Biblioteca de graficación estereoscópica

Marc Serrano Leiva

Resumen—En los últimos años se ha producido un gran avance en la tecnología 3D como se puede observar tanto en productos destinados al público como proyectos internos de grandes empresas. Este trabajo se centra en la generación de diversos tipos de gráficos haciendo uso de la estereoscopia para dotarlos de profundidad y darle al usuario la sensación que el gráfico no solamente está proyectado en las dos dimensiones de la pantalla sino que se ayuda de la tercera dimensión para crear el efecto de que realmente tiene profundidad. Para que el usuario pueda interactuar con el gráfico se implementarán eventos para que el usuario pueda interactuar con él. Como trabajo futuro se pueden implementar más tipos de gráficos, añadir animaciones en la creación de los gráficos e implementar diferentes tipos de eventos para interactuar con el gráfico de diversas formas.

Palabras clave—estereoscopia, 3D, paralaje, infografía, gráfico, eventos, canvas.

Abstract— In recent years there has been a breakthrough in 3D technology as seen in products for public and internal companies projects. This work focuses on the generation of various types of charts using stereoscopy to give them depth and give to the user the feeling that the chart not only is projected in two dimensions of the screen but it helps by the third dimension to create an effect that really has depth. So that the user can interact with the chart, events will be implemented to enable the user to interact with it. As future work can be implemented more chart types, add animation to create the charts and implement different types of events to interact with the chart in several ways.

Index Terms—stereoscopy, 3D, parallax, infographics, chart, events, canvas

1 Introducción

Este trabajo de final de grado nace de la necesidad de crear una biblioteca que permita graficar de manera estereoscópica ya que actualmente no existe ninguna librería que permita realizarlo. La propuesta de este trabajo consiste en crear una biblioteca de fácil acceso y utilización para que los usuarios que deseen incorporar un gráfico estereoscópico en su web puedan hacerlo sin dificultad.

El objetivo principal del proyecto consiste en permitir crear diversos tipos de gráficos como pueden ser:

- Gráficos de barras:
- Gráficos circulares
- Gráficos de línea
- Gráficos de área

Antes de empezar es necesario definir algunos conceptos que hay que tener muy claros para poder entender el proyecto:

¿Qué es un gráfico?

Un gráfico es un tipo de representación de datos que utiliza recursos gráficos como pueden ser líneas, superficies, signos,… para manifestar visualmente la correlación estadística o la relación matemática que guardan los datos entre sí.

Un gráfico también puede ser definido como un conjunto de puntos que se representan en unas coordenadas cartesianas y nos permiten analizar el comportamiento de un proceso. [1]

¿Qué es la infografía?

La infografía es la representación visual de los propios textos, en la que pueden intervenir descripciones, narraciones o interpretaciones representadas de manera gráfica. La infografía nació como medio para transmitir la información de manera gráfica. [2]

¿Qué es el 3D?

En física, geometría y análisis matemático podemos determinar que un objeto es tridimensional si tiene tres dimensiones. En otras palabras, un objeto es tridimensional si podemos localizarlo especificando tres puntos dentro de un cierto rango.

En computación el 3D representa las tres dimensiones, altura, anchura y profundidad. Técnicamente el único mundo 3D es el real, ya que la computadora solo puede simularlo ya que trabaja en 2D. [3]

- *E-mail de contacte: mark_bk12@hotmail.com*
- *Mención realizada: Ingeniería de Computadores*
- *Trabajo tutorizado por: Jordi Carrabina / Diego González*
- *Curso 2014/15*

¿Qué es la estereoscopia?
La estereoscopia es una técnica utilizada para habilitar la tercera dimensión creando la ilusión de profundidad en una imagen 2D. [4]

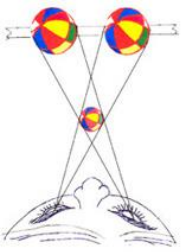

Figura 1 – ¿Qué es la estereoscopia?. Fuente: http://individual.utoronto.ca/iizuka/research/cellophane.htm

¿Qué es el parallax?
Parallax se define como la posición relativa de la imagen de un objeto en un conjunto de imágenes. [5]

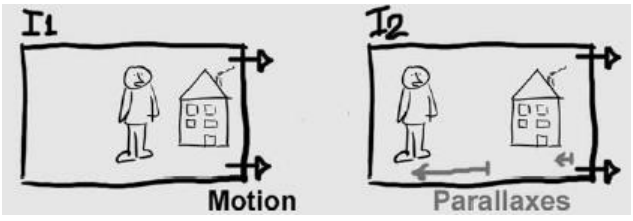

Figura 2 – ¿Qué es el parallax?. Fuente: Bernard Mendiburu. *3D Movie Making – Stereoscopic Digital Cinema from Script to Screen* (p. 15). Oxford: Elsevier, 2009

HTML5 / Canvas
HTML, el lenguaje básico de la World Wide Web se encuentra actualmente en su quinta revisión. Esta versión es la primera en la que se han desarrollado en paralelo HTML i XHTML. Respecto a la revisión anterior incorpora diversas novedades entre las que cabe destacar las siguientes:

- Incorporación de nuevos tags:
  - Canvas 2D/3D, audio, video
- Tags para gestionar grandes volúmenes de datos.
- Mejoras en los formularios
- Nuevos visores
- Drag & Drop

En cuanto al tag "canvas", fue introducido por Apple en 2004 para mejorar sus widgets y su navegador (Safari).
El canvas consiste en una región dibujable definida en HTML con atributos de altura y anchura. Actualmente es soportado por la mayoría de los navegadores:
- Internet Explorer a partir de su versión 9.0
- Firefox
- Chrome
- Opera (Desktop & Mobile)
- Safari (Desktop & Mobile)
- Android browser a partir de si versión 2.1

La biblioteca estará enfocada a la web, es decir las funciones están pensadas para poder dibujar gráficos en páginas web. La biblioteca debe poderla utilizar todo el mundo, por lo que debe ser accesible y de fácil utilización.

