Universidad Nacional de la Patagonia San Juan Bosco

Facultad de Ingeniería

Tesina de grado

Representación de Objetos del Mundo Real mediante Realidad Aumentada y Modelos 3D

Alumnos:

Litterio Sassaroli Marcos Damián

Ginzery Axel Christian

Tutor: Saenz Lopez Marta

Tesina presentada a la Facultad de Ingeniería de la Universidad Nacional de la Patagonia San Juan Bosco como parte de los requisitos para la obtención del título de Licenciado en Informática

Trelew - 2017

Agradecimientos

“Vive como si fueras a morir mañana.
Aprende como si fueras a vivir siempre.”
Mahatma Gandhi

Agradecemos a nuestra familia por el apoyo durante toda nuestra carrera.
A Marta Saenz Lopez en particular que fue quien nos guió durante este proceso.
A nuestros compañeros y profesores de la carrera que formaron parte fundamental para nuestro desarrollo como profesionales y como personas.

Índice

Índice de figuras

Índice de tablas

Glosario de Términos

API - Application Programming Interface (*Interfaz de Programación de Aplicaciones*).
ART - Android *Runtime* (E*ntorno de Ejecución de Android*).
BUA - Bottom-Up Approach (*Aproximación de Abajo hacia Arriba*).
CAD - Computer Aided Design (*Diseño Asistido por Computadora*).
CRT - Cathode Ray Tube (*Tubo Rayos Catódicos*).
DVM - Dalvik Virtual Machine (*Máquina Virtual Dalvik*).
EV- Entorno Virtual (*Virtual Environment*).
GPS - Global Positioning System (*Sistema de Posicionamiento Global*).
GUI - Graphical user interface (*Interfaz Gráfica de Usuario*).
HAL - Hardware Abstraction Layer (*Capa de Abstracción del Hardware*).
HMD - Head Mounted Display (*Visor Montado sobre la Cabeza*).
HUD - Heads-Up Display (*Visualización sin Bajar la Vista*).
IDE - Integrated Development Environment (*Entornos de Desarrollo Integrado*).
JDK - Java Development Kit (*Kit de Desarrollo Java*).
JVM - Java Virtual Machine (*Máquina Virtual Java*).
MIT - Massachusetts Institute of Technology (*Instituto de Tecnología de Massachusetts*).
MMS - Multimedia Messaging Service (*Servicio de Mensajería Multimedia*).
NURBS - Non Uniform Rational B-splines (*B-splines Racionales no Uniformes*).
OHA - Open Handset Alliance.
RA - Reality Augmented (*Realidad Aumentada*).
SAGE - Semi Automatic Ground Environment (*Operador Ambiental Semiautomático*).
SDK - Software Development Kit (*Kit de Desarrollo de Software*).
SMS - Short Message Service (*Servicio de Mensajes Cortos*).
SRU - System Reference Universal (*Sistema Coordenadas Universal*).
SSL - Secure Socket Layer (*Capa de Sockets Seguro*).
TDA - Top-Down Approach (A*proximación de Arriba hacia Abajo*).
XML - Extensible Markup Language (*Lenguaje de Etiquetado Extensible*).

1. Preliminares

1.1. Motivación

Muchas veces realizamos búsquedas de productos para comprar en la web (como televisores, computadoras, muebles, etc.) y nos encontramos con gran incertidumbre a la hora de decidir qué producto elegir: ¿Entrará en el lugar en que lo quiero ubicar? ¿Combina con el resto del mobiliario que tengo en mi hogar? O, simplemente, no sabemos bien de qué se trata ese producto. Esta problemática es muy común hoy en día debido a la expansión de los Sistemas de Compra Web.

En busca del tema para abordar en la Tesina, nos encontramos con una temática muy interesante como lo es la Realidad Aumentada (RA), que permite crear visualmente objetos virtuales sobre el mundo real y que tiene un espectro de aplicación muy amplio. Analizando su campo de aplicación, notamos que es capaz de resolver las inquietudes planteadas anteriormente cuando se realizan compras de productos vía web.

Pero aún nos estaba faltando la forma de representación de los objetos virtuales. Leyendo al respecto, encontramos que junto con RA se pueden integrar objetos en tres dimensiones mediante el Modelado 3D.

Con estos primeros conceptos de RA y Modelado 3D nos planteamos que sería de mucha utilidad crear una aplicación que resuelva el problema, y no sólo eso, sino que también brinde comodidad al usuario al poder utilizarla en un dispositivo móvil.

El crecimiento en la tecnología de los dispositivos móviles nos permite tener buenas prestaciones de hardware al alcance de una gran cantidad de usuarios, logrando que pueda funcionar de manera fluida una aplicación que requiere un procesamiento gráfico elevado, como es el caso del renderizado 3D[1] en conjunto con la RA.

Además, hoy en día los sistemas operativos que tienen los distintos dispositivos móviles nos brindan su propia plataforma de distribución digital masiva de aplicaciones, denominadas tiendas virtuales. Esto permite que el software pueda ser descargado directamente por los usuarios e instalado de manera sencilla.

[1] Renderizado 3D: proceso de generar una imagen o video mediante el cálculo de iluminación GI (global ilumination) partiendo de un modelo en 3D, utilizándolo como simulación realista del comportamiento tanto de luces, texturas y materiales, entre otras utilidades.

1.2. Objetivos

A continuación detallamos los objetivos de la presente Tesina:

Objetivos generales:

- Ofrecer un software para dispositivos móviles, que pueda ser utilizado por los Sistemas de Compra Web, o por catálogos impresos, y sus clientes para visualizar, de forma sencilla e interactiva, los productos ofrecidos de forma virtual en una escala y ubicación acorde al ambiente.

- Proporcionar una plataforma de descarga para la aplicación de fácil acceso para los usuarios.

Objetivos específicos:

- Desarrollar una aplicación para dispositivos móviles, bajo el sistema operativo Android, que muestre un producto en el mundo real, representado mediante un modelo 3D a escala real, utilizando tecnología de Realidad Aumentada y la cámara de un dispositivo móvil.

- Desarrollar los modelos 3D de algunos productos.

- Proporcionar un conjunto de impresiones de códigos de los productos a los vendedores, a fin de ser distribuidos a los posibles clientes.

- Aportar a la comunidad de código abierto un desarrollo que pueda ser utilizado y adaptado por aquellos interesados en resolver problemáticas similares.

1.3. Estado del arte

En la actualidad, con el avance de la globalización y el uso de las tecnología, ha variado mucho el hábito de las personas al momento de adquirir productos. Muchas personas, en diferentes lugares del mundo, están adoptando métodos de compra distintos al tradicional (compra in situ), como la compra vía Internet mediante tiendas oficiales, plataformas de venta como OLX, Mercado Libre, Ebay, etc., incluso compran a través de redes sociales o por teléfono.

De forma paralela a estas nuevas tendencias de compras, ha crecido el uso de dispositivos móviles. El consumo masivo de éstos ha logrado generar cierta dependencia en los usuarios, ya

que tienen al alcance de la mano un conjunto de aplicaciones amplio donde pueden acceder a mapas, navegadores de Internet, redes sociales, tiendas virtuales, juegos, cámaras de fotos, entre otras.

Por otro lado, el crecimiento del uso de los dispositivos móviles exige que éstos cada vez sean más potentes. Hoy en día, el hardware que poseen los dispositivos permite que se pueda realizar desde edición de imágenes o videos, hasta utilizar motores gráficos para juegos. Una de las aplicaciones de juego más relevante en el último tiempo fue el lanzamiento del juego Pokemon Go, basado en RA, donde los usuarios de la aplicación deben capturar criaturas virtuales creadas en base a modelos en 3D, que aparecen aleatoriamente en el mundo real, siguiendo la posición geográfica que brinda el GPS del dispositivo móvil. Lo innovador de este juego fue la incorporación de RA, lo cual le da cierta “sensación” al usuario que las criaturas virtuales están en el mundo real.

A través de la historia existieron muchos dispositivos de RA pero no brindaban comodidades a la hora de utilizarlos por el gran tamaño que tenían. Con el avance de la tecnología y las capacidades de hardware, que permitieron tener más potencia de procesamiento en dispositivos más pequeños, la RA comenzó a volverse una realidad al alcance de casi cualquier persona. Actualmente, con los dispositivos móviles, como tablets o celulares, sumado a las prestaciones del sistema operativo Android, existen aplicaciones de RA para una gran variedad de usos diferentes, por ejemplo en el marco educativo los estudiantes pueden utilizar plataformas interactivas que les permiten manipular objetos en 3D como si existieran en la realidad.

2. Marco Teórico

A lo largo del presente capítulo mostramos los conceptos principales utilizados en nuestro desarrollo, con el fin proporcionar una base de conocimiento que enmarque las cuestiones técnicas inherentes a la problemática planteada en esta Tesina.

Primero partimos de la definición del sistema operativo elegido como base para el uso de la aplicación, con el fin de comprender las necesidades o limitaciones de lo que consideramos el punto de partida para el desarrollo de la aplicación.

Posteriormente, explicamos el concepto de Realidad Aumentada, entendiendo, analizando y conociendo esta tecnología para aplicarla en el desarrollo del presente proyecto.

Por último, realizamos una introducción al desarrollo de Modelos 3D, ya que si bien es una tarea asociada a diseñar gráficamente modelos, éstos son utilizados por la aplicación para que los usuarios puedan verlos dentro del entorno.

2.1. Android

"Si hay varias maneras para realizar la misma tarea,
elija sólo una. Tener dos o más formas
de hacer lo mismo es buscarse problemas."
Andrew Tanenbaum

Figura 1: Logo Android

Fuente: http://www.freepik.es/vector-gratis/logo-de-android_683826.htm

El primer paso para comenzar a hablar de Android es comprender qué es y cómo está conformado. Android es un sistema operativo y una plataforma de software, desarrollado por Google, basado en una versión modificada del kernel de Linux. Orientado en sus comienzos a

dispositivos móviles de pantalla táctil, como celulares o tablets, pero que en la actualidad, gracias a su elevado crecimiento, su versatilidad y disponibilidad, tiene módulos desarrollados para televisores, computadoras, relojes y hasta electrodomésticos, entre otros (Figura 2).

Figura 2: Variedad de aplicaciones en Android

Fuente: https://www.elandroidelibre.com/wp-content/uploads/2012/07/multitaskingEnAndroid.png

La interfaz de usuario de Android está basada principalmente en la manipulación directa de la pantalla táctil mediante gestos que se corresponden con acciones del mundo real, tales como el deslizamiento o la simple pulsación con un dedo, utilizados para manipular los distintos objetos de la pantalla. Además, cuenta con un teclado virtual para brindar la opción de introducción de texto de una manera eficiente.

Para los desarrolladores, Android proporciona todas las herramientas y frameworks necesarios para desarrollar aplicaciones móviles de una forma rápida y sencilla. A la hora de comenzar a desarrollar, una de sus ventajas es que no es necesario ningún dispositivo físico, ya que el SDK[2] de Android incluye hasta un emulador del mismo, por lo que sólo el SDK es todo lo que se necesita para comenzar a incursionar en la programación para dispositivos móviles. Existen numerosas herramientas, tal como Android Studio, por ejemplo, que se pueden utilizar y ayudan al desarrollo de aplicaciones.

Uno de los puntos fuertes que lo diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets, o incluso, modificar el propio sistema operativo, dado que Android es de código libre.

El código fuente de Android está disponible bajo diversas licencias de software libre y código abierto. Google liberó la mayoría del código de Android bajo la licencia Apache. Todo esto permite que un desarrollador no sólo pueda modificar su código sino también mejorarlo. A través de esas mejoras puede publicar el nuevo código y, con él, ayudar a mejorar el sistema operativo para futuras versiones.

[2] SDK: sigla inglés de Software Development Kit que significa Kit de Desarrollo de Software.

2.1.1. Arquitectura

Una vez realizada la introducción para comprender Android y antes de comenzar con cualquier tipo de desarrollo, es esencial conocer cómo está estructurado este sistema operativo con el cual vamos a trabajar.

Android tiene una arquitectura que está formada por varias capas, que le brinda una gran facilidad a los desarrolladores a la hora de crear las aplicaciones. Este tipo de distribución les permite acceder a las capas más bajas mediante el uso de librerías, sin necesidad de que deban programar a bajo nivel las funcionalidades necesarias para que la aplicación haga uso de los componentes de hardware de los dispositivos móviles.

