

WebGL Based 3D Dashboard for Tracking Software Development

Adrián Alonso Barriuso

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN
Universidad Rey Juan Carlos

2016



Contenido

- 1 Introducción
 - Descripción del problema
 - Objetivo principal
- 2 Tecnologías utilizadas
- 3 Desarrollo
 - Metodología
 - Iteraciones
- 4 Diseño y resultados
 - Arquitectura
 - Funciones principales
 - Tipos de gráficas
 - Filtros
 - Paneles
- 5 Conclusiones
 - Conocimientos aplicados
 - Lecciones aprendidas
 - Trabajos futuros
- 6 Referencias



Dashboards y seguimiento de desarrollo de software

Un **dashboard** es una interfaz de usuario que nos permite manejar y gestionar un tipo determinado de software o de hardware. En nuestro caso trabajamos con dashboards de visualización de datos de desarrollo de software. Algunos ejemplos de dashboards y de librerías para la visualización de datos basadas en software libre son:

- **Kibana**
- **Freeboard**
- **dc**



Kibana

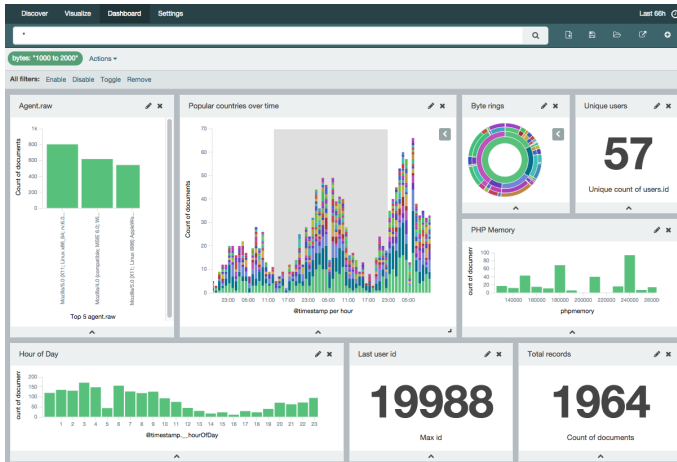


Figura: ejemplo de Kibana



Freeboard



Figura: Ejemplo de freeboard



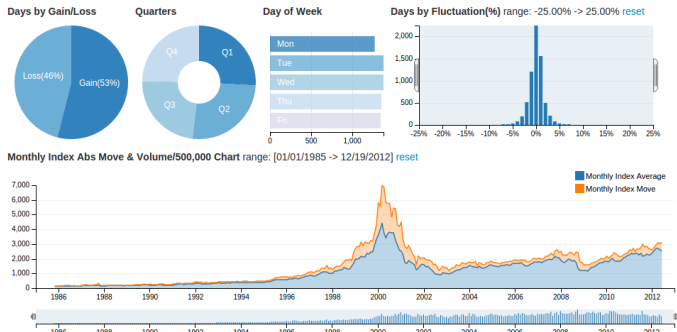


Figura: ejemplo de uso de dc.js

Dashboards en 2D

Ventajas

- 1 Simplicidad de representación.
- 2 No requieren aceleración gráfica.
- 3 Impacto mínimo en el rendimiento.

Inconvenientes

- 1 No se pueden representar gráficos con mas de 2 dimensiones, por razones obvias.
- 2 Las posibilidades de visualización y colocación son limitadas.

Objetivo principal

Crear una librería que nos permita crear visualizaciones y filtrar datos de desarrollo de software en 3D, dentro de cualquier navegador.

Requisitos

- 1 Conseguir una interfaz y funcionalidad tan parecidos a los de dc.js como sea posible.
- 2 Tener varios tipos de gráficas y ser capaces de filtrar.
- 3 Utilizar un framework de WebGL
- 4 Respuesta rápida de los filtros.
- 5 Debemos ser capaces de hacer zoom, desplazarnos y arrastrar las gráficas.
- 6 Debemos poder colocar gráficas tanto de forma independiente como dentro de paneles.
- 7 Tener una estructura basada en programación orientada a objetos.