¿Qué son los eventos?
Podemos definir un evento cómo la manera que tenemos de controlar como se va a comportar nuestra página web cuando el usuario realice una acción sobre ella. [6]

2 ESTADO DEL ARTE

Actualmente el concepto de 3D está muy de moda, aunque se trate de un concepto antiguo (en el cine su creación data de 1838 y la primera proyección en 3D data del 1922 con la película "l'arrivée de train", filmada en 1903) [7] a día de hoy se están desarrollando diversos proyectos e investigando en cómo mejorar esta tecnología.
Aunque esta tecnología pueda parecer más famosa en el mundo del cine, hay muchos otros campos en los que se puede aplicar como pueden ser los videojuegos, en el diseño de nuevos productos, en salud,…
Hoy en día ya no solo se habla de 3D como una ilusión óptica qua ya implementan muchos modelos de televisores Smart TV que no solo incorporan el 3D sino que muchos de ellos son capaces de convertir una imagen 2D en una 3D, sino que se está trabajando en gafas de realidad virtual como pueden ser las Oculus Rift o en gafas de realidad aumentada como las HoloLents de Microsoft. En otros campos que no son simplemente la visión en 3D también se ha efectuado un gran avance en los últimos años, por ejemplo en las impresoras 3D que son capaces de imprimirnos (construirnos) un modelo 3D directamente enviado des de nuestro ordenador.

¿Qué software de graficación existe actualmente?
Actualmente no existe ningún software de graficación estereoscópica, pero sí que podemos encontrar diferentes bibliotecas de graficación que están en funcionamiento actualmente:
Para Microsoft encontramos el Microsoft Chart Control [8] para .NET que nos permite controlar los gráficos creados con .NET.
Para JAVA tenemos el JFREECHART [9], una librería 100% gratuita para poder representar gráficos utilizando el lenguaje de programación JAVA.
Para WEB tenemos el Chart.js [10], una librería que nos permite dibujar hasta 6 tipos de gráficos distintos. Basada en HTML5 y utilizando el tag "canvas", adaptable a diferentes dispositivos, modular e interactivo.
La motivación para desarrollar este proyecto radica en el hecho de que todas estas bibliotecas nos permiten representar gráficos sobre la pantalla de nuestro ordenador, pero ninguno de ellos nos permite introducir el factor estereoscópico para dotar a los gráficos de una cierta profundidad.

3 METODOLOGÍA

La metodología que vamos a seguir para el desarrollo del proyecto va a ser la siguiente:
La primera fase del proyecto va a constar de diferentes lecturas para iniciarnos en el mundo de la estereoscopia así como para conocer su funcionamiento y las diferentes características que tiene y que podremos aprovechar en nuestro proyecto.
Una vez consolidada esa información el siguiente paso a seguir es el análisis de la biblioteca 3D SXS3DCNV[11] que utilizaremos para poder dibujar de manera estereoscópica utilizando el tag canvas.
Después de analizar y comprender el funcionamiento de la biblioteca el siguiente paso a seguir es la realización de diversos ejemplos con la finalidad de familiarizarnos con la biblioteca y las diferentes funciones que nos ofrece. Una vez realizados los ejemplos los validaremos en una pantalla 3D para comprobar que los resultados son satisfactorios.
Una vez familiarizados con la biblioteca ya podemos empezar con los gráficos, concretamente con el grafico de barras. Para este primer ejemplo de grafico no tenemos una estructura de datos bien definida por lo que optamos por una estructura en forma de Array, simplemente para comprobar que realmente se puede graficar con esta biblioteca. Una vez desarrollado el grafico lo testeamos con la ayuda de la pantalla 3D y de las gafas.
Después de comprobar que realmente podemos graficar de forma estereoscópica es hora de pensar en una estructura de datos concreta y bien definida, la cual podremos utilizar para todos los tipos de gráficos. Una vez concebida esa estructura es horade empezar con el desarrollo del primer tipo de gráfico, el de barras y su posterior comprobación para asegurarnos que funciona de manera adecuada. Una vez comprobado el correcto funcionamiento de este tipo de grafico podemos seguir (utilizando el mismo método de desarrollo más comprobación) con los siguientes tipos de gráficos. Una vez finalizados todos los tipos dotaremos estos gráficos de eventos, para que utilizando algún periférico como pueden ser el ratón o el teclado los gráficos tengan algún tipo de transformación o animación.

4 TRABAJO REALIZADO

En este apartado se explican las herramientas utilizadas, que se ha implementado y como se ha hecho.

4.1 Herramientas utilizadas

Las herramientas que vamos a necesitar para la realización del proyecto son las siguientes:
Herramientas de desarrollo:

- Navegador Web (Firefox o Chrome)
- Editor de texto (SublimeText, Brackets, MVSCode, …)
- Biblioteca 3D SXS3DCNV[11]

Herramientas de validación:

- Pantalla 3D (Asus VG278H) [12]
- Gafas 3D (Nvidia Vision 2) [13]

4.2 Fase inicial: SXS3DCNV

La fase inicial del proyecto consiste en ampliar el conocimiento sobre 3D, como podemos simular el efecto tridimensional en una pantalla estereoscópica, porque funciona,… Para realizar esta primera tarea se han leído diferentes libros y artículos para entender su definición y sobretodo su funcionamiento. Una vez adquirido el conocimiento sobre el 3D es el turno de la librería SXS3DCNV. Una vez descargada se procede a su examen para poder conocer su estructura y funcionamiento.