A este tipo de arquitectura se le conoce como pila, ya que cada una de las capas utiliza elementos de la capa inferior para realizar sus funciones. Para entenderlo mejor, podemos observar la Figura 3, en la que se visualizan todas las capas y componentes principales de la plataforma Android:

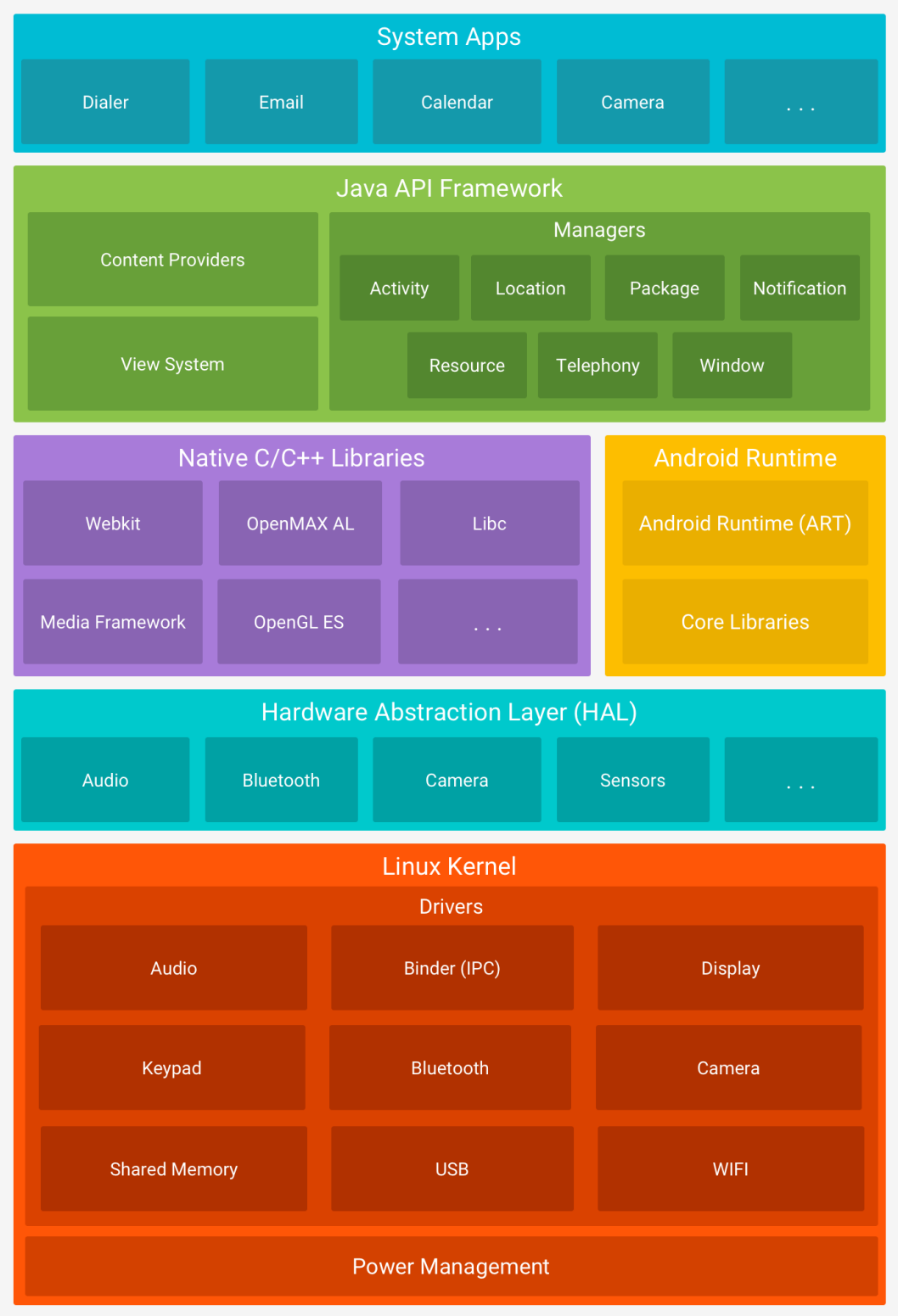


Figura 3: Capas y componentes principales de la plataforma Android

Fuente: https://developer.android.com/guide/platform/index.html

A continuación realizamos una breve descripción de las distintas capas y sus componentes [1], presentados en la Figura 3.

2.1.1.1. System Apps (Aplicaciones)

En esta capa se incluyen todas las aplicaciones del dispositivo, tanto las que tienen interfaz de usuario como las que no, las nativas (programadas en C o C++), las que vienen preinstaladas en el dispositivo y aquellas que el usuario ha instalado. Dentro de estas aplicaciones se encuentran la casilla de correo, mensajería SMS, calendarios, navegadores de Internet, contactos, entre otras.

También se halla la aplicación principal del sistema: inicio (home) o lanzador (launcher), que permite ejecutar otras aplicaciones. El launcher presenta distintas formas para acceder al resto de las aplicaciones, como los diferentes escritorios, con sus correspondientes accesos directos y/o widgets, y un menú de aplicaciones con una lista de las mismas.

2.1.1.2. Java API Framework

Esta capa, antiguamente denominada Framework de Aplicación, está formada por todas las clases y servicios que las aplicaciones utilizan directamente para realizar sus funciones. La mayoría de los componentes de esta capa son un conjunto de herramientas y librerías Java que permiten su reutilización. Los desarrolladores tienen acceso completo a todos los componentes de esta capa, y los pueden reemplazar y aplicar a su manera.

Sus principales componentes son:

- **Contents Provider**: Esta librería crea una capa que encapsula los datos que se compartirán entre aplicaciones para tener control sobre cómo se accede a la información. De esta forma, le permite a cada aplicación acceder a datos de otras aplicaciones o compartir sus propios datos.

- **View System**: Ayuda en la construcción de la aplicación, más precisamente en la creación de las interfaces de usuario: botones, cuadros de texto, listas y hasta elementos más avanzados, como un navegador web o un visor de Google Maps.

- **Activity Manager**: Se encarga de administrar la pila de actividades de cada aplicación, así como su ciclo de vida.

- **Location Manager**: Determina la posición geográfica del dispositivo Android mediante el Sistema de Posicionamiento Global (GPS) o mediante el uso de las redes disponibles, permitiendo la utilización de los distintos puntos geográficos para el trabajo en conjunto con mapas.

- **Package Manager**: Esta librería permite obtener información sobre los paquetes instalados en el dispositivo Android, además de gestionar la instalación de nuevos paquetes. Con paquete nos referimos a la forma en que se distribuyen las aplicaciones Android. Cada paquete contiene el archivo .apk, que a su vez incluye los archivos .dex con todos los recursos y archivos adicionales que necesite la aplicación, para facilitar su descarga e instalación.

- **Notification Manager**: Engloba los servicios que le permiten a una aplicación emitir alertas para notificar al usuario cuando algo requiera su atención, mostrando alertas en la barra de estado. Esta librería también permite la ejecución de sonidos, activar el vibrador o hasta utilizar los LEDs del teléfono (en caso de tenerlos).

- **Resource Manager**: Esta librería permite gestionar todos los elementos que forman parte de la aplicación y que están fuera del código, es decir, cadenas de texto traducidas a diferentes idiomas, archivos de imágenes, sonidos o layouts[3].

- **Telephony Manager**: Administra las comunicaciones, permitiendo realizar llamadas o enviar y recibir SMS/MMS, aunque no deja reemplazar o eliminar la actividad que se muestra cuando una llamada está en curso.

- **Window Manager**: Organiza lo que se mostrará en pantalla. Básicamente crea las superficies en la pantalla que posteriormente pasarán a ser ocupadas por las actividades.

2.1.1.3. Native C/C++ Libraries (Librerías Nativas)

Esta capa se sitúa justo sobre la capa HAL y está compuesta por las librerías nativas de Android, también llamadas bibliotecas. Estas librerías están escritas en C y/o C++ y compiladas para la arquitectura de hardware específica de cada dispositivo. Tienen como objetivo proporcionar funcionalidad a las aplicaciones para tareas repetidas con frecuencia, logrando un

[3] Layout: Estructura visual para una interfaz de usuario, es decir, aquello que hace de intermediario entre el dispositivo móvil y el usuario.

mejor desempeño y evitando código redundante. Exponen su funcionalidad a través de la API del framework de Java.

Entre las librerías incluidas y más conocidas, podemos mencionar a OpenGL (motor gráfico 3D), SSL (cifrado de comunicaciones), bibliotecas multimedia (formatos de audio, imagen y video), Webkit (navegador), FreeType (fuentes de texto), SQLite (base de datos) y SGL (motor gráfico).

2.1.1.4. Android Runtime - ART (Entorno de Ejecución Android)

Android Runtime no se considera una capa en sí, sino que es una subcapa situada dentro de la capa de Librerías. Se trata de un software de ejecución de procesos, encargado de simular una computadora para ejecutar los programas que han sido creados en Java.

Se basa en una variación de la máquina virtual de Java (JVM)[4], llamada Máquina Virtual Dalvik (DVM). Esta máquina proporciona un entorno de ejecución independiente de la plataforma de hardware y el sistema operativo.

Como DVM produce un gasto rápido de la memoria del dispositivo, a partir de la versión 5.0 de Android (Lollipop), fue reemplazado por ART (Android Run Time), con el fin de aumentar la velocidad y el rendimiento de las aplicaciones, la duración de la batería, y mejorar el funcionamiento general del sistema operativo.

ART tiene las mismas funciones que su predecesor pero con mejoras, debido a que posee un sistema de compilación diferente: compila los procesos y guarda el caché[5] desde el momento de la instalación de la aplicación. En Dalvik la aplicación se va compilando a medida que se trabaja con ella.

2.1.1.5. Hardware Abstraction Layer - HAL (Capa de Abstracción del Hardware)

[4] JVM (Java Virtual Machine): conocida como máquina virtual de Java, es un entorno de ejecución que posibilita la abstracción de la ejecución de los programas desarrollados en Java, permitiendo así la portabilidad entre diferentes sistemas operativos.

[5] La memoria caché de un procesador es un tipo de memoria volátil, de velocidad muy alta, que tiene la función de almacenar instrucciones y datos a los que el procesador debe acceder continuamente (datos que sean de acceso instantáneo para el procesador), ya que se trata de información relevante y que debe ser accedida de manera muy fluida.

Como su nombre lo indica, esta capa sirve como una interfaz estándar que le permite al sistema operativo Android hacer uso de los distintos componentes de hardware sin necesidad de conocerlos en profundidad, ni conocer la implementación a bajo nivel de los controladores de los mismos.

Está compuesta por varios módulos de biblioteca, cada uno de los cuales posee una implementación de una interfaz para un tipo específico de componente de hardware, como el módulo para la cámara o el del GPS. Cada vez que el framework de una API realiza una llamada que requiera el acceso al hardware del dispositivo, el sistema Android, a través de esta capa, carga el módulo específico para el componente de hardware en cuestión.

2.1.1.6. Linux Kernel (Núcleo Linux)

Como especificamos anteriormente, la base de la plataforma Android, o sea el núcleo del sistema operativo, está basado en el kernel de Linux versión 2.6, similar al que puede usar cualquier distribución de Linux, como Ubuntu, por ejemplo, sólo que Android está adaptado a las características del hardware de los dispositivos móviles en el que se ejecutará.

La funcionalidad de esta capa es muy importante ya que actúa como abstracción entre el hardware del dispositivo y el resto de las capas de la arquitectura. El desarrollador no accede a esta capa directamente, sino que debe hacer uso de las distintas librerías que se disponen en las capas superiores. De esta forma se abstrae de las características precisas de cada dispositivo móvil, permitiéndole desligarse de esa tarea y, así, enfocarse sólo en el desarrollo de la aplicación. Para ejemplificar esta situación tomemos el caso en el que el programador desea utilizar la cámara del dispositivo: gracias a Android no es necesario que conozca las características o funcionamiento de la misma a bajo nivel, sino que mediante la realización de una solicitud en la capa superior, es el sistema operativo el que se encarga de utilizar el controlador indicado para la cámara.

En Android se tiene un controlador (driver) dentro del kernel por cada elemento de hardware del dispositivo, que permite que éste sea utilizado desde el software.

Esta capa, además, tiene la función de gestionar los diferentes recursos del teléfono (memoria, energía, etc.) y los propios del sistema operativo en sí (elementos de comunicación, procesos, etc.).

2.1.2. Historia

La historia de Android se remonta al año 2003, cuando el sistema fue desarrollado por Android Inc., una empresa pequeña que en 2005 fue comprada por Google, quien continuó

desarrollando el sistema junto a los miembros de la Open Handset Alliance (liderada por Google).

La presentación oficial fue el 5 de noviembre de 2007 junto a la Fundación de la Open Handset Alliance (OHA), una alianza comercial de numerosas compañías de hardware, software y telecomunicaciones, comprometidas con la promoción de estándares abiertos para dispositivos móviles.

Entre el momento de la adquisición de Android Inc. y 2007, Google liberó las versiones Alfa y Beta, que no fueron versiones finales y ni siquiera existían dispositivos comerciales que pudieran utilizarlas, aunque en su momento sí se hicieron públicos los SDK para que la comunidad de desarrolladores comenzara a probarlos y a programar aplicaciones para ellos. Por entonces no existía ninguna versión comercial del sistema y Google continuó liberando actualizaciones del código hasta septiembre de 2008, momento en que libera la primer versión comercial de Android.

2.1.2.1. Versiones

Antes de comenzar cualquier desarrollo en Android es necesario elegir la versión del sistema para la cual se va a desarrollar la aplicación. Es muy importante destacar que existen clases y métodos que sólo están disponibles a partir de cierta versión, por lo que si se los va a usar, se ha de conocer la versión mínima necesaria.

