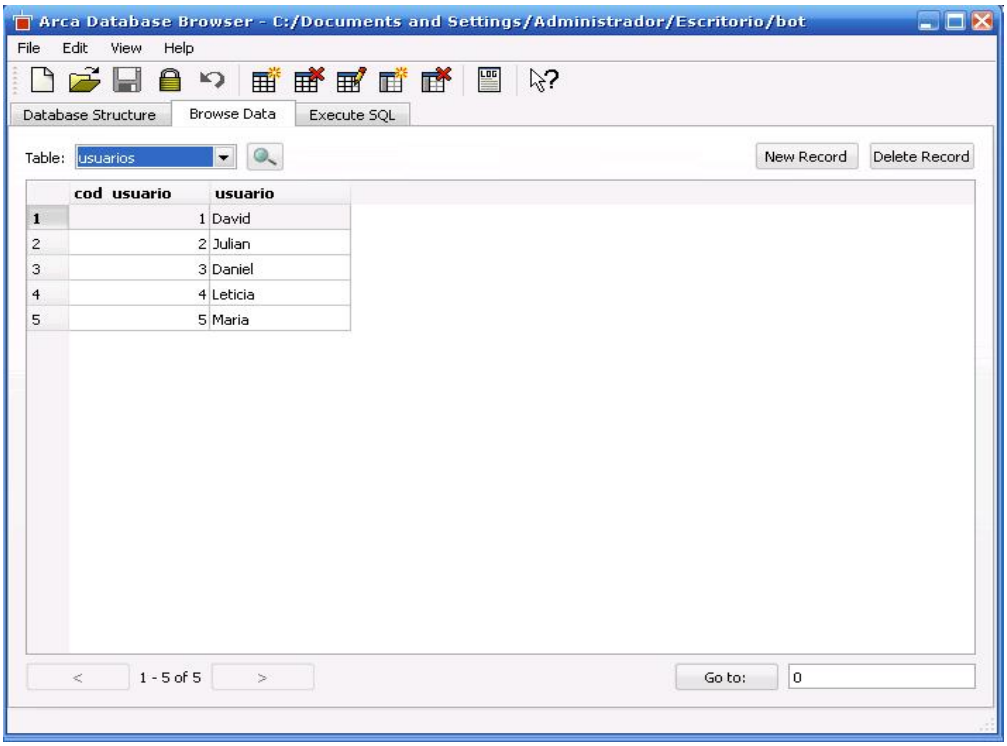
## 4.7.2 ARCA DATABASE BROWSER

Para modelar nuestra base de datos, y hacerla compatible con Arca Database Xtra, utilizamos la herramienta Arca Database Browser y creamos las 3 tablas necesarias para nuestra base de datos.

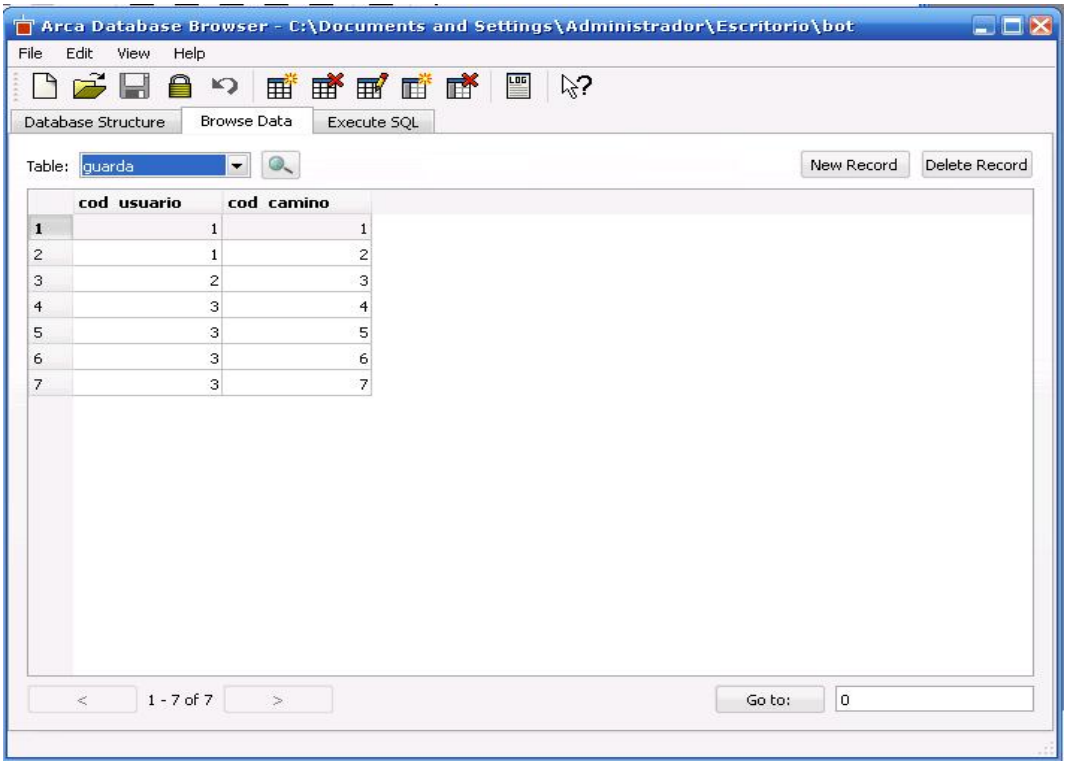



La aplicación está programada para que desde ella se pueda poblar y mantener la base de datos. A continuación se muestra una instancia de cada una de las tablas pobladas desde BeeBot:

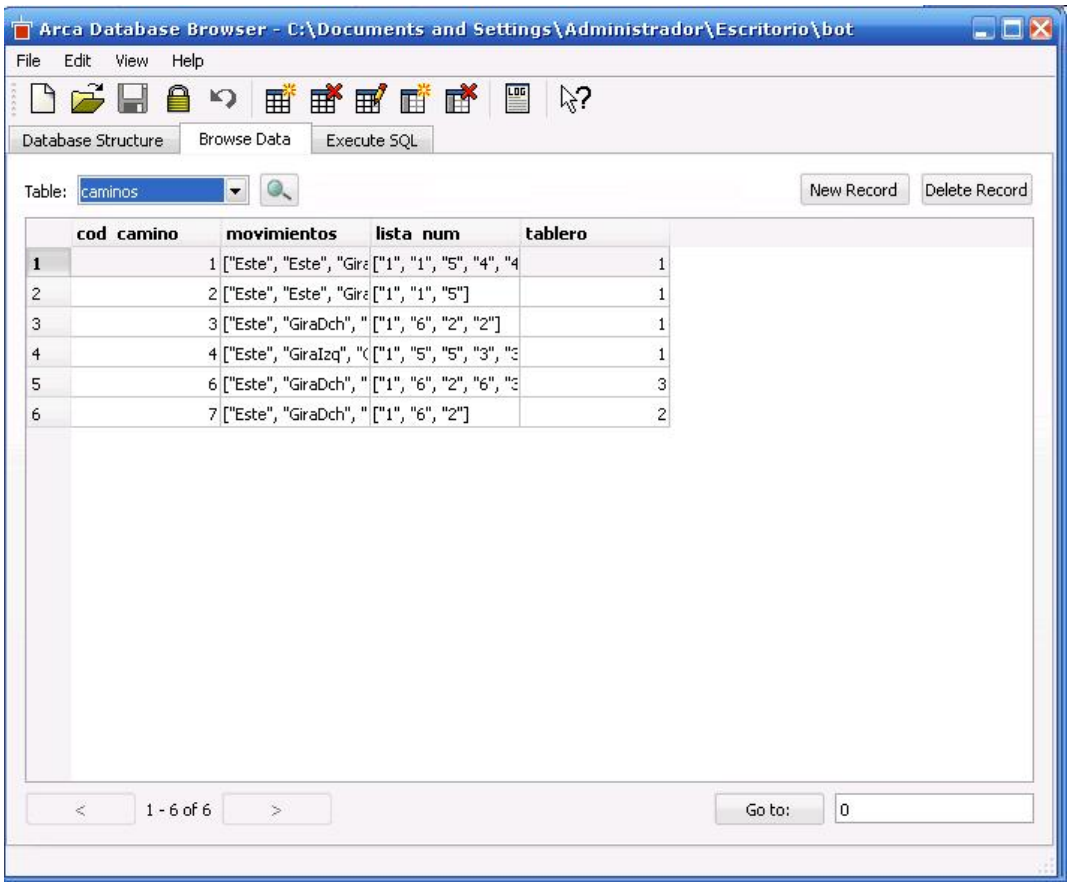
- Tabla usuarios



- Tabla guarda




- Tabla caminos



# 5. CONCLUSIONES

Como conclusión, se ha realizado una aplicación multimedia que simula la programación del robot BeeBot. Con este simulador, los niños aprenden a programar desde pequeños y coger hábitos que en un futuro les será útil. En definitiva, este podría ser su primer lenguaje de programación cuando todavía no saben leer ni escribir.

Para aplicar este tipo de disciplinas educativas que BeeBot nos ofrece, debemos comprar el robot. La compra del robot con su software nos supondría un gasto de unos 65€. Y los tableros BeeBot Mats varían entre 2.5€ y 25€. Además el software nos costaría 35€ por licencia. En total un pack básico puede costar unos 135€, dependiendo de el material que adquiramos.

La idea de este proyecto consiste en la realización de un software de distribución libre, utilizando la herramienta Macromedia Director. Además es el inicio para unir a otros proyectos como son la creación del propio robot, con robots educativos como Lego o Arduino.

Con la creación de este software libre, y la futura creación del propio robot obtendremos una forma más económica de aprovechar las capacidades de aprendizaje que este tipo de robots nos aportan para la educación.

# 6. LINEAS FUTURAS

Para la ampliación de este proyecto hay varias opciones o facetas que resultan interesantes.

- Podrían crearse nuevos tableros con otras disciplinas de aprendizaje para fomentar nuevas habilidades de los niños.

- El tablero del BeeBot es de un tamaño determinado, 5x5. Cuando creemos los diferentes tipos de juegos no tienen porque tener todos el mismo tamaño, cada juego podría tener diferentes tamaños de tablero. Es decir, debería crearse una forma dinámica de crear los tableros, para que el BeeBot pueda moverse correctamente por esos tableros y además no se salga de los límites de cada uno de ellos.

- El robot siempre sale desde la posición central del tablero, celda (3,3). Podría programarse para que el BeeBot salga desde la posición que el usuario desee.

- Una vez creado el software de BeeBot, sería interesante la creación del robot con herramientas como Arduino o Lego, para así tener todo el "kit" de aprendizaje. Por un lado el simulador del robot y por otro el robot físico.

- Aunque no sea muy necesario quedaría mucho más visual, desde el punto de vista del diseño, si se añadiese la posibilidad de poder ver el BeeBot en 3D.

- A la hora de poder utilizar la aplicación en distintas plataformas da errores, debido a que Arca Database Xtra está solo para el sistema operativo Windows. Para solucionar este problema y dar soporte a todas las plataformas, se podría crear el proyecto en http para colgarlo en la red. Además sería necesario subir también la base de datos a un servidor, para dar independencia física.
  Otra forma de solucionar este problema sería utilizar otro "xtra" que soporte todos los sistemas operativos.

- Para finalizar con el proyecto se podría haber hecho unas pruebas a los niños y profesores con la herramienta, para ver los posibles ventajas e inconvenientes de la aplicación.

# 7. BIBLIOGRAFÍA


- [1] USING MACROMEDIA DIRECTOR MX. Gary Rosenzweig. Que publishing 2003.
- [2] DIRECTOR MX Y LINGO. Phil Gross. Anaya 2003
- [3] http://www.director-online.com/
- [4] http://www.vtc.com/products/Macromedia-Director-MX-Espanol-tutorials.htm
- [5] http://www.videoaprendizaje.net/2008/09/videotutoriales-de-director-mx-2004.html
- [6] http://www.infor.uva.es/~descuder/docencia/multim/director/director.pdf
- [7] http://www.desarrollomultimedia.es/cd-dvd-interactivo/manual-director.html
- [8] http://xtras.tabuleiro.com/products/arca/index.tdb
- [9] http://www.tts-group.co.uk/Bee-Bot
- [10] http://scratch.mit.edu/
- [11] http://www.comikit.se/

# ANEXO

En este apartado se incluirán todos los scripts utilizados para la programación del robot educativo.

- Script Funciones: En este script se encuentran las funciones que afectan al movimiento del BeeBot.

  La función Este, Norte, Oeste y Sur cambia la posición horizontal locH y la posición vertical LocV 114 pixeles, lo que hace que el BeeBot se mueva en la dirección deseada. La función wait se utiliza dentro de un bucle for para que el robot no salte de una posición a la otra, sino que vaya progresivamente. La variable punto contiene la posición del robot en la matriz, esta variable se actualiza en función del movimiento que se realice.

```
on Este
  global locH, punto

      --locH: posición horizontal del robot
      --punto: punto del tablero donde se encuentra el robot

  punto.setAt(2, punto.getAt(2)+1)
  repeat with i=1 to 114
    sprite(8).locH = sprite(8).locH + 1
    wait()
    updateStage
  end repeat
end

on Norte
  global  locV, punto

      --locV: posición vertical del robot
      --punto: punto del tablero donde se encuentra el robot

  punto.setAt(1, punto.getAt(1)+1)
  repeat with i=1 to 114
    sprite(8).locV = sprite(8).locV - 1
    wait()
    updateStage
  end repeat
end
```

```
on Oeste
  global locH, punto
      --locH: posición horizontal del robot
      --punto: punto del tablero donde se encuentra el robot
  punto.setAt(2, punto.getAt(2)-1)
  repeat with i=1 to 114
    sprite(8).locH = sprite(8).locH - 1
    wait()
    updateStage
  end repeat
end

on Sur
  global locV, punto
      --locV: posición vertical del robot
      --punto: punto del tablero donde se encuentra el robot
  punto.setAt(1, punto.getAt(1)-1)
  repeat with i=1 to 114
    sprite(8).locV = sprite(8).locV + 1
    wait()
    updateStage
  end repeat
end
```