La estructura de la librería es la siguiente:

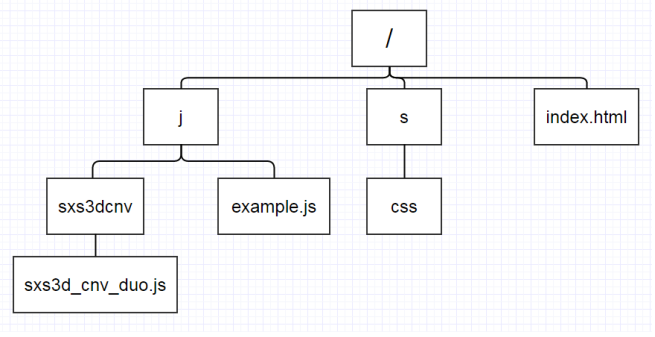

Figura 3 – Estructura SXS3DCNV

sxs3d_cnv_duo.js: librería que implementa todas las funciones que hacen posible dibujar de manera estereoscópica.
example.js: fichero preparado con la función "main" donde podemos implementar nuestras funciones utilizando las funciones de la librería.
index.html: fichero html donde se crea el canvas y es el encargado de permitirnos ver el contenido del example.js en el navegador.

Una vez comprendida la estructura de la librería y el funcionamiento de esta pasamos a realizar algunos ejemplos simples [14].
El primer ejemplo consiste en la creación de una espiral:

Figura 4 – Espiral. Captura tomada en una pantalla convencional

Un edificio:

Figura 5 – Edificio. Captura tomada en una pantalla convencional

Un árbol embrujado:

Figura 6 – Arbol. Captura tomada en una pantalla convencional

Una nube:

Figura 7 – Nube. Captura tomada en una pantalla convencional

Ases de una baraja:

Figura 8 – Ases. Captura tomada en una pantalla convencional

4.3 Creación gráficos

Una vez realizada esta prueba de concepto del funcionamiento de la librería podemos empezar con los desarrollos específicos y estructurados del proyecto.

4.3.1 Estructura de datos

Antes de empezar con lo que comportaría propiamente la creación de los diferentes tipos de gráficos se ha definido diferentes estructuras de datos para las opciones de los gráficos y para los datos (muestras) de estos.
Las opciones del gráfico vienen definidas en una estructura (objeto JavaScript) llamada *tipodegraficoSettings* que contiene diferentes parámetros para cada tipo de gráfico como veremos a continuación. Por su parte la estructura de datos que contiene la información que se va a representar en el gráfico viene definida con el nombre de ***Samples*** para los gráficos de barras, línea y circular y con el nombre de ***AreaSamples*** para el gráfico de área.

4.3.2 Gráfico de barras

El primer tipo de gráfico que se va a tratar en este proyecto es el gráfico de barras. El primer ejemplo de gráfico que se implementó fue el siguiente:

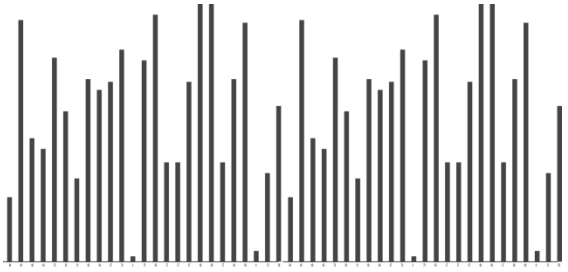

Figura 9 – Gráfico de barra 1.
Captura tomada en una pantalla convencional

Este gráfico estaba creado a partir de valores que se iban generando aleatoriamente y sin utilizar ningún tipo de estructura de datos, simplemente dibujando rectángulos con una pequeña separación.
Posteriormente se procedió a la creación de unos ejes de coordenadas que se podrán reutilizar (sin apenas cambios) en los gráficos de línea y de área.
Los ejes de coordenadas se visualizan de la siguiente forma:

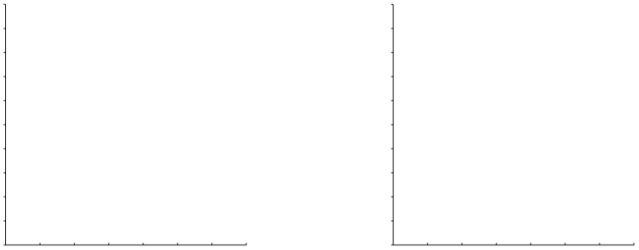

Figura 10 – Ejes de coordenadas.
Captura tomada en una pantalla convencional

Una vez creados los ejes ya podemos dibujar las barras del gráfico pero antes vamos a definir que estructura de datos utiliza el gráfico de barras:
Opciones de gráfico:

```
BarSettings = { w: predw, h: predh, pos: predpos, mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: predlincol, nlabsy: predlabsy, nsamples: npredsamples }
```

w: anchura del gráfico, expresada en píxeles
h: altura del gráfico, expresada en píxeles.
pos: posición del grafico en pantalla, expresada como un string. Puede tener diversas opciones:

- **center:** centrado en pantalla
- **right:** a la derecha de la pantalla
- **left:** a la izquierda de la pantalla
- **man:** posición impuesta manualmente

mw: en el caso de que pos = "man", indica la posición (pixel) horizontal donde va a empezar nuestro gráfico.
mh: en el caso de que pos = "man", indica la posición (pixel) vertical donde va a empezar nuestro gráfico.
dlinesy: booleano que indica si queremos líneas de división en el eje y.
dlinesx: booleano que indica si queremos líneas de división en el eje x.
dlinecol: color de las líneas de división, expresado como un string.
nlabsy: número de divisiones que queremos en el eje y.
nsamples: número de muestras que va a contener el gráfico.

Como se ha comentado anteriormente (en el punto 4.3.1) el gráfico de barras utiliza la estructura de datos ***Samples*** que será descrita en el próximo apartado (4.3.3 Grafico de línea).

