Pixels&Code

NOVIEMBRE 2011 > Nº 07 -

LAS MEJORES TIENDAS VIRTUALES **PREDISEÑADAS**

SEGUNDA PARTE DE **PROGRAMACIÓN DE VIDEOJUEGOS**

Diseños limpios

Te enseñamos a cuidar la estética de tus webs con un estilo que marca tendencia

Plataforma para **Resellers**

¡El negocio lo hacés vos!

Tu propio negocio de hosting con todo lo que necesitás.
Con más de 8 años de experiencia, entendemos tus necesidades. Por eso creamos una plataforma con productos y herramientas para que tu negocio crezca con la mayor rentabilidad.

***www.dattatec.com**/resellers*

ENTERATEQUETENGO.COM

- ✓ Muchas Minitas
- ✓ Vacaciones en Montecarlo
- ✓ T.V. 75 pulgadas
- ✓ Velero con fripobar
- ✓ Loft vista al mar
- ✓ Loft vista al mar
- ✓ Colección completa de muñecos Jack
- ✓ Reabrir Studio 54 por una noche

Conocé el secreto de mi éxito

WWW.ENTERATEQUETENGO.COM

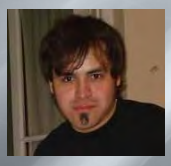
dattatec.com

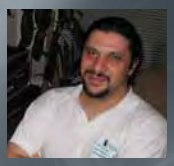
dattatec.com
Tu Hosting hecho Simple!

JEFA DE REDACCIÓN

Débora Orué

COLUMNISTAS



Adrián Ortiz
Cristian Hernán Gaitano Ornia
Gabriel Walter Gaitano Ornia
Juan Gutmann
Roxana Miguel
Natalia Cantero

DISEÑO EDITORIAL Y CREATIVIDAD

www.ampersandgroup.com.ar

REDACCIÓN

lectores@pixelscode.com

COMERCIALIZACIÓN

Dattatec.com SRL - Córdoba 3753, Rosario, Santa Fe
www.dattatec.com

DATOS DE CONTACTO

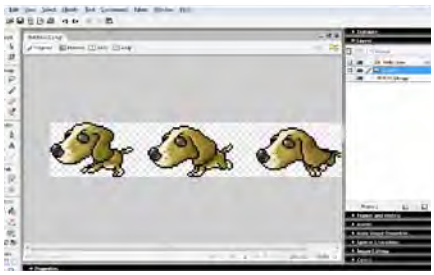
Dirección Comercial:
publicidad@pixelscode.com

Las opiniones expresadas en los artículos son exclusiva responsabilidad de sus autores y no coinciden necesariamente con la opinión de Dattatec.com SRL.

Sumario

NOTA CENTRAL

TENDENCIA:
Los secretos para diseños limpios



Entry point

Por Gabriel Gaitano Ornia

gabriel.gaitanoornia@pixelscode.com

IT NEWS:

DOS PARTIDAS QUE DUELEN MUCHO

STEVE JOBS (1955-2011)

Si bien la exacta dimensión del impacto que Steve Jobs ha tenido en el mundo IT aún no ha sido cuantificada, sólo parece ir agrandándose a medida que se revisa su aporte. Por sí solo revolucionó la informática moderna con sus innovaciones en varias ocasiones (recordemos solamente la Mac, las interfaces gráficas, el uso amplio del mouse, los muy actuales iPad y iPhone), resucitó la empresa que una vez había creado cuando parecía que su futuro estaba sentenciado y sentó la base de una leyenda. Su partida deja un vacío importante en Apple y sin duda en todo el ambiente informático. ¡Buen viaje, Steve! Te extrañaremos.

DENNIS RITCHIE (1941-2011)

El mismo mes que perdimos a Jobs, se fue otro grande la informática. Dennis Ritchie, padre del sistema operativo UNIX y del lenguaje C, falleció a los 70 años luego de una larga enfermedad. Su trabajo junto a Ken Thompson en los laboratorios de Bell entre los 60' y 70', sigue inspirando a programadores de todo el mundo, que trabajan en nuevas versiones de sistemas operativos basados en UNIX o en GNU/Linux. Y C sigue siendo un lenguaje sumamente importante (el segundo más usado a nivel mundial), y su espíritu vive en C++ y Java.

¡Hasta pronto, Dennis! Mantendremos vivo tu legado.

EL GADGET DEL MES:

CARGADOR SOLAR

¿Tu celular se está quedando sin carga? ¿La batería de tu netbook no soporta un largo viaje? Si tenés acceso al sol, esos ya no serían problemas a tener en cuenta con los cargadores solares que están apareciendo a pasos agigantados en la industria, como podemos apreciar aquí:

Ray, como es llamado el dispositivo, es una ingeniosa combinación de cargador solar con una ventosa adhesiva, lo que significa que puede ser fácilmente pegado en una ventana soleada o en el exterior de un coche, para convertir la luz solar en energía para nuestros dispositivos. Este interesante aparato incluye una batería, permitiendo almacenar carga durante el día para liberarla cuando la necesitemos.

WEB DESTACADA:

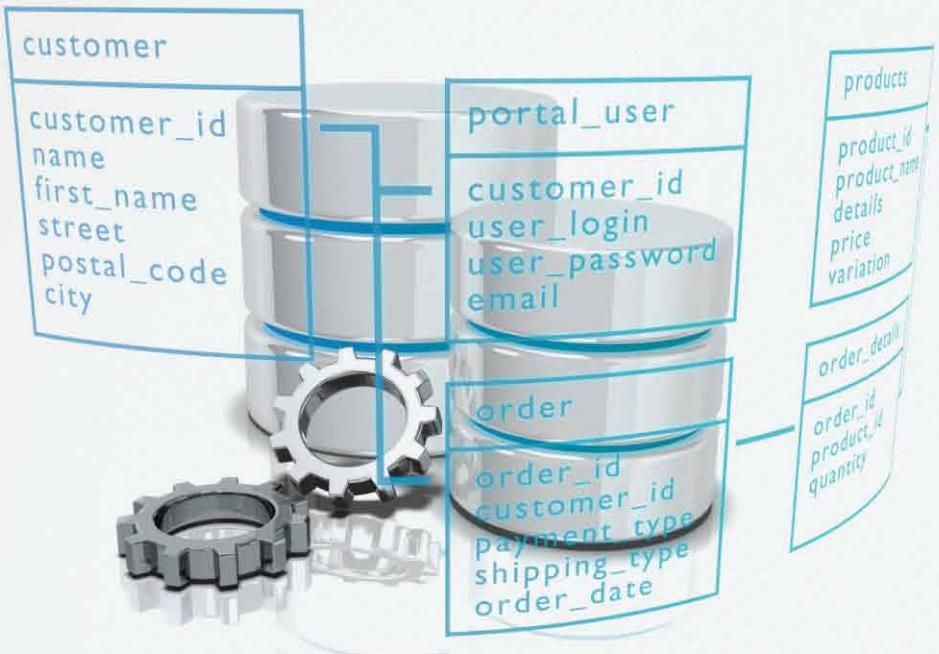
TABLA PERIÓDICA DE HTML 5

En el sitio http://joshduck.com/periodic-table.html podemos ver una interesante aplicación de HTML5 que actúa, a su vez, como una útil página de consultas. Docenas de tags html están disponibles, clasificadas, ordenadas y detalladas, dando al diseñador o programador una excelente hoja de referencia en su trabajo. ¿Qué más podemos pedir?

MÁS ALLA DE LA PC:

LA PRIMERA TABLET DE AMAZON

Kindle Fire, la nueva tableta-lector de libros de Amazon, está comenzando a lucir como una amenaza real al iPad, por lo cual, con un toque de humor, podríamos decir que Apple ya debería empezar a buscar una forma de demandarlos. Revisando las estadísticas de preórdenes (o reservas de equipos), se realizaron más de 250.000 en la semana siguiente a su presentación. El nuevo producto tiene 7 pulgadas, una pantalla de alta resistencia, permite leer ebooks y conectarse a través de una red inalámbrica a internet para navegar y ver películas. Su precio es más accesible que el del iPad, pero carece de cámara o micrófonos y está centrado en el almacenamiento en la nube.



Introducción a las bases de datos relacionales (VII)

Por Juan Gutmann

juan.gutmann@pixelscode.com

En la edición anterior continuamos trabajando con Stored Functions, pero también utilizamos "Triggers", un tipo de procedimiento especial, asociado a la ocurrencia de un evento en particular sobre una tabla. En esta ocasión volvemos a recurrir a ellos, en el marco de un problema muy particular, que también discutimos en el inicio de este cursillo, al hacer foco sobre el aspecto académico de las bases de datos relacionales: la generación de claves primarias. PostgreSQL cuenta con características especiales para ayudarnos en esta tarea, las cuales aplicaremos en esta práctica, que bastan y sobran para circunstancias normales. Sin embargo, como todo programador que trabaje en el "backend" sabe bien, cada empresa es un mundo y demasiado a menudo nos encontraremos con requerimientos que "rompen" los diseños tradicionales y nos obligan a recurrir a algunas triquiñuelas para encontrarles una solución práctica y a la vez elegante. Los "Triggers" son ideales para aplicar en estos casos. Veamos por qué.

UN BREVE REPASO

Recordemos que en un modelo relacional correctamente diseñado, toda entidad -es decir, toda tabla- debe tener una "clave primaria", compuesta por uno o más campos, que se utilizan para individualizar unívocamente el registro. Estas claves pueden ser de dos tipos: naturales y artificiales. Como claves naturales entendemos las que están formadas por atributos pertenecientes al dominio con el que estamos trabajando, que por su naturaleza son aptos para este fin. Por ejemplo, en una entidad donde almacenaremos datos de personas, una clave natural puede formarse mediante los campos "tipo de documento" y "número de documento", que nos asegurarían la unicidad. En cambio, una clave artificial casi siempre se compone de un único campo, el cual contiene un valor generado artificialmente con el objetivo de individualizar un registro. Lo habitual en estos casos es el uso de valores números consecutivos.

GENERACIÓN AUTOMÁTICA DE CLAVES

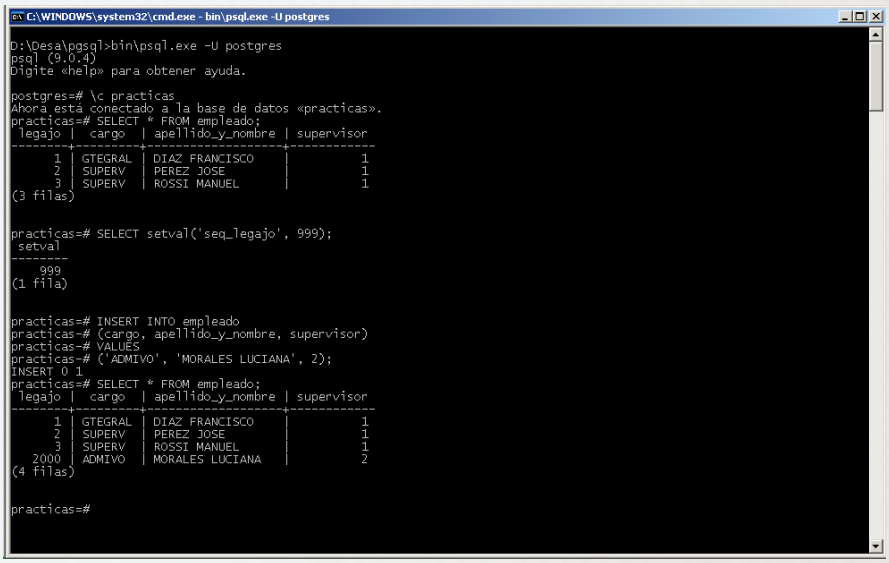
Aunque existen partidarios de ambos tipos de claves, la mayoría de los especialistas suele coincidir en que es más conveniente trabajar con claves primarias artificiales. Entre otras razones, se suele citar que una clave artificial numérica ocupa menos lugar en la base de datos (pensemos que los primeros 256 registros necesitan apenas un byte para un identificador de este tipo, mientras que de emplear una clave natural serían necesarios varios bytes para todos los registros), y también posibilita que los índices sean más chicos y puedan recorrerse con mayor velocidad al realizar una búsqueda. En algunas ocasiones se recurre a una clave híbrida: aunque el campo corresponde a un atributo real del dominio modelado, es generado automáticamente por el sistema, siendo su modalidad de uso equivalente al de una clave artificial. Este es el caso, por ejemplo, del atributo "legajo" que identifica a cada registro de la tabla "empleado", con la que trabajamos en las prácticas anteriores. Si quisiéramos generar automáticamente el número de legajo de un empleado, PostGres cuenta con un tipo de objeto creado específicamente para este fin: la "secuencia" (SEQUENCE). Su creación y uso es muy sencillo, gracias a una serie de funciones dispuestas especialmente para ello.

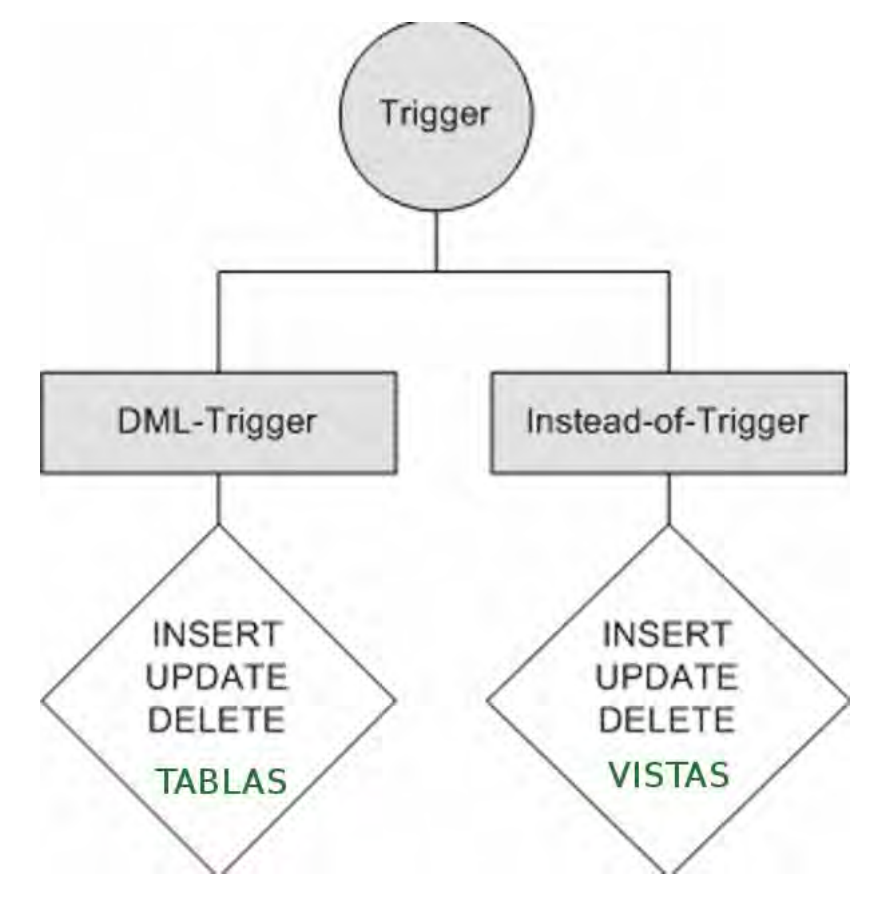
CREANDO UNA SECUENCIA

Nos dirigimos al cliente PostGres de nuestra preferencia, ya sea ésta el de consola o el de interfaz gráfica, e ingresamos lo siguiente:

```
CREATE SEQUENCE seq_prueba
```

> *"Aunque existen partidarios de ambos tipos de claves, la mayoría de los especialistas suele coincidir en que es más conveniente trabajar con claves primarias artificiales."*






```
  INCREMENT BY 1
  NO MINVALUE
  NO MAXVALUE
  START WITH 1
  CACHE 1
  NO CYCLE
  OWNED BY NONE;
```

En la práctica, todas las cláusulas (opciones) que hemos especificado son las que se asumen por defecto, y por ende son redundantes. Este comando es equivalente a haber escrito solamente "CREATE SEQUENCE seq_prueba". No obstante, haberlas escrito explícitamente nos es útil para detallar para qué sirve cada una de ellas. INCREMENT BY establece el intervalo numérico con el que trabajará la secuencia. Casi siempre lo usual es que vaya de uno en uno, pero si necesitamos que cada nuevo valor generado vaya de dos en dos (1, 3, 5, 7 ...) o con cualquier otro intervalo, puede lograrse con esta opción. MINVALUE y MAXVALUE definen los valores mínimos y máximos que puede generar la secuencia, siendo éstos por defecto para una secuencia ascendente 1 y (2∧63 -1), respectivamente. START WITH define el primer valor que devolverá la secuencia. CACHE sirve para "reservar" una cantidad de elementos de la secuencia para cada sesión activa de PostGres; si aquí especificamos "10" y tenemos dos sesiones activas, con una secuencia nueva se reservarán para la primera conexión los valores del 1 al 10; si la segunda sesión solicita valores de la secuencia mientras la primera todavía está activa, recibirá los que se encuentran entre el 11 y el 20. CYCLE y NO CYCLE indican en el primer caso que al llegar al valor máximo de la secuencia la misma regresa a su valor inicial y sigue numerando normalmente desde allí, mientras que en el segundo caso una vez alcanzado el valor máximo, solicitar uno nuevo a la secuencia deriva en un mensaje de error. Por último, con OWNED BY podemos asociar la secuencia a una columna de una tabla en particular, permitiendo que si la columna es eliminada (DROP) la secuencia desaparezca automáticamente junto con ella.

