Cápsula 2: Fuerzas

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta hablaré sobre distintas opciones de fuerzas que provee D3.js para agregar a simulaciones.

En la última cápsula definimos nuestra primera simulación de fuerzas, y para ello agregamos múltiples fuerzas para comenzar. En esta cápsula revisaremos detalles de estas para entender mejor cómo se obtuvo el resultado anterior.

En términos simples, las tres fuerzas entregadas inicialmente hacen cosas distintas. Respectivamente: generan una fuerza de resorte entre nodos que estén conectados mediante enlaces; genera una fuerza de repulsión entre todo par de nodos; y alinea todos los nodos alrededor de un punto central en el sistema coordenado.

En base a estas distintas fuerzas que simulan fuerzas físicas, en cada iteración de actualización la simulación toma cada fuerza y nodo, y computa cambios de posición y velocidad sobre el nodo. Eso altera en cada actualización sus atributos “x”, “y”, “vx” y “vy”; y ese resultado es lo que vemos cambia en el tiempo.

Comenzaremos revisando la última fuerza, que es probablemente la más simple: “d3.forceCenter”. Como mencioné, esta fuerza ayuda a atraer cada nodo alrededor de un punto central. Esto es particularmente útil para asegurar que las posiciones se encuentren dentro de la región espacial que le dedicamos en la pantalla.

Si omito esta fuerza y probamos el código, vemos que muchos nodos no se ven, y que se organiza cerca del origen coordenado. Por eso agregamos antes una que pone el punto central a la mitad del espacio disponible tanto horizontalmente y verticalmente. Podemos alterar estos valores a otros, y veríamos sus efectos rápidamente.

Por ejemplo, si cambiamos el componente horizontal a un cuarto del espacio horizontal completo, veremos cómo todos los nodos se encuentran desplazados en esa dirección.

Por otro lado, “d3.forceLink” era la fuerza que mantiene distancias específicas entre nodos enlazados. Esta es la fuerza que recibía el arreglo de enlaces cargado, y luego reemplaza las referencias de texto que incluían por los objetos de los nodos mismos.

Para eso es el método “id” de esta fuerza, permite definir cómo determinar entre los objetos de nodo cuál es cuál, y particularmente saber a cual se refiere el enlace en sus atributos “source” y “target”. En el caso de nuestro ejemplo, eran los campos “nombre” a los cuales hacían referencia.

Entonces, esta fuerza en cada actualización se asegura que nodos que tengan un enlace que les conecten se mantengan a cierta distancia. Si están más lejos de tal distancia en

algún punto de la simulación atraerá los nodos, y si están más cerca los alejará de forma proporcional a la diferencia de distancia.

La distancia base es de 30 píxeles, y se puede alterar mediante el método “distance”. Si colocamos 50 por ejemplo, notaremos que al actualizar los nodos enlazados están más lejos entre sí. El valor de distancia no tiene porqué ser constante, y puede también ser una función que se evalúa enlace por enlace, dando incluso posibilidad para distintos valores dependiendo de condiciones sobre el enlace.

Por otro lado, “d3.forceManyBody” simula una fuerza de repulsión o atracción entre todos los nodos de la red. Cuando es de repulsión, funciona como fuerzas de carga eléctrica que es más potente a menor distancia. Esta fuerza es parte de porque la red visual mantiene una estructura ordenada por lo general: los nodos intentan escapar unos de otros, lo que genera menos cúmulos visuales y potencialmente menos oclusión entre ellos.

El poder de la fuerza tiene un valor por defecto de -30, cuyo signo indica es de repulsión. Si aumentamos la repulsión a -60 mediante el método “strength” y probamos el código, veremos que se alejan con más convicción los nodos unos de otros.

En cambio, si cambiamos el signo a un valor de 60, podemos ver cómo ocurrirá lo contrario y se intentan juntar los nodos. En caso de simulación de redes esto probablemente no es tan útil, pero si para gráficos de burbujas acumuladas tal vez.

El paquete provee definiciones de más fuerzas para otras situaciones. Por ejemplo, hasta el momento los nodos se tratan como puntos internamente y no como objetos que ocupan espacio. Los círculos que vemos visualmente los agregamos nosotros y les definimos un radio arbitrario. No ocupan realmente ese espacio en la simulación, produciendo que los círculos puedan intersectarse eventualmente y pasar unos sobre otros.

Una de las fuerzas del paquete sirve para esta situación: “d3.forceCollide”. Esta permite definir una fuerza que especifica un radio de colisión alrededor de cada nodo, de manera que otros nodos no se puedan acercar a esa distancia. Esto permite simular que los nodos ocupan espacio circular.

Si probamos con un valor de 10, se simulará que ocupan un radio de 10. Aprovecharé de aumentar el radio, también, de los círculos a ese valor y probar el código. Veremos que tal vez no cambia mucho ya que la mayoría de los nodos ya están alejados, pero se alcanza a ver al final dos nodos que se acercan colisionan sin traspasarse.

Además de estas hay otro tipo de fuerzas que proporciona “d3-force”, pero también es posible definir fuerzas por uno mismo, siguiendo las reglas e indicaciones que deja la documentación.

Es un buen ejercicio cambiar los atributos de fuerzas para ajustarlos a tu *dataset*, ya que pequeños cambios pueden hacer la diferencia. Pero al mismo tiempo pequeños cambios pueden romper la estabilidad lograda.

En pantalla vemos una simulación de fuerzas alterable en un cuaderno de Observable, creado por un ex-ayudante del curso, Hernán Valdivieso. Este permite alterar algunos atributos de fuerzas que acabamos de revisar, como por ejemplo: la distancia de enlaces; la posición del punto central; o el radio de colisión.

Vemos que en todos estos cambios leves la simulación cambia bastante. También la combinación de valores entre fuerzas es un componente a considerar. Un radio de colisión mayor a distancia de enlaces por ejemplo produce algo bastante extraño.

¡Por eso es importante entender que produce cada fuerza, y analizar qué valores pueden ayudar a tu situación!

Con eso termina el contenido de esta cápsula. Recuerda que si tienes preguntas, puedes dejarlas en los comentarios del video para responderlas en la sesión en vivo de esta temática. ¡Chao!