Para dibujar el gráfico lo que se procede a realizar es dibujar un rectángulo con la base situada en el eje y=0, la posición x correspondiente y la altura proporcionada que debe tener teniendo en cuenta el valor que tiene esta muestra y el máximo valor que tenemos en el gráfico que corresponde a la altura máxima.
El resultado final del gráfico utilizando estas estructuras de datos se mostrara en la sección de resultados.

4.3.3 Gráfico de línea

El segundo tipo de gráfico que se va a realizar va a ser el de línea. En este segundo gráfico se utilizan funciones implementadas para el gráfico anterior con pequeñas variaciones como pueden ser la posición donde vamos a mostrar los marcadores "valores de x" o donde vamos a dibujar los "valores de y". Para dibujar este gráfico se utiliza la siguiente estructura de datos para las opciones:

```
LineSettings = { w: predw, h: predh, pos: predpos, mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: predlincol, linewidth: predwidth, linecolour: predlinecolour, nlabsy: predlabsy, nsamples: npredsamples }
```

w: anchura del gráfico, expresada en píxeles
h: altura del gráfico, expresada en píxeles.
pos: posición del grafico en pantalla, expresada como un string. Puede tener diversas opciones:

- **center:** centrado en pantalla
- **right:** a la derecha de la pantalla
- **left:** a la izquierda de la pantalla
- **man:** posición impuesta manualmente

mw: en el caso de que pos = "man", indica la posición (pixel) horizontal donde va a empezar nuestro gráfico.
mh: en el caso de que pos = "man", indica la posición (pixel) vertical donde va a empezar nuestro gráfico.
dlinesy: booleano que indica si queremos líneas de división en el eje y.
dlinesx: booleano que indica si queremos líneas de división en el eje x.
dlinecol: color de las líneas de división, expresado como un string.
linewidth: grosor de la línea, expresado en pixeles.
linecolour: color de la línea, expresado como un string.
nlabsy: número de divisiones que queremos en el eje y.
nsamples: número de muestras que va a contener el gráfico.

Como se ha comentado anteriormente el gráfico de línea también utiliza la estructura de datos ***Samples*** para guardar la información del gráfico. La estructura de datos ***Samples*** es un Array de longitud variable que podemos modificar desde cualquiera de las estructuras de datos *xxxxxSettings* en el atributo ***nsamples***. Este Array contiene un objeto JavaScript en cada una de sus posiciones y cada objeto representa una muestra del gráfico. Vamos a ver por qué está formado este objeto:

```
Samples[X] = { cat: "2007", value: 750, prof: predprof, colour: predcolour };
```

cat: valor x de la muestra, expresado como un string.
value: valor y de la muestra expresado como un int.
prof: profundidad de la muestra (paralaje), expresada como un int.
colour: color de la muestra expresado como un string.

Para dibujar el gráfico se procede a calcular el valor y de la primera muestra, movernos hasta él, siempre respetando la posición x de cada una de las muestras. Una vez situados en la posición inicial del gráfico se van calculando la siguiente y se traza una línea hasta ellas. También se ha añadido un pequeño rectángulo que simboliza el punto exacto que se está representando para hacer más amigable el gráfico.
El resultado final del gráfico utilizando estas estructuras de datos se mostrara en la sección de resultados.

4.3.4 Gráfico de área

El gráfico de área utiliza muchas de las funciones del gráfico de línea solo que en este caso al finalizar el último punto hay que desplazarse hasta la base del gráfico para poder volver al inicio siguiendo el eje x para poder cerrar el path y de esta manera poder pintar el área. Este gráfico es diferente al resto en el aspecto que utiliza su propia estructura de datos para las muestras. Pero antes vamos a repasar la estructura utilizada para las opciones del gráfico:

```
AreaSettings = { w: predw, h: predh, pos: predpos, mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: predlincol, linewidth: predwidth, linecolour: predlinecolour, nlabsy: predlabsy, nsamples: npredsamples }
```

w: anchura del gráfico, expresada en píxeles
h: altura del gráfico, expresada en píxeles.
pos: posición del grafico en pantalla, expresada como un string. Puede tener diversas opciones:

- **center:** centrado en pantalla
- **right:** a la derecha de la pantalla
- **left:** a la izquierda de la pantalla
- **man:** posición impuesta manualmente

mw: en el caso de que pos = "man", indica la posición (pixel) horizontal donde va a empezar nuestro gráfico.
mh: en el caso de que pos = "man", indica la posición (pixel) vertical donde va a empezar nuestro gráfico.
dlinesy: booleano que indica si queremos líneas de división en el eje y.
dlinesx: booleano que indica si queremos líneas de división en el eje x.
dlinecol: color de las líneas de división, expresado como un string.
linewidth: grosor de la línea, expresado en pixeles.
linecolour: color de la línea, expresado como un string.
nlabsy: número de divisiones que queremos en el eje y.
nsamples: número de muestras que va a contener el gráfico.

Como se puede observar utiliza la misma estructura de opciones que el gráfico de línea ya que como se ha comentado anteriormente son muy similares.
En cuanto a la estructura de datos que contiene la información del gráfico este es el único que utiliza la suya propia ya que este gráfico permite representar hasta 3 conjuntos diferentes en el mismo gráfico. La estructura correspondiente a este gráfico es la siguiente:

```
AreaSamplesX[0] = { cat: "2007", value: 750, prof: predprof, colour: predareacolor };
```

La X corresponde al conjunto de datos que estamos representando, puede tener los valores (1, 2,3).
cat: valor x de la muestra, expresado como un string.
value: valor y de la muestra expresado como un int.
prof: profundidad de la muestra, expresada como un int.
colour: color del area expresado como un string.