TRABAJANDO CON LA SECUENCIA

Aunque en PostGres existen cuatro funciones en total diseñadas para trabajar con secuencias, emplearemos casi exclusivamente dos: nextval() y currval(). La primera genera un nuevo valor acorde con el estado actual de la secuencia y lo devuelve, mientras que la segunda retorna el último valor generado en la sesión actual para dicha secuencia. Existen otras dos funciones menos utilizadas: setval(), que permite reasignar a una secuencia su valor actual, y lastval(), que retorna el último valor devuelto por cualquier secuencia. Veamos cómo trabajan nextval() y currval():

```
SELECT currval('seq_prueba');
ERROR:  currval de la secuencia «seq_prueba» no está definido en esta sesión
```

```
SELECT nextval('seq_prueba');
 nextval
---------
       1

SELECT nextval('seq_prueba');
 nextval
---------
       2

SELECT currval('seq_prueba');
 currval
---------
       2
```

Para lograr que la clave primaria de una tabla se genere en forma automática en base a una secuencia que hayamos creado, debemos asignar a la columna clave como valor por defecto la función nextval('nombre_secuencia'). Recordemos que para el modelo de datos de práticas, habíamos creamos así la tabla empleado:

```
CREATE TABLE empleado (
legajo            NUMERIC(5)        NOT NULL,
cargo             VARCHAR(10)       NOT NULL,
apellido_y_nombre VARCHAR(100)      NOT NULL,
...
);
```

Para generar automáticamente el valor de legajo, podemos hacerlo de esta manera:

```
CREATE SEQUENCE seq_legajo MAXVALUE 99999;

CREATE TABLE empleado (
legajo            NUMERIC(5)        DEFAULT nextval('seq_legajo') NOT NULL,
cargo             VARCHAR(10)       NOT NULL,
apellido_y_nombre VARCHAR(100)      NOT NULL,
...
);
```

Como sabemos por definición que la empresa nunca tendrá más de 99.999 empleados (una cifra inalcanzable para una empresa chica o mediana), en lugar de redefinir el tipo de datos de legajo como BIGINT -el tipo que devuelven nextval() y currval()- mantenemos nuestra elección de NUMERIC(5), pero indicando "por si las moscas" a la secuencia que su valor máximo no puede superar el mayor número que puede contener este campo. Si la tabla va a contener millones de registros, entonces sí el tipo de datos correspondiente al campo clave autogenerado deberá ser BIGINT obligatoriamente.

Al momento de insertar una nueva fila, omitimos especificar el valor del atributo "legajo" y será tomado automáticamente de la secuencia. Como la entidad empleado incluye una jerarquía (es decir, almacena la información de quién es el jefe de quién), debe ser cargada respetando un orden. El primer registro que contenga debe ser el del empleado de máxima jerarquía, ya que será su propio supervisor. Aunque sepamos a ciencia cierta que su legajo será el "1" por ser éste el valor inicial de la secuencia, es más prolijo especificar como valor del atributo supervisor

```
-- Function: f_trg_empleado_ins()

-- DROP FUNCTION f_trg_empleado_ins();

CREATE OR REPLACE FUNCTION f_trg_empleado_ins()
  RETURNS trigger AS
$BODY$
   DECLARE
       v_legajo empleado.legajo%TYPE;

   BEGIN
       v_legajo := nextval('seq_legajo');

           IF v_legajo = 1000 THEN
           SELECT setval('seq_legajo', 1999);
                   v_legajo := nextval('seq_legajo'); -- Retornará 2000
       END IF;

           NEW.legajo := v_legajo;

       RETURN NEW;
   END;
$BODY$
  LANGUAGE plpgsql VOLATILE
  COST 100;
ALTER FUNCTION f_trg_empleado_ins() OWNER TO postgres;
```

> *"En lugar de definir al campo legajo con un llamado a nextval() como valor por defecto, vamos a recurrir a un Trigger y a su correspondiente Stored Function..."*

el valor generado por la secuencia para ese mismo legajo, mediante currval().

```
# INSERT INTO empleado
(cargo, apellido_y_nombre, supervisor)
VALUES
('GTEGRAL', 'DIAZ FRANCISCO', currval('seq_legajo'));
```

```
# SELECT * FROM empleado;
 legajo |  cargo  | apellido_y_nombre | supervisor
--------+---------+-------------------+------------
      1 | GTEGRAL | DIAZ FRANCISCO    |          1
```

EL TIPO DE DATOS SERIAL

PostGres cuenta con un tipo de datos que nos permite facilitar todavía más este proceso, en el caso de que no tengamos condiciones especiales que satisfacer (por ejemplo, un valor máximo de 99.999, como en el caso de seq_legajo). Al crear una tabla, podemos especificar:

```
CREATE TABLE mitabla (
    nombre_columna_clave SERIAL
);
```

```
Esto equivale a lo siguiente;
```

```
CREATE SEQUENCE nombre_columna_clave_seq;
```

```
CREATE TABLE mitabla (
    nombre_columna_clave DEFAULT nextval('nombre_columna_clave_seq') NOT NULL);

ALTER SEQUENCE nombre_columna_clave_seq
  OWNED BY mitabla.nombre_columna_clave;
```

GENERANDO CLAVES DESDE UN TRIGGER

Ahora, supongamos que existen reglas de negocio que imponen ciertas características a la generación automática de valores clave. Por ejemplo, el departamento de RRHH podría haber determinado por alguna razón que no deben asignarse los números de legajo entre 1000 y 1999. Esto no nos impide el uso de secuencias para automatizar esta tarea, pero sí debemos trabajar un poco más para resolverlo. En lugar de definir al campo legajo con un llamado a nextval() como valor por defecto, vamos a recurrir a un Trigger y a su correspondiente Stored Function para resolver este requerimiento. Asumiendo entonces que "legajo" fue definido como un simple NUMERIC(5) NOT NULL, si la tabla ya contiene registros, creamos la secuencia con la cláusula START WITH, indicándole que empiece por el número siguiente al último existente, o bien la creamos para que empiece por 1 (valor por defecto) y avanzamos manualmente la secuencia hasta ese valor de esta forma:

```
SELECT setval('seq_legajo', (SELECT MAX(legajo) FROM empleado)::BIGINT);
 setval
--------
      2
```

Noten la conversión explícita, declarada con los cuatro puntos "::". Ocurre que el tipo de datos del campo "legajo" es NUMERIC, y el segundo argumento de setval() debe ser un BIGINT. De no recurrir a la conversión explícita, esta sutil diferencia derivaría en este error:

```
ERROR:  no existe la función setval(unknown, numeric)
LÍNEA 1: SELECT setval('seq_legajo', (SELECT MAX(legajo) FROM emplead...
                  ^
```

SUGERENCIA: Ninguna función coincide en el nombre y tipos de argumentos. Puede desear agregar conversión explícita de tipos.

Afortunadamente, el mismo PostGres nos indica que la solución es recurrir a una conversión explícita, en este caso totalmente factible, ya que ambos tipos de datos son enteros numéricos.

Una vez cumplidos estos pasos, escribimos la Stored Function que genere automáticamente el valor de legajo, teniendo en cuenta la salvedad impuesta por la regla de negocio, y creamos su correspondiente trigger:

```
CREATE OR REPLACE FUNCTION f_trg_empleado_ins() RETURNS trigger AS $$
   DECLARE
      v_legajo empleado.legajo%TYPE;

   BEGIN
      v_legajo := nextval('seq_legajo');

      IF v_legajo BETWEEN 1000 AND 1999 THEN
            PERFORM setval('seq_legajo', 1999);
            v_legajo := nextval('seq_legajo'); -- Retornará 2000
      END IF;
```

```
      NEW.legajo := v_legajo;

      RETURN NEW;
   END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_empleado_ins
   BEFORE INSERT ON empleado
   FOR EACH ROW
   EXECUTE PROCEDURE f_trg_empleado_ins();
```

Asignar valores a uno o más campos mediante un trigger de INSERT o UPDATE sólo puede realizarse en un trigger de "BEFORE". Los triggers pueden programarse para que se ejecuten antes o después del evento correspondiente, pero en los triggers de "AFTER" no es posible realizar este tipo de procesamiento. Los triggers en PostGres soportan también una cláusula "INSTEAD OF", que sólo puede aplicarse en vistas (VIEWs), y habilitan al trigger a reemplazar completamente con su propia lógica las modificaciones solicitadas a los datos de la vista por las sentencias DML INSERT, UPDATE o DELETE. En este caso, el trigger no corre antes (BEFORE) ni después (AFTER) de la sentencia SQL causante del evento, sino "en lugar de".

Para probar el correcto funcionamiento de la autoasignación del legajo, lo hacemos así:

```
# INSERT INTO empleado
(cargo, apellido_y_nombre, supervisor)
VALUES
('SUPERV', 'ROSSI MANUEL', 1);
```

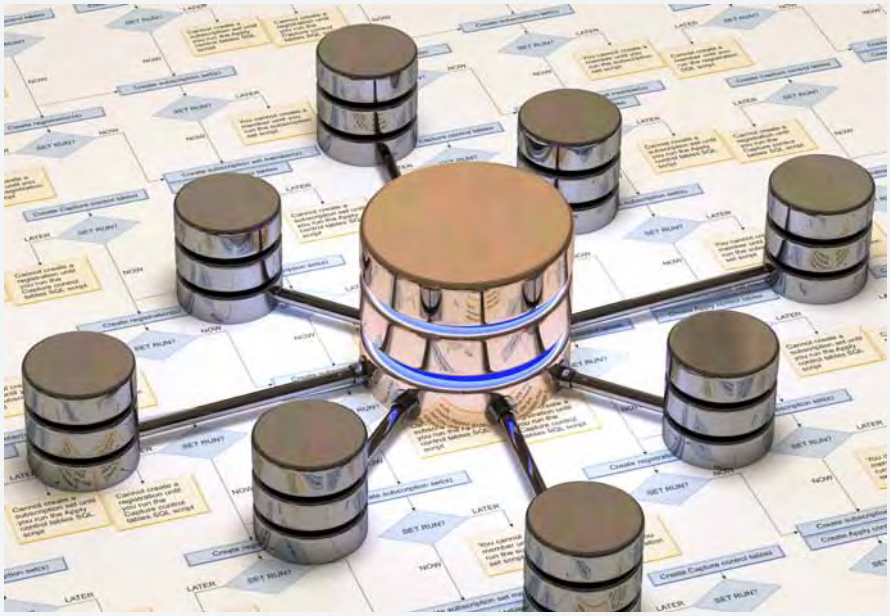
```
# SELECT * FROM empleado;
 legajo |  cargo  | apellido_y_nombre | supervisor
--------+---------+-------------------+------------
      1 | GTEGRAL | DIAZ FRANCISCO    |          1
      2 | SUPERV  | PEREZ JOSE        |          1
      3 | SUPERV  | ROSSI MANUEL      |          1
```

En la tabla ya existían los empleados con legajo 1 y 2, y el trigger le asignó correctamente el valor "3" al atributo legajo del nuevo empleado. Ahora comprobemos si se cumple que no se asignarán legajos entre 1000 y 1999. Para ello, forzamos el valor actual de la secuencia a 999:

```
# SELECT setval('seq_legajo', 999);
```

```
# INSERT INTO empleado
(cargo, apellido_y_nombre, supervisor)
VALUES
('ADMIVO', 'MORALES LUCIANA', 2);
```

```
# SELECT * FROM empleado;
 legajo |  cargo  | apellido_y_nombre | supervisor
--------+---------+-------------------+------------
      1 | GTEGRAL | DIAZ FRANCISCO    |          1
      2 | SUPERV  | PEREZ JOSE        |          1
      3 | SUPERV  | ROSSI MANUEL      |          1
   2000 | ADMIVO  | MORALES LUCIANA   |          2
```



> *(...) las secuencias son herramientas extremadamente útiles a la hora de generar valores para claves primarias, por su sencillez de aplicación y transparencia para las aplicaciones cliente."*

SECUENCIAS Y TRANSACCIONES

Como ya explicamos en la cuarta entrega de esta introducción, PostGres soporta transacciones, que permiten tratar como una única unidad lógica una serie de modificaciones realizadas a los datos, confirmándolas (COMMIT) si todo fue bien, o deshaciéndolas (ROLLBACK) en el caso de que se presente un error. Esto impide que un grupo de actualizaciones que deben realizarse en forma conjunta se ejecute en forma parcial, provocando inconsistencia en la información almacenada en la base de datos. Sin embargo, debe tenerse en cuenta que las secuencias no trabajan en forma transaccional. Una vez solicitado un nuevo valor a una secuencia mediante nextval(), o modificado su valor actual con setval(), estas operaciones no tienen vuelta atrás, aunque se ejecute un ROLLBACK. Es decir, ante algún inconveniente, pueden producirse "baches" en la numeración generada, ya que si un valor no llegara a grabarse, se perderá. Esto no presenta mayores inconvenientes cuando el único objetivo de la secuencia es generar un identificador único. Sin embargo, en ocasiones donde es imperativo que no existan huecos en la numeración -por ejemplo, al generar números de factura- este problema torna inviable el empleo de secuencias, y nos obliga a buscar soluciones alternativas. En un sistema monousuario, podríamos escribir un trigger que obtenga el último valor utilizado con un SELECT MAX() y lo incremente en uno. En un sistema multiusuario debemos recurrir a métodos más complicados, ya que si un usuario obtiene el último valor de la clave con un MAX() y otro hace lo propio antes que la transacción del primero llegue a confirmarse, ambos obtendrán el mismo valor y se generará una violación de clave primaria. Una solución típica en estos casos es recurrir a una tabla que almacene secuencias, y bloquear (LOCK) el registro para evitar que sea leído por otros usuarios en el momento en el que se obtiene el último valor, liberando el bloqueo luego de generada la nueva clave y actualizada la tabla de secuencias.

