DEFINICION FUNCIONAL ANIMACIONES

Descripción breve

Presentación de propuesta para la incorporación de animaciones en el producto móvil de Veritran.

Jorge Alberto López Márquez
jlopez@veritran.net

1. INTRODUCCION

Después de una serie de reuniones con Diego Acevedo, en el cual se revisaron cuáles eran las animaciones más "comunes" en las aplicaciones móviles, se determinó que no era tan fácil deducirlas ya que a nivel de prototipo se pueden hacer animaciones muy complejas a través de herramientas como after efects, pero llevar esto a la implementación real en los dispositivos móviles y con lenguaje nativo no es muy fácil. Además como se sabe nosotros no hacemos aplicaciones a la medida, por lo tanto se tiene que pensar en algo más genérico y que sea configurable.

A partir de este criterio se siguieron evaluando posibilidades y Diego encontró una herramienta para hacer prototipos de animaciones llamada Framer (https://framer.com/), la cual tiene como factor diferenciador, que permite hacer diseños a partir de herramientas visuales (forma clásica), pero también permite hacerlo por medio de código, lo que da un acercamiento de como una animación o efecto puede ser parametrizado.

Como tal la herramienta no permite exportar directamente a un programa de desarrollo como XCode, sin embargo el motor de animaciones está construido de manera muy similar a cómo funciona el motor de animaciones de iOS, lo que permite por ejemplo definir una animación de translación con una"Spring Curve", esta genera una parametria, la cual puede copiarse a la animación nativa de iOS.

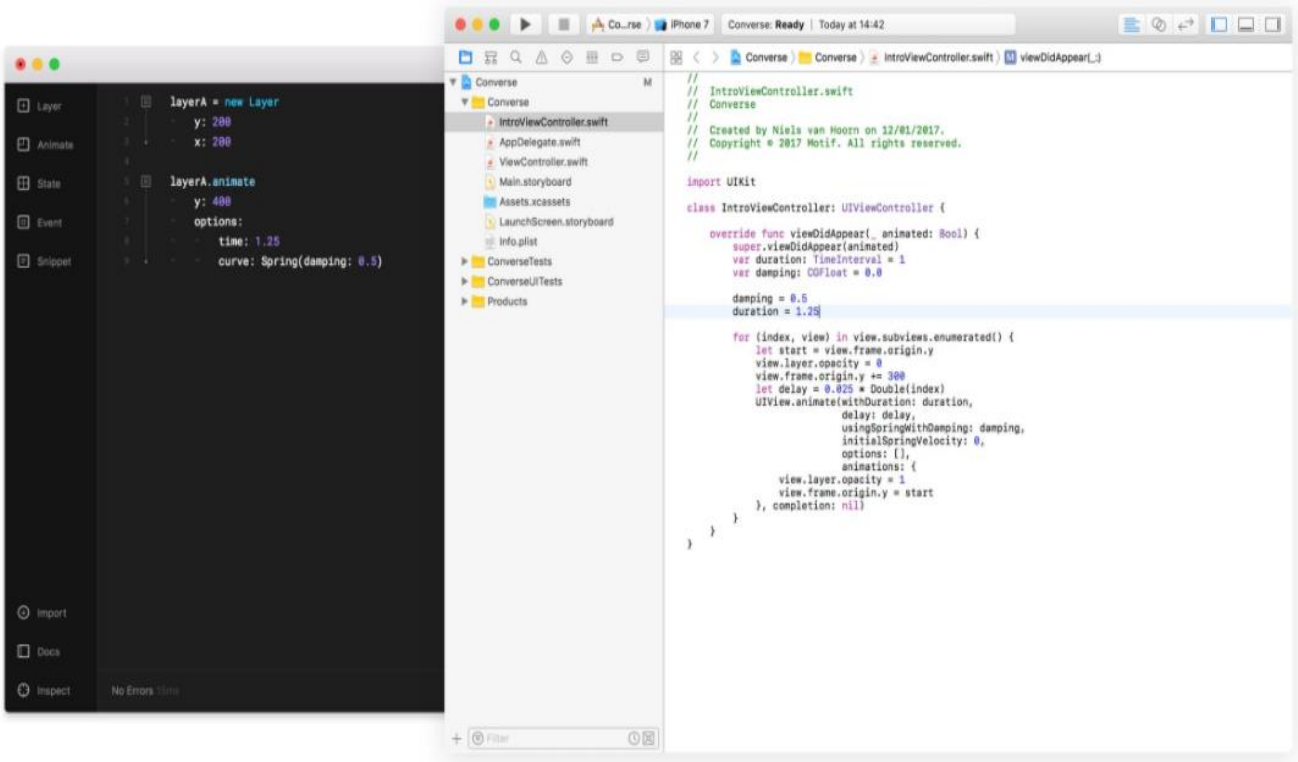

Figura 1

El objetivo como tal de la herramienta es hacer prototipado, y ofrece facilidades para ello basándose principalmente en algo denominado estados y eventos.