El resultado final del gráfico utilizando estas estructuras de datos se mostrara en la sección de resultados.

4.3.5 Gráfico circular

Finalmente el último tipo de gráfico implementado es el gráfico circular. Este gráfico es totalmente diferente a los anteriores y no comparte ninguna de sus funciones. La estructura de datos que tiene para controlar las opciones del gráfico es la siguiente:

```
PieSettings = {radius:predradius, nsamples: npredsamples}
```

radius: indica el radio del círculo, expresado en pixeles.
nsamples: número de muestras que va a contener el gráfico.

Como se ha comentado anteriormente (en el punto 4.3.1) el gráfico de barras utiliza la estructura de datos ***Samples*** descrita anteriormente.

Para dibujar este gráfico se procede a calcular el total correspondiente a la suma de todos los valores y de las diferentes muestras. Una vez obtenido el total podemos calcular que porcentaje del círculo le corresponde a cada muestra. Con este porcentaje se calcula el ángulo correspondiente. Una vez calculado se procede a dibujar situando un punto inicial al cual se le va incrementando el ángulo correspondiente para cada muestra.

El resultado final del gráfico utilizando estas estructuras de datos se mostrara en la sección de resultados.

4.4 Creación segundo contexto

Para mejorar el rendimiento de la aplicación se ha decidido crear un segundo contexto de dibujo. En el primer contexto de dibujo, el que ya teníamos creado se van a dibujar los diferentes gráficos estáticos de manera que ese contexto quedara ocupado por el gráfico en sí. En el segundo contexto vamos a implementar los eventos que se describirán en el siguiente punto. La razón por la que se ha decidido utilizar dos contextos de dibujo diferentes es porque los eventos necesitan limpiar (borrar) la pantalla cada vez que se detecta un evento o se deja de detectar uno. Si tenemos los eventos en el mismo contexto que el gráfico cada vez que se modifique el estado del evento debemos limpia la pantalla y volver a dibujar el gráfico mientras que si utilizamos dos contextos solo borramos el contexto donde se encuentran los eventos y el gráfico solamente se dibuja una vez al principio de la ejecución.

4.5 Implementación eventos

Una vez finalizados y comprobados todos los tipos de gráficos la implantación de eventos para dotar a los gráficos de cierta interacción con los usuarios pasa a ser un punto muy importante. Para poder crear eventos lo primero que se debe implementar es una clase eventos que contenga todas las funciones necesarias para poder utilizar los eventos a posteriori. Las funciones que vamos a utilizar principalmente son la de crear un objeto "Evento", el Begin i Close Path para poder definir las regiones donde va a actuar nuestro evento, El listener para que nuestro evento este siempre "escuchando" y las funciones que vamos a utilizar en este proyecto que son las de mouseover (el ratón esta encima) y mouseout (el ratón esta fuera).

```
this.canvas.addEventListener("mouseover", function(evt)
this.canvas.addEventListener("mouseout", function(evt)
```

Una vez implementada la clase eventos tenemos que implementar las funciones para que estos eventos funcionen. Para ello debemos modificar las funciones de creación de los gráficos para incluir las diferentes regiones de manera que el código sigua el siguiente esquema:

```
BeginRegion();
//Codigo que dibuja una de las muestras
this.canvas.addEventListener("mouseover", function(evt)
{
   Accion();
}
this.canvas.addEventListener("mouseout",  function(evt)
{
   LimpiarPantalla();
}
CloseRegion();
```

La acción implementada es la creación de un rectángulo con el borde del mismo color que la muestra y con el contenido del valor x e y de la misma:

Figura 11 – Infobox. Imagen del rectángulo que se crea al activar un evento y que muestra la información de la muestra.

5 RESULTADOS

En esta sección se van a analizar los resultados obtenidos para cada uno de los tipos de gráficos implementados en este proyecto. Las imágenes han sido tomadas de un monitor convencional (sin 3D) por lo que la imagen se ve duplicada.

5.1 Gráfico de barras

Vamos a empezar con el gráfico de barras. Este resultado se va a mostrar utilizando los siguientes parámetros:

```
BarSettings = { w: 600, h: 600, pos: "center", mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: "grey", nlabsy: 10, nsamples: 7 }
```

I modificando el color de los ***Samples*** para obtener un mejor efecto visual.

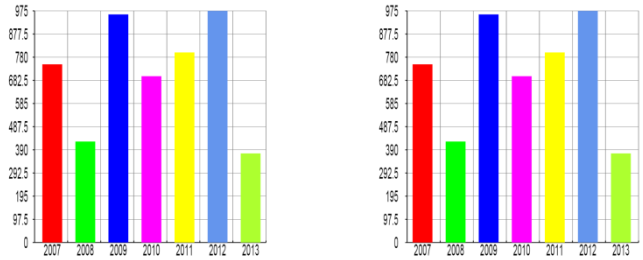

Figura 12 – Captura de pantalla correspondiente a el gráfico de barras

Si nos situamos encima de alguna de las barras aparece una ventana indicándonos la información de esta (interacción con los eventos.

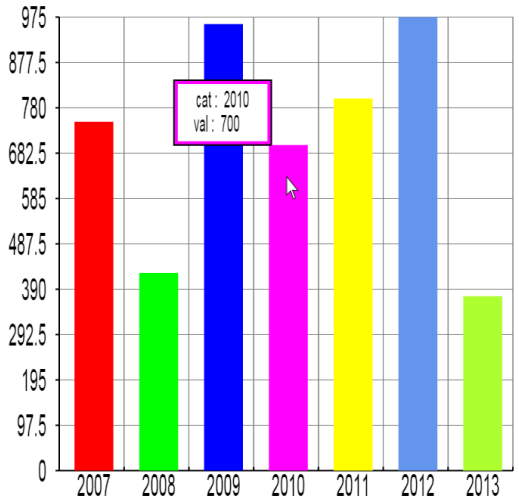

Figura 13 – Captura de pantalla correspondiente a el gráfico de barras con la implementación de los eventos al situar el mouse encima de una de las muestras.