Siempre que se ha lanzado una nueva versión ha sido compatible con las anteriores. Esto quiere decir que sólo se añaden nuevas funcionalidades, y en el caso de que se modifique alguna funcionalidad, no se elimina, sino que sólo se etiqueta como obsoleta pero, aún así, se puede continuar utilizándola.

A lo largo de la historia de Android se presentaron diferentes versiones. Cada versión se identifica de tres formas alternativas: versión, nivel de API y nombre comercial. Para los nombres comerciales se han elegido postres, en orden alfabético, asociados al número de versión, que detallamos en la Tabla 1. Como se puede ver en dicha tabla, las dos primeras versiones, que hubieran correspondido a las letras A y B, no recibieron nombre.

Nombre	Versión	API	Fecha de lanzamiento
Android (Letra A)	1.0	1	23 de Septiembre - 2008
Android (Letra B)	1.1	2	9 de Febrero - 2009
Cupcake	1.5	3	27 de Abril - 2009
Donut	1.6	4	15 de Septiembre - 2009
Eclair	2.0 - 2.1	5 - 7	26 de Octubre - 2009
Froyo	2.2 - 2.2.3	8	20 de Mayo - 2010
Gingerbread	2.3 - 2.3.7	9 - 10	6 de Diciembre - 2010
Honeycomb	3.0 - 3.2.6	11 - 13	22 de Febreo - 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	18 de Octubre - 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	9 de Julio - 2012
KitKat	4.4 - 4.4.4 - 4.4w - 4.4w.2	19 - 20	31 de Octubre - 2013
Lollipop	5.0 - 5.1.1	21 - 22	12 de Noviembre - 2014
Marshmallow	6.0 - 6.0.1	23	5 de Octubre - 2015
Nougat	7.0 - 7.1.2	24 - 25	22 de Agosto - 2016

Tabla 1: Versiones Android

A la hora de seleccionar la versión mínima utilizada para el desarrollo de nuestra aplicación tuvimos en cuenta, no sólo el rendimiento, sino además la popularidad de la misma. En el caso del rendimiento, la versión 5.0 “Lollipop” es adecuada por la mejora conseguida en los tiempos de ejecución gracias a la introducción de ART, mencionada anteriormente. En cuanto a la selección por popularidad, analizando los datos de la Tabla 2 y la Figura 4, podemos observar que justo esa versión es la segunda más usada a nivel histórico, después de la versión 6.0 “Marshmallow” que se impuso este último año. Teniendo en cuenta todo esto, optamos por la versión 5.0 “Lollipop” para tener un margen más amplio de dispositivos que puedan utilizar la aplicación.

Version	Nombre	API	Distribución
2.3.3 - 2.3.7	Gingerbread	10	0.7%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.7%
4.1.x - 4.2.x - 4.3	Jelly Bean	16	2.8%
		17	4.1%
		18	1.2%
4.4	KitKat	19	17.1%
5.0 - 5.1	Lollipop	21	7.8%
		22	22.3%
6.0	Marshmallow	23	31.8%
7.0 - 7.1	Nougat	24	10.6%
		25	0.9%

Tabla 2: Distribuciones Android

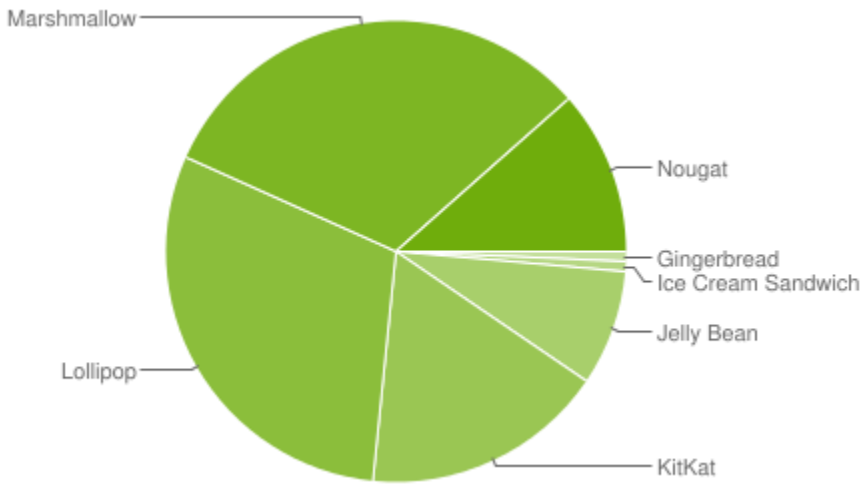


Figura 4: Gráfico Distribución
Fuente: https://developer.android.com/about/dashboards/index.html

2.1.3. Google Play Store

Es la plataforma de distribución digital para las aplicaciones desarrolladas para el sistema operativo Android, que cuenta con una tienda online desarrollada y operada por Google. La plataforma tiene un sistema de navegación que permite la búsqueda y descarga de aplicaciones, juegos, música, libros, revistas y películas.

En los inicios de las versiones comerciales de Android, Google creó el 28 de agosto de 2008 una aplicación llamada *Android Market*, que fue puesta a disposición de los usuarios unos meses más tarde, el 22 de octubre de 2008. En ese momento aún no contaba con soporte para aplicaciones pagas, lo cual fue introducido el 13 de febrero de 2009 para los desarrolladores de Estados Unidos y el Reino Unido.

Unos años más tarde, el 6 de marzo de 2012, fue lanzado nuevamente el servicio pero con un cambio de apariencia y con el nombre que actualmente conocemos: *Google Play Store*. Este cambio no fue solamente de nombre y apariencia, además se incorporaron nuevos servicios como Play Libros, Play Juegos, Play Kiosco, Play Video, entre otros, que antes no se contaban en *Android Market.*

En Google Play Store se pueden subir aplicaciones de distribución gratuita, como así también aplicaciones pagas. Las formas de pago que acepta son las transacciones mediante tarjeta de crédito o débito de las empresas Visa, MasterCard y Visa Electron.

Brinda una plataforma de distribución de aplicaciones a nivel mundial, pero a la hora de realizar el proceso administrativo de subir la aplicación a la plataforma cuenta con ciertas restricciones. Dentro de éstas existen países a los cuales se les permite o no realizar un registro de desarrolladores o registro como comerciante. En el caso de Argentina, tiene permitido realizar ambas acciones, y el tipo de moneda que se utiliza es el dólar (esto puede variar según el país). Esta información más detallada y el Acuerdo de Distribución para Desarrolladores de Google Play se pueden consultar en detalle en el Anexo I y el Anexo II, respectivamente.

Actualmente, la plataforma tiene más de 700.000 aplicaciones y juegos disponibles para la descarga.

2.2. Realidad Aumentada

"Nada es verdad, nada es mentira,
todo depende del cristal con que se mira"

William Shakespeare

Desde el comienzo de los tiempos la vida ha existido plenamente dentro de los límites de un mundo físico, compuesto por objetos reales y escenas que se pueden ver, oler o sentir. A través de la historia se ha ido modificando la percepción, la forma en que se percibe el mundo físico, cómo los objetos se mueven en el espacio o hasta cómo se sienten en las manos. Con la llegada de las computadoras se comenzaron a crear mundos virtuales, en los cuales se puede interactuar de diferentes formas, creando así una nueva realidad, lo irreal, lo virtual.

Con el avance de los años, las nuevas tecnologías han permitido crear aplicaciones con gráficos virtuales cada vez más reales, los cuales van acompañados del gran crecimiento en materia de hardware, lo que permite que, día a día, se tengan dispositivos más pequeños y más potentes, capaces de ejecutar todas estas aplicaciones que se van desarrollando con el tiempo.

Respecto a esta nueva perspectiva es necesario comenzar a ver las cosas con un enfoque distinto, tratando a lo real y lo virtual de una forma conjunta. Así nace la Realidad Aumentada.

2.2.1. Definición

La Enciclopedia Británica presenta la siguiente definición: "*Realidad Aumentada, en programación de computadoras, es el proceso de combinar o "Aumentar" las imágenes de video o fotografías, superponiendo las imágenes con datos generados por computadora*." [2].

La RA es una variación del término Entorno Virtual (EV o virtual environment o también denominado Realidad Virtual). Los EV generan a los usuarios la sensación de estar completamente inmersos dentro de un mundo sintético, generado por computadora. Mientras está dentro, el usuario está sumergido completamente en este mundo virtual, se abstrae del mundo real dejando de ver todo lo que lo rodea.

Por el contrario, la RA genera un ambiente intermedio en el que el usuario ve los objetos virtuales superpuestos o mezclados con el mundo real. Es decir, la RA complementa la realidad en lugar de sustituirla por completo, añadiendo información digital a la información física.

Idealmente, al usuario le parece que los objetos virtuales y los reales coexisten en el mismo espacio; esto es similar a los efectos logrados en películas, como en Iron Man. La Figura 5 muestra un ejemplo en donde podemos observar una armadura completamente virtual que está superpuesta dentro de una sala del mundo real, permitiendo además la interacción del humano. Se debe tener en cuenta que los objetos son combinados en 3D, por lo que, tomando como ejemplo la imagen de la película, la armadura virtual cubre lo que se ve del mundo real y el mundo real puede abarcar partes de los modelos 3D.

Figura 5: Imagen de la película Iron Man, un ejemplo de RA

Fuente: https://k61.kn3.net/taringa/1/9/9/9/5/9/73/alex2040r/4FE.jpg

2.2.2. Terminología

El término RA fue acuñado por los investigadores Tom Caudell y David Mizell, en el año 1992, para describir una exhibición digital presentada sobre una pantalla montada sobre la cabeza, que usarían los técnicos electricistas de la empresa aeronáutica Boeing para el armado de grandes paquetes de cables electrónicos para las aeronaves, que mezclaba gráficos virtuales sobre las imágenes reales [3]. La Figura 6 presenta el sistema de RA implementado por Caudell y Mizell:

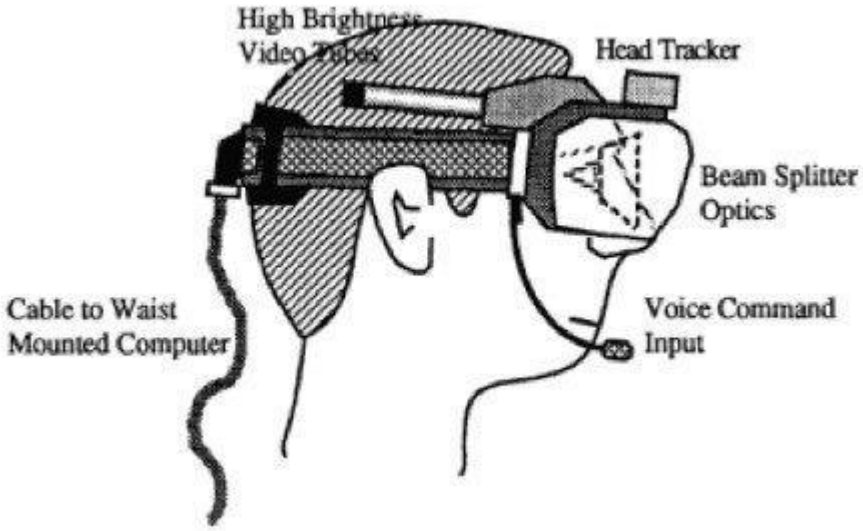


Figura 6: Sistema de RA implementado por Caudell y Mizell

Fuente: https://s-media-cache-ak0.pinimg.com/564x/87/41/9f/87419fbc3438a7b272f0ab31e6244053.jpg

Más tarde, en 1994, Paul Milgram escribe su primer artículo “Taxonomía de la Realidad Mixta”, en el cual define la “Realidad-Virtualidad continua” o “Continuo de Virtualidad” [4], también llamada “Realidad Mixta”. Un punto extremo contiene el entorno real (realidad) y el otro extremo presenta el entorno virtual (virtualidad). Todo lo que está entre ellos es Realidad Mixta (Figura 7). [5]

Figura 7: Taxonomía de Realidad Mixta según Milgram

La Realidad Mixta es un sistema subdividido en dos, la RA (más cercana a la realidad) y la virtualidad aumentada (más próxima a la virtualidad pura), en el que se fusionan el mundo real y los mundos virtuales para producir un nuevo ambiente donde objetos físicos y digitales coexisten e interactúan entre sí. En este contexto, el término Realidad toma el concepto del ambiente físico, el cual es visto directamente o a través de una pantalla de video.

En 1997, Ronald Azuma publica el estudio “A Comprehensive Survey on Augmented Reality” [6] y, debido al rápido desarrollo en el área, produce un nuevo estudio en 2001 [7]. En ellos realiza una definición más actual de RA identificando a estos sistemas mediante la relación de tres conceptos claves:

- Combinación de elementos virtuales y reales.
- Interactividad en tiempo real.
- Información almacenada en 3D.

Tanto Milgram como Azuma definieron a la Realidad Mixta como una formación unidimensional, que va desde el Ambiente Real al Ambiente Virtual. Sin embargo, ésto puede ser extendido a lo largo de una segunda dimensión. En 2002, Mann [8] agregó un segundo eje a la continuidad definida por Milgram, para cubrir otro tipo de alteraciones, esta nueva formación bidimensional, donde el eje vertical representa la cantidad de mediación o filtrado que se está realizando en la vista del usuario, sobre el entorno real o el virtual. De esta forma, definió el concepto de Realidad Media y Virtualidad Media (Figura 8), filtrando o modificando la visión del mundo real, en lugar de simplemente añadir contenido como se hace con la RA.

