GIG AR - IR Augmented Reality - Image Recognition

GIGIGO-ECONOCOM

31 JANUARY 2019

EDUARDO PARADA PARDO

Table of contents

01

Limitaciones

- Modelo de móvil, versión mínima: **iphone 6s**
- Versión mínima de sistema operativo para *IR*: **iOS 11.3**
- Versión mínima de sistema operativo para traqueo de imágenes: **iOS 11.3**
- Versión mínima de sistema operativo para *AR*: **iOS 12**
- Versión mínima de sistema operativo para traqueo de objetos: **iOS 12**

02

Creación de modelos

a) Crear una carpeta. *Ejemplo*: CreacionModelo.

b) Añadir los archivos copySceneKitAssets y scntool en la carpeta.

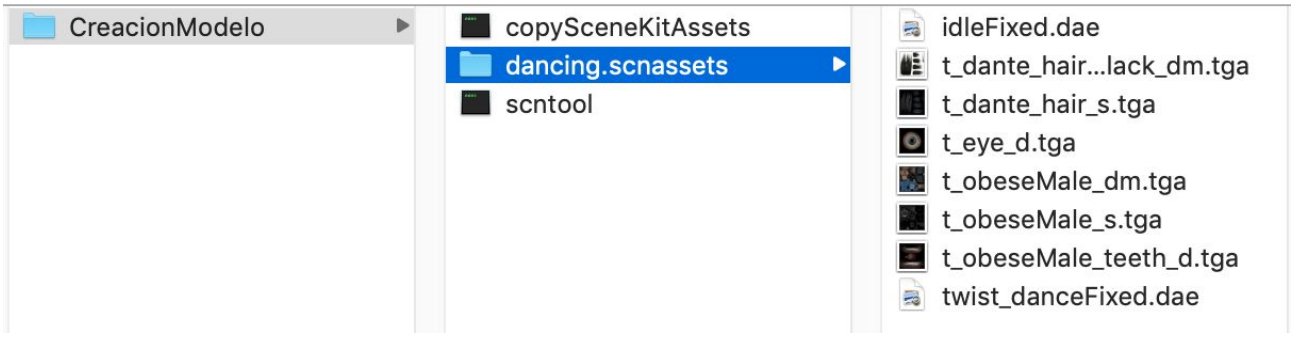
kitArchivos.zip

c) Dentro crear una carpeta acabada en .scnassets con el nombre del modelo.

Ejemplo: dancing.scnassets

d) Añadir dentro de esa carpeta el modelo, animaciones y texturas al mismo nivel:

Ejemplo:



e) Abrimos el terminal en la raíz de nuestra carpeta, y escribimos lo siguiente:

Estructura:

```
./copySceneKitAssets NombreCarpetaAssets -o NuevaCarpetaOptimizada
```

Ejemplo:

```
./copySceneKitAssets dancing.scnassets -o dancing-optimizated.scnassets
```

f) Creamos un archivo comprimido de tipo zip, de esta carpeta y es **muy importante**, no cambiar el nombre del archivo zip.

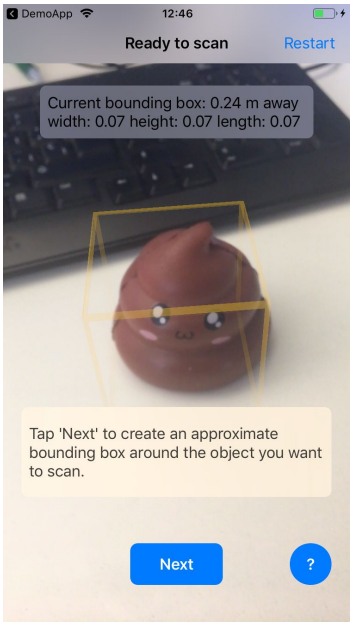
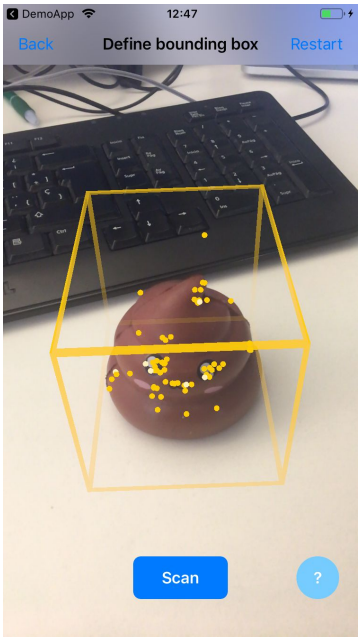
d) Este archivo lo subiremos al servidor.