La función GiraIzq y GiraDch sirven para hacer que el robot gire hacia la izquierda y derecha respectivamente. Al igual que para moverse en cualquier dirección, necesitamos un bucle "for" con la función wait para que el robot no gire de golpe, sino que lo haga poco a poco. La posición es una variable global en donde se guarda un número del 1 al 4 indicando la dirección a la que apunta el BeeBot. El valor de cada número está explicado en el apartado 4.4 Dirección y posicionamiento.

```
on GiraIzq
  global posición
       --posicion: posición hacia donde apunta el BeeBot

  posicion = posicion-1
  if posicion<1 then
    posicion = 4
  end if
  valor=90
  repeat with i=1 to valor
    sprite(8).rotation= sprite(8).rotation - 1
    wait()
    updateStage
  end repeat
end
```

```
on GiraDch
  global posicion
  --posicion: posición hacia donde apunta el BeeBot

  posicion = posicion +1
  if posicion>4 then
    posicion = 1
  end if

  valor=90
  repeat with i=1 to valor
    sprite(8).rotation= sprite(8).rotation + 1
    wait()
    updateStage
  end repeat
end
```

La función wait ha sido programada para añadir un retardo a la hora de moverse el BeeBot por el tablero. Para ello ejecuta una operación matemática.

```
on wait
  repeat with k=1 to 15000
    m = k*(15000/144)+32
  end repeat
end
```

La función comprobar evalúa si el robot puede dar el siguiente paso, para ello comprueba la dirección que tiene el BeeBot y la posición del tablero donde se encuentra. Si se puede realizar el siguiente movimiento la función devuelve un 1 y si no un 0.

```
on comprobar
  global boolean, punto, camino_numeros, num

--boolean: variable que indica si se puede realizar el movimiento o no
--punto: punto donde se encuentra el robot en el tablero
--camino_numeros:
--num: cuenta el numero de veces que se ha movido el coche, para ver donde se ha
quedado

  put "en comprobar"
  put punto
  put camino_numeros
  put "fin comprobar"
  if getAt(value(punto),2)=5 and getAt(value(camino_numeros),num+1)=1 then
    boolean = 0
  else if getAt(value(punto),2)=1 and getAt(value(camino_numeros),num+1)=3 then
    boolean = 0
  else if getAt(value(punto),1)=1 and getAt(value(camino_numeros),num+1)=2 then
```

```
    boolean = 0
  else if getAt(value(punto),1)=5 and getAt(value(camino_numeros),num+1)=4 then
    boolean = 0
  else
    boolean = 1
  end if
end
```

- Script Registrar usuario

  Este script está programado para el botón que hace que se registre un usuario. Guarda el nombre de usuario en la base de datos. Han sido depurados errores como la introducción de un nombre vacío, o de un usuario que ya existe en la base de datos.

```
on mouseDown
  global gDB, user

--gDB: instancia de Arca Database
--user: nombre del usuario

  puppetSound 2, "click"
  user = Field "usuario"
  if user = "" then
    Alert "Introduce un nombre de usuario"
    go to frame "Registro usuario"
  else


    myresult = gDB.executeSQL("SELECT * FROM Usuarios WHERE
Usuario='"&user&"'")
    x= count(myResult.rows)
    if x=0 then
      gDB.executeSQL("INSERT INTO usuarios VALUES(NULL, '"&user&"')")
      Alert "El usuario ha sido creado"
      go to frame "Robot"
    else
      Alert "El usuario ya existe. Carga el usuario en el menu principal"
      go to frame "Inicio"
    end if
  end if
end
```

- Script crear instancia de arca

Para poder conectarte a la base de datos y poder hacer consultas y modificaciones sobre la misma, debemos crear una instancia del Arca Database Xtra.

```
on exitFrame me
  global gDB, tablero

--gDB: instancia de Arca Database
--tablero: variable para cargar uno de los tableros

  _global.clearGlobals()
  tablero = 1
  gDB = new (xtra "arca")
  gDB.openDB(the moviepath & "bot")

  if not objectp(gDB) then
    alert ("No se ha podido crear la instancia de arca")
    return
  else
    put ("Instancia arca creada")
  end if

end
```

- Script mostrar usuarios

Este script sirve para mostrar en un field los usuarios que ya hay registrados en la base de datos.

```
on exitFrame me
  global gDB

  --gDB: instancia de Arca Database

  put "" into Field "field_resul"
  put "" into Field "field_cod"

  myResult = gDB.executeSQL("SELECT * FROM usuarios")
  put myResult
  repeat with i=1 to count (myresult.rows)
    thisRecord = myresult.rows[i]
    user = thisRecord[2]
    cod_user = thisRecord[1]
    put cod_user && " - " && user && return after field "field_resul"
  end repeat
end
```

- Script botón cargar usuario

  Una vez mostrados todos los usuarios que existen, elegimos uno de la lista. Si el código del usuario introducido no existe o dejamos el input en blanco, la aplicación nos mostrará un error.

```
on mouseUp me
  global gDB
  global user
  global cod_user

--gDB: instancia de Arca Database
--user: nombre del usuario
--cod_user: codigo del usuario

  puppetSound 2, "click"
  cod_user = Field "field_cod"
  if cod_user="" then
    Alert "Introduce un numero de usuario"
  else

    myRes = gDB.executeSQL("SELECT * FROM usuarios WHERE
cod_usuario='"&cod_user&"'")
    x= count(myRes.rows)
    if x=0 then
      Alert "El usuario no existe. Escribe un numero de usuario valido de la lista de
arriba."
      go to frame "Seleccion usuario"
    else
      thisRecord = myRes.rows[1]
      user= thisRecord[2]
      go to frame "Seleccion camino"
    end if
  end if
end
```

- Script mostrar caminos de cada usuario

  Este script sirve para mostrar en un field los caminos registrados en la base de datos del usuario seleccionado.

```
on exitFrame me
  global cod_user
  global user
  global gDB
  global cod_camino
  global camino_nombres
  global camino_numeros
  global carga
```

```
  global tablero

--gDB: instancia de Arca Database
--user: nombre del usuario
--cod_user: codigo del usuario
--cod_camino: codigo del camino
--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
-- carga=1 me dice si hay un camino cargado o no
--tablero: variable para cargar uno de los tableros

  carga=0

  put "" into Field "Field_resul"
  put "" into Field "Field_cod"
  put user into Field "titulo"

  myRes = gDB.executeSQL("SELECT * FROM guarda WHERE
Cod_usuario="&cod_user)

  if count (myres.rows)=0 then
    Alert "No hay ningun juego guardado para este usuario. Guarde en la proxima
partida."
    go to frame "robot"
  else


    repeat with i=1 to count (myres.rows)
      thisRecord = myres.rows[i]
      cod_camino=thisRecord[2]
      put cod_camino

      myResult = gDB.executeSQL("SELECT * FROM caminos WHERE
cod_camino="&cod_camino)
      repeat with j=1 to count (myresult.rows)
        thisRecord = myresult.rows[j]
        camino_nombres = thisRecord[2]
        camino_numeros = thisRecord[3]
        cod_camino = thisRecord[1]
        tablero = thisRecord[4]
        put cod_camino && " - " && camino_nombres && " - Tablero "&& tablero
&& return after field "field_resul"
        put "                " && return after field "field_resul"
      end repeat
    end repeat
  end if
end
```