5.2 Gráfico de línea

El segundo gráfico que se va a mostrar es el correspondiente al gráfico de línea. Este resultado ha sido tomado con los siguientes parámetros:

```
LineSettings = { w: 600, h: 600, pos: "center", mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: "grey", linewidth: 2, linecolour: "black", nlabsy: 6, nsamples: 7 }
```

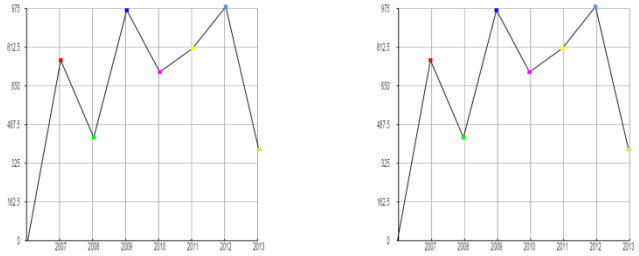

Figura 14 – Captura de pantalla correspondiente a el gráfico de linea

Situando el ratón encima (o en una posición aproximada) de alguno de los puntos del gráfico nos aparece una ventana mostrándonos la información correspondiente a esa muestra.

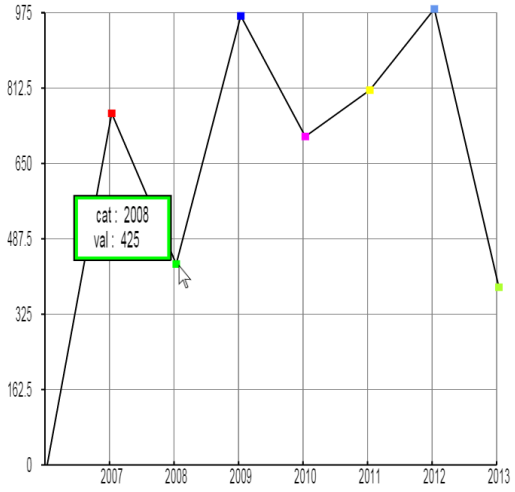

Figura 15 – Captura de pantalla correspondiente a el gráfico de linea con la implementación de los eventos al situar el mouse encima de una de las muestras.

5.3 Gráfico de área

El tercer resultado que se va a observar es el correspondiente al gráfico de área. Para este gráfico vamos a observar diferentes resultados obtenidos utilizando uno, dos y tres conjuntos de muestras. Para el primer caso utilizaremos únicamente un conjunto de muestras (el mismo que anteriormente hemos utilizado en el gráfico de línea y con los mismos parámetros de opciones.

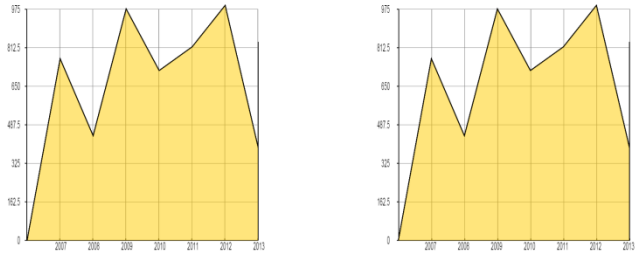

Figura 16 – Captura de pantalla correspondiente a el gráfico de área con un solo conjunto de datos

Mostrando un evento:

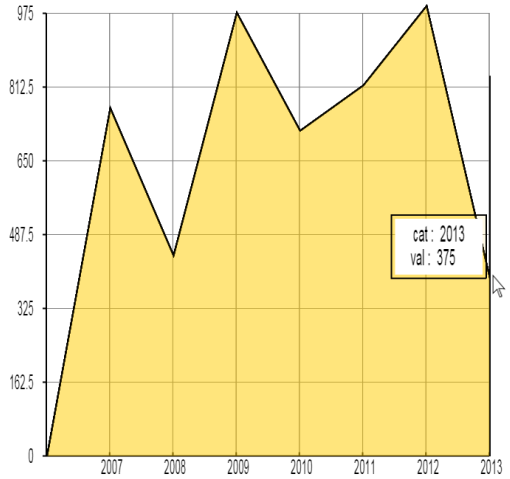

Figura 17 – Captura de pantalla correspondiente a el gráfico de área con un solo conjunto de datos y con la implementación de los eventos al situar el mouse encima de una de las muestras.

El segundo resultado es el obtenido utilizando dos conjuntos de datos (con valores diferentes). Y los siguientes parámetros:

```
AreaSettings = { w: 600, h: 600, pos: "center", mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: "grey", linewidth: 2, linecolour: "black", nlabsy: 10, nsamples: 8 }
```

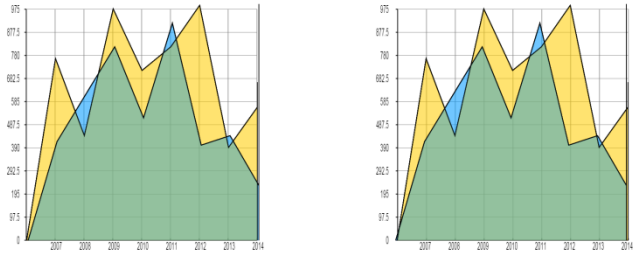

Figura 18 – Captura de pantalla correspondiente a el gráfico de área con dos conjuntos de datos

Cabe destacar que los colores utilizados en el gráfico de área son RGB con transparencia. En este tipo de gráfico con dos conjuntos de datos podemos obtener información mediante eventos de cualquiera de ellos.