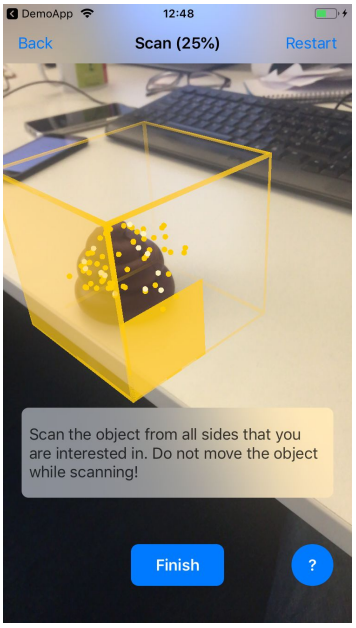
03

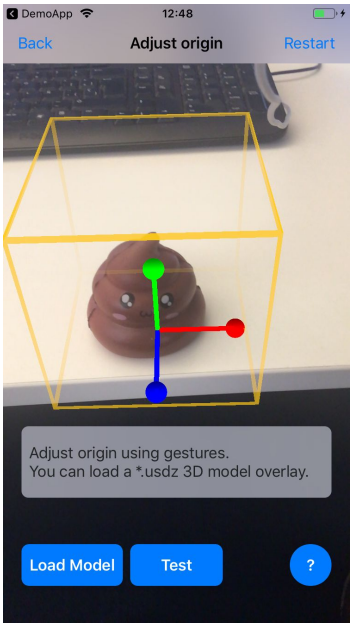
Escaneo de objetos 3D

Para poder escanear un objeto y poder realizar un reconocimiento o trackeo de un objeto 3D, debemos crear este objeto, para ello usaremos la app oficial de Apple que encontraremos en la carpeta de documentation -> ScaningObject.

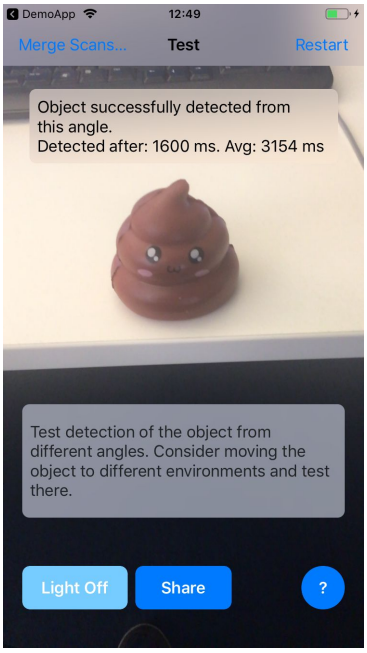
Pasos para uso:

<table>
<tr>
<td>1. Detectar el objeto
 
</td>
<td>2. Ajustar ancho, importante, ampliar el tamaño para que tenga aproximadamente 2 cuadrículas, si el área es pequeña cuando llegue al final del proceso os informará que no tiene suficientes puntos para generarlo.
 
</td>
</tr>
<tr>
<td>3. Escanear el objeto por sus 4 caras</td>
<td>4. Ajustar el eje de coordenadas</td>
</tr>
</table>

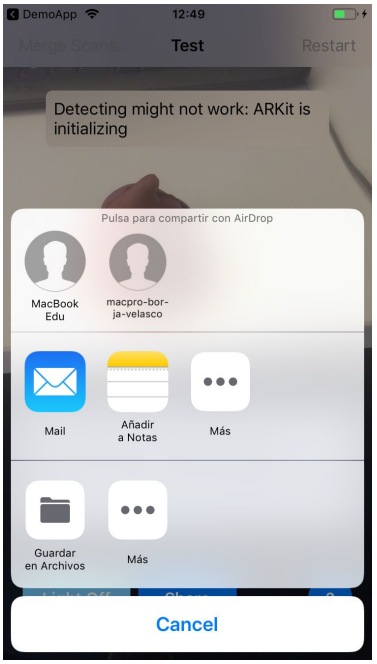




5. Informe de que el objeto se ha generado correctamente.



6. Descargarlo compartiéndolo con el email o similar.



7. El archivo resultante será un archivo con formato: *.arobject*

04

GIG Ar-Ir

1. Instalación

a. A través del Cartfile, añadir la siguiente referencia:
Añadir referencia

b. Descargar solo la plataforma de ios:
carthage update --platform iOS

c. Agregar en embebed binary los siguientes frameworks:
Añadir imagen del proyecto

d.