EN RESUMEN

Para finalizar la presente entrega, les brindamos como conclusión que las secuencias son herramientas extremadamente útiles a la hora de generar valores para claves primarias, por su sencillez de aplicación y transparencia para las aplicaciones cliente. También hemos visto otras funcionalidades a la hora de trabajar con Triggers, cuya versatilidad permite implementar con poco esfuerzo complejas reglas de negocio. En el próximo número, seguiremos trabajando con este tipo de procedimientos almacenados, demostrando cómo pueden ser empleados para funciones de auditoría, como mantener un histórico de cambios para cada una de las tablas del modelo de datos, posibilitando de esta manera tanto un control exhaustivo sobre quienes realizan cambios a los datos, como la recuperación de información modificada o eliminada por error. ¡Hasta la próxima!

Programación de videojuegos Parte 2

La primera entrega de este fascinante artículo orientado al back-office del pasatiempo preferido por muchos, nos sirvió para conocer los orígenes del mundo de los videojuegos, conocer también a los primeros pioneros que marcaron un hito en este mundo virtual y también saber con qué herramientas involucrarnos en el inicio de la programación de aventuras gráficas sin morir en el intento. En esta segunda parte terminaremos de darle vida a nuestro videojuego puliendo los detalles estéticos que hacen a una aventura gráfica más interesante.

Por Fernando Luna
fernando.luna@pixelscode.com

"Conocer la historia nos sirve para saber en dónde estamos parados hoy". Por ello, la introducción al apasionante mundo de los videojuegos que vivimos en el número de Septiembre 2011 de Pixels&Code, nos sirvió para conocer el papel importante que cada gigante de hoy ocupó en los inicios de este mundo virtual, lo cual de alguna manera nos sirve como experiencia y para aprender a conocer un poco la mente y objetivos de aquellos que nos marcaron el camino que hoy queremos seguir trazando. Para no morir en el intento de programar videojuegos, siempre es bueno buscar herramientas que nos faciliten las cosas al principio, y que a su vez sean ricas en contenido web, para poder aprender y tener a quienes consultar para saber mejor cómo superar los obstáculos que nos surgen en el camino. Conocimos que Game Maker es la herramienta perfecta para dar el salto hacia el mundo de los videojuegos y no morir en el intento, ya que combina simpleza para resolver problemas, una interfaz intuitiva, un editor de imágenes básico y fácil de manejar, y un corazón poderoso basado en el lenguaje propio GML, que sirve para programar cualquier acción que deseamos que nuestros personajes realicen dentro de la partida.

El juego que planificamos (Pixels&Space) consiste en una batalla espacial donde nosotros manejamos una nave y debemos combatir con los enemigos que vienen de frente, al mejor estilo Gálaga. Hasta ahora sólo teníamos el escenario principal, algunos planetas orbitando y nuestra nave, la cual se desplaza de manera horizontal por la pantalla.

Ahora agregaremos enemigos, crearemos los disparos para nuestra nave, incorporaremos una Health Bar, y un contador. Al finalizar el videojuego, agregaremos una pantalla de inicio personalizada y otros detalles finales.

ENEMIGOS AL ATAQUE

El objetivo es que naves enemigas vengan hacia nosotros, con lo cual manejaremos el desplazamiento de los gráficos de manera inversa a como va el desplazamiento orbital, y con un sólo modelo de nave replicaremos la misma por toda la pantalla, creando así un batallón.

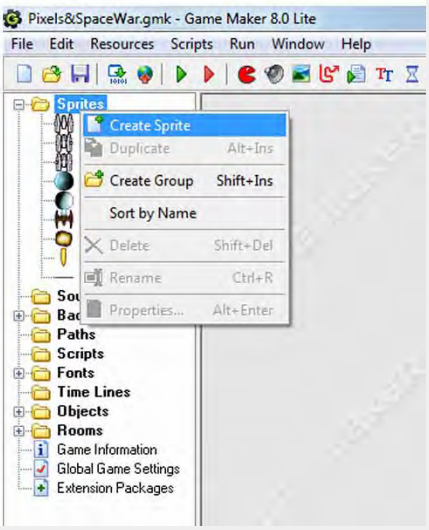
Para que los malos sean partícipes de nuestro juego, descarguemos nuevamente el Pack de archivos de imágenes desde el siguiente link: www.f-digital.com.ar/pixelscode/recursos_gm.rar
Copiemos el contenido de la carpeta resources a la carpeta de nuestro proyecto. Luego, desde Game Maker, abrimos el proyecto del videojuego y en el menú de carpetas lateral seleccionamos sobre la carpeta Sprites/Create New Sprite. Sobre la ventana que se abre, seleccionamos Edit Sprite, y nos abrirá una segunda ventana: Sprite Editor.

Allí presionamos CTRL + A (agregar imagen desde archivo), y luego de la carpeta Resources seleccionamos la imagen: vulture_droid_b_1.png. Una vez agregada, realizamos este mismo paso, seleccionando la imagen: vulture_droid_b_2.png. Tildando el checkBox Show Preview, lograremos obtener una vista previa del efecto que crea la nave enemiga. Acabamos de crear un Sprite animado.

Aceptamos todas las ventanas abiertas y procedemos a crear a continuación un Object. Sobre la carpeta lateral, clic con el botón derecho del mouse, y seleccionamos Create Object. Tildamos las opciones Visible y Solid. A continuación le agregamos los eventos, presionando Add Event/Create, luego desde la pestaña Move, agregamos Set the vertical Speed = 3. En la pestaña Main2, seleccionamos Set Alarm 0 to 20. Add Event/Step/Step, desde la pestaña Control, agregamos Test Variable y cargamos los campos Variable = y,

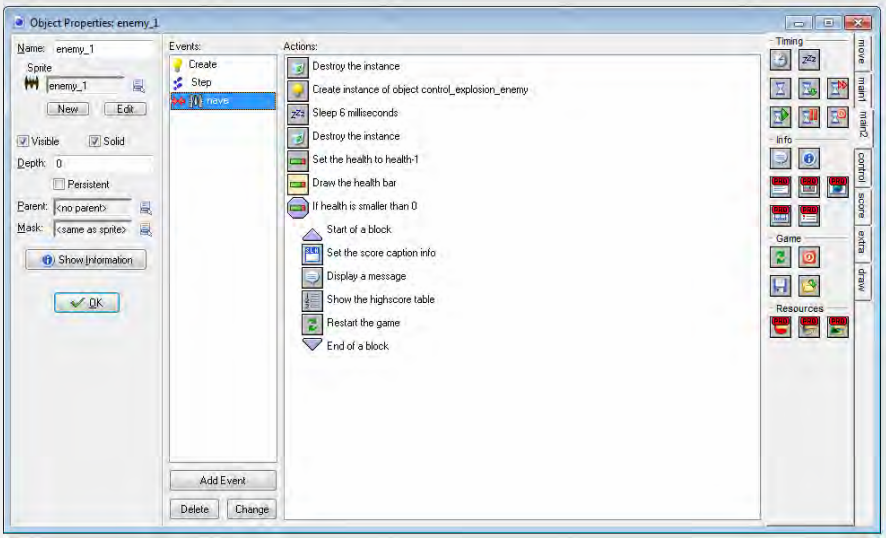
La creación de este nuevo Sprite, permitirá incorporar un enemigo, que se replicará automáticamente mediante la configuración de eventos en los objetos

Value = room_height+32, Operation = Larger than, y presionamos Aceptar. Agregamos desde la pestaña Move/Jump to position y completamos los campos: x = random(room_width), y = -32. Luego, desde la pestaña Main2/Set Alarm 0 = 100.



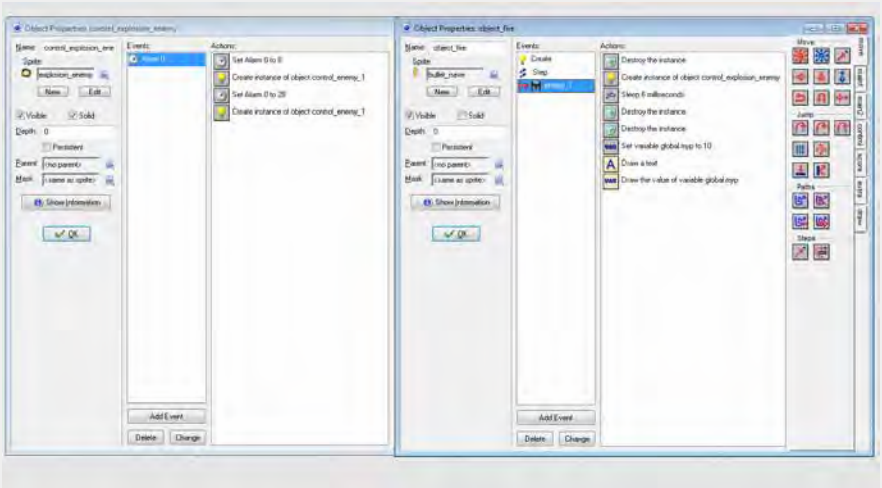
Vamos con uno más: Add Event/Collision, en la pestaña Main1/Destroy the instance, Create Instance/object Explosion_enemy, en la pestaña Main2/Sleep = 6 miliseconds, nuevamente agregamos el evento Destroy the instance/Object = control_explosion_enemy, desde la pestaña Score/Set Health = health-1, y Draw Health bar en las posiciones (20, 20, 140,

Los sprites animados se pueden realizar a través de varias imágenes, utilizando una misma dividida en secciones, o a través de Gifs animados. Estos crearán una verdadera interactividad en nuestro juego.



Create Instance of Object control_enemy_1 (Object = control_enemy_1, x = random(room_width-32), y = -32), Set Alarm 0 to 28, y Create Instance of Object control_enemy_1 copiando las acciones de la primer instancia creada, cambiando solo el valor de y = -95. Con esto generamos aleatoriamente la aparición de naves en la pantalla.
Ahora creamos el objeto object_fire, que será nuestro disparo contra los enemigos. Y vamos con los eventos: Add Event/Create, seteando su

La mayor parte del control de las acciones del videojuego están incluidas sobre el objeto Nave, quien controla la energía, colisiones y otros eventos importantes de la partida



40), seteando a continuación Back Color = none y Bar Color = green to red. Sin irnos de la pestaña Score, seleccionamos Test health: Value = 0, Operation = smaller than. Por último, en la pestaña Control/Start of a Block y End of a Block, en el medio de ambos, desde la pestaña Main2/Display a message = Su nave ha quedado sin energía. El juego ha terminado!, luego Score/Show the highscore table, y Restart the game.

EFECTOS EXPLOSIVOS

Cuando una nave enemiga colisiona contra nuestra nave, ésta se destruye, al igual que cuando le disparamos a la nave. Para eliminarla de la pantalla, crearemos un efecto de explosión. Nuevamente creamos un nuevo Sprite, y agregamos la imagen explosion1_strip6.png. Esta imagen contiene un fotograma de seis imágenes que ejemplifican una explosión. Aceptamos y cerramos, grabando este sprite con el nombre explosion_enemy. Agregamos otro Sprite, seleccionando la imagen spr_laser.png y grabando el mismo como bullet_nave.
Ahora vamos por los objetos que utilizarán estos Sprites. Creamos un nuevo Objeto al cual llamamos control_explosion_enemy. Seleccionamos como Sprite a explosion_enemy, y le agregamos los eventos: Add Event/Alarm 0. Luego cargamos las acciones: set Alarm 0 to 8,

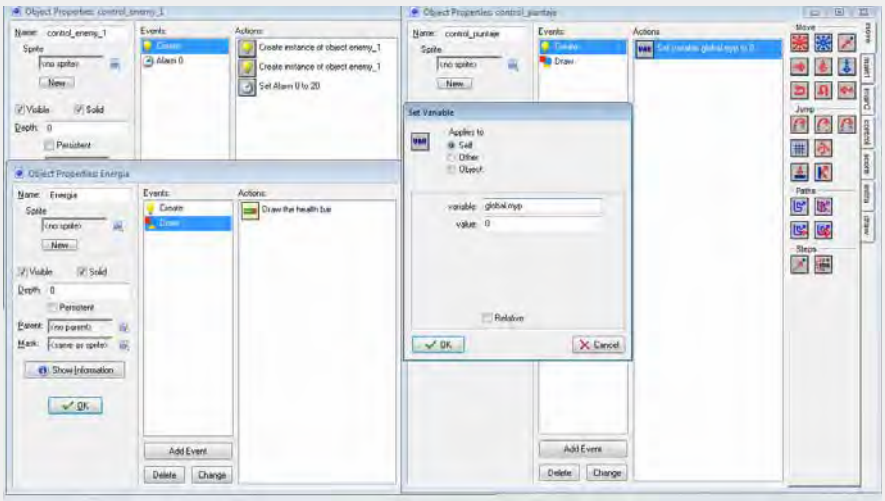
Dos objetos que controlan partes importantes del juego: Los enemigos y nuestros disparos

velocidad vertical en -10. Esto indica la velocidad con la que saldrá el rayo disparado desde nuestra nave. Add Event/Step, y agregamos un control de variable, preguntando si y < -16. Luego agregamos la destrucción de la instancia Self, lo que hará que una vez el rayo salga del escenario visible del juego, se destruya su instancia,

liberando así la memoria. Por último Add Event/Collision, configurando que chequee si colisiona nuestro rayo contra el objeto Enemy_1, ocurra lo siguiente: Destruya su propia instancia (Self), cree una instancia del objeto control_explosion_enemy, Sleep = 6 milisegundos, destruya la instancia del otro objeto, en este caso enemy_1 y a su vez, nuevamente una instancia de destrucción aplicada a control_explosion_enemy. Luego creamos una variable, que llamaremos global.myp, seteando la misma en 10 de manera incremental. (Esto será nuestro puntaje). Agregamos Draw a text, lo que hará que se refleje en pantalla la palabra Puntaje: (Text = Puntaje: , x = 410, y = 20), y los puntos que tenemos: Draw the value of variable: (Variable = global.myp, x = 500, y = 20).
En el proceso de destrucción de enemigos y puntaje en pantalla creamos una variable global.myp. Toda variable que creemos con la palabra global, seguida de un punto y un nombre de variable a elección nuestra, será una variable que mantendrá su valor para leerse o actualizarse dentro de cualquier instancia del juego. Si no intercedemos la palabra global., cualquier variable que creemos se usará y destruirá dentro del evento donde la creamos, perdiendo en cada ejecución del evento su valor seteado.