- Script cargar camino

Una vez mostrados todos los caminos que existen, elegimos uno de la lista. Éste camino quedará guardado en las variables correspondientes para cargarlo posteriormente y poder simular dicho camino. Si el código del camino introducido no existe o dejamos el input en blanco, la aplicación nos mostrará un error.

```
on mouseDown
  global camino_nombres, camino_numeros, gDB, carga, cod_user, tablero,
cod_camino
 --gDB: instancia de Arca Database
--cod_user: codigo del usuario
--cod_camino: codigo del camino
--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
-- carga=1 me dice si hay un camino cargado o no
--tablero: variable para cargar uno de los tableros

  puppetSound 2, "click"
  carga = 1
  cod_camino = Field "field_cod"
  if cod_camino="" then
    Alert "Introduce un numero de camino"
  else
    put "usuario :" && cod_user
    put "cod_camino :" && cod_camino

    myRes = gDB.executeSQL("SELECT * FROM guarda WHERE
cod_camino='"&cod_camino&"' AND cod_usuario ='"&cod_user&"'")

    if count(myRes.rows)=0 then
      Alert "El camino no existe. Escribe un numero de camino valido de la lista de
arriba."
      go to frame "Seleccion camino"
    else
      myResult = gDB.executeSQL("SELECT * FROM caminos WHERE
cod_camino='"&cod_camino&"'")
      thisRecord = myresult.rows[1]
      camino_nombres = value(thisRecord[2])
      camino_numeros = value(thisRecord[3])
      tablero = value(thisRecord[4])
      put "tablero" tablero
      go to frame "Robot"
    end if
  end if

end if

end
```

- Botón borrar usuario

  Este botón se encarga de elegir el código del usuario para borrar. Si el input está vacío o si el usuario no existe saldrá un error.

```
on mouseUp me
  global gDB, cod_user
  --gDB: instancia de Arca Database
--cod_user: codigo del usuario

  cod_user = Field "field_cod"

  if cod_user="" then
    Alert "Introduce un numero de usuario"
  else
    myRes = gDB.executeSQL("SELECT * FROM usuarios WHERE
cod_usuario='"&cod_user&"'")
    x= count(myRes.rows)
    if x=0 then
      Alert "El usuario no existe. Escribe un numero de usuario valido de la lista de
arriba."
      go to frame "Seleccion usuario"
    else
      thisRecord = myRes.rows[1]
      user= thisRecord[2]
      put user
      go to frame "Borrar usuario"
    end if
  end if
end
```

- Script borrar usuario

  Este script es el encargado de borrar de la base de datos el usuario seleccionado. Además de la tabla usuarios, también hay que borrarlo de la tabla guarda para que no queden datos inútiles en nuestro almacen.

```
on mouseUp me
  global cod_user, gDB

--gDB: instancia de Arca Database
--cod_user: codigo del usuario

  myResult = gDB.executeSQL("SELECT * FROM guarda WHERE
cod_usuario="&cod_user)
  repeat with i=1 to count (myresult.rows)
    thisRecord = myresult.rows[i]
    cod_camino=thisRecord[2]
```

```
    gDB.executeSQL("DELETE FROM caminos WHERE
cod_camino=?",[cod_camino])
  end repeat

  gDB.executeSQL("DELETE FROM usuarios WHERE cod_usuario=?",[cod_user])
  gDB.executeSQL("DELETE FROM guarda WHERE cod_usuario=?",[cod_user])


  Alert "Usuario borrado"
  go to frame "Seleccion usuario"

end
```

- Script botón gira dch

Con este botón vamos modificando la dirección a la que apunta el BeeBot. Si en ese momento el BeeBot tiene dirección (1 – Este), al pulsar este botón la dirección cambiara a (2 - Sur) y quedará guardado en las variables correspondientes, para posteriormente reproducir el camino.

```
on mouseDown
  puppetSound 2, "click"
  global posicion, camino_nombres, camino_numeros

--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
--posicion: posición hacia donde apunta el BeeBot

  posicion = posicion + 1
  if posicion > 4 then
    posicion = 1
  end if
  add(camino_nombres,"GiraDch")
  add(camino_numeros,"6")
  put value(camino_nombres) into field "lista"
end
```

- Script botón gira izq

Con este botón vamos modificando la dirección a la que apunta el BeeBot. Si en ese momento el BeeBot tiene dirección (1 – Este), al pulsar este botón la dirección cambiara a (4 - Norte) y quedará guardado en las variables correspondientes, para posteriormente reproducir el camino.

```
on mouseDown
  global posicion, camino_nombres, camino_numeros

  --camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
```

```
--posicion: posición hacia donde apunta el BeeBot

 puppetSound 2, "click"

 posicion = posicion - 1
 if posicion < 1 then
  posicion = 4
 end if
 add(camino_nombres,"GiraIzq")
 add(camino_numeros,"5")
 put value(camino_nombres) into field "lista"
end
```

- Script botón UP

  Este botón añade a la lista de movimientos hacia donde tiene que avanzar el robot, según la dirección a la que apunte.

```
on mouseDown

 global posicion, camino_nombres, camino_numeros
 puppetSound 2, "click"

 --posicion: direccion del coche
 --camino_nombres: lista de movimientos a realizar escritas en forma de nombre
 --camino_numeros: lista de movimientos a realizar escritas en forma de numeros

 --1 Este
 --2 Sur
 --3 Oeste
 --4 Norte
 --5 Gira Izq
 --6 Gira dch


 if posicion = 1 then
  add(camino_nombres,"Este")
  add(camino_numeros,"1")
  put value(camino_nombres) into field "lista"

 else if posicion = 4 then
  add(camino_nombres,"Norte")
  add(camino_numeros,"4")
  put value(camino_nombres) into field "lista"


 else if posicion = 3 then
  add(camino_nombres,"Oeste")
  add(camino_numeros,"3")
  put value(camino_nombres) into field "lista"
```

```
  else if posicion = 2 then
    add(camino_nombres,"Sur")
    add(camino_numeros,"2")
    put value(camino_nombres) into field "lista"

  end if
end
```

