Cápsula 3: Reactivación de simulación

Hola, bienvenidxs a una cápsula del curso Visualización de Información. En esta hablaré sobre desactivación y reactivación de simulaciones.

Como habrás notado, automáticamente la simulación comienza y eventualmente termina, las posiciones de nodos dejan de actualizarse y se detienen. Ese es el comportamiento base de las simulaciones propuestas por “d3-force”, y lo usa cómo estrategia para que la simulación no funcione indefinidamente.

Como comentamos en una cápsula anterior, situaciones pueden aparecer donde la red es tan compleja y hay tantas fuerzas actuando en contra de unas de otras que esperar a un estado “estable” puede tomar mucho tiempo, y cómputo además. Por eso prefiere la opción de simulación por medio de un método de “enfriamiento” del sistema.

La simulación cuenta con un valor interno “alpha” que comienza con valor de 1 por defecto, y se modifica su valor en cada actualización de la simulación. En cada actualización se modifica el valor de “alpha” de manera que se acerque un poco a un valor objetivo “alphaTarget”, que es otro valor interno de la simulación, y por defecto es 0. La simulación continuará llamando actualizaciones y modificando “alpha” hasta que su valor sea menor que el atributo “alphaMin”, que por defecto es 0.001.

Podemos apreciar esto en vivo si se agrega una impresión en cada llamada de actualización. El código en pantalla imprimirá el valor del atributo “alpha” y también un valor booleano que indica si tal “alpha” es menor al atributo “alphaMin”.

Si lo probamos y revisamos la consola, podemos apreciar que las impresiones comienzan cerca del 1, y luego se reducen y reducen, intentando llegar a 0. Finalmente se detienen junto a los nodos de la simulación cuando este valor efectivamente es menor a 0.001*.

Saber esto es bueno ya que nos permite saber cómo funciona internamente, y eventualmente modificar estos valores para obtener algo distinto. Por ejemplo, una persona eventualmente puede no verse satisfecha con el resultado por la cantidad de actualizaciones que se logran por defecto, y quiere darle una oportunidad y ver qué pasa si se deja actualizando más tiempo.

Una forma directa de hacer eso es modificar el atributo “alphaTarget” de manera que sea mayor a “alphaMin”. Como la simulación intenta acercar “alpha” a “alphaTarget” nunca pasará el límite mínimo, y por lo tanto no se detendría la simulación. Podemos probar esto, y apreciamos rápidamente que nuestros nodos siguen moviéndose lentamente, pero de forma indefinida.

También podemos apreciar en las impresiones que se llega a un valor muy cercano al colocado como objetivo, 0.3, y en cada iteración siguiente “alpha” se mantiene constante en ese valor, pero aún actualizando.

Esto también sirve para reiniciar la simulación en caso de que por código o por interacción de usuario se deba reiniciar una simulación detenida. Pondremos esto en práctica agregando una de las interacciones más comunes con este tipo de *idiom*: el arrastrado de nodos.

Similar a otras definiciones de comportamientos, D3 provee una función “d3.drag” que define un objeto para arrastrar elementos. En este caso le entregaremos tres funciones gatilladas para tres eventos importantes en el arrastrado: el inicio (o “start”), un arrastre siendo ejecutado (o “drag”) y el final del arrastre (o “end”).

Llamaremos este objeto sobre la selección de círculos, que son los elementos a arrastrar. Antes de probar esto en ejecución, revisaremos las funciones gatillantes que fueron agregadas al comportamiento.

La intención es arrastrar nodos de forma que el usuario tenga algo de control por sobre la posición actual de los nodos, pero de forma que las posiciones relativas con el resto de los nodos respeten esta nueva posición y las fuerzas que les involucran.

Primero la función que se gatilla al inicio de un gesto de arrastre. Como potencialmente la simulación se detuvo, el inicio del arrastre es un buen momento para reiniciar la simulación en caso de que esta ya se haya detenido.

Esto hace la primera sección de código, revisa que el número de gestos de arrastres activos, además de este, sean cero, y modifica el el valor de “alphaTarget” a un valor mayor que “alphaMin”. Luego llama a “restart” que vuelve a comenzar la simulación en caso de que se haya detenido.

Luego altera atributos especiales de “subject” del evento, que hace referencia al dato asociado al elemento que se está arrastrando. En este caso, hace referencia a un objeto de nodo, y establece el valor de atributos “fx” e “fy”.

Estos atributos corresponden a posiciones especiales en simulaciones que se interpretan como fijas, y si un nodo las tiene, se considera como un nodo no movible. Lo que hace esta implementación es fijar el nodo en las posiciones que nos indique el arrastre, y luego se soltará al final. Al comienzo empieza con las coordenadas actuales “x” e “y” del nodo.

Si seguimos con esa idea, la función de arrastre que se llama mientras se arrastra al elemento lo que hace es actualizar los atributos mencionados según las posiciones indicadas por el evento.

Y finalmente la función de término devuelve el valor de “alphaTarget” a cero de forma que una vez terminada la interacción, la simulación continua y se enfría. Con el punto de inicio y

final contamos que la simulación seguirá funcionando mientras el gesto de arrastre esté en funcionamiento. Finalmente también se vuelven nulos los valores de los atributos especiales usados, para que el nodo ya no se considere fijo.

Si probamos el código, esperaré a que se enfríe la simulación inicial, y comenzaré arrastrando un nodo. Podemos ver que la simulación vuelve a comenzar por las impresiones, y que los nodos se actualizan de posición respetando las fuerzas definidas.

Podemos apreciar que mientras se arrastra el nodo, el valor de alpha se estanca cerca de 0.3, y al soltarse, empieza a volver a bajar y el nodo se ajusta rápidamente por las fuerzas.

Como mencionado, se hace uso de atributos especiales “fx” e “fy”, que si están presentes en un nodo, una simulación los considera como fijos, reduce su velocidad a 0 y deja sus coordenadas según el valor de ellos. Podemos intentar no usar estos atributos y usar los atributos regulares “x” e “y”. Pero al probarlo notarás que se pone bastante inestable debido a las fuerzas que calcula la simulación e intentan mover al nodo.

Por eso es mejor utilizar “fx” y “fy” en estos contextos, ya que en la simulación se consideran estáticos esos nodos y no son afectados por las fuerzas al mismo tiempo. Solo el resto de los nodos son afectados. Incluso podemos intentar no anulando sus valores al final, y al probarlo veremos que estos nodos quedarán estáticos por siempre.

Esto puede ser conveniente en algunos contextos: el permitirle al usuario ordenar la red por su cuenta. El problema es que ahora no es posible soltar los nodos, por lo que se debería agregar otra interacción que los “des-fije”. Intenta implementar esto sobre el código base entregado, pueden hacerlo mediante doble clic sobre un nodo, por ejemplo.

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!