Controlando la interactividad

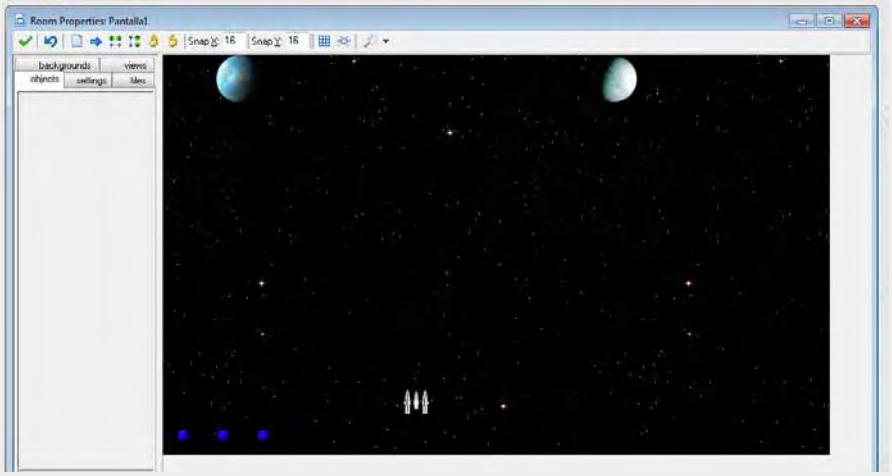
Al momento hemos creado los objetos pertenecientes a todos los protagonistas de nuestro videojuego (Nave, enemigos, disparos, planetas orbitando y espacio en movimiento). Ahora nos queda comenzar a controlar los eventos que harán que el juego en sí cobre vida. Para ello,



debemos crear otros objetos, sin vincularlos con Sprites determinados. Serán objetos vacíos que se ejecutarán con sólo ubicarlos en pantalla. Vamos a crear el primer objeto: control_enemy_1. Agregamos el evento Create, y dentro de éste las siguientes acciones: Create Instance (Object = Enemy_1, x = random(room_width-64)), y = -32). Nuevamente la acción Create Instance (Object = Enemy_1, x = random(room_width-110)), y = -32). Set Alarm 0 to 20. Nuevamente agregamos un evento Alarm 0, repetimos la acción Create Instance (Object = Enemy_1, x = random(room_width-64)), y = -32), y Set Alarm 0 to 20.
Agregaremos una barra de energía, la cual disminuirá cada vez que colisionemos con una nave. Creamos el objeto Energia, le agregamos el evento Create, y la acción Set health = 100, lo que nos permitirá arrancar la partida con el 100% de energía. Luego agregamos el evento Draw, y dentro de este la acción Draw the health bar con los siguiente valores (x1 = 20, y1 = 20, x2 = 160, y2 = 30, Back color = black, bar color = green to red).
Por último, agregamos un objeto llamado control_puntaje, con el evento Create y la acción Set Variable con los valores (variable = global.myp, Value = 0) y por último el evento Draw con las acciones (Set the color = orange),

La creación de Objects también incluye a elementos no visibles, pero vitales para la ejecución del videojuego, como son el control de las naves enemigas, la barra de energía y el puntaje acumulado de la partida

> *"El juego que planificamos (Pixels&Space) consiste en una batalla espacial donde nosotros manejamos una nave y debemos combatir con los enemigos que vienen de frente, al mejor estilo Gálaga."*



> *Cuando una nave enemiga colisiona contra nuestra nave, ésta se destruye, al igual que cuando le disparamos a la nave. Para eliminarla de la pantalla, crearemos un efecto de explosión.”*

la acción Draw a text con los valores (Text = Score: , x = 410, y = 20, y la acción Draw a value of variable con los valores (variable = global.myp, x = 500, y = 20). Aceptamos la ventana, ¡y a terminar de armar el juego!

Terminar de ensamblar las piezas

Abrimos nuevamente el escenario del juego, Rooms, y seleccionamos el único escenario que creamos. En la pestaña Objects, desplegamos el menú de selección “Object to add with left mouse”, y seleccionamos control_puntaje. Luego hacemos un clic en cualquier parte de la pantalla. Repetimos este paso, seleccionando previamente el objeto control_enemy_1, y lo mismo para el objeto Energia. Nos deben quedar en pantalla tres pequeños íconos de color azul con un signo de interrogación en su interior.

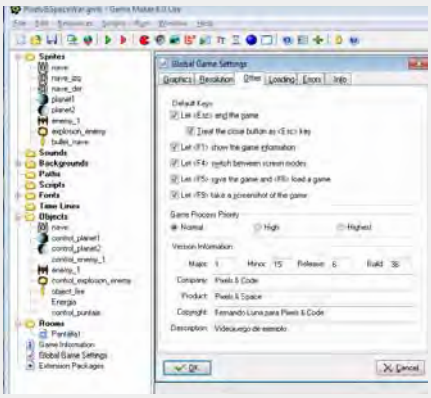
Abrimos nuevamente el objeto nave y le agregamos el evento press <Space>, y dentro de este las acciones: create instance of object_fire y Set Alarm 0 to 15. Antes de ejecutar nuestro videojuego terminado y comenzar a disfrutar de él, hagamos los ajustes finales para poder compilarlo y luego subirlo a nuestra red social favorita para compartirlo con nuestros amigos. Configuraremos la información principal del videojuego, accediendo al menú lateral y seleccionando Global Game Settings. En esta ventana encontraremos varias pestañas para configurar. En la pestaña Graphics, tildamos la opción Start in full-screen mode, Freeze the game when the form loses focus, y destildamos la opción Display the cursor. Lo que estamos haciendo, básicamente, es que el juego se inicie a pantalla completa, no muestre el cursor del mouse mientras jugamos y que se pause automáticamente si cambiamos de pantalla mediante ALT+TAB. Aunque esté a pantalla completa, recordemos que presionando la tecla F4, podemos intercalar entre modo ventana y pantalla completa. La siguiente pestaña a configurar es Other, donde tildamos Let <Esc> end the game, destildamos Let <F1> show the game information, y los campos inferiores correspondientes a la información de versión. Company, Product, Copyright, Description, con los datos que deseamos que figuren.

Luego, vamos a la pestaña Loading, donde tildamos Show your own image while loading, y presionamos el botón Change Image, seleccionando el archivo splash_screen.jpg de la carpeta Resources.

> *Un punto importante es incorporar los objetos invisibles al escenario del juego. Esto es vital para que todo marche sobre ruedas durante el desarrollo del videojuego.*

Tildamos la opción Default loading progress bar y el botón Change Icon, si deseamos un ícono en particular que creamos para la oportunidad. El videojuego mostrará una portada, o Splash Screen, personalizada junto con una barra de progreso mientras se carga el juego. Recordemos que en la versión Paga de Game Maker, el Logo de la compañía no es mostrado cuando ejecutamos nuestros juegos. Si queremos trabajar de manera profesional y crear juegos competitivos en el mercado, con tan sólo 45 dólares podremos obtener la versión oficial de esta aplicación, junto con el soporte técnico y la actualización de versiones que publica el equipo de Yoyo Games.

Con estos datos completos y marcados, ya estamos listos para compilar



La información del videojuego es importante, ya que si queremos comercializar o hacernos conocidos, siempre es importante completar esto para que los interesados puedan contactarnos.

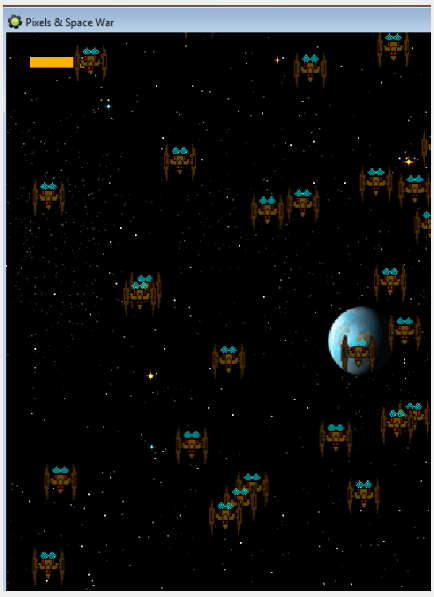
y ejecutar nuestro juego. Compilemos el mismo, generando el ejecutable, seleccionando el botón Create stand-alone Executable de la barra de herramientas.
Una vez compilado el ejecutable, podemos visualizar en las propiedades de éste que los datos configurados de Copyright y la versión del juego, entre otros, ya están incluidas en el archivo EXE.

Sólo nos resta el doble clic para comenzar a disfrutar de nuestra aventura gráfica, apreciando previamente la ventana personalizada de carga del videojuego. Al ejecutar el juego veremos que las naves enemigas vendrán hacia nosotros, y deberemos esquivar las mismas para no perder energía, y a su vez dispararles presionando la barra espaciadora del teclado para poder destruirlas. Sobre el extremo superior izquierdo veremos nuestra barra de energía, y en el extremo superior derecho el puntaje que llevamos.

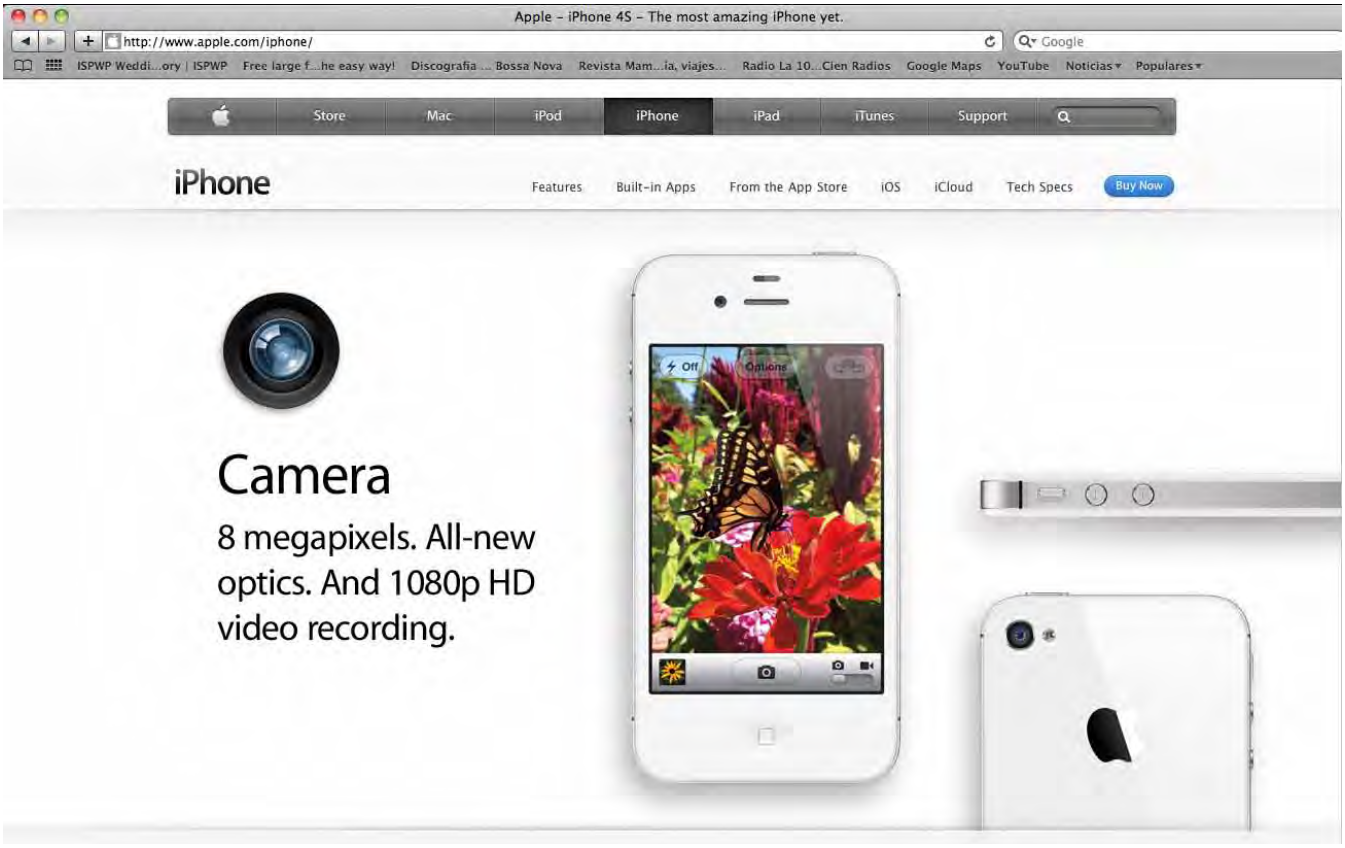
UN MUNDO DE POSIBILIDADES

Este juego, si bien no está completo, ya nos da una idea de cómo se crea una aventura gráfica con Game Maker. Podemos aprovechar la comunidad Game Maker (http://www.comunidadgm.org/) para preguntar a los usuarios más experimentados cómo realizar, por ejemplo, un menú de inicio del videojuego, o niveles más complejos, manteniendo siempre nuestro puntaje a medida que avanzamos de nivel. Cómo personalizar la ventana de High Score final, que se muestra cuando perdemos la partida, o cómo agregar objetos que nos permitan recuperar el nivel de energía. También podemos modificar nosotros el videojuego, experimentando con que nuestros enemigos también nos disparen, elevando así la complejidad, o incorporar del pack de recursos de imágenes, otros enemigos más avanzados, que puedan cambiar el desplazamiento en la pantalla, dificultando aún más la partida. La comunidad Game Maker también nos podrá ayudar a incorporar sonido al videojuego, que fue algo que no realizamos por una cuestión de no extender más esta nota. Las posibilidades son infinitas, sólo nos resta perfeccionarlo al nivel que deseemos.

El inicio de nuestro videojuego ya terminado, con una portada de Lujo, que le da profesionalismo a nuestro trabajo



¡Y la acción comenzó! Ahora a disfrutar de nuestra propia creación y a distribuirla entre amigos y conocidos para poder perfeccionarla en base a las sugerencias de terceros.



Tendencia: Los secretos para diseños limpios

Por Roxana Miguel
roxana.miguel@pixelscode.com
@roxanamiguel

Un website hoy no requiere de demasiados artilugios que llamen la atención, sino más bien de la selección de unas pocas y sencillas funciones, acompañadas por un diseño simple. Aquí conoceremos algunos tips de tipografías, imágenes y colores.

Cuando tener una página web se convirtió en un poco más que una moda, sino una necesidad para que las compañías ganen su terreno en la red, podíamos incursionar en el mundo de las doble v y descubrir a cada clic cientos de sitios llamativos, con botoneras ruidosas, menúes coloridos y hasta logos con animaciones que se presentaban como intro. Lo cierto es que tanta alharaca terminaba por ser molesta para los visitantes que, según la conexión con la que contaran, podían esperar un buen rato para finalmente entrar a la página.

Algunos sitios no podían descargarse correctamente, o resultaban tan

pesados en cantidad de información en imágenes que el usuario se distraía esperando el entretenimiento del movimiento antes de lograr entender cuál era el contenido real de aquel sitio. Desde aquel momento hasta hoy han pasado muchas cosas en el mundillo de los diseñadores web, una y quizás la más importante es descubrir que el circo de funciones no es justamente lo más funcional.

Podría decirse que actualmente la tendencia pone sobre el tapete lo que se empieza a conocer como diseños limpios, y esto no es otra cosa que sitios simples, con contenidos objetivos y mensajes directos. La página no ofrece una intoxicación de información y predominan los espacios neutros con colores definidos, primordialmente, el blanco. De esta manera sitios como Apple, Dattatec o Macromedia, por nombrar sólo algunos, pueden ser referentes de esta forma de pensar la presentación de una empresa en la web.

Quienes se resisten a esta tendencia, que muchas veces es el mimo cliente o también diseñadores, la señalan como vaga, parecida a los blogs y poco atractiva. Lo cierto, y sin ánimo de subestimar sus opiniones, es que las claves de los diseños limpios conlleva un estudio del mensaje, la tipografía y la gama de color clara, las imágenes no pixeladas, la jerarquía visual y el uso coherente de los espacios. Esto nada tiene que ver con las plantillas predeterminadas que pudieran ofrecer los blogs.

LAS MALAS INTERPRETACIONES DEL DISEÑO LIMPIO

Para aprender los principios de esta tendencia es necesario quitar de nuestro imaginario algunos conceptos que pueden generar confusiones:

Blanco: Aunque veremos que predomina el uso de los espacios blancos, esto no significa que sea un sitio limpio. Como hemos dicho anteriormente, hay que utilizar los espacios con una lógica que no siempre evoca un mismo color, de la misma forma tampoco nos debe intranquilizar un diseño donde predominan los campos vacíos, siempre y cuando esto tenga que ver con lo que la página quiera comunicar. Las páginas de Apple, que elige una base blanca, y la base gris oscura de Macromedia son buenos ejemplos.

