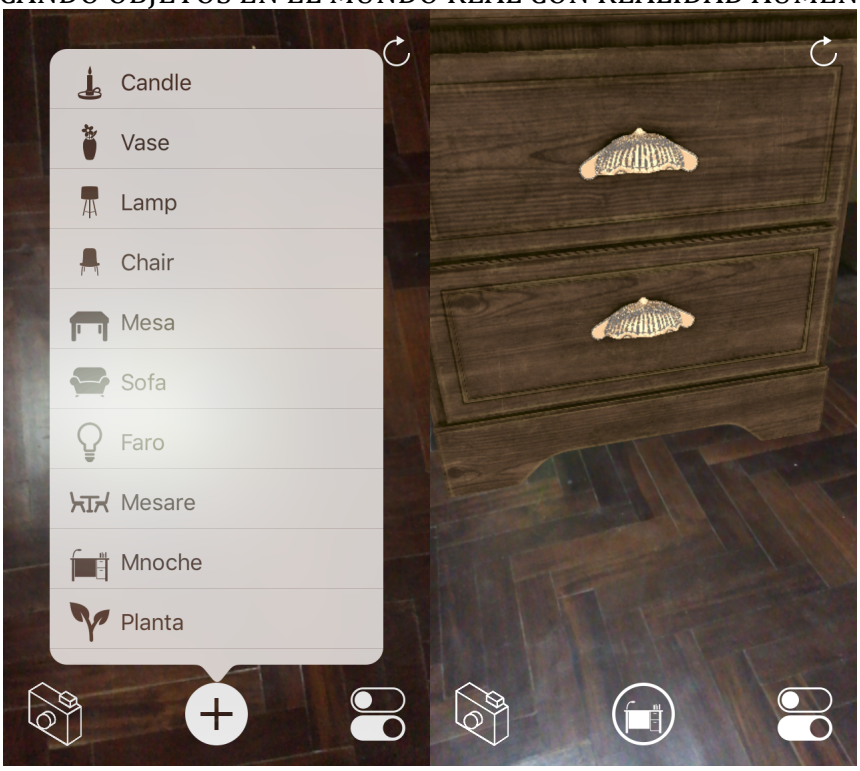
ARKit Maker
COLOCANDO OBJETOS EN EL MUNDO REAL CON REALIDAD AUMENTADA



Instalación

Solo clona el repositorio

https://github.com/lasmc/ARKit-Maker.git

Requisitos
Utilizar XCode 9 cualquier version

ARKit está disponible en cualquier dispositivo con iOS 11, pero las funciones de rastreo mundial que permiten experiencias de AR de alta calidad requieren un dispositivo con el chip A9 o un procesador posterior.

Lista de dispositivos disponibles

* iPhone 6s
* iPhone 6s Plus
* iPhone SE
* iPhone 7
* iPhone 7 Plus
* iPhone 8
* iPhone 8 Plus
* iPhone X / iPhone 10
* El iPad 2017 de 9.7 pulgadas
* Todas las variantes del iPad Pro

NOTAS: Este proyecto fue probado en un iPhone SE
Para los objetos 3D se utilizo blender

Visión de conjunto
La realidad aumentada ofrece nuevas formas para que los usuarios interactúen con contenido 3D real y virtual en su aplicación. Sin embargo, muchos de los principios fundamentales del diseño de la interfaz humana siguen siendo válidos. Las ilusiones convincentes de AR también requieren una cuidadosa atención al diseño y renderización de activos en 3D. Al seguir las pautas de este artículo para los principios de la interfaz humano AR y experimentar con este código de ejemplo, puede crear experiencias inmersivas e intuitivas de realidad aumentada.

Realimentación

** Ayude a los usuarios a reconocer cuándo su aplicación está lista para interacciones en el mundo real. **
El seguimiento del entorno del mundo real implica algoritmos complejos cuya puntualidad y precisión se ven afectadas por las condiciones del mundo real.

La clase `FocusSquare` en este proyecto de ejemplo dibuja un contorno cuadrado en la vista AR, dando al usuario pistas sobre el estado del world tracking de ARKit. El cuadrado cambia de tamaño para reflejar la profundidad de escena estimada, y cambia entre estados abierto y cerrado con una animación de "bloqueo" para indicar si ARKit ha detectado un plano adecuado para colocar un objeto.