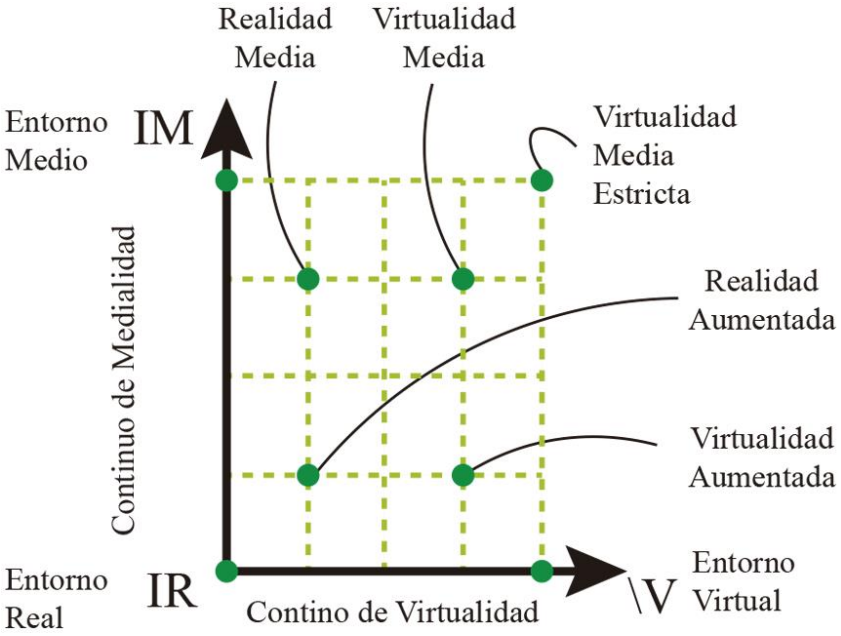


Figura 8: Continuo de Medialidad/Virtualidad de Mann

En la Realidad Media la percepción que tiene una persona sobre la realidad es manipulada de una u otra forma. Un sistema puede cambiar la realidad de diferentes maneras, puede añadir algo (Realidad Aumentada), quitar algo (Realidad Disminuida), o alterarla (Realidad Modulada). Mann también presentó las relaciones entre éstas mediante un diagrama de Venn, como se puede apreciar en la Figura 9. En la Realidad Disminuida se eliminan componentes reales que existen en el ambiente, por ejemplo un edificio, árboles, etc. En cierto modo, la Realidad Disminuida es lo opuesto de la Realidad Aumentada.

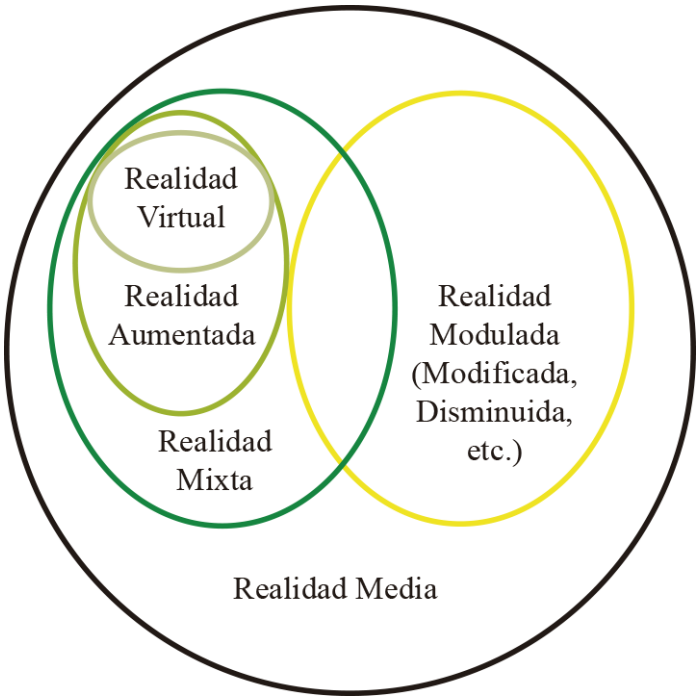


Figura 9: Realidad Media de Mann

En el año 2009 se crea el logo oficial de la RA (Figura 10), con el fin de estandarizar la identificación de la tecnología aplicada en cualquier soporte o medio por parte del público general.

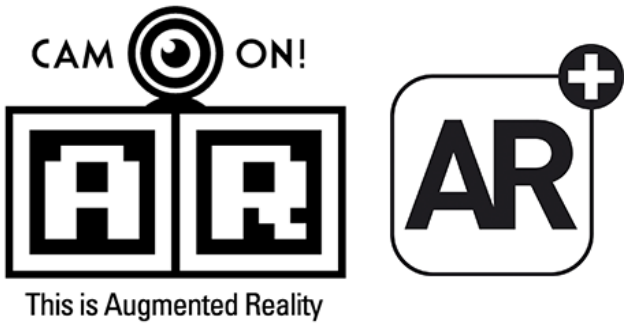


Figura 10: Logos oficiales estandarizados de RA

Hoy en día, la mayoría de las definiciones de RA y Realidad Mixta están basadas en las definiciones presentadas por Milgram, Azuma y Mann. Sin embargo, la categorización se vuelve imprecisa y la demarcación entre diferentes áreas es a menudo difícil o volátil y, a veces, incluso

contradictoria. Por ejemplo, Mann definió la Realidad Virtual como una subárea de la Realidad Mixta, mientras que Azuma separa completamente la Virtualidad Total de la Realidad Mixta.

2.2.3. Tipología

Según diversos informes o autores, existen diferentes maneras de clasificar la RA teniendo en cuenta las formas de implementación que presentan.

Una primera clasificación, está definida por las características del hardware que utilizan [9], dividiendo los sistemas de RA en dos tipos: fija y móvil. Los **sistemas móviles** permiten al usuario la posibilidad de efectuar grandes desplazamientos sobre un medio determinado, mientras que los **sistemas fijos** no aportan flexibilidad en este aspecto.

Dejando de lado las características del hardware, otra clasificación existente se basa en el reconocimiento o lectura que realiza el sistema [10][11], rigiéndose por los niveles o grados de complejidad que posee el objeto marcador o de reconocimiento. Se pueden emplear diferentes tipos de objetos reconocibles (o trackeables), los cuales son:

- **Marcadores (Markers)**: Se caracterizan por su simplicidad gráfica. Son derivaciones de matrices de pixeles con profundidades de color de 1 bit, de tal modo que facilitan el reconocimiento cuando se emplean cámaras digitales.

- **Imágenes**: En este caso, el objeto reconocible pasa a ser imágenes que actúan como marcadores, pero con una composición gráfica más compleja, con diferentes dimensiones, texturas y colores.

- **Objetos**: Actualmente, las aplicaciones son capaces de reconocer objetos completos, tridimensionales y con texturas complejas, como así también caras o edificaciones. Esta clasificación presenta un gran salto evolutivo ya que permite la posibilidad de prescindir de una gráfica impresa.

- **Localización**: Existen sensores que hoy en día poseen la mayoría de los celulares, como el GPS, la brújula y el acelerómetro, y que permiten trabajar con información de posicionamiento. Estos dispositivos, en conjunto, permiten determinar en qué punto del planeta, a qué altura y en qué dirección está apuntando la cámara del móvil, por lo que se utiliza esa información para representar el contenido de RA.

Los diferentes tipos de reconocimiento están directamente relacionados con las altas prestaciones del hardware existente, donde los niveles de procesamiento habilitan la captura de imágenes en alta definición y el uso de sensores externos, que permiten experiencias menos complejas a la hora de su implementación y con un mayor grado de presencia en el despliegue de la información digital.

Una última clasificación existente de los sistemas de RA se encuentra definida por la interacción que posee el usuario con la aplicación. Hoy en día, la mayoría de las interacciones que se realizan en RA tienen como objeto la interoperabilidad mediante elementos visuales. De esta manera es posible encontrar una variedad de métodos para llevar a cabo actividad en RA [9]:

- **Interfaces de Usuario Tangibles (TUI)**: Son sistemas que generan una combinación más profunda entre la realidad y los datos digitales. Incluyen elementos hápticos (tacto), que se combinan con la información virtual superpuesta a la información real, lo que permite una mejora significativa en la interacción física con los elementos virtuales.

- **Interfaces de RA Colaborativas**: En este tipo de sistemas de RA se utilizan múltiples pantallas para visualizar la información virtualizada, para poder realizar una experiencia compartida y que puede ser utilizada de manera remota por diversos usuarios.

- **Interfaces Híbridas**: Estos sistemas combinan diferentes interfaces que se complementan entre sí, que permiten al usuario interactuar con el contenido en diversas formas. El objetivo es implementar una plataforma flexible de interacción que sea interoperable entre distintos tipos de hardware.

- **Interfaces Multimodales**: Son sistemas que combinan una serie de métodos de interacción con los datos virtualizados. Esta interacción se realiza con objetos reales de manera natural y fluida, como sucede con el habla, el tacto, los gestos naturales del cuerpo, o la mirada.

Hasta la fecha, las únicas limitaciones en torno a la visualización y la interacción con información en RA provienen de las características del hardware, y hay una dependencia directa de los elementos de portabilidad y procesamiento que requiere esta tecnología. En un futuro, seguramente, surgirán nuevas categorías de clasificación de RA, como, por ejemplo, una categoría que no necesite del uso de un dispositivo de visualización y la información virtual esté combinada con el mundo real y a la vista del ojo humano.

2.2.4. Áreas de aplicación

Si bien la RA es una idea bastante antigua, también es una realidad emergente que, con el paso del tiempo y las tecnologías, es posible ir implementando en distintas actividades y campos profesionales.

En septiembre de 2009, la prestigiosa revista británica The Economist publicó un artículo en el que especificó que "*intentar imaginar cómo se utilizará la Realidad Aumentada es como intentar predecir el futuro de la tecnología web en el año 1994*" [12]. La agencia de investigaciones de mercado Juniper Research en su informe "Mobile Augmented Reality: smartphones, tablets and smart glasses 2013-2018" [13], predica a la tecnología de RA como la plataforma de comunicación que se convertirá en indispensable para dispositivos móviles y el comercio. Según los datos del informe, para el año 2018 se estima que el número de usuarios en uso de aplicaciones de RA se aproxime a los 200 millones.

Existe una multitud de áreas de aplicación para la RA, como lo describió Azuma en su revisión del estado del arte en el año 1997 [6]. Si bien el desarrollo propuesto en esta Tesina tiene como fin un uso en el ámbito comercial, realizamos a continuación una presentación de algunas de las áreas de aplicación que tienen mayor crecimiento en la actualidad:

- La **medicina** es un campo que se ve muy beneficiado por el uso de la RA, ya sea para su utilización en cirugías como, también, para la educación de nuevos profesionales. Por ejemplo, existen resonancias magnéticas o tomografías que recaban datos del interior del paciente para ser representados mediante modelos 3D, los cuales pueden ser superpuestos sobre el cuerpo físico del paciente dando resultados mediante una visión ampliada y en tiempo real, permitiendo así estudios u operaciones más eficientes. Otro ejemplo lo constituyen las gafas de RA creadas por un grupo de científicos de la Escuela de Medicina de la Universidad de Washington, que pueden distinguir las células cancerígenas de las sanas, marcando la diferencia en los procedimientos quirúrgicos para extirpar los tumores de pacientes con cáncer, facilitando así el trabajo de los cirujanos para operar con mayor precisión las áreas afectadas por la enfermedad.

- Dentro del ámbito **comercial** y el de la **publicidad** una de las tareas fundamentales es la de captar la atención, por lo que las empresas o comercios buscan formas diferentes o novedosas para realizar la promoción de sus productos. Utilizando RA, se le ofrece a los usuarios / clientes la posibilidad de tener experiencias visuales llamativas, por ejemplo mediante representaciones virtuales del producto y proporcionando, además, audios, videos o enlaces web a los que pueden acceder para obtener más información del producto real (que está visualizando a través de la cámara del dispositivo).

- En aplicaciones sobre tecnologías **militares**, el Heads-Up Display (HUD) constituye un ejemplo típico de RA. Una pantalla transparente se coloca directamente en la vista del piloto de un caza, en donde se le muestran los datos de la altitud, la velocidad aerodinámica y la línea del horizonte, entre otros. El término "heads-up" se aplica porque el piloto no tiene que mirar hacia abajo para obtener los datos que necesita (hacia los instrumentos de la aeronave). Los dispositivos montados sobre la cabeza (HMD) son utilizados por las tropas de tierra para visualizar datos críticos como, por ejemplo, la posición de tropas enemigas que se encuentren dentro del radio de visión del soldado que utiliza el dispositivo. Esta tecnología también se usa, como sucede en otros campos de aplicación, para la formación de nuevos profesionales.