Intuitiva: Pon a un usuario frente al site que estás diseñando sólo durante cinco segundos y pídele que te explique de qué trata aquella página. Algunos entienden por diseño limpio una portada vacía de imágenes ruidosas, pues no, el foco tiene que ser claro y contener un punto referencial que indique al usuario de qué se trata la página, sin que este tenga que investigar, pudiendo usar para ello fotografías u objetos gráficos.

Elementos: Algunos nos emocionamos demasiado al darle a un sitio toques de distinción con sombras en los textos o efectos de neón. Hacer un diseño limpio no implica eliminar estos “gustitos” sino que coexistan sin que terminen por ser difusos o entorpezcan el entendimiento del concepto. Con tipo-

> *Podría decirse que actualmente la tendencia pone sobre el tapete lo que se empieza a conocer como diseños limpios, y esto no es otra cosa que sitios simples, con contenidos objetivos y mensajes directos.”*

grafías claras y líneas definidas se pueden conservar estos recursos sin sobrecargar la página.

Innovación: Limpio significa que no tiene mezcla de otra cosa y que está despojado de lo superfluo, accesorio o inútil. Para nada significa ser poco creativo e innovador. Los sitios claros y sencillos pueden ser más que innovadores con las elecciones correctas.

Vistos algunos parámetros que llevan a la confusión, es hora de ponernos a trabajar en un sitio limpio para lo cual recurriremos a una guía que, como todo lo que tiene que ver con la web, es vulnerable a los cambios constantes de tendencias y hasta propuestas individuales. Sólo es referencial y está basada en los ejemplos que utilizaremos para dar más luz sobre esta tendencia.

LAS CLAVES DEL DISEÑO LIMPIO

Lo que quiero que se entienda, se tiene que comprender a simple vista. Esta debiera ser la premisa ante un diseño limpio y para ello nos basaremos en los recursos que los diseñadores webs tenemos sobre nuestra mesa de trabajo:

Tipografías: Si bien el mundo de las letras está lleno de novedades y todas son muy tentadoras para darle al sitio un estilo único, lo ideal es recurrir a lo clásico. Fuentes como Bodoni, Frutiger, Univers, Helvética o Franklin Gothic son comunes aunque no tanto como Times New Roman y permiten generar cuadros de texto claros y estéticamente atractivos. Por otro lado, existe una bolsa de hermosas fuentes finas; algunas hay que pagar para descargarlas, pero otras no. Lo bueno es que por su formato no se trata de tipografías que rompan con el equilibrio que buscamos, algunos ejemplos son: XXII Menga, Circula Medium, Gnuolane Regular, Expressway Regular, Raleway, Balham, Sertig, Edelsan, Candella, Acid, Circle, Tertle, Cantarell, Titillium, Hattori Hanzo, Marge, Mr Jones y muchos más. Ponlas a prueba sobre bases de colores diferentes y verás que no necesariamente hay que caer en los fondos blancos para crear un diseño limpio.

Espacio: Hay un elemento fundamental en una página: el aire. Como ya hemos visto, el uso del mismo genera contradicciones, pero veremos cómo es utilizarlo con coherencia. Ignorar su existencia es un gran error, puesto que las páginas deben contener elementos ubicados adecuadamente, dejando aire entre ellos para que cada uno resalte su significado. Mientras

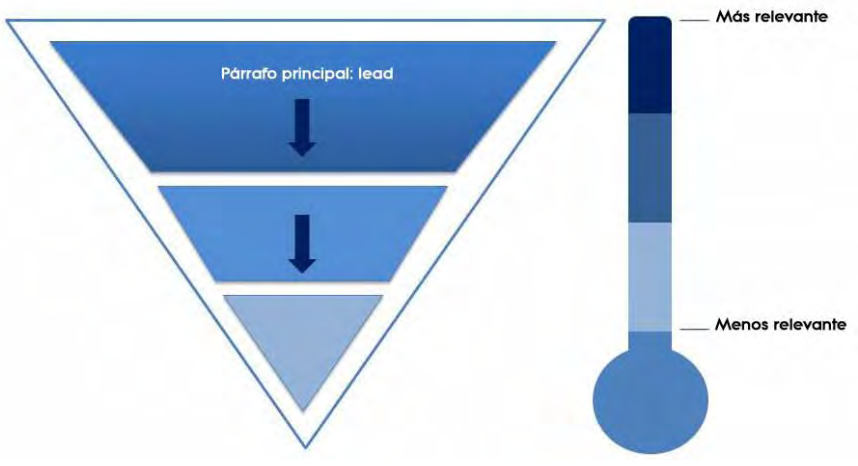
que Apple hace un uso más marcado del aire entre textos e imágenes, en el caso de Macromedia veremos que cada espacio cuenta con su propio ambiente, generando una división entre propuesta y propuesta que permite la convivencia entre varios elementos, pero igualmente en un espacio armónico.

Pirámide invertida: No se trata de hacer una web geométrica, sino de ubicar los elementos correctamente. Esto tiene que ver con el orden en que normalmente se lee un sitio. Lo más importante se coloca arriba y en un tamaño más destacado, lo menos importante irá en orden decreciente abajo y cada vez con menor tamaño. Algunos prefieren ubicar en el centro lo más importante suponiendo que es lo primero que se ve, bastaría con hacer un pequeño ejercicio: cuando vemos una imagen llamativa en el centro de un site, ¿hacia dónde dirigimos la mirada para entender de qué se trata?

> ***"Sitios como Apple, Dattatec o Macromedia, por nombrar sólo algunos, pueden ser referentes de esta forma de pensar la presentación de una empresa en la web."***

LA IMAGEN NO ES TODO

En cambio la primera impresión sí lo es, y en la web lo que llega al ojo humano es una imagen compuesta por color, texto y dibujo. Por ello en un diseño limpio, si vamos a utilizar una fotografía o una secuencia de fotos debemos procurar, aunque esto parezca una obviedad, que esté trabajada en la mejor resolución posible para web sin compresiones que compliquen su visualización al estar online.



Siempre tenemos que tener presente para qué tipo de usuario estamos creando el site; es posible entonces que la abundancia de imágenes sea inevitable para lo que debemos transmitir. En este sentido lo aconsejable es poner en orden el material de trabajo y proponernos un método de exposición que no sea invasivo.

En estos casos, se pueden crear archivos de flash secuenciales con una transición suave. Actualmente, la tecnología web nos ofrece otras opciones de desarrollo con imagen por medio de capas. En sitios como el de Papyros Digitales o TBM vemos buenos ejemplos de ello donde no hay un exceso de exposiciones fotográficas. Un punteado que indica cuántas imágenes hay para ver será propicio para no aburrir al usuario. El paseo fotográfico es armonioso y apoya el objetivo del sitio.

Cuando las fotos que nuestro cliente nos proporciona para la página son extremadamente coloridas, es conveniente trabajar con una base de color neutro que permita el aire que recomendábamos anteriormente. La tipografía fina también es aconsejable, siempre utilizando tamaños más grandes en la parte superior de modo que el usuario no espere encontrar sólo en las imágenes las respuestas al sitio.
Siempre que sea necesario contar con elementos como videos, música o sonido, es recomendable ofrecerle

> *Podría decirse que actualmente la tendencia pone sobre el tapete lo que se empieza a conocer como diseños limpios, y esto no es otra cosa que sitios simples, con contenidos objetivos y mensajes directos.”*

al usuario la posibilidad de silenciarlos. No existe nada más molesto que estar de visita en un lugar donde los ruidos nos aturden, por lo general, esperamos el momento de irnos. Frente a una web, la puerta siempre estará abierta para marcharnos y no volver más.

Si podemos evitar la combinación de una secuencia fotográfica con gráficos y animaciones, todo al mismo tiempo, los usuarios agradecerán con su visita. Resulta demasiado cargoso cuando una página cuenta con una mixtura de dinamismos que terminan por ser confusos y lo que logran finalmente es distraer a los internautas. Más adelante veremos ejemplos de cómo trabajar estos elementos combinados sin entorpecer el website.

Una vez que sabemos quién es nuestro cliente, qué tenemos para contar y quiénes son nuestros visitantes, entonces lo ideal es definir con pocas palabras lo que tenemos para mostrar.

BUENOS EJEMPLOS

En algún momento fue muy común ver sitios cuyos menúes superiores diseñados con algún efecto de sonido o movimiento, estaban repetidos en forma estática en la parte inferior para los que no podían ver lo superior. En aquellos casos, siempre lo de abajo resultaba en una letra demasiado chica y antiestética respecto al resto del diseño. La pregunta obligada es, ¿para qué contar dos veces lo que se puede decir de una sola vez?

Quizás no nos animábamos a usar esa tipografía simple para una botonera superior o todavía no contábamos con algunos recursos web que con el tiempo se fueron perfeccionando; hoy los diseños limpios nos obligan a ser claros. InfinVision es un ejemplo de diseño web limpio donde con una tipografía tan simple como Serif, un fondo blanco vejo y una gráfica también simple, se genera un entorno web sencillo, claro y conciso.

Image Mechanics es otro ejemplo de que se puede diseñar un sitio interactivo, elegante y limpio, con tanta información como aire. El cliente desarrolla aplicaciones móviles y ofrece servicios de diseño por lo que está en su propia salsa.

> *"Sitios como Apple, Dattatec o Macromedia, por nombrar sólo algunos, pueden ser referentes de esta forma de pensar la presentación de una empresa en la web."*

Al entrar al site está toda la información en una misma interfaz, el color blanco, letras destacadas en negro y un distinguido logo generan una identidad.

Una forma de dejar en claro que muchas de las mañas que los diseñadores tenemos no hay que encarpetarlas en el olvido es echar un vistazo por el site Two Designers; sólo hay que utilizar una coherencia que permita la claridad, palabra que tanto usamos en este artículo. En este caso el sitio cuenta con efectos gráficos simples, un fondo con relieve de puntos, dinamismo en los cuadros de imagen y mezcla de tipografías. Cualquiera diría que esto no es un diseño limpio, sin embargo la distribución de los espacios dan el suficiente aire como para generar un ecosistema elegante y claro.

LOS CAMBIOS DE LA TENDENCIA

Como podremos ver en los ejemplos citados y a lo largo del artículo, los diseños limpios utilizan algunas tecnologías que últimamente están muy en boga, dejando poco espacio para lo que antes se resolvía utilizado Flash, ahora hay más lugar para HTML5 y esto tiene que ver con la optimización del site para lograr objetivos SEO. Por otra parte, las soluciones gráficas se pueden resolver con CSS3.

La paleta de colores no debe intimidar al diseñador que empieza a incursionar en el diseño limpio; tanto los blancos, como negros son muy útiles para lograr estos resultados, sin embargo, hay que animarse a otros colores que sorprenderán, como también así jugar con los colores de las fotografías o gráficas que nos proporcione el cliente.

El todo que compone un diseño terminará por ser limpio cuando el objetivo esté claro y el internauta que visite la página entienda en pocos segundos cuál es la información que está buscando. No se trata de una tarea sencilla puesto que dependeremos de con cuánta información contamos para armar el diseño y cómo distribuirla. Afortunadamente, los recursos web hoy nos permiten jugar sin perder creatividad, por lo que el rol del diseñador web emprende un camino desafiante: poner la mente en blanco y con cada diseño empezar varias veces de cero hasta lograr un resultado limpio.

Desarrollo de juegos con HTML5 Segunda parte

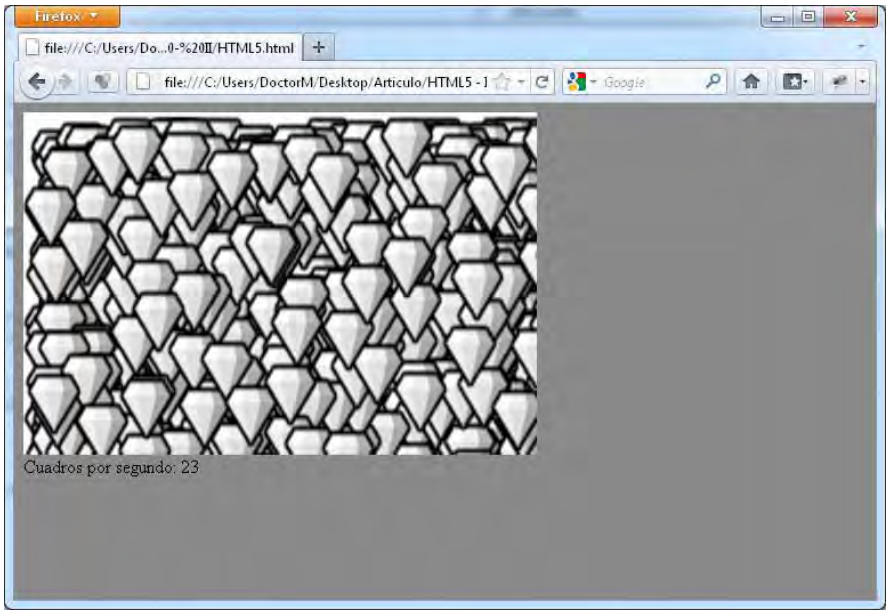
Por Matías Iacono

matias.iacono@pixelscode.com

En la primera parte de esta serie de artículos, pudimos sentar las bases para la creación de juegos con HTML5 y JavaScript. Utilizando una imagen, un objeto canvas y algo de código, interactuamos con la misma, mostrándola en el navegador. Finalmente, creamos un rudimentario sistema para el manejo de las teclas con las que el usuario interactuaría y movimos la imagen dentro del contenedor. En esta entrega aprenderemos a crear animaciones para nuestros personajes, el concepto de buffer doble y cómo detectar colisiones entre los objetos.

BUFFER DOBLE

El concepto del buffer doble es muy conocido y utilizado en el desarrollo de video juegos, incluso, en el desarrollo de aplicaciones que requieran



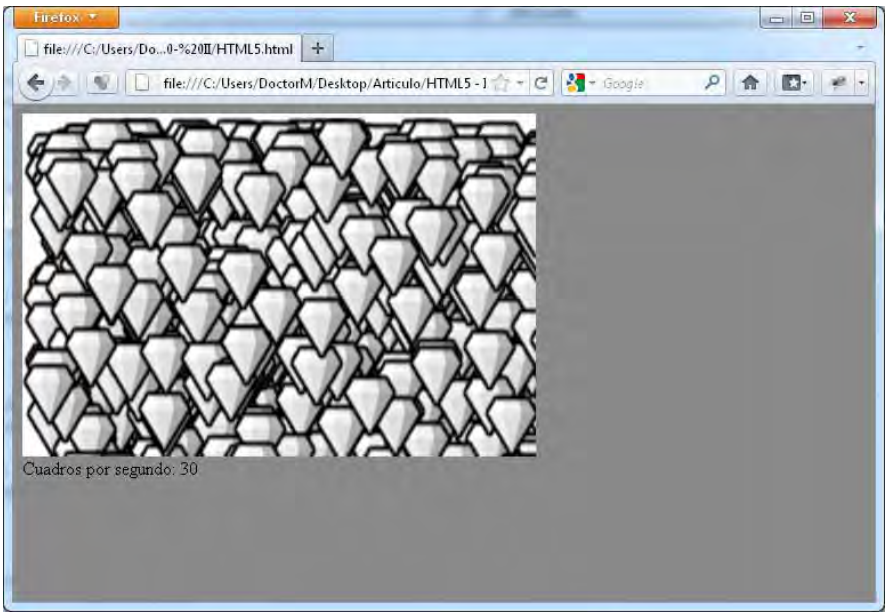
En esta imagen podemos ver los cuadros por segundo en base a un muestreo de 3000 imágenes.

gran procesamiento de gráficos, pero, antes de hablar sobre esta técnica necesitamos entender por qué es necesaria.