Al hacer una revisión más profunda de la herramienta descubrimos que a nivel de código utiliza un lenguaje (la librería está hecha en javascript), que facilita su lectura y migración a otros lenguajes.

A continuación se muestra un ejemplo de cómo se configura un par de animaciones. Esta animación permite rotar, trasladar, escalar y hacer aparecer un objeto.

```
circulo.opacity= 0

circulo.animate
    rotation: -90
    x: 50
    scale: 1.8
    opacity: 1.00
    options:
        time: 1
        curve: Bezier.easeInOut

circulo.animate
    y: 100
    options:
        time: 2
        delay: 1
        curve: Spring(damping: 0.23)
```

Figura 2

Los parámetros que conforman la animación son casi los mismos parámetros que se usan en iOS para configurar una animación de forma nativa, solamente hay que “copiar” y “pegar” ciertos parámetros para lograr el mismo efecto.

2. Parámetros para lograr animaciones en Framer

Framer cuenta principalmente con dos secciones para parametrizar una animación.

En la primera sección se encuentran los diferentes parámetros que se pueden animar del objeto en cuestión. Estos se pueden modificar conjuntamente logrando que se animen diferentes propiedades a la vez.

En la segunda sección se encuentran los parámetros propios de la animación, donde se puede configurar entre otros su duración, la curva (aceleración) etc.

A partir de esto se puede lograr por ejemplo una animación de 1 segundo, donde se pueda modificar la posición, la rotación y el escalamiento de un objeto.

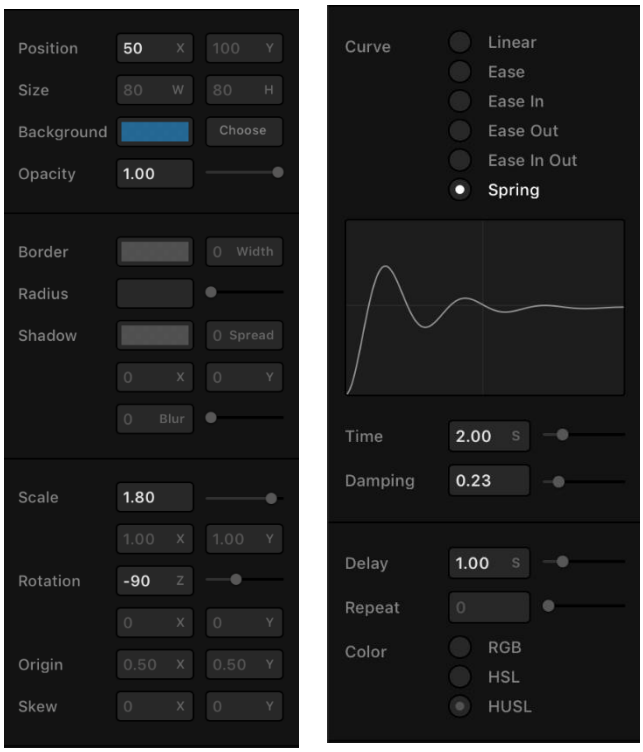

Figura 3

Con animaciones de este tipo se pueden lograr comportamientos como los siguientes (Haz Click aquí para ver la animación):

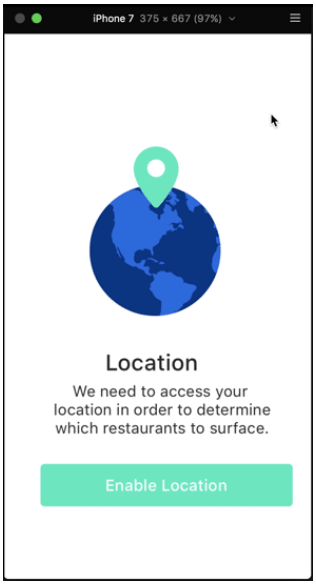

Figura 4

Si queremos comportamientos un poco más complejos podemos crear varias animaciones con diferentes tiempos y decidir si queremos que las animaciones se ejecuten simultáneamente o una después de la otra.

3. Animaciones en las plataformas móviles (Nativas)

A partir de estos análisis procedimos a realizar una investigación y pruebas de concepto en las plataformas como tal y ver nativamente como se construyen las animaciones (Revisamos Android y iOS por el momento). En ambas plataformas se tiene en común que se pueden animar las siguientes propiedades:

- Position→ Permite desplazar un objeto. (Animaciones de translación).
- Scale → Permite escalar un objeto, es decir aumentar o disminuir su tamaño inicial. (Animaciones de escalamiento)
- Rotation → Permite "rotar" un objeto. (Animaciones de rotación)
- Alpha → Permite modificar la opacidad de un objeto. (Animaciones tipo FadeIn, FadeOut)

Notas Importantes:

- Las rotaciones y escalamientos en esencia son Transformaciones, hay que hacer más prueba de concepto para validar que pasa cuando se hacen estas transformaciones en vistas que tienen subvistas.
- Como ya se ha dicho en iOS, el motor de animaciones funciona con casi los mismos parámetros que framer, en Android el asunto es un poco diferente pero la mayoría de los atributos tienen su equivalente, estamos en investigación con el tema de las curvas (Aceleraciones), pero esperamos poder lograr el mismo comportamiento que en iOS de una manera transparente para el configurador.