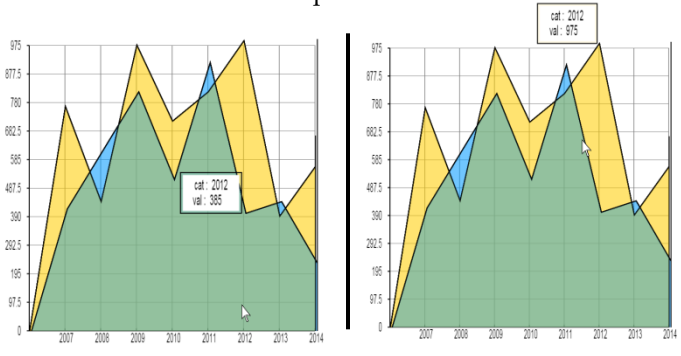

Figura 19 – Captura de pantalla correspondiente al gráfico de área con dos conjuntos de datos y con la implementación de los eventos al situar el mouse encima de una de las muestras.

Finalmente utilizando 3 conjuntos de datos con los siguientes parámetros:

```
AreaSettings = { w: 600, h: 600, pos: "center", mw: predmw, mh: predmh, dlinesy: true, dlinesx: true, dlincol: "red", linewidth: 2, linecolour: "brown", nlabsy: 4, nsamples: 10 }
```

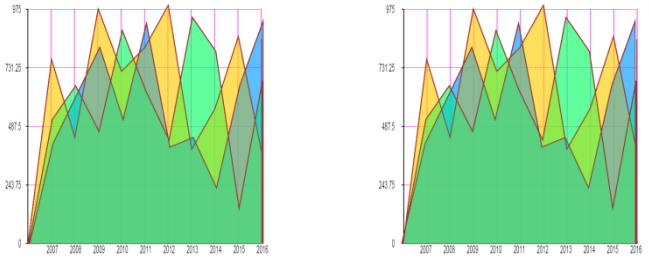

Figura 20 – Captura de pantalla correspondiente a el gráfico de área con tres conjuntos de datos

Igual que en el caso anterior (con dos muestras) los colores son RGB con transparencia.
Aplicando eventos al gráfico:

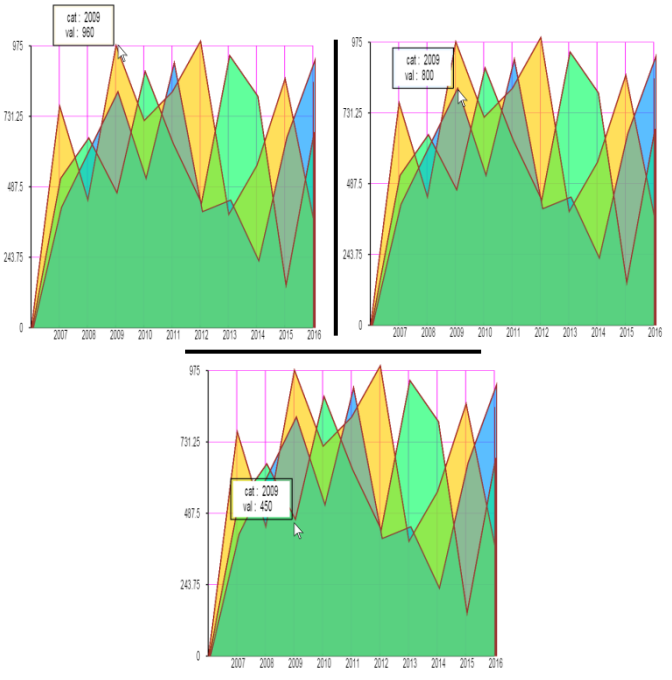

Figura 21 – Captura de pantalla correspondiente al gráfico de área con tres conjuntos de datos y con la implementación de los eventos al situar el mouse encima de una de las muestras.

5.4 Gráfico circular

Finalmente el último tipo de gráfico implementado es el gráfico circular.
El resultado obtenido utilizando los siguientes parámetros de opciones es:

```
PieSettings = {radius:300, nsamples: 4}
```

Figura 22 – Captura de pantalla correspondiente a el gráfico circular

Cabe destacar que aunque en esta imagen apreciemos un ovalo realmente se trata de un circulo perfecto, aunque al observarlo en una pantalla convencional se vea deformado al activar el 3D i combinar las dos imágenes queda en forma circular.
En el gráfico circular también podemos aplicar eventos solo que en este caso en vez de salir una ventana con la información esta sale junto al gráfico representada en el mismo color de la muestra.

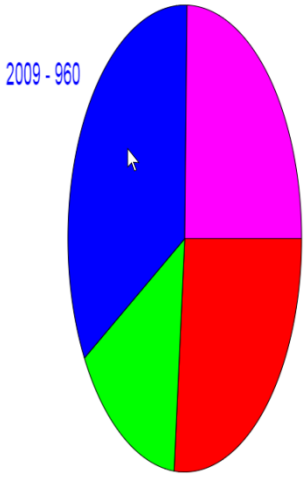

Figura 23 – Captura de pantalla correspondiente a el gráfico circular con la implementación de los eventos al situar el mouse encima de una de las muestras.

5.5 Compatibilidad

Compatibilidad con los diferentes navegadores (se especifica la versión en el apéndice)

Gráfico de barras	✓	✓	✓	✓	✓
Gráfico de línea	✓	✓	✓	✓	✓
Gráfico de área	✓	✓	✓	✓	✓
Gráfico circular	✓	✓	✓	✓	✓

Tabla 1: Compatibilidad de los distintos tipos de gráfico en los distintos navegadores.