- Script boton GO

  Es el botón encargado de empezar con la simulación del camino guardado. El Bebió no se puede salir del tablero, para ello la función comprobar() se encarga de que esto no suceda devolviendo un número 1 si es valido el movimiento, o 0 si no es válido. En el caso de que no sea válido se borrarán todos los movimientos siguientes y se dejarán solamente los que son válidos.

```
on mouseDown
  global camino_nombres, camino_numeros, num, pl, posicion, boolean, punto

--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
--posicion: posición hacia donde apunta el BeeBot
--pl: flag para ver si es la segunda vez que se pulsa play, para continuar donde
estaba el coche
--num: cuenta el numero de veces que se ha movido el coche, para ver donde se ha
quedado
--boolean: si esta a 1 el movimiento esta permitido si es 0 no
--punto: lugar donde se encuentra el BeeBot en el tablero
  if pl = 0 then
    repeat with x=1 to count(value(camino_numeros))
      dato = getAt(value(camino_numeros), x)
      comprobar()
      if boolean=1 then
        if dato = "4" then
          Norte()
          posicion = 4


        else if dato = "2" then
          Sur()
          posicion = 2


        else if dato = "1" then
          Este()
          posicion = 1
```

```
    else if dato = "3" then
      Oeste()
      posicion = 3


    else if dato = "5" then
      GiraIzq()


    else if dato = "6" then
      GiraDch()


    end if
    num=num+1

  else

    x=count(value(camino_numeros))

    repeat while num <> x
      deleteAt(value(camino_nombres), x)
      deleteAt(value(camino_numeros), x)
      x=x-1
      posicion= getAt(value(camino_numeros),num-1)

    end repeat

    put camino_nombres
    put camino_numeros
    put posicion

    put value(camino_nombres) into field "lista"

  end if

 end repeat
 pl = pl+1


##########################################################################
##########################################################################

 else if pl>0 then

  repeat with x=num+1 to count(value(camino_numeros))
    dato = getAt(value(camino_numeros), x)
    comprobar()
    if boolean=1 then
```

```
      if dato = "4" then
        Norte()
        posicion = 4


      else if dato = "2" then
        Sur()
        posicion = 2


      else if dato = "1" then
        Este()
        posicion = 1


      else if dato = "3" then
        Oeste()
        posicion = 3

      else if dato = "5" then
        GiraIzq()


      else if dato = "6" then
        GiraDch()


      end if
      num=num+1

    else

      x=count(value(camino_numeros))

      repeat while num <> x
        deleteAt(value(camino_nombres), x)
        deleteAt(value(camino_numeros), x)
        x=x-1
        posicion= getAt(value(camino_numeros),num-1)

      end repeat

      put value(camino_nombres) into field "lista"
    end if
  end repeat
 end if

 puppetSound 2, "click"
end
```

```
on mouseEnter
  sprite(5).member = "go_sel"
end

on mouseLeave
  sprite(5).member= "go"
end
```

- Script para borrar todas las variables globales

```
on exitFrame me
  _global.clearGlobals()
end
```

- Script para inicializar el juego

Este script se encarga de la inicialización del juego. Hay dos opciones que diferenciar. La primera es cuando el usuario decide programar un juego nuevo, en ese instante la variable carga estará a 0, y las variables del robot se inicializarán para crear un camino nuevo. La segunda opción se produce al cargar un camino ya guardado por un usuario, la variable carga en ese momento estará a 1 y se cargarán los valores del camino cargado para comenzar con su simulación.

```
on exitFrame me
  global camino_numeros, camino_nombres, posicion, punto, user, gDB, cod_user,
carga, tablero, pl, num

--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
--posicion: posición hacia donde apunta el BeeBot
--pl: flag para ver si es la segunda vez que se pulsa play, para continuar donde
estaba el coche
--gDB: instancia de Arca Database
--num: cuenta el numero de veces que se ha movido el coche, para ver donde se ha
quedado
--boolean: si esta a 1 el movimiento esta permitido si es 0 no
--punto: lugar donde se encuentra el BeeBot en el tablero
--user: nombre del usuario
--cod_user: codigo del usuario
--carga: variable para ver si hay que cargar el camino o iniciar uno nuevo
--tablero: variable que identifica que tablero hay que cargar

  if carga = 0 then
    put " " into field "lista"
    num = 0
    pl = 0
    posicion = 1
    camino_nombres=[]
    camino_numeros=[]
```

```
    punto=[3,3]
    myRes = gDB.executeSQL("SELECT * FROM usuarios WHERE
usuario='"&user&"'")
    thisRecord = myRes.rows[1]
    user= thisRecord[2]
    cod_user = thisRecord[1]
    put "Usuario" && user
    put "Cod_usuario" && cod_user
   else
    put camino_nombres into field "lista"
    put camino_numeros
    punto=[3,3]
    posicion = 1
    pl = 0
    num = 0
   end if

   if tablero =1 then
    sprite(7).member = "tablero1"
    updateStage
   else if tablero =2 then
    sprite(7).member = "tablero2"
    updateStage
   else if tablero =3 then
    sprite(7).member = "tablero3"
    updateStage
   end if

  end
```

- Script guardar camino

  Este script nos permite guardar el camino deseado en la base de datos, para poder cargarlo posteriormente.

```
on mouseDown
 global gDB, camino_nombres, camino_numeros, cod_user, user, tablero

--camino_nombres: lista de movimientos a realizar escritas en forma de nombre
--camino_numeros: lista de movimientos a realizar escritas en forma de numeros
--gDB: instancia de Arca Database
--user: nombre del usuario
--cod_user: codigo del usuario
--tablero: variable que identifica que tablero hay que cargar


 myresult = gDB.executeSQL("SELECT * FROM Usuarios WHERE
Usuario='"&user&"'")
```

```
  gDB.executeSQL("INSERT INTO caminos
VALUES(NULL,'"&camino_nombres&"','"&camino_numeros&"','"&tablero&"')")

  myresult = gDB.executeSQL("SELECT * FROM Caminos WHERE
Usuario='"&user&"'")

  gDB.executeSQL("INSERT INTO guarda VALUES('"&cod_user&"',NULL)")
  Alert ("Juego guardado")

  puppetSound 2, "click"

end
```

- Botón borrar camino

  Este script se encarga de elegir el código del camino para borrar. Si el input está vacío o si el usuario no existe saldrá un error.

```
on mouseUp me
  global gDB, cod_user, cod_camino
--gDB: instancia de Arca Database
--cod_camino: codigo del camino
--cod_user: codigo del usuario
  cod_camino = Field "field_cod"
  myRes = gDB.executeSQL("SELECT * FROM caminos WHERE
cod_camino='"&cod_camino&"'")
  if cod_camino = "" then
    Alert "Introduce un numero de camino valido"
  else
    if count(myRes.rows)=0 then
      Alert "El camino no existe. Escribe un numero de camino valido de la lista de
arriba."
      go to frame "Seleccion camino"
    else
      go to frame "Borrar camino"
    end if
  end if
end
```