4. Definición Funcional animaciones Veritran.

A partir de todo lo anterior proponemos la siguiente definicional funcional para agregar animaciones a nuestra plataforma.

1. Permitir animar cualquier VTComponent.
2. Agregar a todos los VTComponent las siguientes propiedades:
    a. InitialAlpha: → Alpha inicial con el que se será dibujado el VTComponent en la pantalla. Este valor será de 0 a 1, siendo 0 el valor que da más transparencia.
    b. InitialRoration: → Rotación inicial con la que será dibujado el VTComponent en la pantalla. Su valor será dados en Grados.
    c. InitialScale: → Escalamiento inicial de la imagen. Este valor indica el factor de escalamiento de una imagen, siendo 1 la escala natural del objeto.
3. Tener una sección en el designer para la configuración de animaciones, donde se podrán crear, editar y eliminar.
4. Las animaciones se crearán independientemente a los VTComponents, es decir estas podrán ser creadas de tal forma que puedan ser invocadas por cualquier VTComponent a través de un mensaje. (Ver punto 6) permitiendo su reutilización.
5. Al momento de crear una animación se solicitarán los siguientes parámetros:

Parámetro	Tipo	Observación
DeltaX		Posición inicial en x, de la posición donde partirá la animación. Su valor será un delta respecto al parámetro aleft del designer, el cual es el parámetro de la posición final de la animación. (Animación de translación)
DeltaY		Posición inicial en y, de la posición donde partirá la animación. Su valor será un delta respecto al parámetro atop del designer, el cual es el parámetro de la posición final de la animación. (Animación de translación)
Alpha	Double	Parámetro que indica la opacidad final del objeto. Este valor ira de 0 a 1. Donde 0 es transparencia. La animación será una transición entre el valor que haya sido configurado en el parámetro InitialAlpha (Ver punto 2a de esta sección)
Scale	Double	Factor que indica la escala final del objeto después de realizar la animación. Este valor estará en el rango de 0 a 2, donde 0 indica que el objeto tendrá una escala de 0, 1 si queremos que el objeto tenga su tamaño normal (el configurado con el width y height), si queremos que el objeto tenga un tamaño más grande podemos poner cualquier valor superior a 1. Si configuramos 2 quiere decir que queremos que el objeto tenga el doble de su tamaño original. La animación será una transición entre el valor que haya sido configurado en el parámetro InitialScale (Ver punto 2c de esta sección)
Rotation	double	Parámetro que indica el grado de rotación final del objeto después de la animación. Este valor podrá tener tanto valores positivos como negativos que irán

		del 0 al 360. Siendo 0 el valor que indica que el objeto no tendrá ninguna rotación. La animación será una transición entre el valor que haya sido configurado en el parámetro initialRotation (Ver punto 2b de esta sección)
Duration	Doublé	La duración total de la animación medida en segundos. Si se especifica un valor negativo o igual a 0, no se ejecutara con animación.
Delay	Double	La cantidad de tiempo medida en segundos, que se espera antes de que comience la animación. Especificar un valor de 0 para iniciar la animación inmediatamente.
Curve	String	Curva a utilizar en la animación. (Este punto hay que mirarlo bien si aplica para todas las animaciones)
Repeat	String	Número de veces que se quiere repetir la animación. Si se especifica un valor INFINITE, la animación se ejecutara indefinidamente.

   a. Nota: Se está evaluando la posibilidad de manejar los parámetros de ScaleX y ScaleY, para hacer escalamiento independiente en los ejes.
   b. Se está evaluando la posibilidad de tener un WidthFinal y un HeightFinal para tener animaciones de crecimiento de los objetos. (Estas animaciones tendrían sentido en un comienzo en los Paneles)

6. Además de los parámetros del punto anterior, las animaciones tendrán un groupID y un nameID, las cuales podrán ser invocadas por medio de mensajes tipo G, o E. Permitiendo que puedan ser llamas cuando el implementador lo requiera y permitiendo invocar varias animaciones de varios componentes de manera simultánea.

Con esta primera parte estaríamos cubriendo el tema de las animaciones simples, con lo que se lograría comportamientos como este o este.

Si queremos permitir hacer animaciones más complejas, en las cuales se necesite agregar más de una animación a la vez sobre un VTComponent, se debe tener una sección denominada “Complex Animations” o “Effects”, en la cual podamos crear una animación compleja la cual estará conformada por animaciones simples. Cada que se cree una animación simple, esta tendrá un parámetro adicional denominado nextAnimation, donde se indique el nameID de la animación que se inicializará justo después de terminar la actual. Si se quiere que las animaciones se ejecuten simultáneamente, o con un delay especifico, no se colocara nada en el campo nextAnimation.

Las animaciones y/o efectos que se pueden crear con esta son del tipo:

https://github.com/daltoniam/DCAnimationKit.

Si quieres ver estos o más ejemplos haz clic aquí, para ver una presentación más interactiva.