Utilice el método de delegado [`session (_: cameraDidChangeTrackingState:)` (https://developer.apple.com/documentation/arkit/arsessionobserver/2887450 session) para detectar cambios en la calidad del seguimiento, y presente comentarios al usuario cuando baja- las condiciones de calidad se pueden corregir (por ejemplo, al decirle al usuario que se mueva a un entorno con mejor iluminación).

Use términos específicos que un usuario pueda reconocer. Por ejemplo, si da retroalimentación textual para la detección de plano, un usuario que no esté familiarizado con las definiciones técnicas podría confundir la palabra "plane" con una referencia a un avión.

Retroceda con elegancia si el seguimiento falla y permita al usuario restablecer el seguimiento si su experiencia no funciona como se espera. Vea el botón y método `restartExperience` en la clase` ViewController` de este ejemplo. La variable `use3DOFTrackingFallback` controla si se cambia a una configuración de sesión de menor fidelidad cuando la calidad del seguimiento es deficiente.

** Ayude a los usuarios a comprender la relación entre el contenido virtual de su aplicación y el mundo real. ** Use señales visuales en su UI que reaccionen a los cambios en la posición de la cámara en relación con el contenido virtual.

El cuadro de enfoque desaparece después de que el usuario coloca un objeto en la escena y reaparece cuando el usuario apunta la cámara lejos del objeto.

La clase `Plane` en este ejemplo maneja la visualización de planos del mundo real detectados por ARKit. Sus métodos `createOcclusionNode` y` updateOcclusionNode` crean una geometría invisible que oscurece de forma realista el contenido virtual.

Manipulación directa

** Proporcione gestos comunes, familiares para los usuarios de otras aplicaciones iOS, para interactuar con objetos del mundo real. ** Consulte la clase `Gesto` en este ejemplo para implementaciones de los gestos disponibles en esta aplicación de ejemplo, como arrastrar un dedo hacia mueve un objeto virtual y rota dos dedos para girar el objeto.

Asigna gestos táctiles a un espacio restringido para que el usuario pueda controlar los resultados más fácilmente. Los gestos táctiles son intrínsecamente bidimensionales, pero una experiencia de AR implica las tres dimensiones del mundo real. Por ejemplo:

- Limite el objeto arrastrando al plano bidimensional en el que se encuentra el objeto. (Especialmente si un avión representa la tierra o el suelo, a menudo tiene sentido ignorar el alcance del avión mientras se arrastra).

- Limita la rotación de objetos a un solo eje a la vez. (En este ejemplo, cada objeto descansa sobre un plano, por lo que el objeto puede rotar alrededor de un eje vertical).

- No permita que el usuario cambie el tamaño de los objetos virtuales u ofrezca esta habilidad solo con moderación. Un objeto virtual habita el mundo real de manera más convincente cuando tiene un tamaño intrínseco intuitivo. Además, un usuario puede confundirse en cuanto a si están cambiando el tamaño de un objeto o cambiando su profundidad con respecto a la cámara. (Si proporciona el cambio de tamaño del objeto, use gestos de pellizco).

Mientras el usuario está arrastrando un objeto virtual, suavice los cambios en su posición para que no parezca saltar mientras se mueve. Consulte el método `updateVirtualObjectPosition` en la clase` ViewController` de este ejemplo para ver un ejemplo de suavizado según la distancia percibida desde la cámara.
Establezca umbrales para los gestos para que el usuario no active un gesto accidentalmente, pero modere sus umbrales para que los gestos no sean demasiado difíciles de descubrir o desencadenar intencionadamente. Consulte la clase `TwoFingerGesture` para ver ejemplos del uso de umbrales para elegir dinámicamente los efectos de gestos.

Proporcione un área lo suficientemente grande donde el usuario pueda tocar (o comenzar un arrastre) en un objeto virtual, de modo que aún pueda ver el objeto

mientras lo mueve. Vea cómo se calcula el booleano `firstTouchWasOnObject` en la clase` TwoFingerGesture` para ver ejemplos.

** Diseñe interacciones para situaciones donde las ilusiones de AR pueden ser más convincentes. ** Por ejemplo, coloque contenido virtual cerca de los centros de los planos detectados, donde es más seguro asumir que el plano detectado coincide con la superficie del mundo real. Puede ser tentador diseñar experiencias que utilicen la superficie completa de una mesa, donde los elementos virtuales de la escena pueden reaccionar o caerse de los bordes de la mesa. Sin embargo, el seguimiento mundial y la detección de plano pueden no estimar con precisión los bordes de la tabla.