- Script  borrar camino

  Este script es el encargado de borrar de la base de datos el camino seleccionado.

```
on mouseUp me
  global cod_camino, gDB

--gDB: instancia de Arca Database
--cod_camino: codigo del camino
```

```
    gDB.executeSQL("DELETE FROM caminos WHERE
cod_camino=?",[cod_camino])
    go to frame "Seleccion camino"
    Alert "Camino "&&cod_camino&&" borrado"
    go to frame "Seleccion camino"

end
```

- Script botón de carga del tablero 1

```
on mouseUp me
  global tablero, carga

--carga: variable para ver si hay que cargar el camino o iniciar uno nuevo
--tablero: variable que identifica que tablero hay que cargar

  carga = 0
  tablero = 1
  go to frame "robot"
end
```

- Script botón de carga del tablero 2

```
on mouseUp me
  global tablero, carga

--carga: variable para ver si hay que cargar el camino o iniciar uno nuevo
--tablero: variable que identifica que tablero hay que cargar

  carga = 0
  tablero = 2
  go to frame "robot"
end
```

- Script botón de carga del tablero 3

```
on mouseUp me
  global tablero, carga

--carga: variable para ver si hay que cargar el camino o iniciar uno nuevo
--tablero: variable que identifica que tablero hay que cargar

  carga = 0
  tablero = 3
  go to frame "robot"
end
```

# SIMULACIÓN DE UN ROBOT PROGRAMABLE CON MACROMEDIA DIRECTOR

David Cordón Blanco

# INTRODUCCIÓN

- El objetivo de este proyecto es ofrecer una aplicación para el desarrollo del aprendizaje de los niños, mediante la robótica educativa.
- Actualmente vivimos en una época en la que la tecnología está a la orden del día. Por ello, con la realización de esta aplicación, fomentamos el aprendizaje de los más pequeños desde unas edades muy tempranas.

# INTRODUCCIÓN

- La inspiración para la realización de este proyecto ha sido un robot creado para la educación llamado Bee-Bot.
- Además existen los Bee-Bot Mat, que son unos tableros con numerosos ejercicios y actividades que fomentan la resolución de problemas, pensamiento crítico y habilidades para tomar decisiones.

# Bee-Bot

- Bee-Bot es un robot programable que permite dar a los estudiantes instrucciones con diferentes grados de dificultad para desarrollar una amplia gama de áreas.
- Sus principales características son el desarrollo de habilidades de motricidad mediante el uso de los botones de dirección, el juego imaginativo a través del uso de diferentes tableros, y demostrar las habilidades del estudiante de una manera que un enfoque tradicional no podrían demostrarse.

# Objetivos del proyecto

- El objetivo de este proyecto es la creación de un software libre que se asemeje al Bee-Bot y permita desarrollar las mismas habilidades de aprendizaje.
- Esta herramienta consistirá en una aplicación multimedia y multiusuario.
- Además con la realización de este proyecto se pretende crear un punto de inicio para futuros proyectos, como puede ser la modelización y creación de un robot. Mejora de este software añadiéndole nuevas funcionalidades. O incluso pruebas con niños y profesores una vez creado el robot y el software.

# Etapas del proyecto

El PFC está estructurado en 3 etapas fundamentalmente:

- La primera etapa consiste en estudiar las posibles herramientas y el lenguaje de programación a utilizar.
- En la segunda etapa se procedería a la propia programación de la aplicación con Lingo. Lo cual lleva una nuevo estudio del lenguaje de programación para solucionar los diferentes problemas que supone esta etapa.
- En la última etapa, una vez programada la aplicación habría que realizar una interfaz amigable, que sea fácil de usar para un niño y además sea atractiva para ellos.

# Etapa 1: Lenguajes

**COMIKIT**

- ComiKit es un “toolkit” para niños que usa un lenguaje de programación visual basado en tiras de cómics.
- La propuesta de ComiKit es hacer la programación mas fácil para niños convirtiéndola en más visual.

# Etapa 1: Lenguajes

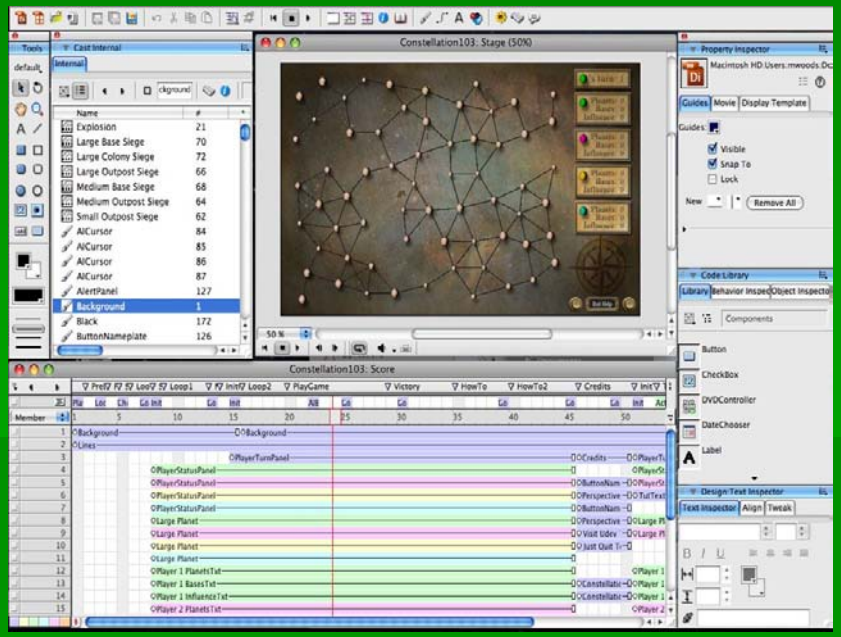
## SCRATCH

- Scratch es una herramienta informática destinada principalmente a los niños que utiliza etiquetas para programar
- El lenguaje y entorno de desarrollo es muy intuitivo y fácil de aprender para los niños que no tengan experiencia previa en programación.

# Etapa 1: Lenguajes

**DIRECTOR**

- Macromedia Director es una aplicación de Desarrollo de Software Multimedia destinado para la producción de programas ejecutables ricos en contenido multimedia.
- Es capaz incorporar imágenes, audio, video digital, películas flash, y un motor 3D.
- El lenguaje de programación que utiliza es Lingo o JavaScript.



# Etapa 2: Programación

- La aplicación está programada como una secuencia de pantallas. Nos movemos por ella a través de diferentes botones de una forma muy intuitiva.