Tecnologías utilizadas

- HTML5
- Javascript
- **webGL**
- **Three.js**
- Crossfilter
- THREEEx.DomEvents
- Orbit controls
- dat.GUI



Metodología Scrum

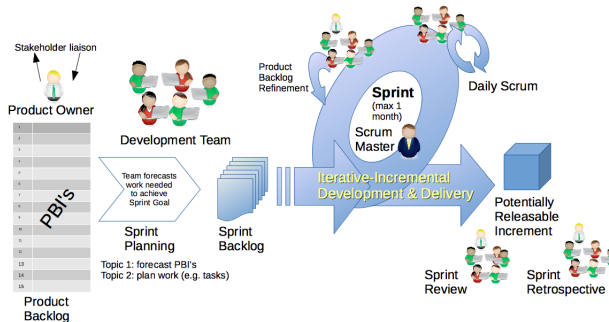


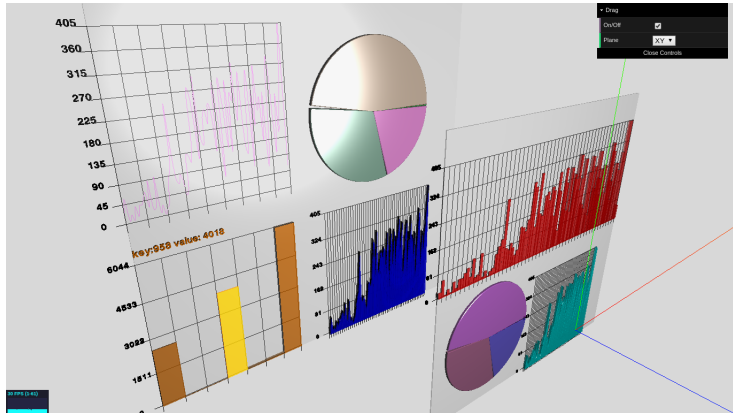
Figura: Metodología Scrum

Iteraciones

- **Iteración 0:** Investigación y aprendizaje
- **Iteración 1:** Primeras demos
- **Iteración 2:** Añadir interactividad
- **Iteración 3:** Añadir posibilidades de filtrado dimensional.
- **Iteración 4:** Creación de la biblioteca
- **Iteración 5:** Añadir paneles y otras características.



Ejemplo al final de la iteración 5



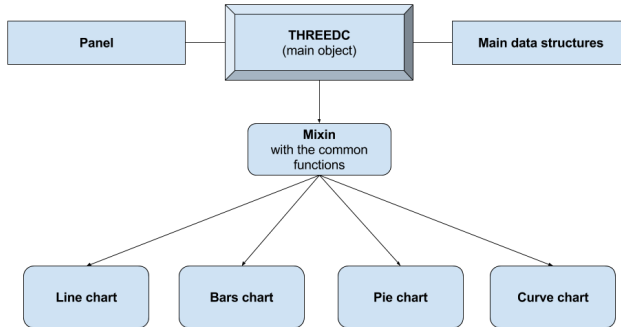


Figura: Arquitectura de la librería

Métodos en cadena

- **group**: Establece el grupo de Crossfilter.
- **dimension**: Establece la dimension de Crossfilter.
- **width**: El ancho de la gráfica.
- **height**: El alto de la gráfica.
- **gridson**: Activa los grids
- **gridsoff**: Descactiva los grids.
- **numberOfXLabels**: Establece el numero de etiquetas del eje x.
- **numberOfYLabels**: Establece el numero de etiquetas del eje y.
- **color**: Establece el color de la gráfica.
- **depth**: Establece la profundidad de la gráfica.
- **opacity**: Establece la la opacidad de la gráfica



Una vez creada la escena con las condiciones de cámara y luz deseadas, hacemos uso de la librería:

```
var line = THREEDC.lineChart([0,0,0]);  
line.group(groupByMonth)  
  .dimension(dimByMonth)  
  .width(500)  
  .numberOfXLabels(10)  
  .numberOfYLabels(15)  
  .gridsOn()  
  .height(200)  
  .color(0x0000ff)  
  .depth(20);  
THREEDC.renderAll();
```