- Las aplicaciones de **navegación** son, posiblemente, las que usamos más asiduamente en nuestra vida cotidiana. Los sistemas mejorados de GPS utilizan RA para facilitar el desplazamiento de un punto a otro. Por ejemplo, usando la cámara del teléfono en combinación con el GPS, los usuarios pueden ver indicaciones sobre una ruta seleccionada, sobre una vista en vivo, sin dejar de ver de esta forma lo que se encuentra adelante.

- En forma similar a las aplicaciones de navegación, que utilizan los datos obtenidos sobre la ubicación, existe una gran cantidad de aplicaciones para **turismo** que, haciendo uso de las cámaras y la tecnología de reconocimiento de imagen, permiten a los turistas recorrer sitios históricos y ver hechos y figuras presentadas sobre una superposición en la pantalla de sus dispositivos.

- En los contextos **educativos**, la mayoría de los proyectos buscan una innovación frente a las diferentes formas de enseñar, basando su principal filosofía en aprender por medio de la participación y el entretenimiento. Es por ello que la RA se ha comenzado a utilizar para complementar las formas de estudio. Constituye una plataforma tecnológica muy eficaz en lo relacionado a cómo los estudiantes perciben la realidad física debido a que, gracias a la implementación de gráficos 3D desarrollados mediante computadora, se puede desglosar esta realidad en diversas dimensiones, facilitando la captación de sus diversas particularidades, y complementando cualquier experiencia de aprendizaje. Además, las aplicaciones educativas de RA tienen la capacidad de obtener interacción con los usuarios o estudiantes, respondiendo a las entradas de los mismos, lo que permite interactuar con la información virtual que se esté representando para su estudio.

- En la **arquitectura** la RA tiene un uso actual bastante grande y un futuro prometedor. Por ejemplo, permite visualizar proyectos en detalle como si se

estuvieran presentando con una maqueta real, sólo que ésta es una representación virtual. La ventaja de esta "maqueta virtual" es que se pueden realizar distintos enfoques y mostrar detalles de diferentes aspectos con más precisión, sin tener que estar encasillado a una escala o un sector en particular. Además, este contenido virtual arquitectónico tiene la capacidad de ser animado, reproducir distintos factores naturales dentro de la escena, agregar interactividad o cualquier tipo de contenido virtual que se desee.

- En el campo de la **industria**, la RA se ha introducido en los últimos años a través de las diferentes tecnologías. Hoy existen gafas que permiten visualizar lo que observan los trabajadores ubicados en distintas locaciones, recibiendo asistencia o monitorización remota a través de contenido aumentado. Se utilizan gafas también para la formación de operarios, a través de textos explicativos en las imágenes de RA, permitiendo a los trabajadores entender mejor la tarea que tienen que realizar y reduciendo las incidencias en el mantenimiento y las reparaciones de maquinarias. Además de estos usos, permite en la fabricación de un producto, tener una visión más clara del producto que se está desarrollando. Por ejemplo, en la empresa automovilística Audi ya se incorpora esta tecnología, que faculta a sus ingenieros manipular y ver en 3D las partes que están desarrollando, para tener una visión de su funcionamiento en la vida real.

- Las aplicaciones de RA dedicadas al **entretenimiento** han tenido gran crecimiento y, hoy en día, cuentan con el mayor grado de usabilidad. El principal objetivo de estas aplicaciones es el de dar un nuevo enfoque al entretenimiento, dejando de lado lo completamente virtual para sumergir a los jugadores en una experiencia que permita el entretenimiento virtual en un ambiente real. El ejemplo más claro de este tipo de aplicaciones de RA es el Pokémon Go, que batió la mayoría de los récords de aplicaciones móviles. El juego permite ir caminando por la calle, buscando unas criaturas que se pueden encontrar aleatoriamente o en lugares específicos del mundo. Estas criaturas aparecen a través de la cámara de los dispositivos, representadas mediante 3D en el lugar que el usuario está observando mediante la cámara, y se pueden recolectar, para luego utilizarlas en batallas o simplemente para coleccionar. Como expresó Roger Pastor, cofundador de Pangea Reality, una de las empresas pioneras en RA, la clave del éxito de Pokémon Go radica en que "d*e repente, la frontera entre lo real y lo digital desaparece, dando paso a un modo de juego en el que tú eres el protagonista mientras te mueves por el mundo real. Los Pokémon que marcaron a más de una generación de jugadores están viviendo en el mundo real por primera vez en la historia, y todo gracias a la realidad aumentada*." [14]

Todos estos campos que presentamos utilizan las distintas tecnologías de RA. Pero, para tener una idea más general de sus distintos usos, podemos observar en la Figura 11 los datos presentados por Juniper Research sobre la cantidad de usuarios en uso de aplicaciones de RA, separados en las diferentes categorías de aplicación:

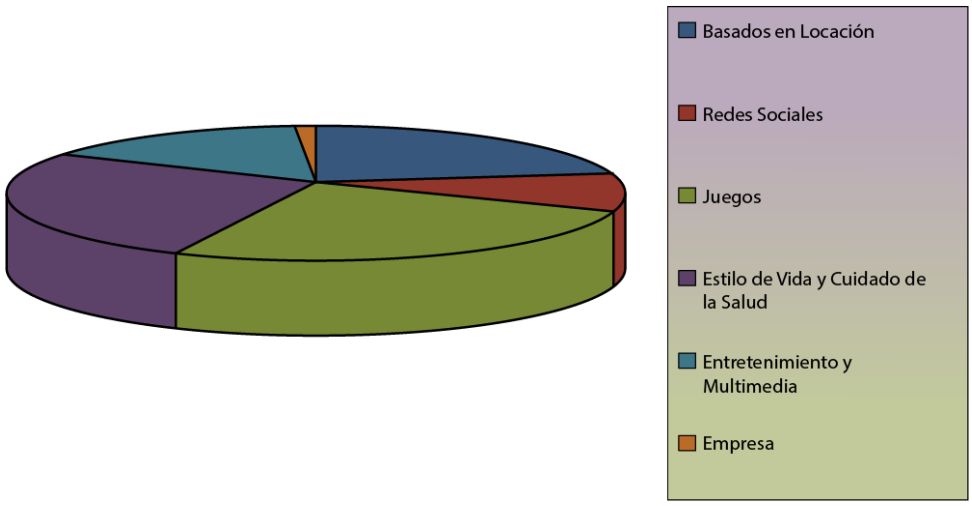


Figura 11: Porcentaje de cantidad de usuarios en las diferentes categorías de aplicación (2018)

2.2.5. Diagrama de una aplicación de RA

Una aplicación de RA está conformada por los siguientes cuatro procesos principales (Figura 12):

- **Captura de la escena real**: La escena real se captura por medio de la cámara de video del dispositivo. Las imágenes de video capturadas son utilizadas para realizar el seguimiento (o tracking).

- **Seguimiento del objeto (tracking)**: Esta tarea de seguimiento puede ser realizada mediante dispositivos específicos o, también, mediante el análisis de la captura de video de la escena real. Se utiliza el análisis de imagen mediante algoritmos para determinar la posición de la cámara o del usuario.

- **Generación de la escena virtual**: Se tiene un mundo virtual con la información de la posición y la orientación del objeto que se va a representar en 3D, con las cuales se genera una vista acorde del mismo.

- **Combinación del mundo virtual con la escena real**: Existen muchos tipos de combinaciones que se pueden realizar para realizar esta unión. En nuestro caso de

estudio, la aplicación genera como resultado un video con la escena real capturada y la escena sintética generada.

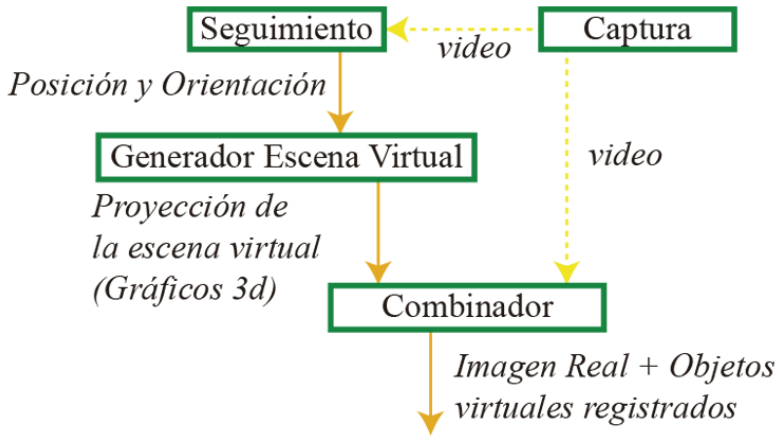


Figura 12: Diagrama de los procesos de una aplicación de RA

2.2.6. Desafíos de una aplicación de RA

En RA se requiere un seguimiento preciso de la posición y de la orientación para alinear, o registrar, la información de los objetos físicos reales junto a los objetos virtuales que se van a representar en la escena. Es por ello que existen dos desafíos principales a los cuales se debe enfrentar toda aplicación de RA: la resolución correcta del registro y del tracking o seguimiento.

2.2.6.1. Registro

A diferencia de la realidad virtual, donde el usuario interactúa en mundo totalmente virtual, la RA tiene la tarea de generar capas de información virtual que deben ser correctamente alineadas con la imagen del mundo real para lograr una correcta sensación de integridad. De aquí surge uno de los principales problemas con el que deben tratar los sistemas de RA, que es el denominado **registro**, que consiste en calcular la posición relativa (posición y rotación) de la cámara real respecto de la escena para poder generar imágenes virtuales correctamente alineadas con esa imagen real. En la Figura 13 puede verse un ejemplo de una imagen con un registro incorrecto y uno correcto:

Figura 13: El problema del registro: registro incorrecto (izquierda) y registro correcto (derecha)

Fuente: http://posgrado.itlp.edu.mx/uploads/587975dd46189.pdf

El registro utiliza los resultados del seguimiento, o tracking, para alinear el contenido virtual y el real entre sí. Esta alineación, como ya comentamos, se realiza respecto a los objetos reales y a la posición de la cámara. Los sistemas de registro pueden ser realizados de dos formas:

- **Registro de bucle abierto**: Se basa únicamente en la información brindada por el sistema de seguimiento, no usa ningún tipo de retroalimentación.

- **Registro de bucle cerrado**: Entrega una información de retroalimentación sobre la precisión durante la tarea de registro, la cual se tiene en cuenta para los cálculos futuros.

Lograr un registro adecuado puede ser complicado porque los usuarios ven lo que sucede en el mundo real y siempre reconocen los desplazamientos entre contenido virtual y los objetos reales. Para algunas aplicaciones no es necesario un registro extremadamente exacto, sin embargo, otras requieren un registro mucho más preciso. Para estas últimas, no sólo sería problemático contener mala información, sino que el uso de esta mala información puede provocar que la aplicación deje de utilizarse. Por ejemplo, si una aplicación turística, de vez en cuando, no brinda correctamente y con precisión la información, su uso aún sería aceptable; pero en una aplicación de medicina, el mal uso de la información podría perjudicar gravemente al paciente.

Azuma afirma que el registro exacto es difícil de lograr ya que existen varias y, muy diferentes, fuentes de errores. Divide estos errores de registro en dos grandes categorías:

- **Errores de registro estáticos**: Se generan cuando se produce una distorsión óptica, errores en el sistema de seguimiento, desajustes mecánicos (por lo general

en los tipos de registro que no son por visión) o cuando hay parámetros de visualización incorrectos.

- **Errores de registro dinámicos**: Se producen cuando hay retrasos del sistema (lag[6]). Estos retrasos existen porque cada componente en un sistema de RA requiere un tiempo para realizar su trabajo.

Existen varias tecnologías que permiten realizar el registro, tales como sensores mecánicos, ultrasónicos, magnéticos, inerciales y los basados en visión. Debido a que la mayoría de estos métodos tienen grandes costos o se necesitan conocimientos avanzados para realizar su aplicación, de aquí en adelante sólo nos basaremos en la tecnología que hace uso del análisis de los datos de los flujos de video, denominados **métodos de visión**, que son los que utilizamos para nuestra aplicación. Son los más baratos, los de mayor uso y los más fáciles de desplegar. [15]

2.2.6.2. Seguimiento (Tracking)

El **tracking o seguimiento**, como mencionamos anteriormente, es el encargado de obtener una estimación de la posición y la orientación de la cámara[7] o participante dentro del entorno del mundo real. El seguimiento es una parte fundamental de cada aplicación de RA; sin la correcta interpretación de la ubicación de la cámara, el contenido virtual no se puede colocar en relación con elementos reales o la perspectiva de la cámara.

El **seguimiento o tracking basado en visión**, que utiliza las cámaras de video, es un subcampo del tracking 3D en el que se emplean distintas técnicas de visión por computadora para obtener el posicionamiento de la cámara mediante seis grados de libertad (tres grados para la posición y tres de la orientación).

Para poder calcular la posición de la cámara respecto al mundo real es necesario contar con un conjunto de referencias tridimensionales. Estas referencias pueden ser marcas con una descripción geométrica conocida de antemano u objetos previamente modelados. Comparando con lo que se percibe mediante la cámara del mundo real, es posible obtener el posicionamiento relativo de estas referencias.