# Etapa 2: Programación

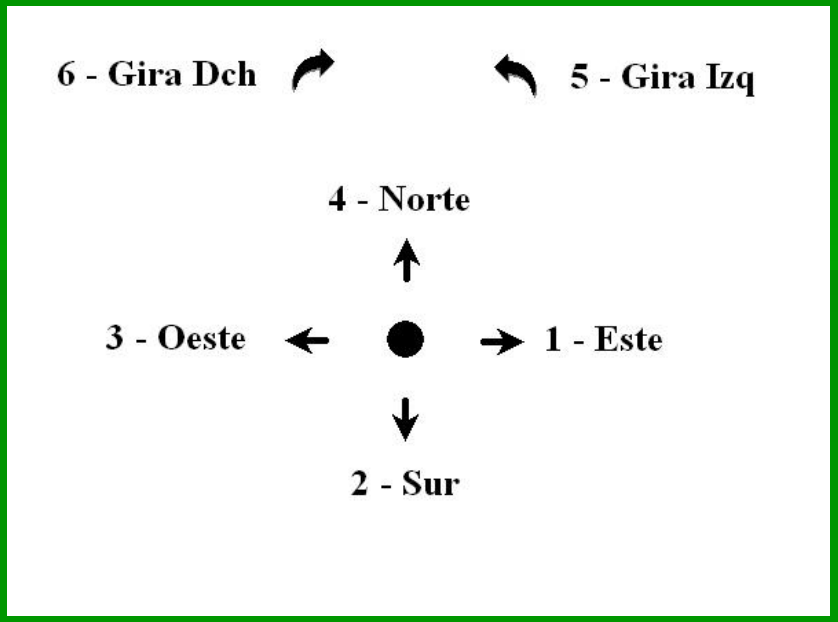
**DIRECCIÓN Y POSICIONAMIENTO**

- La simulación del camino programado se realiza en el tablero. Para determinar en que posición se encuentra el robot y que sepa en que dirección tiene que ir, se ha diseñado una matriz 5x5, donde en cada punto puede ir en 4 direcciones (Norte, Sur, Este, Oeste). Exceptuando los bordes, ya que el BeeBot no podrá salir del tablero.

| | | | | |
|---|---|---|---|---|
| (1,1) | (1,2) | (1,3) | (1,4) | (1,5) |
| (2,1) | (2,2) | (2,3) | (2,4) | (2,5) |
| (3,1) | (3,2) | (3,3) | (3,4) | (3,5) |
| (4,1) | (4,2) | (4,3) | (4,4) | (4,5) |
| (5,1) | (5,2) | (5,3) | (5,4) | (5,5) |

# Etapa 2: Programación

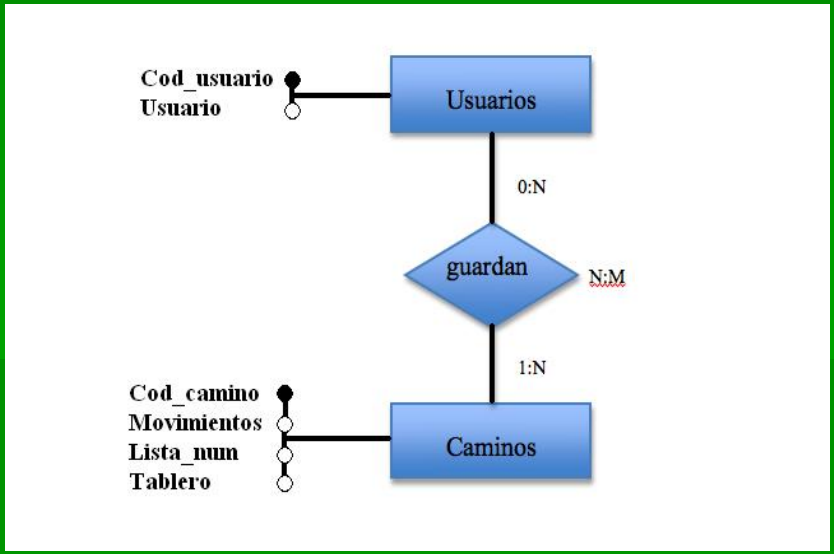
- Cada vez que pulsamos en una flecha la aplicación guarda en una lista todos los movimientos programados.
- El robot siempre parte desde la posición (3,3) del tablero y apuntando hacia la dirección Este, que equivale al número 1.
- La flecha de navegación “arriba” es la que hace que el robot avance según en la dirección que apunte, y las flechas “izquierda” y “derecha” hacen que la dirección del robot cambie.

# Etapa 2: Programación

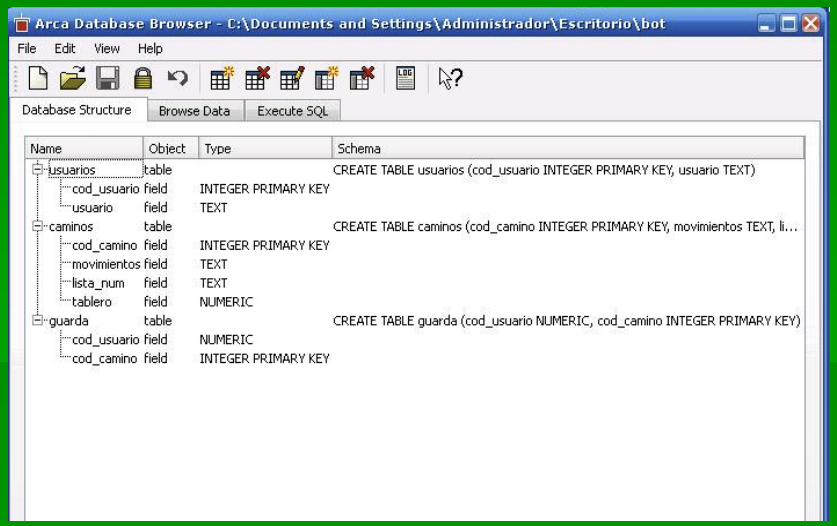
**BASE DE DATOS**

- Para la realización de nuestra aplicación multiusuario necesitamos crear una base de datos en la que poder almacenar los usuarios que vayan a utilizar el BeeBot. Además para cada usuario quedarán guardados los caminos que haya ido programando.
- Esta base de datos será dinámica ya que el contenido irá cambiando constantemente según los usuarios que usen la aplicación y los caminos que se deseen guardar.



# Etapa 2: Programación

- Director no ofrece ninguna herramienta para la creación y mantenimiento de una base de datos, por eso hay que añadir xtras.
- Arca Database Xtra es el encargado de hacer funcionar todo lo referente a la base de datos en Director.
- Viene con una aplicación complementaria para crear, revisar y modificar bases de datos, el Arca Database Browser.



# Etapa 3: Diseño

- El Bee-Bot es una herramienta dirigida a niños. Tanto los colores como los gráficos han sido escogidos para que sean llamativos.
- Está demostrado que los juguetes que tienen mucho color y son atractivos son buenos, ya que consiguen captar la atención de los más pequeños.