lineChart

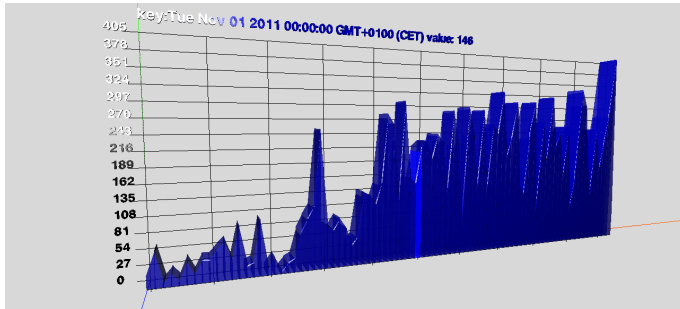


Figura: Ejemplo de lineChart

pieChart

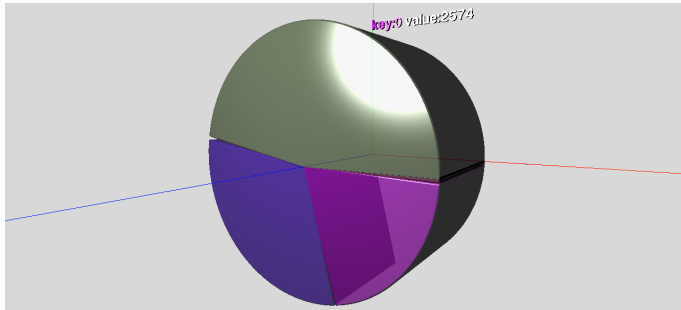


Figura: Arquitectura de la librería

barsChart

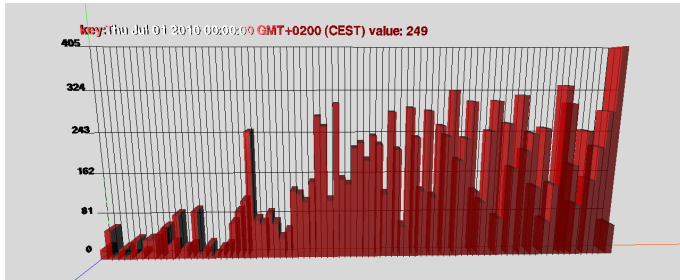


Figura: Arquitectura de la librería

smoothCurveChart

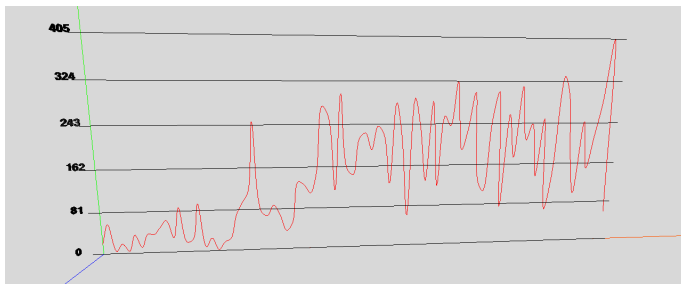


Figura: Arquitectura de la librería

Filtrado

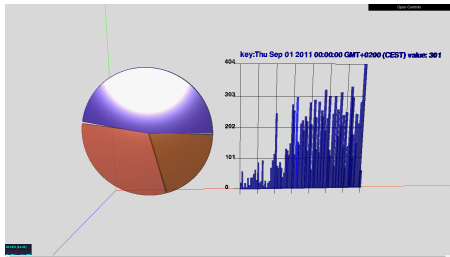


Figura: Antes de filtrado

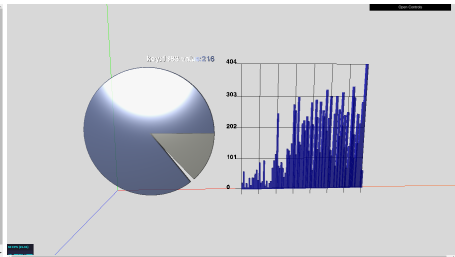


Figura: Después del filtrado

Paneles

Los **paneles** tienen como objetivo contener las gráficas relacionadas, sus parámetros son:

- 1 coords
- 2 numberOfCharts
- 3 size

Un ejemplo de instanciación de panel con 3 gráficas:

```
var panel=THREEDC.addPanel([0,0,0],3,[200,200]);
```



Ejemplos de colocación

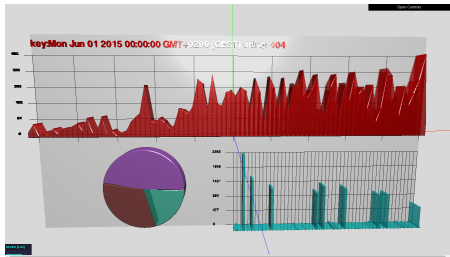


Figura: Gráfica dentro del panel

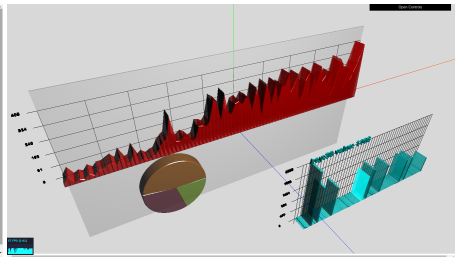


Figura: Gráfica fuera del panel

Ejemplos de colocación

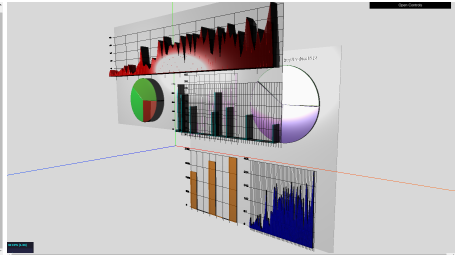
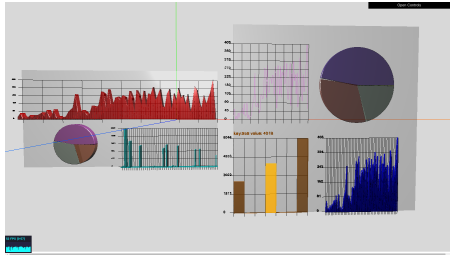