Control de usuario

** Esfuércese por lograr un equilibrio entre ubicar con precisión el contenido virtual y respetar la información del usuario. ** Por ejemplo, considere una situación en la que el usuario intente colocar contenido que debería aparecer sobre una superficie plana.

- Primero, intente colocar contenido utilizando el método [`hitTest (_: types:)`] (https://developer.apple.com/documentation/arkit/arframe/2875718-hittest) para buscar una intersección con un ancla de plano Si no encuentra un ancla de plano, todavía puede haber un plano en la ubicación de destino que aún no se haya identificado por detección de plano.
- Al carecer de anclaje del plano, puede realizar pruebas de impacto contra las características de la escena para obtener una estimación aproximada de dónde ubicar el contenido de inmediato y refinar esa estimación a medida que ARKit detecta los planos.
- Cuando la detección de plano proporciona una mejor estimación de dónde colocar el contenido, utilice la animación para mover sutilmente ese contenido a su nueva posición. Tener contenido colocado por el usuario de repente saltar a una nueva posición puede romper la ilusión AR y confundir al usuario.
- Filtra los resultados de la prueba de aciertos que están demasiado cerca o muy lejos. En la mayoría de los escenarios, existe un límite razonable de cuán lejos se puede colocar el contenido virtual. Para evitar que los usuarios accidentalmente coloquen contenido virtual demasiado lejos, puede utilizar la propiedad `distance` de` ARHitTestResult` para filtrar las pruebas de aciertos que exceden el límite.

** Evite interrumpir la experiencia de AR. ** Si el usuario realiza una transición a otra IU de pantalla completa en su aplicación, la vista de AR podría no ser un estado esperado al volver.

Utilice la presentación de Popover (incluso en iPhone) para los controladores de vista auxiliares para mantener al usuario en la experiencia de AR mientras ajusta las configuraciones o realiza una selección modal. En este ejemplo, las clases `SettingsViewController` y` VirtualObjectSelectionViewController` usan la presentación de popover.

Prueba

Para probar y depurar las experiencias de AR, ayuda tener una visualización en vivo del procesamiento de la escena que realiza ARKit. Vea el método `showDebugVisuals` en la clase` ViewController` de este proyecto para la visualización del world tracking, y la clase `HitTestVisualization` para una demostración de los métodos de detección de características de ARKit.

Mejores prácticas y limitaciones

El world tracking es una ciencia inexacta. Este proceso a menudo puede producir una precisión impresionante, lo que lleva a experiencias realistas de AR. Sin embargo, depende de los detalles del entorno físico del dispositivo que no siempre son consistentes o son difíciles de medir en tiempo real sin algún grado de error. Para crear experiencias AR de alta calidad, tenga en cuenta estas advertencias y consejos.

** Diseñe experiencias de AR para condiciones de iluminación predecibles. ** El seguimiento mundial implica el análisis de imágenes, que requiere una imagen clara. La calidad del seguimiento se reduce cuando la cámara no puede ver detalles, como cuando la cámara apunta a una pared en blanco o la escena es demasiado oscura.

** Use la información de calidad de seguimiento para proporcionar comentarios de los usuarios. ** El world tracking correlaciona el análisis de imágenes con el movimiento del dispositivo. ARKit desarrolla una mejor comprensión de la escena si el dispositivo se está moviendo, incluso si el dispositivo se mueve solo sutilmente. El movimiento excesivo, demasiado rápido, excesivo o tembloroso, produce una imagen borrosa o demasiada distancia para el seguimiento de las características entre los marcos de video, lo que reduce la calidad del seguimiento. Los ARCamera class proporciona información sobre el estado de seguimiento de la información, que puede usar para desarrollar la interfaz de usuario que le dice a un usuario cómo resolver situaciones de seguimiento de baja calidad.

** Permita tiempo para que la detección de plano produzca resultados claros y desactive la detección de plano cuando tenga los resultados que necesita. ** Los resultados de detección de plano varían con el tiempo: cuando se detecta por primera vez un plano, su posición y extensión pueden ser inexactas. A medida que el plano permanece en la escena a lo largo del tiempo, ARKit refina su estimación de posición y extensión. Cuando hay una gran superficie plana en la escena, ARKit puede seguir cambiando la posición, el alcance y la transformación del anclaje plano una vez que ya haya utilizado el plano para colocar el contenido.