# Etapa 3: Diseño

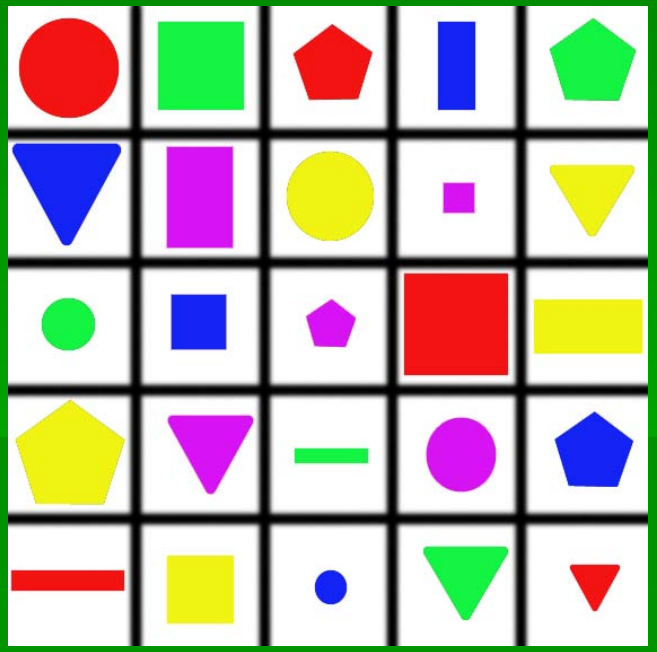
**DISCIPLINAS DE APRENDIZAJE**

- Los juegos divertidos son muy útiles ya que los niños tienden a recordar las lecciones aprendidas cuando se están divirtiendo.
- Para fomentar las habilidades de los niños se han creado 3 tableros para el Bee-Bot.

# Etapa 3: Diseño

Tablero para explorar las formas.
Sirve para conocer conceptos matemáticos además de habilidades de reconocimiento de formas y colores.

- Describir las diferencias de formas, colores y tamaños.
- ¿Pueden los niños describir las similitudes y diferencias de las formas?
- Elige una forma determinada y da pistas sobre la forma elegida ¿Pueden los niños adivinar cual es la forma seleccionada?
- ¿Cuántos triángulos pueden ver?
- ¿Cuántos colores pueden reconocer?
- Elegir una forma determinada. Observar para cada niño cual ha sido el camino elegido.
- Contar cuantos pasos han sido necesarios.
- Recorrer todos rectángulos del tablero.
- Programar el BeeBot para que vaya desde el pentágono morado, al pentágono amarillo.



# Etapa 3: Diseño

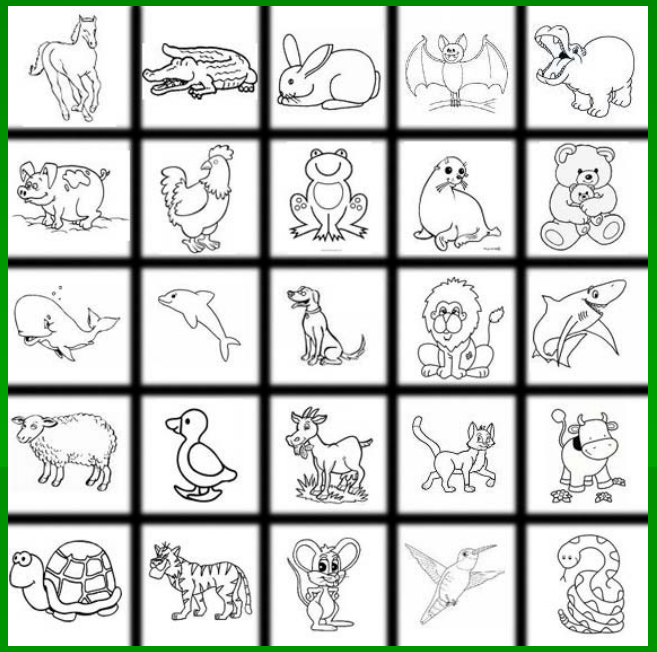
Tablero para trabajar con números. Fomenta el desarrollo de las habilidades matemáticas.

- Proponer sencillos problemas matemáticos y programar el BeeBot para que vaya a la solución.
- Programar el robot para resolver cuentas, sumas, restas, multiplicaciones y divisiones.
- Recorrer todos los números del 1 al 25.
- Seleccionar los múltiplos de 3.
- Recorrer todos los números de color azul de forma creciente.

# Etapa 3: Diseño

Tablero para aprender sobre los animales. Fomenta las capacidades cognitivas del estudiante.

- Reconocer todos los animales del tablero y aprender los nombres.
- Programar el BeeBot para que recorra los animales que puedan volar.
- Elegir los animales mamíferos del tablero.
- Seleccionar los animales que vivan en el agua.
- Recorrer todos los animales carnívoros.
- Recorrer todos los animales herbívoros.
- Programar el robot para que recorra los animales omnívoros.
- Elegir el camino para seleccionar tu mascota favorita.



# Conclusiones

- Se ha realizado una aplicación multimedia que simula la programación del robot BeeBot.
- Para aplicar este tipo de disciplinas educativas que BeeBot nos ofrece, debemos comprar el robot. La compra del robot nos supondría un gasto de unos 65€. Y los tableros BeeBot Mats varían entre 2.5€ y 25€. Además el software nos costaría 35€ por licencia. En total un pack básico puede costar unos 135€, dependiendo de el material que adquiramos.
- La idea de este proyecto consiste en la realización de un software de distribución libre, utilizando la herramienta Macromedia Director.
- Además es el inicio para unir a otros proyectos, como la creación del propio robot, con robots educativos como Lego o Arduino.
- Con la creación de este software libre, y la futura creación del propio robot obtendremos una forma más económica de aprovechar las capacidades de aprendizaje que este tipo de robots nos aportan para la educación.

# Líneas futuras

- Podrían crearse nuevos tableros con otras disciplinas de aprendizaje para fomentar nuevas habilidades de los niños.
- El tablero del BeeBot es de un tamaño determinado, 5x5. Cuando creemos los diferentes tipos de juegos no tienen porque tener todos el mismo tamaño, cada juego podría tener diferentes tamaños de tablero. Es decir, debería crearse una forma dinámica de crear los tableros, para que el BeeBot pueda moverse correctamente por esos tableros y además no se salga de los límites de cada uno de ellos.
- El robot siempre sale desde la posición central del tablero, celda (3,3). Podría programarse para que el BeeBot salga desde la posición que el usuario desee.
- Aunque no sea muy necesario quedaría mucho más visual, desde el punto de vista del diseño, si se añadiese la posibilidad de poder ver el BeeBot en 3D.
- A la hora de poder utilizar la aplicación en distintas plataformas da errores, debido a que Arca Database Xtra está solo para el sistema operativo Windows. Para solucionar este problema y dar soporte a todas las plataformas, se podría crear el proyecto en http para colgarlo en la red. Además sería necesario subir también la base de datos a un servidor, para dar independencia física. Otra forma de solucionar este problema sería utilizar otro “xtra” que soporte todos los sistemas operativos.
- Una vez creado el software de BeeBot, sería interesante la creación del robot con herramientas como Arduino o Lego, para así tener todo el “kit” de aprendizaje. Por un lado el simulador del robot y por otro el robot físico.
- Para finalizar con el proyecto se podría haber hecho unas pruebas a los niños y profesores con la herramienta, para ver los posibles ventajas e inconvenientes de la aplicación.