[6] Lag: Es un retraso del sistema de extremo a extremo. Se define como la diferencia entre el momento en que el sistema de seguimiento mide la posición y la orientación del punto de vista, hasta el momento en que las imágenes generadas aparecen en pantalla.

[7] Conocer la posición de la cámara es la esencia de todos los seguimientos en video a través de RA. En dispositivos HMD el seguimiento generalmente se centra en determinar la posición de la cabeza del usuario, mientras que en términos de seguimiento y registro en esta Tesina siempre se refiere a la cámara.

El seguimiento debe ser lo más preciso y robusto posible para crear la ilusión que el contenido virtual es una parte del mundo real. Idealmente, un proceso de seguimiento calcula los resultados en tiempo real sin verse afectado por diferentes entornos o diferentes circunstancias, por ejemplo los cambios en las condiciones de iluminación; pero estos sistemas de seguimiento perfecto son difícilmente alcanzados. Por lo tanto, la mayoría de los sistemas de seguimiento son especializados y funcionan mediante condiciones y sus propias restricciones.

Como mencionamos cuando hablamos del registro, los métodos que están basados en visión tienen el mayor nivel de uso, debido a que la información necesaria puede ser obtenida del flujo de video y de esta forma se logran sistemas con un costo más bajo y una aplicación más sencilla que los que utilizan cualquier otro dispositivo. Sin embargo, las aplicaciones que emplean estos métodos, deben enfrentarse a tres clases de problemas:

1. **Identificación:** Son causados por el propio uso de las cámaras de video, es decir los problemas del punto de vista que tiene la cámara, los problemas de oclusión o las condiciones de iluminación, entre otros.

2. **Seguimiento**: Este grupo de inconvenientes está asociado con los problemas al analizar los flujos de video. Son producidos por dificultades al no encontrar referencias para representar el modelo 3D.

3. **Entornos no controlados**: Son generados cuando el trabajo de mantener la correcta alineación entre las distintas imágenes (real y virtual) se centra en realizar el seguimiento sin referencias conocidas previamente.

Existen dos métodos de tracking basado en visión según Marimon [16], las aproximaciones Bottom-Up (BUA) y las Top-Down (TDA), que describimos a continuación.

2.2.6.2.1. Aproximaciones Bottom-Up

Las BUA tratan de obtener la posición a partir de lo que percibe la cámara: se elige un rastreador que depende de una referencia conocida, el rastreador detecta esta referencia y produce una estimación de la posición de la cámara en cada fotograma.

Los seis grados de libertad que se utilizan para conseguir la posición de la cámara se calculan a partir de la obtención de características geométricas co-nocidas de objetos y sus relaciones geométricas 3D (como, por ejemplo, un cuadrado, un plano, etc.).

Además, la estimación de la posición se realiza mediante la acumulación de estimaciones de movimiento entre tramas o el uso de fotogramas clave.

Existen dos tipos de estas aproximaciones:

1. **Tracking basado en marcas**: Este método es el más utilizado actualmente y se basa en el uso de patrones específicos. Se utilizan marcas cuadradas o circulares que pueden detectarse fácilmente gracias a su alto contraste, marcas que contienen diferentes colores, marcas que codifican su identificador mediante un sistema de código de barras, entre otras (Figura 14). El framework pionero en esta tecnología es ARToolkit, sin embargo, hoy en día existen diversos entornos de seguimiento basados en marcas.

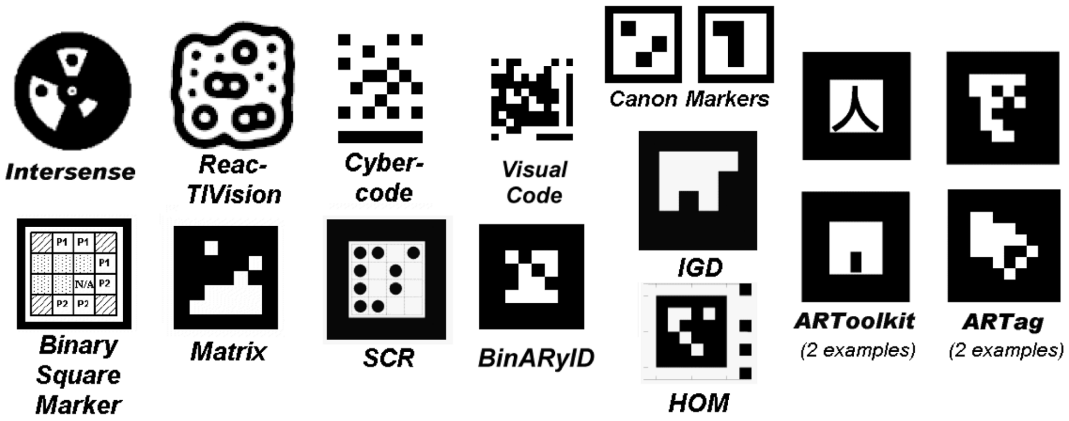


Figura 14: Ejemplos de marcas utilizadas en diferentes sistemas de tracking

Fuente: http://doi.ieeecomputersociety.org/cms/Computer.org/dl/trans/tp/2010/07/figures/ttp20100713173.gif

2. **Tracking sin marcas**: Este método de seguimiento, a diferencia del basado en marcas, emplea únicamente el reconocimiento de características naturales de la escena, es decir, estructuras o formas físicas que son fáciles de detectar, como por ejemplo los bordes de una mesa o las ventanas de un edificio. No requiere de entornos totalmente preparados para trabajar, pero debe contar con algunos modelos geométricos específicos o esperados. Las diferentes técnicas que existen para este método son:

    - **Estructuras Planas**: Utilizan información específica sobre áreas planas detectadas en la escena, como sectores del suelo, fachada de un edificio, carteles en pared, entre otros (Figura 15). Tienen altos requerimientos de procesamiento. [17]

Figura 15: Tracking basado en estructuras planas

Fuente: https://www.robots.ox.ac.uk/~vgg/publications/2000/Simon00/simon00.pdf

- **Basadas en Modelos**: Utilizan un modelo CAD[8] de partes de la escena, por lo general es un objeto, y de este objeto se detectan las aristas y los puntos de interés mediante el video de entrada, estableciendo de este modo la posición de la cámara relativa al objeto seleccionado (Figura 16). [18]

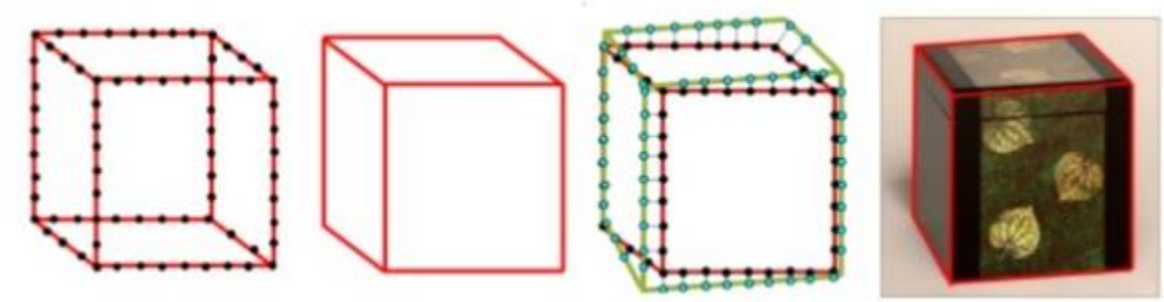


Figura 16: Tracking basado en modelos

Fuente: https://image.slidesharecdn.com/presentacion-170201070726/95/sistemas-de-tracking-en-realidad-aumentada-para-patrimonio-46-638.jpg?cb=1485932976

- **Escenas sin Restricciones**: Existen también algunos escenarios de aplicación donde no es posible contar con conocimiento previo sobre estructuras planas o modelos. En general este tipo de técnicas hace uso de restricciones epipolares[9] de las cámaras en movimiento. Su mayor problema es el elevado costo computacional, por lo que recién en estos últimos años se comenzó a utilizar en tiempo real [19][20].

[8] CAD (computer-aided design): Diseño asistido por computadora.

[9] La geometría epipolar es aquella dada por la intersección del plano de cada imagen con una familia de planos, donde todos los planos de esta familia contienen la línea base que une los centros ópticos de las cámaras.

2.2.6.2.2. Aproximaciones Top-Down

Las TDA pueden estimar robustamente los parámetros de la cámara utilizando el contexto y múltiples hipótesis: intentan estimar si desde la posición actual se perciben los resultados esperados (una vez que se estableció la posición, se trata de identificar las referencias que se esperaban obtener).

Utilizan el contexto e inducen la geometría de la escena a partir de este contexto. Más concretamente, estas técnicas se basan en la estimación mediante modelos de movimiento basados en filtros bayesianos para predecir la postura de la cámara y, a partir de esta predicción, se buscan referencias en la escena. Con la detección de estas referencias, se corrige la predicción y, adicionalmente, se puede deducir la geometría del entorno. De hecho, la posición 3D de referencias, tales como bordes o puntos, y su retroproyección 2D en el plano de la imagen están relacionadas por restricciones geométricas en la pose de la cámara. Tales restricciones inducen la corrección necesaria de la pose calculada. Por lo tanto, se le realiza un seguimiento a la cámara durante un tiempo largo ejecutando un ciclo de predicción / corrección en cada trama.

En el ciclo de predicción / corrección existen dos cuestiones diferentes que hay que conocer: el filtrado y la asociación de datos. El filtrado está relacionado con el modelo de movimiento utilizado y las limitaciones de asumir dicho modelo. La asociación de datos se ocupa de la localización de la/s referencia/s de acuerdo con la pose predicha, para una posterior corrección de la postura.

En las BUA el trabajo de inicialización y recuperación cuando se pierde el seguimiento se realiza automáticamente, sin embargo el principal problema viene dado por la dificultad de no poder localizar una referencia completa, cuando el proceso de seguimiento no puede obtener una posición válida. En este caso, las TDA sacan una ventaja y funcionan mejor, aún cuando se tienen referencias parciales, el seguimiento realiza una estimación de la posición. Sin embargo, las TDA no permiten la inicialización automática de una posición inicial y acumulan errores parciales de las estimaciones, sumado a que al no contar con una referencia visual se debe realizar más procesamiento de la imagen de video para analizar los distintos puntos referenciales. [21][22]

2.2.7. Generación de la escena virtual

Como comentamos en la sección anterior los desafíos a los que se enfrenta cualquier aplicación de RA son cómo combinar el mundo real y el mundo virtual en único entorno, y de una forma correcta. Para mantener la ilusión al usuario de que los objetos virtuales son también parte del mundo real, se requiere una fusión coherente, buscando una alineación especial entre el mundo virtual y las imágenes captadas del mundo real, dentro de un espacio tridimensional.

Para que esta alineación entre ambos mundos se pueda realizar, se deben analizar los factores involucrados en las transformaciones especiales encargadas de relacionar los distintos

sistemas de referencia, que se tienen en los diferentes espacios, así como, también, analizar las transformaciones básicas que se pueden realizar a los componentes virtuales.

2.2.7.1. Sistemas de coordenadas

Dentro del proceso de formación de las imágenes que se van a representar por la aplicación de RA, existen:

- Una **escena**, la cual está compuesta por diversos objetos, que tienen sus correspondientes posiciones y orientaciones.

- Una **cámara**, encargada de captar las imágenes que serán utilizadas por la aplicación, que, al igual que los objetos, tiene su propia posición y orientación dentro de la escena.

- Una **imagen**, resultante de proyectar la escena de acuerdo a la cámara.

Además de estos elementos, se debe tener en cuenta que hay modelar la escena virtual (la teoría del modelado 3D la detallamos en la sección 2.3.).

Cuando se habla de la posición y orientación de un objeto o de la cámara que capta las imágenes, se debe tomar un sistema de referencia en base al cual se van a expresar. En RA, al hablar de las imágenes registradas (registro) nos referimos a que tanto las imágenes sintéticas como las del mundo real están en referencia al mismo sistema de coordenadas. Por lo tanto, es necesario definir los siguientes sistemas de coordenadas de visualización para hacer referencia a los componentes que intervienen en la formación de las imágenes:

- **Sistema de coordenadas local, de modelo u objeto**: Es un sistema de coordenadas 3D que se utiliza para referenciar los puntos de un objeto en particular. Para interpretarlo mejor, al modelar un objeto se estará utilizando un sistema de coordenadas que será local al objeto, situado en algún punto del mismo. Un ejemplo sería si se modela un árbol, éste tendría su sistema de coordenadas en su centro o en la base del mismo, con su eje alineado al tronco.

- **Sistema de coordenadas del mundo o universal**: Todos los objetos partícipes de una escena de RA poseen una posición y una orientación, las cuales están en relación con un sistema de coordenadas 3D que está situado en algún lugar del mundo. Cuando se modela la escena completa se utiliza un único sistema de coordenadas que es común entre todos ellos, denominado coordenadas mundo. De

esta forma todos los objetos integrantes de la escena están posicionados, orientados y escalados según la relación establecida por dicho sistema.