Algo que debemos tener bien claro es que la capacidad de mostrar imágenes rápidamente en pantalla no es algo que se le dé muy bien a los ordenadores. Si bien la potencia actual de los mismos nos permite tener cada vez más juegos con gráficos espectaculares, muchos de estos gráficos, o mejor dicho, mucha de la potencia de cómputos está enfocada en calcular cómo se deben ver más que el hecho de mostrar la imagen propiamente dicha en pantalla. Así podemos tener juegos casi realistas, con texturas perfectas y movimientos de personajes similares a los de una persona, pero esto nada tiene que ver, en sí mismo, con la capacidad de mostrar todo este cálculo en la pantalla.

Mientras que estamos creando los gráficos y enviando los mismos a la pantalla, otros procesos son ejecutados, incluidos los de sincronización entre el monitor y el hardware de la computadora. Estos procesos exceden la potencia de cómputos de nuestro ordenador y solemos encontrarnos con un cuello de botella. De cualquier manera, no es a donde queremos llegar, por lo menos hoy.

En todo caso, pensemos que un juego necesita poder enviar muchos cuadros de imágenes a la pantalla en un lapso de un segundo, por lo tanto, necesitamos que el envío de estas imágenes se haga rápidamente. Ahora pensemos que al usar el lienzo de dibujo en el artículo anterior, estábamos enviando, cada vez que ocurría una actualización, la imagen directamente hacia este objeto, y este objeto directamente a la pantalla. Posiblemente no veríamos ningún problema ya que la imagen es mostrada de forma correcta, incluso cuando se desplazaba mediante el presionado de las teclas. Pero, a medida que fuéramos agregando nuevas imágenes al lienzo, el proceso de copiado de la imagen hacia la pantalla sería cada vez mayor, y por lo tanto impactaría directamente en el rendimiento de este proceso.



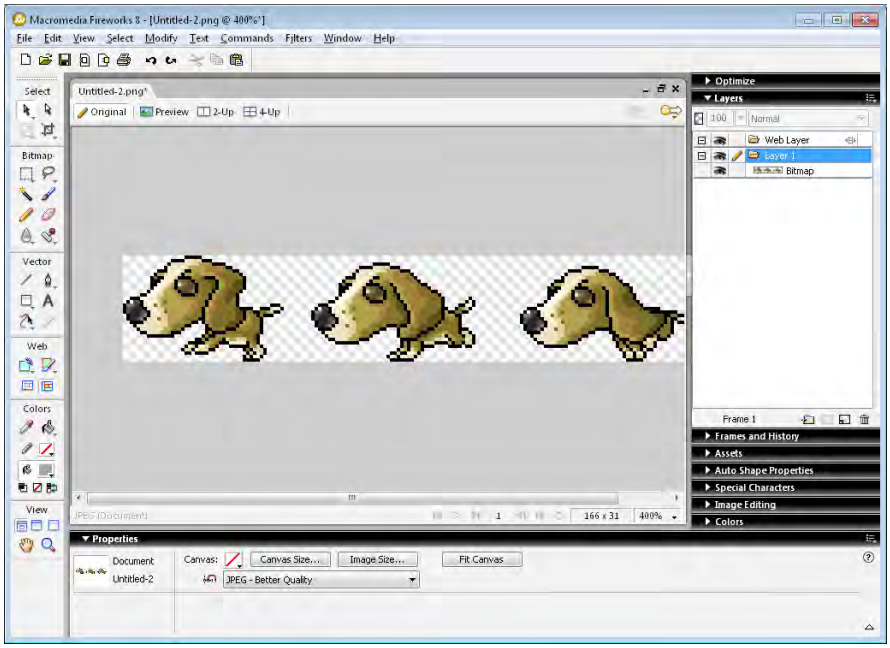
Al implementar el buffer doble vemos un aumento significativo en el rendimiento del juego aumentado la cantidad de cuadros por segundo de muestreo.

Si el problema aun no nos parece obvio, no nos preocupemos, suele tomar algo de tiempo entenderlo. Para ejemplificarlo de forma que quede un poco más claro (y simplificado), digamos que para dibujar 10 imágenes en un nuestro lienzo pretendemos hacerlo en un lapso de 41 milisegundos (esto nos daría unos aproximados 24 cuadros por segundo), por lo tanto, tenemos esta ventana de tiempo para recorrer todos los objetos a dibujar y enviar cada uno de ellos al lienzo para que sean mostrados en la pantalla. Pero el problema es que el proceso de copiado de esa imagen hacia el lienzo requiere de un tiempo, además, cada vez que estamos copiando uno de ellos, como está siendo copiado directamente a la pantalla requerirá de procesos y tiempo adicional para gestionar esto. Digamos que en esa ventana de tiempo estaríamos ordenándole al lienzo (y a la pantalla) que tome cada pixel de la primera imagen y la dibuje pixel a pixel en la pantalla, luego, que haga lo mismo con la segunda, y la tercera, y la cuarta hasta llegar a la última. Esto requeriría de una capacidad gigantesca de procesamiento y coordinación para poder, por ejemplo, sincronizar todo esto y que al final quede una imagen final en la pantalla. Por lo tanto, mientras más imágenes agreguemos, la ventana de 46 milisegundos podría verse sobrepasada, resultando en una pérdida de cuadros por segundo para generar la animación de 24 cuadros mínimos.

Si bien la lógica para mostrar las 3000 imágenes es rudimentaria y básica, podemos notar una disminución en los cuadros por segundo. En este caso, habíamos configurado el redibujado con un promedio de 30 cuadros por segundo. Tengamos en cuenta que no se está realizando ningún cálculo complejo del objeto a dibujar. En juegos avanzados el promedio de cuadros por segundo se vería aun más disminuido. Veamos cómo hemos creado este simple medido de cuadros por segundos.

```
<canvas id="lienzo"
style="width:450px;
```

```
height:300px; background-
color:white"></canvas><br
/>
Cuadros por segundo:
<span id="frames"></span>
...
```



> *"El concepto del buffer doble es muy conocido y utilizado en el desarrollo de video juegos, incluso, en el desarrollo de aplicaciones que requieran gran procesamiento de gráficos."*

Mediante diferentes programas de edición de imágenes podemos crear nuestras líneas de animación para los juegos.

```
...
<script
language="javascript">
    var contador = 0;
    var fra-
mes = document.
getElementById("frames");
    ...
    ...
```

```
    setInterval(Dibujar,
1000 / 30);
    setInterval(Frames,
1000);
    ...
    ...
    function Frames() {
        frames.innerHTML
= contador;
        contador = 0;
    }
    ...
    ...
    function Dibujar() {
        contador++;
        context.fillStyle
= "rgb(255,255,255)";
        context.fillRect
(0, 0, canvas.width, can-
vas.height);
```

```
        for (var i = 0; i
< 3000; i++) {
            context.
drawImage(imagen, Math.
floor(Math.random() *
450),
                Math.
floor(Math.random() *
300), imagen.width, ima-
gen.height);
        }
    }
```

Ahora que podemos ver los problemas de rendimiento que presenta este modelo podremos entender el concepto de buffer doble.
El principal problema radica, entonces, en el envío de las imágenes de forma directa para su visualización en pantalla, por lo tanto, deberemos buscar un mecanismo que nos permita manejar la misma cantidad de

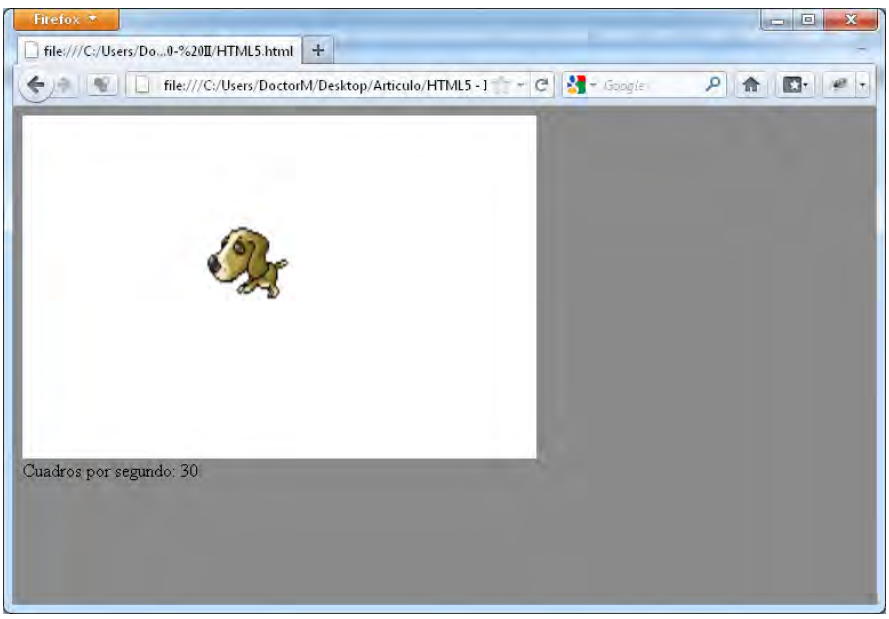
> *Una de las técnicas más comunes para obtener animaciones en los juegos es la de tener una línea con cada uno de los cuadros de la animación. Esto es, una única imagen con la animación cuadro a cuadro separada por espacios iguales."*

elementos en pantalla, pero con la menor cantidad de envío de información hacia la misma. El buffer doble, entonces, intenta imitar el comportamiento del lienzo de dibujo pero en la memoria de la computadora, de esta forma todas las imágenes son dibujadas en este lienzo pero sin tener que viajar hacia la salida de video, reservando esta acción al momento en que todos los objetos del juego han sido recreados y actualizados. De esta forma, obtenemos una única imagen con todos los objetos sobre ella y copiamos la misma, una única vez, al lienzo que tiene contacto con la pantalla.

Podemos ver en la imagen anterior que los cuadros por segundo aumentan significativamente. Entonces, para implementar este segundo lienzo en memoria podemos escribir el siguiente código.

```
var backBuffer = document.createElement("canvas");
backBuffer.width = canvas.width;
backBuffer.height = canvas.height;
var backBufferContext2D = backBuffer.getContext("2d");
```

Por un lado, desde JavaScript creamos un nuevo elemento HTML del tipo canvas. Este elemento es creado en memoria y no físicamente en la pantalla. Para dimensionarlo usamos los valores del lienzo original, el que sí está en la pantalla. Por último, creamos un contexto de dibujo para este lienzo. El siguiente paso es remplazar la función de dibujado por una que lo haga sobre el buffer en memoria, para luego volcar todo el resultado al lienzo final.



Cuando presionamos las teclas para mover el personaje, además de desplazarse por la pantalla se genera una animación cuadro a cuadro.


```
backBufferCon-
text2D.fillStyle =
"rgb(255,255,255)";
backBufferContext2D.
fillRect(0, 0, backBu-
ffer.width, backBuffer.
height);
```

```
for (var i = 0; i < 3000;
i++) {
      backBufferCon-
text2D.drawImage(imagen,
Math.floor(Math.random() *
450),
                Math.
floor(Math.random() *
300), imagen.width, ima-
gen.height);
}
context.
drawImage(backBuffer, 0,
0);
```

La última línea es la que vuelca todo el contenido al lienzo en pantalla. De cualquier manera, esta técnica no es perfecta ya que el buffer doble que se aloja en memoria ocupará tanta memoria como el lienzo que tenemos para visualizar las imágenes en su forma final, por lo que el consumo de memoria aumentará significativamente. Posiblemente un precio que vale pagar a costa de un mejor rendimiento en la representación de las imágenes.

ANIMACIONES

Son pocos los juegos en los cuales los elementos del mismo no contienen aunque sea un cuadro de animación. Incluso aquellos puzles clásicos contenían más de un gráfico para las diferentes partes del juego. Y ver que nuestro personaje o nave se mueve no sólo por la pantalla, sino que además puede cambiar su forma mientras lo hace es parte del atractivo en los videojuegos.
Una de las técnicas más comunes para obtener animaciones en los juegos es la de tener una línea con cada uno de los cuadros de la animación. Esto es, una única imagen con la animación cuadro a cuadro separada por espacios iguales.

Cuando dibujamos una imagen en un lienzo no sólo podemos especificar el lugar donde será visualizada, sino que también po-

> *Todas las imágenes, por más que veamos sólo su contenido, son representaciones rectangulares, por lo tanto, una forma de detectar colisiones entre dos de estas imágenes es mediante este rectángulo y el momento en que ambos se superponen.”*

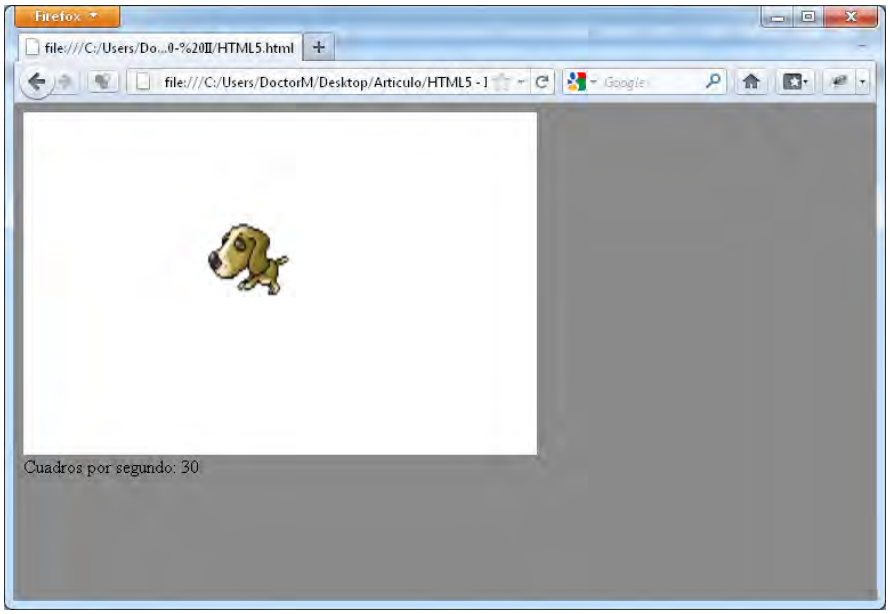
dremos seleccionar de la imagen original qué parte de esta es la que queremos mostrar. Por lo tanto, si tenemos nuestra línea con los cuadros de la animación sólo deberemos ir saltando entre fragmentos de la misma tras cada cuadro de animación.

```
function Dibujar() {
    contador++;
    backBufferCon-
text2D.fillStyle =
“rgb(255,255,255)”;
    backBufferContext2D.
fillRect(0, 0, backBu-
ffer.width, backBuffer.
height);
    backBufferContext2D.
drawImage(perro, cuadro *
50, 0, 50, 31, x, y, 50,
31);
    context.
drawImage(backBuffer, 0,
0);
}
```

El método drawImage entonces, recibe la imagen de nuestra línea de cuadros. Cada cuadro está separado por 50 píxeles de distancia, por lo que mediante un contador de cuadros en base a las teclas presionadas desplazamos el punto inicial de recorte a ese lugar (cuadro * 50), luego definimos el tamaño de la imagen que en este caso será de 50 x 31 píxeles, y finalmente el destino de la misma en el lienzo mediante las coordenadas X e Y.

A estas animaciones simples podemos agregarle algo de complejidad, por ejemplo, intercambiando entre cuadro y cuadro sólo cuando transcurra cierta cantidad de tiempo; esto generaría un movimiento más suave. Todos los demás movimientos, como saltos, disparos, y un largo etcétera, dependerán de los gráficos con los que dispongamos, aunque la lógica para animarlos pueda ser potencialmente la misma.

A pesar de que los dos dibujos no se tocan realmente, sí lo hacen sus contenedores rectangulares, por lo tanto, podemos ver que la validación es verdadera.



