RE:CICLO. DESARROLLO DE APP DE JUEGO CON INTERACCIONES GESTUALES EN SWIFT

ALBERTO BOTANA[1]

ÍNDICE

ÍNDICE DE FIGURAS

[1] *Master en Sistemas Inteligentes, Universidad de Salamanca, España. Correo:* alberto.botana@usal.es

1 INTRODUCCIÓN

En este informe se elabora una recapitulación del proceso de diseño y construcción de *Re:Ciclo*, un juego basado en reciclaje para iPad. Para ello se ha utilizado el entorno Xcode 11 y el lenguaje de programación Swift 5. El objetivo de este juego es que el usuario obtenga el mayor número de puntos en un tiempo determinado, para lo cual tendrá que llevar a cabo distintas interacciones gestuales, que le permitirán conseguir mayor cantidad de puntos cuanto mayor sea la complejidad de la tarea.

1.1 Reglas del juego

El objetivo de *Re:Ciclo* es simple: Una vez empieza la partida, el jugador dispone de 60 segundos en los que tiene que obtener el mayor número de puntos posible. Para ello se le presentan tres tipos de basura: Una botella de vidrio, un envase ligero (En este caso, una bolsa de patatas fritas) y un avión de papel. Al arrastrar estos objetos al contenedor correspondiente a cada material (azul para el papel, verde para el vidrio y amarillo para los envases ligeros) el jugador obtendrá un número base de puntos: 75 para el papel y los envases ligeros y 100 para el vidrio. Esto es debido a que, para poder introducir el vidrio en el contenedor, el jugador deberá voltearlo boca abajo.

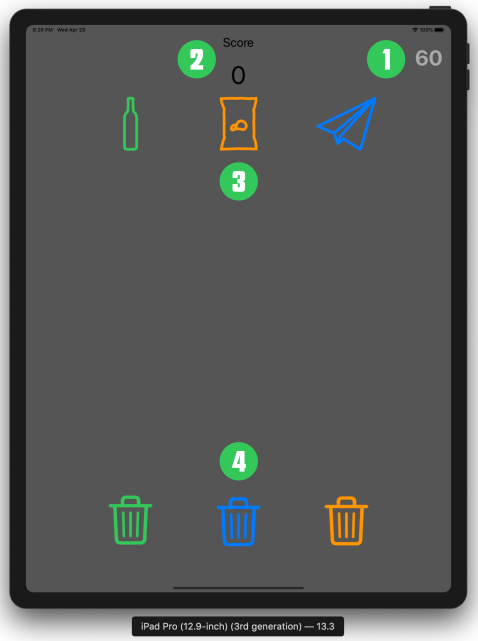

Figura 1: Captura de la pantalla inicial del juego, con sus elementos marcados con un número. 1: Contador de tiempo. 2: Marcador de puntuación. 3: Basura (Vidrio, envase ligero y papel, respectivamente). 4: Contenedores (Correspondientes a vidrio, papel y envase ligero, respectivamente.)

Si el jugador quiere conseguir todavía más puntos, tendrá la opción de ampliar el tamaño de la basura haciendo un gesto de pellizco. De esta forma, se multiplicará el número de puntos por el tamaño actual de la basura, pero deberá tener cuidado, dado que si la basura es demasiado grande, no cabrá en el contenedor y no podrá ganar puntos. Si esto ocurre, el jugador tendrá la oportunidad de reducir un trozo de basura a la mitad de tamaño haciendo doble toque en el objeto en cuestión.

Cuando el jugador arrastre el trozo de basura a su contenedor correspondiente, éste desaparecerá, momento en el cual el jugador podrá tocar la pantalla para generar más, pero no podrá generar más de un elemento por cada tipo de basura. Una vez el contador de tiempo llegue a cero, se habrá acabado la partida y no podrá depositar más basura en los contenedores.

En la figura 1 se presenta un esquema resumido de los elementos de juego presentes y su función.