- **Sistema de coordenadas de la cámara o de visualización**: Este sistema de coordenadas de la cámara 3D viene dado por el centro óptico de la cámara. El eje de proyección de la cámara pasa por el centro óptico y es perpendicular al plano de formación de la imagen. El eje Z de este sistema está alineado con el eje de proyección, con Z+ hacia donde se ubica la escena. El eje Y tiene dirección vertical y el eje X dirección horizontal.
  El sistema de coordenadas mundo puede expresarse en relación al sistema de coordenadas cámara mediante transformaciones geométricas. De forma equivalente, el sistema de coordenadas cámara puede expresarse en relación al sistema de coordenadas mundo.

- **Sistema de coordenadas de la imagen (imagen plana)**: Los puntos de la imagen se expresan en relación a un sistema de coordenadas 2D, con los ejes alineados con los bordes de la imagen.

Para entender mejor el proceso de formación de la imagen podemos observar la Figura 17, donde además de representar los sistemas de coordenadas se establecen las relaciones de objeto (local) a mundo `O´, de mundo a cámara `C´, y de cámara a una imagen plana `P´. De objeto a mundo hace referencia a la posición y orientación de los objetos virtuales respecto al sistema de coordenadas del mundo definido por la escena real. La del mundo a la cámara, define la posición de la cámara que toma las imágenes del mundo real. Por último, de la cámara a una imagen plana, especifica la proyección que la cámara debe realizar para crear una imagen 2D respecto a la escena real en 3D. [23]

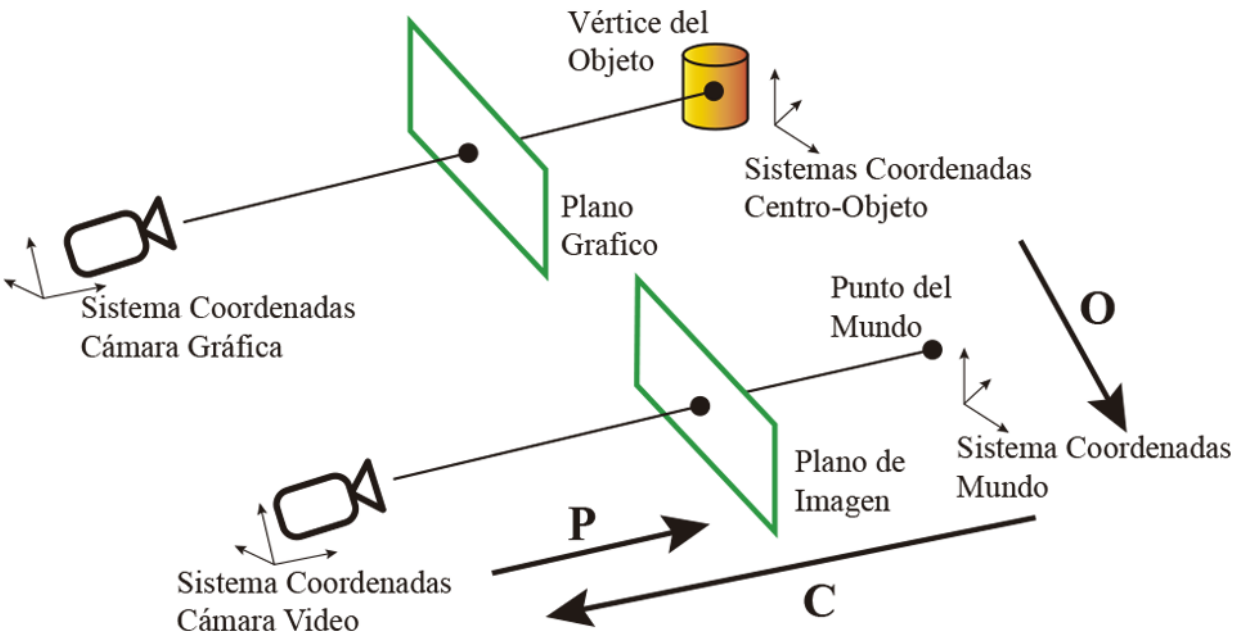


Figura 17: Sistemas de coordenadas en los distintos espacios y sus relaciones

2.2.7.2. Transformaciones geométricas

Para la generación de la escena virtual en RA se utiliza la representación de gráficos en 3D, por lo que es necesario comprender las transformaciones básicas de los objetos virtuales. Los objetos generados mediante modelado 3D están conformados por mallas poligonales (ver Sección 2.3.). Las operaciones que se aplican a cada uno de los triángulos de esas mallas para cambiar la posición, la orientación o el tamaño, se denominan **transformaciones geométricas**. A continuación, detallamos las operaciones básicas en 2D para trabajar con RA y su respectiva notación matricial [24].

2.2.7.2.1. Traslación

Esta es la transformación dimensional básica más sencilla, utilizada para realizar el traslado dentro del plano bidimensional de un punto o un objeto.

Para poder realizar la traslación de un punto y obtener las coordenadas de la nueva posición, se suma un vector de desplazamiento a las coordenadas iniciales de dicho punto de inicio.

También es posible trasladar un objeto, aplicando la misma suma de traslación de un punto a todos los puntos que conforman el objeto, por lo que se estaría trasladando el objeto entero de una posición a otra.

De este modo, se puede definir a la traslación como la suma de un vector libre *p'* (Figura 18), la cual se puede expresar matemáticamente de la siguiente forma:

$$p'_x = p_x + t_x \qquad p'_y = p_y + t_y$$

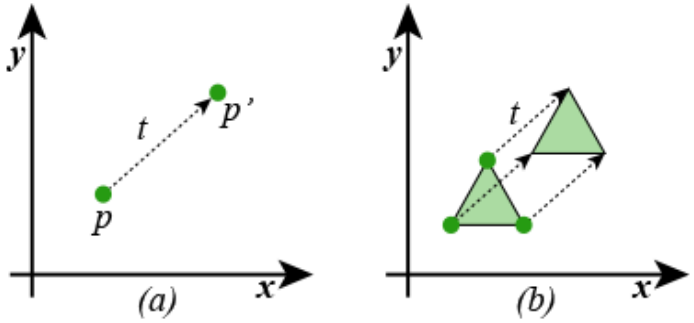


Figura 18: (a) Traslación de un punto (b) Traslación de un objeto.

2.2.7.2.2. Rotación

De la misma forma que se definió la traslación, se puede expresar la rotación de un punto $p = (x,y)$ a una nueva posición rotando un ángulo θ respecto de las coordenadas de origen, especificando el eje de rotación y el ángulo θ (Figura 19).

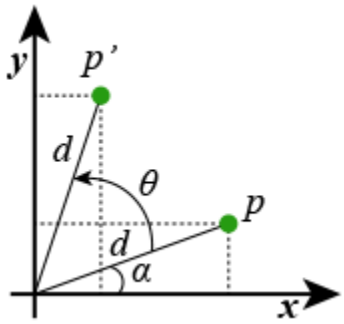


Figura 19: Rotación del punto p un ángulo respecto del origen de coordenadas

Las coordenadas iniciales del punto se pueden expresar como:

$$p_x = d \cos \alpha \qquad p_y = d \operatorname{sen} \alpha$$

Siendo *d* la distancia entre el punto y el origen del sistema de coordenadas.

Usando identidades trigonométricas, las coordenadas transformadas se pueden expresar como la suma de los ángulos del punto original α y el que se desea rotar θ como:

$$p'_x = d \cos(\alpha + \theta) = d \cos\alpha \cos\theta - d \operatorname{sen}\alpha \operatorname{sen}\theta$$
$$p'_y = d \operatorname{sen}(\alpha + \theta) = d \cos\alpha \operatorname{sen}\theta - d \operatorname{sen}\alpha \cos\theta$$

Si se sustituyen en la fórmula de traslación se obtiene:

$$p'_x = p_x \cos\theta - p_y \operatorname{sen}\theta \qquad p'_y = p_x \operatorname{sen}\theta - p_y \cos\theta$$

2.2.7.2.3. Cambio de escala

En forma similar a las dos operaciones anteriores, el cambio de escala aplicado a un objeto bidimensional, se puede realizar multiplicando los componentes *x,y* del objeto a escalar, por el factor de escala $S_x.S_y$ en cada eje (Figura 20). El escalado se puede expresar matemáticamente de la siguiente forma:

$$p'_x = p_x S_x \qquad p'_y = p_y S_y$$

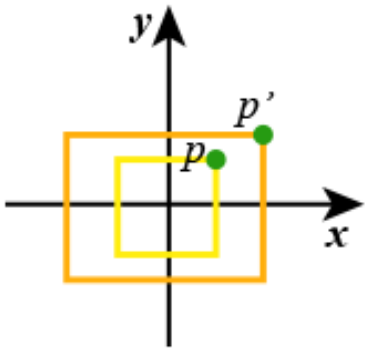


Figura 20: Conversión de un cuadrado a un rectángulo empleando los factores de escala

2.2.7.2.4. Representación matricial

En RA cuando se desea realizar un movimiento de uno de los objetos 3D, por lo general, se debe especificar más de una operación, por ejemplo, una combinación de traslación y rotación, por lo que se necesita disponer de una representación que nos permita combinar

transformaciones de una forma eficiente. Es por ello que se implementan representaciones matriciales.

En casi todas las aplicaciones gráficas, la mayoría de los objetos se deben transformar geométricamente, de una forma constante. En el ámbito de la RA, aunque un objeto 3D esté asociado a un marcador o una imagen y éste permanezca inmóvil, la cámara virtual deberá cambiar su posición para que se ajuste a los movimientos que está realizando el usuario de la aplicación.

Si bien son correctas las operaciones de transformaciones que planteamos anteriormente, es necesario poder combinar las transformaciones de modo que la posición final de las coordenadas de cada punto se obtenga directamente a partir de las coordenadas de inicio. Por cuestiones de eficiencia en la implementación de la concatenación de transformaciones, se necesita una representación homogénea de la traslación, la rotación y el escalado. Hasta el momento la representación no es homogénea, ya que la traslación se expresa como una suma, mientras que la rotación y el escalado se representan como multiplicaciones de matrices. Si se reformula la escritura de dichas ecuaciones para que todas las operaciones se puedan realizar mediante la multiplicación, se pueden homogeneizar las transformaciones y, de esta forma, lograr una mayor eficiencia a la hora de realizar las distintas concatenaciones de transformaciones.

Entonces, para obtener una representación homogénea, se recurre a las denominadas coordenadas homogéneas[10], agregando un término extra, un parámetro homogéneo *h*, a la representación del punto dentro del espacio *(x,y)*. De este modo, se obtiene una representación homogénea de la posición descrita como (x_h, y_h, h). Este parámetro h es un valor que difiere de cero, de forma que $x=xh/h$, $y=yh/h$. Por lo tanto, existen infinitas representaciones homogéneas que son equivalentes para cada par de coordenadas; si bien normalmente se utiliza el valor $h = 1$ existen casos en que no es así.

Si se agrega el parámetro homogéneo a la operación de traslación, se puede expresar esta operación de forma matricial como:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_x \\ 0 & 1 & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Al resolver la multiplicación matricial, se obtiene un conjunto de ecuaciones equivalentes a las primeras ecuaciones que fueron definidas.

Las operaciones de rotación T_r y escalado T_s también tienen su equivalente matricial homogéneo, como se muestra a continuación:

[10] Las coordenadas homogéneas permiten representar las ecuaciones de transformación geométricas como multiplicación de matrices y, de esta forma, se usa el mismo formato de trabajo que el método estándar utilizado en gráficos por computadora (el cual soporta el hardware de aceleración de video).

$$T_r = \begin{bmatrix} cos\theta & -sen\theta & 0 \\ sen\theta & cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad T_s = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Para obtener las transformaciones inversas sólo hay que realizar un cambio de signo en el caso de la traslación y rotación (distancias y ángulos negativos), y mediante la obtención del recíproco en el caso de la escala $1/S_x$, $1/S_y$.

Las transformaciones en el espacio 3D requieren, simplemente, añadir el parámetro homogéneo y describir las matrices. De esta forma, las traslaciones T_t y escalados T_s en 3D pueden representarse de forma homogénea mediante las siguientes matrices:

$$T_t = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad T_s = \begin{bmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

En el caso de las operaciones de rotación, se requiere distinguir el eje sobre el que se realizará la rotación. Las rotaciones positivas que se realizan alrededor de un eje, toman el sentido opuesto a las agujas del reloj cuando se está mirando a lo largo de la mitad positiva del eje hacia el origen del sistema de coordenadas (Figura 21). Las expresiones matriciales de las rotaciones son las siguientes:

$$R_x = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sen\theta & 0 \\ 0 & sen\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_y = \begin{bmatrix} cos\theta & 0 & sen\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sen\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad R_z = \begin{bmatrix} cos\theta & -sen\theta & 0 & 0 \\ sen\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

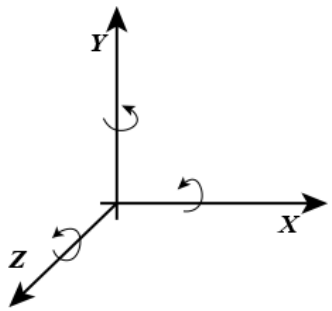


Figura 21: Sentido de las rotaciones positivas respecto de cada eje de coordenadas

Todas las transformaciones que analizamos (traslación, rotación y escalado) son ejemplos de transformaciones afines, donde cada una de las coordenadas transformadas se puede expresar como una función lineal de la posición origen y una serie de constantes determinadas por el tipo de transformación.