2. Uso

a. Importar el framework de ARKit y crear una instancia de IRAR

b. Llamar al constructor IRAR(id:) con el identificador del proyecto y asignarle su delegado. Opcionalmente puedes definir el logLevel.

c. El método **Start,** descargara la configuración de la librería y cuando acabe podrás lanzar el evento de *launch* para empezar a utilizar la librería.

d. Existen 2 opciones de uso, que el sdk se ocupe de IR y AR ó que te entregue los modelos según se requiera.

   i. SDK:
      1. Para lanzar el método de **Launch**, antes desde tu storyBoard, hay que crear una instancia de un *ARSCNView*, lincarla a tu ViewController y añadir como parámetro del método.

      2. El método delegado "*func idRecognition(id: String)*" devolverá el Id de la imágen reconocida.

ii. Aplicación integradora:

1. Primero recuperaremos el listado de ID con los que descargar el modelo a través del metodo público:
   open func getListID() -> [String]?

2. Segundo, recuperar el modelo con el identificador dado:
   open func get3DModel(id: String)

   Recuperaremos el modelo a través del método delegado:
   func model3D(result: ResultGet3DModel)

3. Json configuración

A continuación se definirá los diferentes modos en los que puede funcionar el framework.

a. Modo reconocimiento de imágenes (Antiguo Vuforia).

El nodo del json contará con el siguiente formato:

```
{
        "image": "https://s3-eu-west-1.amazonaws.com/Images/cats.jpg",
        "idReco": "IdReconocimiento",
        "type": "ir"
}
```

Image: Es el elemento que busca reconocer
idReco: Es el identificador que devolverá una vez reconocido
type: Identifica el tipo de objeto que busca reconocer.

b. Reconocimiento objetos 3D

Próximamente.

c. IR/AR + Acción.

Todas las acciones, tendrán una serie de variables comunes:

1) type. Tipos de acción: *model3D*, *text*, *video*
2) source

Ejemplo tipo texto:

```
{
	"image": "https://s3-eu-west-1.amazonaws.com/cats.jpg",
	"idReco": "Image_AR3",
	"type": "ir",
	"action": {
		"type": "text",
		"source": "TEXTO DESDE SERVIDOR"
	}

}
```

Ejemplo tipo modelo 3D*:*

idModel: identifica el modelo 3d que queremos mostrar dentro de nuestro assets
idNodo: Identifica el nombre del objeto dentro del modelo.

```
{
	"image": "https://s3-eu-west-1.amazonaws.com/gigigo.jpg",
	"idReco": "Image_with_3DModel",
	"type": "ir",
	"action": {
		"type": "model3D",
		"source": "https://s3.com/dancing-optimizated.scnassets.zip",
		"idModel": "idleFixed.dae",
		"idNodo": "node/11"
	}
}
```

Cuando queremos mostrar modelos 3D, tenemos una serie de elementos opcionales, como **animaciones**, **posición** y **escala**.

Ejemplo modelo 3D versión completa.

```
{
	"image": "https://s3-eu-west-1.amazonaws.com/gigigo.jpg",
	"idReco": "Image_with_3DModel",
	"type": "ir",
	"action": {
		"type": "model3D",
		"source": "https://s3.com/dancing-optimizated.scnassets.zip",
		"idModel": "idleFixed.dae",
		"idModelAnim": "twist_danceFixed.dae",
		"idNodo": "node/11",
```

```
                "idNodoAnim": "twist_danceFixed-1",
                "styles": {
                        "scale": {
                                "x": 0.1,
                                "y": 0.1,
                                "z": 0.1
                        },
                        "position": {
                                "x": 0,
                                "y": 0,
                                "z": 0
                        }
                }
        }
}
```


d.


gigigo
econocom