2 DESGLOSE DEL PROGRAMA

En este programa se incluyen diversos tipos de interacción gestual, para los cuales hubo que recurrir a su propio método de introducción:

- Arrastre con un dedo (*Pan*): Este gesto se utilizará para desplazar los objetos por la pantalla. Para implementar este gesto, se crea una función para cada objeto en la cual se designa la vista sobre la que trabajar, así como la transformación que se producirá. A continuación, se mueve el centro del objeto en las coordenadas X e Y un valor igual al que se ha desplazado la transformación en el último período de muestreo, tras lo cual devuelve el valor de la transformación a 0 para evitar acumulaciones indeseadas. Esto se lleva a cabo mientras haya cambios en la transformación, o lo que es lo mismo, mientras el dedo se esté moviendo por la pantalla. Una vez se levante el dedo, se verifica si el objeto está intersecando con el contenedor de basura correspondiente, en cuyo caso se hace desaparecer el objeto, bajando su opacidad al 0 % al poner su valor de alpha a 1.

- Toque único (*Tap*): Con este gesto el usuario podrá generar nuevos objetos para interaccionar cuando los haga desaparecer al depositarlos en el contenedor correcto. Para implementarlo, se crea una función del tipo `UITapGestureRecognizer` que, al hacer un toque, genere un número aleatorio entre 1 y 3, asignado cada uno a uno de los objetos con los que interaccionar. Con este número se llama a la función correspondiente al objeto, la cual moverá su posición de vuelta a su punto de partida, su tamaño a su escala original y, en el caso de la botella, su rotación a un ángulo aleatorio.

- Pellizco con dos dedos (*Pinch&Zoom*): Mediante este gesto el usuario podrá aumentar o disminuir el tamaño de la basura con la que interaccione. Para implementarlo se crea una función del tipo `UIPinchGestureRecognizer` por cada tipo de objeto. Esta función registrará la presencia de dos dedos, y aumentará el ancho y alto del objeto por un valor proporcional a la separación o acercamiento de los dedos por cada período de muestreo, se delimitan los nuevos límites de la imagen y finalmente se notifica a Swift que el objeto necesita ser redibujado. Tras esto, se inicializa de nuevo la escala para evitar acumulaciones en la transformación. Esto se lleva a cabo mientras se detecten la presencia de los dos dedos y estén ocurriendo cambios entre muestreos.

- Doble toque (*Double Tap*): Este gesto se implementa de forma análoga sl toque único, llamando a una función de `UITapGestureRecognizer` pero configurando esta vez el número de toques necesarios en 2 en lugar de 1. Con este gesto el usuario reduce a la mitad la escala del objeto con el que interaccione. Para ello se lleva a cabo una transformación análoga a la del pellizco, pero multiplicando el tamaño por la cantidad fija de 0,5 en lugar de depender de muestreos anteriores.

- Rotación con dos dedos: Utilizando esta interacción el usuario podrá rotar los objetos para, en este caso, poder introducir las basuras de vidrio en su contenedor correspondiente. El usuario podrá también rotar las demás basuras, pero sin consecuencias algunas. Para implementar este gesto, se crea una función para cada objeto del tipo `UIRotationGestureRecognizer` que, al detectar el toque de dos dedos, seguirá la variación del ángulo de la trayectoria que

une los dos puntos de toque en cada período de muestreo, y rotará el objeto por una cantidad dada por la rotación de la trayectoria antes definida. Esto se hace mientras se detecten cambios en la posición de los dedos en la pantalla.

Los gestos que requieren comprobaciones continuas como el arrastre, la rotación y el pellizco se implementan mediante las funciones descritas y, tras ello, se llaman en la sección `viewDidLoad` del código utilizando otra función auxiliar que añade un delegado para poder reconocer funciones simultáneas, así como actuar de intermediario entre la ejecución del programa y el código. Esta función auxiliar tiene la siguiente forma (En este ejemplo, aplicado a la implementación de gesto de rotación para el objeto de la botella):

```
func addRotateBottle(view: UIView){
        let rotBot = UIRotationGestureRecognizer(target: self, action: #selector(
    ViewController.rotateBottle(sender:)))
        rotBot.delegate = self
        view.addGestureRecognizer(rotBot)
    }
```

Para los gestos de arrastre o pellizco se implementa de forma análoga, refiriendo a la clase correspondiente a su gesto. Cabe destacar que, para poder implementar dos gestos de manera simultánea, como el de rotación a la vez que el pellizco para ampliar, es necesario declarar junto con la clase `ViewController` el conjunto de métodos `UIGestureRecognizerDelegate` junto con una función que declare explícitamente la necesidad de que se puedan recoger dos gestos a la vez, siguiendo la forma:

```
func gestureRecognizer(_ gestureRecognizer: UIGestureRecognizer,
    shouldRecognizeSimultaneouslyWith otherGestureRecognizer: UIGestureRecognizer)
     -> Bool {return true}
```