DETECCIÓN DE COLISIONES

Existen muchas formas de detectar colisiones entre objetos de un juego. La detección de las colisiones nos resultan útiles para otorgarle mayor dinamismo al juego. Por ejemplo, podríamos contar con una plataforma donde nuestro personaje deba pararse y no caer al vacío, o si un misil golpea nuestra nave, esta debe explotar, hasta la posibilidad de recoger monedas para obtener puntos. En cualquier caso, la detección de colisiones se puede realizar de varias maneras, desde las más complejas y precisas hasta simples y rápidas. Nosotros nos enfocaremos en una técnica, simple y rápida, aunque no de lo más precisa.

Todas las imágenes, por más que veamos sólo su contenido, son representaciones rectangulares, por lo tanto, una forma de detectar colisiones entre dos de estas imágenes es mediante este rectángulo y el momento en que ambos se superponen.

```
function Colisionan() {
    if (x + 50 < 10)
        return false;
    if (y + 31 < 40)
        return false;
    if (x > 10 + 15)
        return false;
    if (y > 40 + 15)
        return false;
    return true;
}
```

Esta función sólo puede validar la colisión entre dos elementos. Por un lado, nuestro personaje, y por otro, una gema colocada en una posición estática dentro del juego. Por supuesto, esto puede ser modificado para que su funcionamiento sea genérico, pero sirve para que podamos entender cómo detectar los márgenes de cada objeto. Dentro de la función que muestra las imágenes sólo deberemos llamar a esta otra para saber si los dos objetos se tocan.

```
function Dibujar() {
    ...
    ...
    backBufferContext2D.
drawImage(perro, cuadro *
50, 0, 50, 31, x, y, 50,
31);
    backBufferContext2D.
drawImage(imagen, 10, 40,
15, 15);
    if (Colisionan()) {
        colision.innerHT-
ML = "Sí";
    } else {
        colision.innerHT-
ML = "No";
    }
    ...
    ...
}
```

Otro modelo muy utilizado en juegos de dos dimensiones es el validar las colisiones mediante los píxeles de cada una de las imágenes. Esto permite una mayor precisión en la detección de estas colisiones, pero también suelen ser más costosas a nivel de procesamiento de datos. De cualquier manera, y como hemos visto en este artículo, a veces debemos sacrificar algo de memoria o poder de procesamiento para un mejor resultado, como lo hicimos con el buffer doble.

Algoritmos: Gnome Sort

Por Juan Gutmann

juan.gutmann@pixelscode.com

Siempre es bueno volver a las fuentes. Hace siete ediciones atrás inauguramos esta sección con el algoritmo de ordenamiento de burbuja (Bubble Sort), uno de los más sencillos que existen. Decíamos entonces que aunque debido a su baja eficiencia es muy poco usado, vale la pena estudiarlo por su importancia académica. En la segunda entrega cubrimos el ordenamiento por Inserción (Insertion Sort). En esta ocasión, vamos a ver un algoritmo menos conocido, por ser relativamente nuevo, que se basa en estos dos métodos "clásicos": Gnome Sort. Resulta interesante ver cómo un algoritmo poco eficaz, como el Bubble Sort, se puede transformar en una rutina más poderosa con unos pocos cambios conceptuales. Procedamos a analizar su funcionamiento.

METODOLOGÍA DE GNOME SORT

Este algoritmo debe su nombre a los gnomos de jardín, más conocidos entre nosotros como los tradicionales "enanitos de jardín", esculturas que suelen adornar pequeños parques hogareños. En muchos países europeos, de donde proviene la mitología de la que provienen elfos y gnomos, sostienen medio en broma, medio en serio, que estas criaturas cobran vida por la noche y se dedican a

> *Al comenzar, se ubican ante la segunda maceta, y comparan la cantidad de flores de esa maceta con las de la anterior. Si esa maceta contiene menos flores que la anterior, invierten la ubicación de ambas. Luego pasa a la siguiente maceta, y repite el procedimiento hasta estar ubicado ante la última, donde habrá terminado su trabajo."*

trabajar en el jardín, manteniéndolo cuidado y prolijo. Según uno de los dos expertos a los que se atribuye este algoritmo, el académico holandés Dick Grune -creador del archifamoso sistema de control de versiones CVS-, en su país una de las tareas nocturnas de los gnomos de jardín es mantener en orden las macetas con flores. Por cierto, como la mayoría de los europeos, estos gnomos son extremadamente metódicos, y siguen siempre el mismo sistema. Al comenzar, se ubican ante la segunda maceta, y comparan la cantidad de flores de esa maceta con las de la anterior. Si esa maceta contiene menos flores que la anterior, invierten la ubicación de ambas. Luego pasa a la siguiente maceta, y repite el procedimiento hasta estar ubicado ante la última, donde habrá terminado su trabajo. Con esta breve y pintoresca descripción puede inferirse que al igual que Bubble Sort e Insertion Sort, Gnome Sort pertenece a la familia de algoritmos que trabajan "por comparación", y aunque se recorre la lista en forma similar al ordenamiento por inserción, el "intercambio" de a pares de valores de la lista es idéntico al principal concepto en el que se basa el ordenamiento de burbuja. A continuación, el pseudocódigo de Gnome Sort:

```
Dada una lista L, de N elementos
I = 1
mientras I < N
    si L(I) >= L(I - 1)
        I = I + 1
    si no
        valor = L(I)
```

Una implementación de Gnome Sort en lenguaje Java

> *"Tanto los métodos de inserción como de burbujeo -y por ende también Gnome Sort- ofrecen tiempos de respuesta muy pobres ante listas con una gran cantidad de elementos."*

```
        L(I) = L(I - 1)
        L(I - 1) = valor

        si I > 1
            I = I - 1
        si no
            I = I + 1
        fin si
    fin si
fin mientras
```

Como vemos, la metodología es muy breve y simple tanto de comprender como de implementar en cualquier lenguaje, características compartidas por todos los algoritmos basados en Bubble Sort. Tal como es costumbre, les presentamos ahora una implementación en Python, escrita respetando el flujo y estilo del pseudocódigo, con el habitual agregado de impresión a standard output de los pasos realizados por el programa en cada iteración.

```
# -*- coding: cp1252 -*-
def gnomesort(L):
```

```
public int[] gnomeSort(int[] x){
    int[] y = x;
    for(int i = 1, j = 2; i < y.length; i++){
        if(y[i-1] <= y[i]){
            i = j;
            j++;
            }
        else{
            swap(y[i-1], y[i]);
            i--;
            if(i == 0){
                i = j;
                j++;
                }
            }
        }
    return y;
}

public void swap(int[] x, i, j){
    int temp = x[i];
    x[i] = x[j];
    x[j] = temp;
```

```
    N = len(L)
    I = 1
    itera = 0

    while I < N:
        itera = itera + 1

        if L[I] >= L[I - 1]:
            print L, "iteración =", itera, "I =", I, "Sin cambios"
            I = I + 1
        else:
            print L, "iteración =", itera, "I =", I, "Intercambio de", L[I], "y",
L[I - 1]
            valor = L[I]
            L[I] = L[I - 1]
            L[I - 1] = valor

            if I > 1:
                I = I - 1
            else:
                I = I + 1

    return L

Z = [1, 4, 8, 2, 5, 3, 9, 7, 10, 6]
print gnomesort(Z)
```

A continuación, la salida por pantalla con la lista de prueba:

```
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 1 I = 1 Sin cambios
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 2 I = 2 Sin cambios
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 3 I = 3 Intercambio de 2 y 8
[1, 4, 2, 8, 5, 3, 9, 7, 10, 6] iteración = 4 I = 2 Intercambio de 2 y 4
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 5 I = 1 Sin cambios
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 6 I = 2 Sin cambios
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 7 I = 3 Sin cambios
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 8 I = 4 Intercambio de 5 y 8
[1, 2, 4, 5, 8, 3, 9, 7, 10, 6] iteración = 9 I = 3 Sin cambios
[1, 2, 4, 5, 8, 3, 9, 7, 10, 6] iteración = 10 I = 4 Sin cambios
[1, 2, 4, 5, 8, 3, 9, 7, 10, 6] iteración = 11 I = 5 Intercambio de 3 y 8
[1, 2, 4, 5, 3, 8, 9, 7, 10, 6] iteración = 12 I = 4 Intercambio de 3 y 5
[1, 2, 4, 3, 5, 8, 9, 7, 10, 6] iteración = 13 I = 3 Intercambio de 3 y 4
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 14 I = 2 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 15 I = 3 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 16 I = 4 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 17 I = 5 Sin cambios
```

```
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 18 I = 6 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 19 I = 7 Intercambio de 7 y 9
[1, 2, 3, 4, 5, 8, 7, 9, 10, 6] iteración = 20 I = 6 Intercambio de 7 y 8
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 21 I = 5 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 22 I = 6 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 23 I = 7 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 24 I = 8 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 25 I = 9 Intercambio de 6 y 10
[1, 2, 3, 4, 5, 7, 8, 9, 6, 10] iteración = 26 I = 8 Intercambio de 6 y 9
[1, 2, 3, 4, 5, 7, 8, 6, 9, 10] iteración = 27 I = 7 Intercambio de 6 y 8
[1, 2, 3, 4, 5, 7, 6, 8, 9, 10] iteración = 28 I = 6 Intercambio de 6 y 7
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 29 I = 5 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 30 I = 6 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 31 I = 7 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 32 I = 8 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 33 I = 9 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

La cantidad total de iteraciones nos permite ver rápidamente que Gnome Sort supera claramente a Bubble Sort. Esta versión resuelve el ordenamiento en 33 iteraciones, contra 81 que le llevaría la misma lista al método de Burbuja sin optimizar. Incluso aplicándole a Bubble Sort las dos optimizaciones sugeridas cuando lo analizamos, sigue perdiendo la batalla contra el sistema que nos ocupa en esta ocasión, ya que incorporándole tanto la reducción de iteraciones internas tras cada intercambio como la detección de "lista ya ordenada" y el consiguiente corte de ejecución, la cantidad total de iteraciones baja a 35, dos más que las realizadas por esta versión no optimizada de Gnome.

OPTIMIZACIÓN

Una mejora sencilla de implementar pero con un gran impacto sobre la velocidad de ejecución del algoritmo consiste en almacenar en una variable la posición en la que se realizó el último intercambio, saltando directamente a esta posición al recorrer nuevamente la lista. De esta forma, al llegar a la última maceta, el gnomo se "teletransporta" hacia la próxima maceta que resta ordenar, omitiendo recorrer nuevamente las que ya han sido ordenadas (iteraciones sin intercambios). Esto reduce considerablemente el costo de ejecución del algoritmo. Incorporando esta optimización, el pseudocódigo quedaría así:

```
Dada una lista L, de N elementos
I = 1
X = 0
mientras I < N
    si L(I) >= L(I - 1)
        si X <> 0
            I = X
            X = 0
        fin si

        I = I + 1
```

```
    si no
        valor = L(I)
        L(I) = L(I - 1)
        L(I - 1) = valor

        si I > 1
            si X = 0
                X = I
            fin si

            I = I - 1
        si no
            I = I + 1
        fin si
    fin si
fin mientras
```

El académico holandés Dick Grune es el principal difusor de este método de ordenamiento.

Y la correspondiente implementación en Python:

```
# -*- coding: cp1252 -*-
def gnomesort(L):
    N = len(L)
    I = 1
    X = 0
    itera = 0

    while I < N:
        itera = itera + 1

        if L[I] >= L[I - 1]:
            print L, "iteración =", itera, "I =", I, "X =", X, "Sin cambios"
            if X <> 0:
                I = X
                X = 0

            I = I + 1
        else:
            print L, "iteración =", itera, "I =", I, "X =", X, "Intercambio de",
L[I], "y", L[I - 1]
            valor = L[I]
            L[I] = L[I - 1]
            L[I - 1] = valor

            if I > 1:
                if X == 0:
```

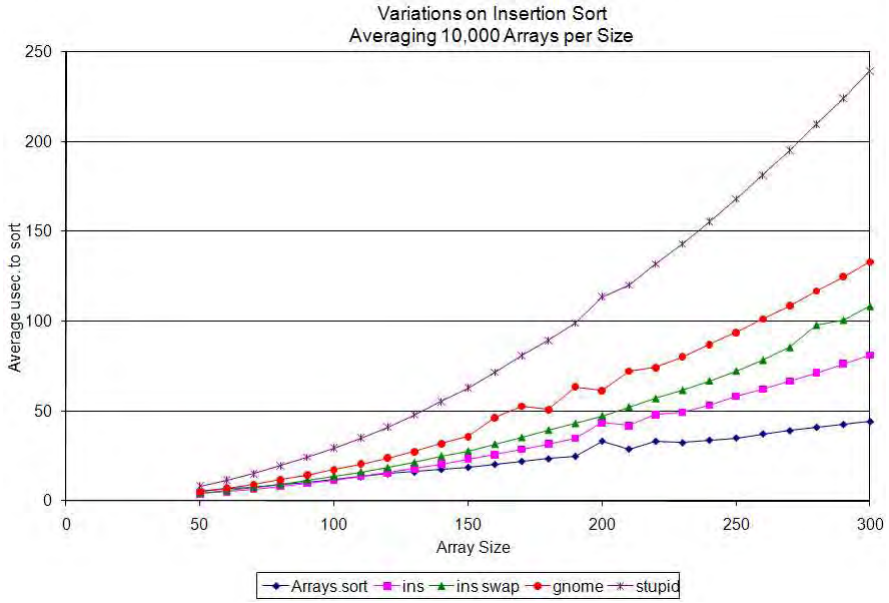
```
                    X = I
                I = I - 1
            else:
                I = I + 1
    return L
Z = [1, 4, 8, 2, 5, 3, 9, 7, 10, 6]
print gnomesort(Z)
```

La salida por pantalla de la versión optimizada muestra que para la lista de prueba se ha conseguido una mejora considerable, reduciendo en 12 la cantidad total de iteraciones, con un total de 21. Esto demuestra la clara superioridad de Gnome Sort contra Bubble Sort cuando a ambos se le han incorporado sus optimizaciones más comunes.

```
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 1 I = 1 X = 0 Sin cambios
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 2 I = 2 X = 0 Sin cambios
[1, 4, 8, 2, 5, 3, 9, 7, 10, 6] iteración = 3 I = 3 X = 0 Intercambio de 2 y 8
[1, 4, 2, 8, 5, 3, 9, 7, 10, 6] iteración = 4 I = 2 X = 3 Intercambio de 2 y 4
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 5 I = 1 X = 3 Sin cambios
[1, 2, 4, 8, 5, 3, 9, 7, 10, 6] iteración = 6 I = 4 X = 0 Intercambio de 5 y 8
[1, 2, 4, 5, 8, 3, 9, 7, 10, 6] iteración = 7 I = 3 X = 4 Sin cambios
[1, 2, 4, 5, 8, 3, 9, 7, 10, 6] iteración = 8 I = 5 X = 0 Intercambio de 3 y 8
[1, 2, 4, 5, 3, 8, 9, 7, 10, 6] iteración = 9 I = 4 X = 5 Intercambio de 3 y 5
[1, 2, 4, 3, 5, 8, 9, 7, 10, 6] iteración = 10 I = 3 X = 5 Intercambio de 3 y 4
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 11 I = 2 X = 5 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 12 I = 6 X = 0 Sin cambios
[1, 2, 3, 4, 5, 8, 9, 7, 10, 6] iteración = 13 I = 7 X = 0 Intercambio de 7 y 9
[1, 2, 3, 4, 5, 8, 7, 9, 10, 6] iteración = 14 I = 6 X = 7 Intercambio de 7 y 8
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 15 I = 5 X = 7 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 16 I = 8 X = 0 Sin cambios
[1, 2, 3, 4, 5, 7, 8, 9, 10, 6] iteración = 17 I = 9 X = 0 Intercambio de 6 y 10
[1, 2, 3, 4, 5, 7, 8, 9, 6, 10] iteración = 18 I = 8 X = 9 Intercambio de 6 y 9
[1, 2, 3, 4, 5, 7, 8, 6, 9, 10] iteración = 19 I = 7 X = 9 Intercambio de 6 y 8
[1, 2, 3, 4, 5, 7, 6, 8, 9, 10] iteración = 20 I = 6 X = 9 Intercambio de 6 y 7
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10] iteración = 21 I = 5 X = 9 Sin cambios
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

De hecho, con esta mejora, los tiempos de ejecución de Gnome Sort son similares a los de Insertion Sort para la misma lista. Aunque uno de sus dos “padres”, el científico de la Universidad de Glasgow Hamid Sarbazi-Azad, lo bautizó como “Stupid Sort”, y el propio Dick Grune lo llama “el algoritmo de ordenamiento más sencillo que existe”, es una variación interesante que debemos conocer por su conjunción de simplicidad y eficacia. Su complejidad es de O(N∧2) para el caso promedio, pero la misma se reduce a O(N) si la lista está

En este gráfico observamos la baja en los tiempos de respuesta de este algoritmo conforme aumenta la cantidad de elementos a ordenar. Aunque no es tan grave como en otros casos, el deterioro es lo bastante significativo como para optar por otros métodos en estos casos.



casi ordenada.