Figura: paneles y gráficas translúcidos



Ejemplos de colocación

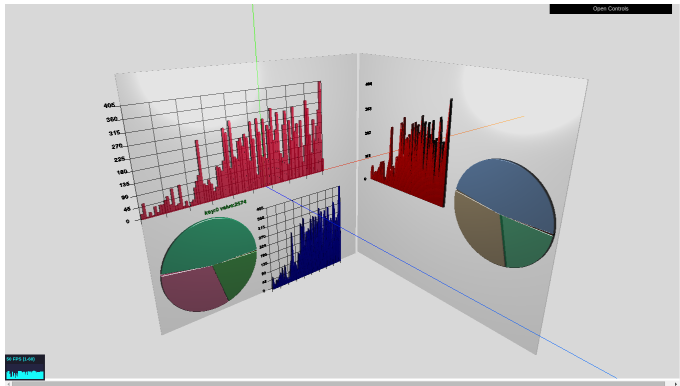


Figura: Ejemplo de charts y paneles rotados

Código para crear la misma gráfica de tarta usando dc.js y THREEDC.js:
Usando dc.js:

```
//HTML5 div to attach the pie
var pieDC = dc.pieChart("#test");

pieDC.radius(100)
    .dimension(dimPerOrg)
    .group(groupPerOrg);
dc.renderAll();
```

Usando THREEDC.js:

```
//Coordinates to attach the pie, a panel could be used too.
var pieTHREEDC = THREEDC.pieChart([0,0,0]);

pieTHREEDC.radius(100)
    .dimension(dimPerOrg)
    .group(groupPerOrg);

THREEDC.renderAll();
```



Comparacion con dc

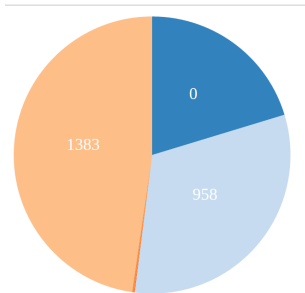


Figura: dc.js

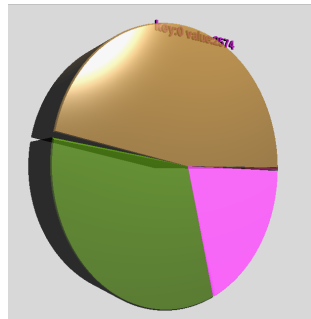


Figura: THREEDC.js

Asignaturas relacionadas

- Desarrollo de aplicaciones telemáticas
- Ingeniería de sistemas de información
- Servicios y aplicaciones telemáticas
- Ingeniería de sistemas telemáticos



Lecciones aprendidas

- Profundización en conocimiento de Javascript y HTML5
- Uso e implicaciones de WebGL y desarrollo de gráficos en 3D
- Uso de dashboard y frameworks de visualización de datos
- Uso de \LaTeX
- Mejora del nivel de inglés escrito



Trabajos futuros

- ❶ Mejoras gráficas.
- ❷ Añadir mas tipos de gráficas
- ❸ Añadir animaciones
- ❹ Añadir posibilidades de personalización en tiempo de ejecución
- ❺ Conseguir integración con Elasticsearch y Kibana
- ❻ Optimización general de rendimiento
- ❼ Implementar la rotación de gráficas y paneles en la UI



Referencias y bibliografía

- ① Javascript:
Marius Haverbeke, *Eloquent JavaScript*. 2014
<http://www.w3schools.com/js/>
- ② Web theoretical reference:
https://en.wikipedia.org/wiki/Main_Page
- ③ Three.js:
<http://threejs.org/docs/>
<https://www.udacity.com/course/interactive-3d-graphics--cs291>
<http://stemkoski.github.io/Three.js/>
- ④ Crossfilter:
<https://github.com/square/crossfilter/wiki/API-Reference>
- ⑤ WebGL:
<https://www.khronos.org/webgl/>
- ⑥ HTML5:
http://www.w3schools.com/html/html5_intro.asp
- ⑦ Domevents:
<https://github.com/jeromeetienne/threex.domevents>



Web del proyecto

<https://adrianalonsoba.github.io/web-THREEDC/>