6 Conclusiones

Después de realizar este proyecto se pueden extraer diversas conclusiones:
La primera y más importante es que el objetivo que trataba de lograr este proyecte se ha cumplido ya que es posible crear una biblioteca para graficar de manera estereoscópica.
Los resultados son bastante satisfactorios aunque sí que es cierto que hay pequeños errores que se podrían mejorar en versiones futuras como puede ser que al aplicar profundidad al gráfico circular las muestras no quedan perfectamente conjuntadas.
Es importante tener una buena estructura de datos que te facilite la creación de los gráficos y sobretodo la modificación e adición de opciones.
Gracias a este proyecto he podido comprobar el gran potencial que tiene el tag <canvas> ya que antes de la realización del mismo no lo había utilizado.
A simple vista (utilizando las gafas 3D) es difícil apreciar que muestra es más profunda ya que aunque sí que es verdad que el gráfico da la sensación de profundidad, el fondo está claramente detrás de las muestras si la diferencia de profundidad de estas es demasiado pequeña puede parecer que todas están en el mismo plano.
El gráfico de área es el que mejor resultado he obtenido ya que se puede apreciar claramente que las diferentes áreas están en profundidades distintas gracias a la transparencia que tienen.
Para una experiencia todavía más satisfactoria se podría implementar un "*mouse*" estereoscópico de manera que podamos modificar también la profundidad del puntero.
Existe un motivo psicológico que hace que esta representación estereoscópica provoque que los humanos nos fijemos más en las cosas con volumen. En otras palabras el hecho de que una barra este por delante (más cerca de nosotros) que las otras hace que nos fijemos más en ella, como si estuviera resaltada. Esto puede ser muy útil si nuestra intención es que el público se fije en un cierto valor ya que resaltando una barra modificando su profundidad llamamos la atención del público.
Los eventos ofrecen al usuario una cierta interacción con el gráfico hecho muy importante ya que le dan al usuario un cierto control sobre este.
En la realización de este proyecto he podido aplicar los conocimientos adquiridos sobre planificación y gestión de proyectos ya que hasta la fecha todos (o casi todos) los trabajos qua había realizado en la universidad habían sido muy pautados y con fechas de entrega relativamente cortas mientras que en este he adquirido los hábitos para ir realizando tareas pequeñas cada semana.
El aspecto que me ha comportado mayor dificultad ha sido el hecho de que no hay demasiada información pública, lo que requiere un mayor tiempo de análisis y en muchos casos la prueba y error para resolver problemas ya que no disponía de ejemplos funcionales donde mirar en que me había equivocado.
En cambio, ha sido muy positivo el hecho de utilizar el entorno estereoscópico desarrollado por el tutor de mi proyecto (Diego González) ya que podía hablar directamente con él para entender su funcionamiento y debatir

sobre el funcionamiento de mi librería 3D al igual que era mucho más fácil solucionar cualquier error.

AGRADECIMIENTOS

Quiero mostrar mi agradecimiento a Jordi Carrabina y a Diego González, tutores de este proyecto. He recibido su soporte durante todo el proyecto en forma de ideas, recomendaciones, lecturas, consejos. Gracias por vuestro apoyo tanto en los informes entregados como en el código implementado. Agradecer también a Diego González la cesión de su librería estereoscópica que me ha facilitado la creación de los gráficos. Gracias a esta ayuda y apoyo he aprendido muchas cosas y mejorado la calidad de este trabajo.

REFERENCIAS

[1] http://definicion.mx/grafico/
[2] http://www.ofifacil.com/ofifacil-infografias-que-es-definicion-como-se-hacen.php
[3] Bernard Mendiburu. *3D Movie Making – Stereoscopic Digital Cinema from Script to Screen* (p. 2-3). Oxford: Elsevier, 2009
[4] Bernard Mendiburu. *3D Movie Making – Stereoscopic Digital Cinema from Script to Screen* (p. 15). Oxford: Elsevier, 2009
[5] http://whatis.techtarget.com/definition/stereoscopy-stereoscopic-imaging
[6] http://www.desarrolloweb.com/articulos/1235.php
[7] http://sabia.tic.udc.es/gc/Contenidos%20adicionales/trabajos/3D/estereoscopia/cine.htm
[8] https://www.microsoft.com/es-es/download/details.aspx?id=14422
[9] http://www.jfree.org/jfreechart/
[10] http://www.chartjs.org/
[11] https://github.com/diekus/SXS3D-CNV
[12] https://www.asus.com/Monitors/VG278H/
[13] http://www.nvidia.es/object/3d-vision-main-es.html
[14] Eric Rowell. *HTML5 Canvas Cookbook*. Birmingham: Packt, 2011
[15] http://www.w3.org/
[16] http://individual.utoronto.ca/iizuka/research/cellophane.htm
[17] Bernard Mendiburu. *3D Movie Making – Stereoscopic Digital Cinema from Script to Screen.* Oxford: Elsevier, 2009
[18] David Flanagan. *Canvas Pocket Reference*. Sebastopol: O'Reilly, 2011
[19] Eric Rowell. *HTML5 Canvas Cookbook*. Birmingham: Packt, 2011
[20] D. González-Zúñiga, A. Chistyakov, and J. Carrabina. *Re-defining a Pattern: Stereoscopic Web Perception.*
[21] Jukka Häkkinen, Takashi Kawai, Jari Takatalo, Reiko Mitsuya and Göte Nyman. *What do people look at when they watch stereoscopic movies?*
[22] Samuel Gateau, Robert Neuman, Marc Salvati. *Stereoscopy, From XY to Z.* Siggraph, 2011.
[23] http://www.sky.com/shop/__PDF/3D/Basic_Principles_of_Stereoscopic_3D_v1.pdf

APENDICE

Version de navegador:
Chrome: 43.0.2357.124 m
Explorer: 11
Firefox: 38.01
Opera: 30.0
Safari: 8.0.5 (10600.5.17)