Inmediatamente después de lanzar la aplicación, lo primero que el usuario ve es un menú, presente en la figura 2, dando la opción de empezar directamente la partida o de leer las reglas antes de hacerlo. Este menú se implementó utilizando la clase `UIAlertController`, pero teniendo en cuenta que, en iPad, estas alertas sólo se pueden implementar utilizando el estilo `.actionSheet`, el cual está pensado para aparecer saliendo de un elemento de interfaz. Como en este programa, no hay ningún elemento del que tenga sentido salir, se creó un elemento virtual invisible en el centro de la pantalla del cual tenga que salir, imitando así una alerta por defecto de iOS. Para ello, es necesario introducir este extracto de código en todas las instancias de alerta del programa:

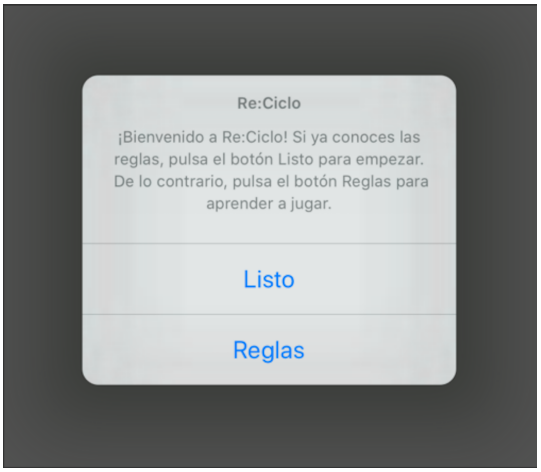

Figura 2: Pantalla de inicio de la aplicación.

```
if let popoverController = alert.popoverPresentationController {
          popoverController.sourceView = self.view
          popoverController.sourceRect = CGRect(x: self.view.bounds.midX, y: self.view
    .bounds.midY, width: 0, height: 0)
          popoverController.permittedArrowDirections = []
        }
```

Por último, es necesario decir que para que la botella apareciese siempre boca arriba, es necesario llevar a cabo una transformación del tipo `.identity` para devolverla a su rotación inicial, dado que esto ha dado problemas en las fases de pruebas al intentar devolverla al estado inicial mediante rotaciones de ángulo inverso. Al intentar hacerlo de esta forma, no sólo se consumía memoria innecesaria, sino que además era un método poco preciso y propenso a dar problemas. Mediante la transformación propuesta, se solventa este problema de forma limpia e inmediata.

3 CONCLUSIONES

En este informe se ha desglosado el proceso de desarrollo de la app *Re:Ciclo*, un juego de iPad con temática de reciclaje, con el objetivo final de implementar una serie de interacciones gestuales entre el usuario y el programa. Se ha llevado a cabo un desglose de las reglas de juego así como de las distintas implementaciones de cada uno de los gestos presentes en el juego: Toque único, toque doble, arrastre, pellizco con dos dedos y rotación con dos dedos. Además, se ha explicado el procedimiento llevado a cabo para poder implementar dos gestos a la vez en un mismo objeto, como en el caso de rotación y pellizco para poder girar y ampliar un objeto a la vez.

Para futuras versiones de la app, sería interesante poder conseguir la funcionalidad originalmente planteada del gesto de doble toque: No sólo reducir la escala de un objeto a la mitad, sino dividirlo en dos copias con este tamaño, "partiéndolo" efectivamente por la mitad. Además, quedó por corregir un problema con los *constraints* de la app, ya que si se ejecutaba la app en un dispositivo distinto del iPad Pro de 12,9 pulgadas, dispositivo para el cual se diseñó originalmente, los elementos de la basura aparecen desplazados. Esto se intentó solucionar colocándoles *constraints* a estos objetos, pero mediante eso, si bien se solucionaba ese problema, se impedía el movimiento de dichos objetos frente a interacciones. Como mejora menor, para futuras versiones podría intentar mejorarse la interfaz gráfica, así como intentar aumentar el tamaño de la pantalla de reglas, dado que la legibilidad de las alertas es un poco reducida debido a su tamaño de letra.