EN CONCLUSIÓN

Pese a las ventajas que ya hemos expuesto, nuestro buen gnomo sufre del mismo déficit que presentan sus fuentes de inspiración. Tanto los métodos de inserción como de burbujeo -y por ende también Gnome Sort- ofrecen tiempos de respuesta muy pobres ante listas con una gran cantidad de elementos. Para este tipo de estructuras, los lectores de esta sección saben bien que es conveniente recurrir a otro tipo de algoritmos, como los que trabajan aplicando recursividad. Sin embargo, estudiar Gnome Sort vale la pena, aunque más no sea para descubrir que, aunque las bases de este tipo de programas han sido sentadas por los grandes próceres de la informática, todavía es posible desarrollar nuevas variantes y derivados interesantes e ingeniosos. Este algoritmo, que tiene unos pocos años de vida (nació con el nuevo milenio) así lo demuestra. P

"La cantidad total de iteraciones nos permite ver rápidamente que Gnome Sort supera claramente a Bubble Sort. Esta versión resuelve el ordenamiento en 33 iteraciones, contra 81 que le llevaría la misma lista al método de Burbuja sin optimizar."

El e-commerce hecho simple

Por Roxana Miguel

roxana.miguel@pixelscode.com
@roxanamiguel

La web es un espacio de comercialización cada vez más importante. Las empresas o comercios que abren sus puertas al público también abren una ventana en la gran nube para el e-commerce. Nadie puede quedarse sin su stand en la feria online. Aquí veremos una manera de incluir los carritos a la web de manera más sencilla.

Preparo un café y enciendo la notebook, cliqueo sobre un paquete de azúcar, otro de hamburguesas y unas planchas de pastas frescas; indico la forma de pago y el destino del envío. Mientras espero que llegue el pedido del supermercado, reviso mis cuentas bancarias y cancelo los pagos de servicios que están por vencer, en otra página cargo en mi carrito de compras unos discos y películas para ver durante el fin de semana. Apago la notebook y salgo a la calle, si me quedó algo por comprar lo hago desde mi smartphone.

Hace un tiempo atrás esto parecía imposible, hoy, cada vez más usuarios realizan sus compras habituales por medio de la red, independientemente de los prejuicios y los miedos a la ciber-piratería de datos. A favor de estas tendencias, muchas compañías suman esta opción a sus páginas que, hasta ese momento,

contaban tanto con un site como con un aviso en el diario.

Desarrollar un sitio con estas condiciones es una tarea bastante compleja; para quienes no quieran meter mano en los códigos siempre se puede recurrir a algunos accesos más simples por medio de frameworks dentro de tiendas prediseñadas. Se trata de herramientas que funcionan como estructuras conceptuales cuyos módulos de software pueden organizar nuestro desarrollo. Con esta arquitectura de software se obtiene una metodología de trabajo que amplía los beneficios de las aplicaciones en los dominios.

Al comenzar la programación de una página cuyo objetivo es comercializar un producto, nos toparemos con el engorroso capítulo de hacer funcional el cuadro e-commerce. No es tarea para nada sencilla y es justamente por esta razón que han sido creadas estas tiendas online listas para usar. Serán nuestros aliados en este trabajo, ya que existen para facilitar el desarrollo del software identificando sus requisitos.

Para trabajar con estos elementos podemos definir una estructura estable que determinará el dinamismo de la página. Actualmente, existe mucho software que proporciona la facilidad de implementar una tienda virtual en la web por medio de servicios gratuitos como Oscommerce, Prestashop o Magento, que son los más utilizados por su antigüedad y utilidades. Los menos consumidos son Zen Cart, Virtuemart y Open Cart, entre otros.

TIPS ANTES DE ABRIR LA TIENDA

Una vez que está abierta la tienda online es como abrir un negocio, todas las reformas pueden traernos algunas frustraciones si no fueron pensadas con anticipación. Al comenzar con el desarrollo del site debemos contar con algunos parámetros que nos serán útiles a futuro, como cuál es el lenguaje que utilizaremos para esta programación, qué tecnología contendrá y cuál será el esquema.
El lenguaje PHP y una base de datos MySql son recomendados para estos programas ya que son simples, de la misma manera que tenemos que buscar que se adapte a niveles CSS y HTML de manera sencilla. El site debe estar diseñado conforme al posicionamiento SEO, puesto que si tenemos productos en venta al que nadie accede, de nada sirve tener un web inmersa en el comercio electrónico, sobre todo cuando esta modalidad está cada vez más en

> *"Actualmente, existe mucho software que proporciona la facilidad de implementar una tienda virtual en la web por medio de servicios gratuitos como Oscommerce, Prestashop o Magento."*

> *Quien desee trabajar con Oscommerce deberá contar con un buen manejo de PHP y generar un gran número de módulos para que el sitio sea captado por los buscadores."*

auge y la web se encuentra rebalsada de estos servicios.
Es muy positivo pensar una tienda virtual que contará con categorías o familias que puedan crecer y hasta reproducirse con el paso del tiempo; con suerte, nuestro comercio trascienda la propuesta inicial y no querremos quedarnos a mitad de camino. Debe coexistir una lógica para esta suerte de prosperidad, como por ejemplo, la opción de combinar artículos, incluir documentos Word, pdf o fotos y videos, incluso relacionarlo con otros artículos para captar el interés del usuario en otras ofertas.
Puestos en el aérea comercial, la estructura debe ofrecer diferentes formas de pago, transferencias y tarjetas, también opciones de envío. Es más que interesante que permita realizar compras de manera fácil y rápida, sin tener que llenar demasiados formularios y que proporcione un orden que visualice precios y nombre del producto.

SOFTWARES POPULARES

Los llamaremos así porque son los más utilizados para generar tiendas virtuales, no sólo por ser gratuitos sino que además son más potentes. Se tratan de Oscommerce, Prestashop y Magento. Conozcámoslos.

Oscommerce, es una aplicación open source gratuita que comenzó a funcionar en marzo del 2000. Desde entonces, fue conformando una comunidad de usuarios que colabora con los fallos de seguridad, bugs y reportes de errores. Admite formas de pago como Ahthorize.net, tarjetas de Crédito, pago contra reembolso, iPayment, cheque, transferencia bancaria, NOCHEX, PayPal, 2CheckOut, PSiGate, SECPay, Visa, Mastercard, entre otros. Aunque en su momento fue uno de los mejores gestores, hoy presenta algunas objeciones que lo debilitan frente a sus competidores, aún así recientemente presentaron la tercera versión que no fue tan aceptada por los webmasters ya que presenta muchos errores que pretenden que sean solucionados por los usuarios.
Cuenta a su favor con que integra distintos idiomas, ofrece una gran cantidad de módulos desarrollados que economizan costos y su desarrollo es sencillo, incluso la instalación. Sin embargo, tiene algunos puntos en contra como que la instalación requiere de varios módulos para que tome forma de tienda, no usa CSS y no se escuchan rumores de que se actualice en este sentido. Quien desee trabajar con Oscommerce deberá contar con un buen manejo de PHP y generar un gran número de módulos para que el sitio sea captado por los buscadores.

Prestashop, también es un software gratuito y open source, pero que a diferencia del anterior cuenta con una importante comunidad de usuarios que crece a pasos agigantados y quizás porque aquel ha cerrado su foro español, de hecho está preparada para recibir los restos de Oscommerce, ya

que cuenta con un módulo para migrar a Prestashop como una manera de mejorar la tecnología.
Actualmente es uno de los preferidos de los desarrolladores porque ofrece una buena indexación, Ajax integrado, funciona con CSS y, por ende, es muy sencillo de personalizar.
Este modelo creado en Francia es liviano, de fácil instalación y profesa que en diez minutos la tienda virtual estará en funcionamiento. Ofrece un panel de administración intuitivo que permite crear categorías, productos, fabricantes y controlar a los clientes. No por nada fue galardonada en el 2010 como la mejor aplicación para comercio electrónico.
Ofrece muchas ventajas para los desarrollos como un bajo consumo de CPU, introducción de códigos de barras, opción de descarga de productos físicos o virtuales, cuenta con módulos ya desarrollados y ya se anuncia una importante actualización con una versión 1.5. Para los que buscan el pelo en la sopa, puede traer inconvenientes que los idiomas predominantes son el francés y el inglés.

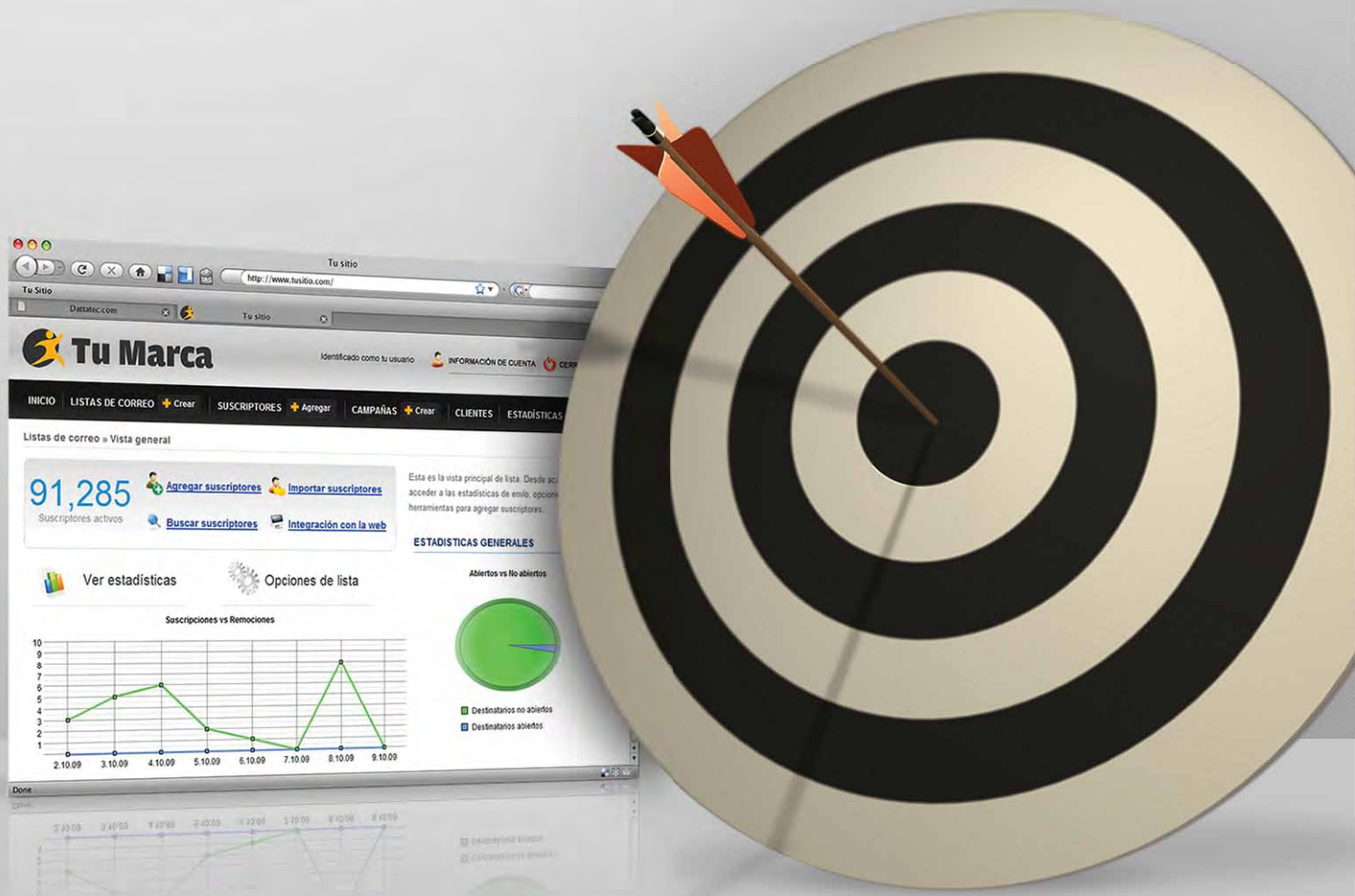
Magento, este proyecto comenzó a funcionar en el año 2007 y ha logrado sumar desde entonces una muy buena reputación. También es open source y gratuito, se destaca porque se puede navegar de manera fácil, cuenta con un código simple, ofrece un muy buen posicionamiento en los buscadores y un diseño de interfaz distinguido.
Aunque es uno de los más utilizados, su gestor de instalación es sumamente complejo respecto a los anteriores, como así también el uso del panel de control. Tu cliente seguramente necesitará que lo orientes para usarlo cómodamente.
Como cuenta con una versión paga, quizás una de las negativas de Magento más trascendentales es que su versión gratuita es cada vez más limitada. Sin embargo, su rendimiento es muy bueno para empresas grandes que requieren de un software completo, en contraposición, para que su funcionamiento no resulte lento deberás proveerte de un VPS o servidor único para hacer correr esta tienda. Es que Magento es muy potente, permite multitiendas, cuenta con sistema de búsqueda en Ajax, se puede personalizar completamente y ofrece un completo panel de control.
No esperes encontrarte con una comunidad muy poblada, por el contrario, los pocos que rondan sólo brindan apoyo en inglés. Debes estar preparado porque consume muchos recursos y prácticamente no ofrece muchos módulos desarrollados. Se trata de un archivo pesado, que exigirá un buen servidor. Dicen que lo bueno cuesta caro, pero también hay que rever si la web en desarrollo requiere de estos costos de inversión.

ELIGE TU PROPIA TIENDA

Ya contamos con una web diseñada, un grupo de productos para comercializar y un software específico para e-commerce. Lo que sigue es más que sencillo: hay que descargar el software y alojarlo en el mismo servidor donde está nuestra página. Desde allí, los framesworks que utilizaremos nos permitirán adaptar nuestra propuesta a la arquitectura de software que tenemos disponible. El consejo más inteligente es entonces que tengamos bien claro qué tipo de comercialización se realizará, con qué tráfico de ventas y consultas esperas encontrarte y entonces elige cuál de todas las aplicaciones que hoy conocimos te será más útil. P

Email Marketing con **tu marca**

Ganá dinero ofreciendo a tus clientes la **mejor herramienta.**



EnvialoSimple.com

dattatec.com