La ventaja de estas transformacion lineales en el ámbito de los gráficos informáticos, radica en que, si se desea transformar todos los puntos de un segmento, sólo es necesario transformar los extremos y luego trazar el segmento transformado uniendo los extremos transformados.

2.2.7.3. Visualización 3D

Existen algunas características que deben ser tomadas en cuenta a la hora de visualizar gráficos 3D que hayan sido desarrollados mediante computadoras y que en esta sección explicaremos brevemente.

Como ya mencionamos (sección 2.2.7.1.), para obtener una escena 3D que esté definida en un sistema de coordenadas universal (SRU), necesitamos la definición de un sistema de referencias de coordenadas para los parámetros de visualización (o también denominados parámetros de cámara). Este sistema de referencia define el plano de proyección (que sería el equivalente al sensor de imágenes de la cámara digitales). Mediante este plano se transfieren los objetos al sistema de coordenadas de visualización y, por último, se proyectan sobre el plano de visualización, como podemos apreciar en la Figura 22:

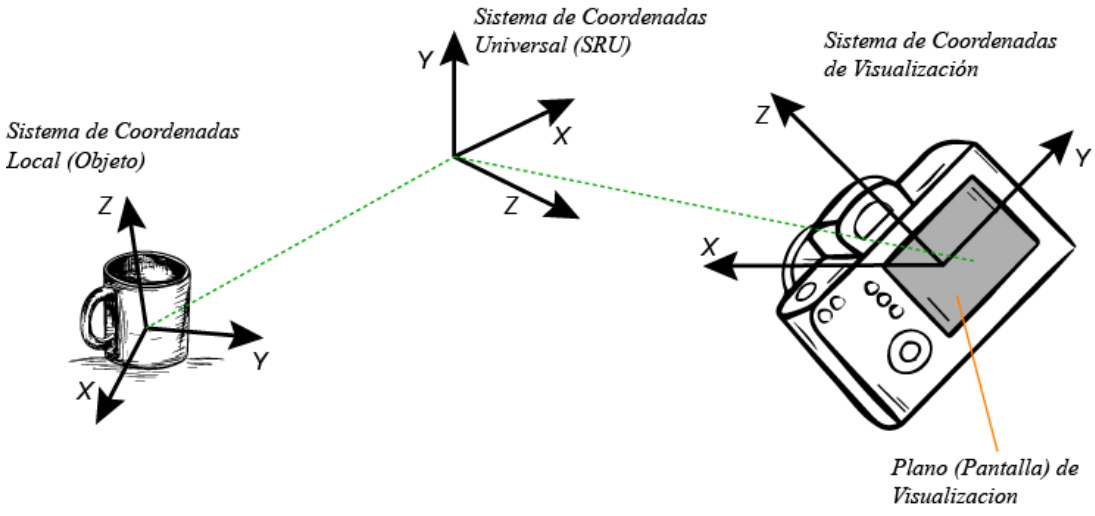


Figura 22: Sistema coordenadas de visualización y su relación con otros sistemas de coordenadas en la escena

Cuando se utilizan gráficos desarrollados por computadora, existe la posibilidad de elegir distintos modelos de proyección que van a tener los diferentes objetos sobre el plano de visualización. Por ejemplo, las aplicaciones CAD usan la proyección de los objetos utilizando líneas paralelas sobre el plano de proyección, lo que se denomina **proyección paralela**. Este modo de proyección conserva las proporciones relativas que existen entre los objetos, independientemente de la distancia que exista entre ellos.

La **proyección de perspectiva** se utiliza para proyectar los puntos hasta el plano de visualización usando trayectorias convergentes en un punto, lo que hace que los objetos que se encuentran a mayor distancia del plano de visualización tengan una representación más pequeña en la imagen. Este modelo de proyección genera imágenes más realistas, ya que lo hace de la misma manera que el ojo humano y de la misma forma en que las cámaras reales toman las imágenes.

Además de la perspectiva usada, para la creación y la transformación de una escena 3D a coordenadas que sean dependientes del dispositivo de visualización se sigue un proceso general, denominado **Pipeline de Visualización** (Figura 23):

Figura 23: Procesos generales del pipeline de visualización tridimensional

Todo objeto 3D sufre las transformaciones a los diferentes sistemas de coordenadas, hasta llegar a la pantalla. Como pudimos observar en la Figura 22, un objeto tiene su propio sistema de coordenadas local, que define las **Coordenadas de Modelo**, por lo que, desde su punto de vista, no está transformado. A todos los vértices de cada uno de los modelos se la aplica la denominada **Transformación de Modelado**, que permite presionarlo y orientarlo según el SRU, lo que posibilita obtener las **Coordenadas Universales** (o coordenadas del mundo, como mencionamos en 2.2.7.1.). Como este sistema de coordenadas es único, una vez que se aplica la transformación de modelado a cada objeto, todas las coordenadas estarán expresadas en el mismo espacio.

Otro aspecto que toma un rol importante dentro de la composición de la imagen, es la posición y orientación de la cámara que va a determinar qué objetos son los que se van a representar en la imagen final. Como la cámara posee sus propias coordenadas universales, el propósito de la **Transformación de Visualización** es posicionar a la cámara en el origen del SRU. Este posicionamiento sitúa a la cámara en dirección negativa al eje Z de coordenadas y con

el eje Y hacia arriba (Figura 24). De este modo se obtienen las **Coordenadas de Visualización** o **Coordenadas Cámara**.

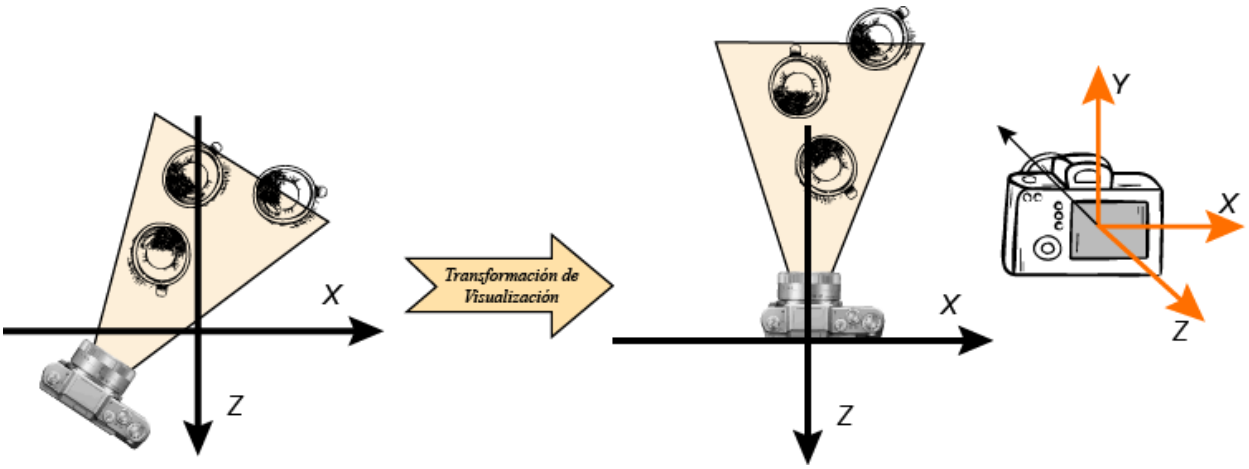


Figura 24: Transformación de Visualización

Luego del proceso de Transformación de Visualización se produce la **Transformación de Proyección**, que convierte el volumen de visualización (zona sombreada en la Figura 24), que se define mediante planos de recorte 3D y determina todos los elemento que serán visualizados, en un cubo unitario[11]. Existen muchos métodos de proyección, pero comentamos anteriormente que los más utilizados son la paralela (u ortográfica) y la de perspectiva. Una vez realizada la Transformación de Proyección, el volumen de visualización se transforma en **Coordenadas Normalizadas**, que se obtienen del cubo unitario, donde los modelos son proyectados de 3D a 2D.

Sólo los objetos que se encuentren dentro del volumen de visualización serán los generados en la imagen final. De esta forma, sólo si el objeto entero se encuentra dentro del volumen de visualización pasará íntegramente a la última etapa del Pipeline, ya que aquellos que se encuentren de forma parcial recibirán un corte generando nuevos vértices en las zonas afectadas. Esta operación se denomina **Transformación de Recorte** y es realizada automáticamente por el hardware de video (Figura 25).

[11] Cubo unitario: Es un cubo en el cual todos sus lados poseen una unidad de longitud. A veces también se llama cubo de lado 1.

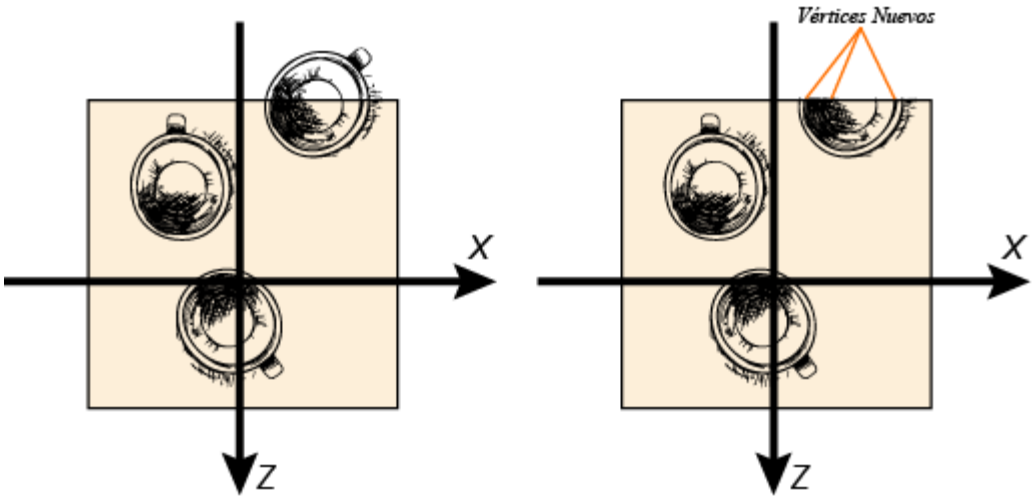


Figura 25: Ejemplo de Transformación de Recorte, objetos dentro y fuera del cubo unitario

Por último se realiza la **Transformación de Pantalla** que toma como entradas las coordenadas de la etapa anterior del Pipeline y produce las denominadas **Coordenadas de Pantalla** que ajustan las coordenadas X e Y del cubo unitario a las dimensiones de ventana finales.

2.2.8. Historia

Si bien RA es un término que parece que hubiera surgido estos últimos años, es algo que se viene desarrollando desde hace bastante tiempo. La primer idea de esta tecnología proviene del año 1901, en donde el escritor Lyman Frank Baum (“El Maravilloso Mago de Oz”) publicó su novela ilustrada “La Llave Maestra” (Figura 26), en la cual el autor imaginó unos lentes de RA. Estos lentes tenían la funcionalidad de ser capaces de ver más allá de lo que los ojos veían y presentaban una carta junto a la persona que reflejaba su verdadera personalidad. Por ello, muchos reconocen a esta novela como el precedente literario más antiguo de la RA.

Figura 26: Tapa del libro “THE MASTER KEY”

Fuente: http://1.bp.blogspot.com/-ivTmFuGLKnM/ULbHJFwez7I/AAAAAAAABrE/kx-CuDJ5Yzg/s320/IMG.jpg

Pero fuera de la ficción, la RA no se hizo realidad hasta 1968, cuando Ivan Sutherland, considerado el precursor del concepto de Mundo o Realidad Virtual, crea el primer sistema de RA (Figura 27), el cual consistía de un sistema óptico montado sobre la cabeza (HMD - Head Mounted Display), que representaba las imágenes mediantes dos monitores de rayos catódicos (CRT) en miniatura junto a dos sensores: uno mecánico y otro ultrasónico, que se utilizaban para poder localizar la posición de la cabeza del usuario dentro de un espacio tridimensional. La idea principal del dispositivo era la de brindarle al usuario una experiencia distinta de la realidad. Para lograr esto se le presentaba al usuario una imagen tridimensional en perspectiva que cambiaba a medida que éste se movía; esta imagen debía cambiar de la misma manera en que lo haría si se estuviera observando el objeto en la realidad, simulando así los movimientos de la cabeza del usuario dentro de ese espacio tridimensional. [25]

Figura 27: HMD - Head Mounted Display

Fuente: https://4.bp.blogspot.com/-JaRJPIEA0As/VtbYLhmx3KI/AAAAAAAAAiU/qEXGGCBwh-8/s1600/Ivan%2BSutherland%2527s%2Bfirst%2BVR%2Bdevice%2B%2528The%2BSword%2Bof%2BDamocles%2529.png

En 1975, Myron Kruger crea una habitacion de realidad artificial, denominada Videoplace, que permitía por primera vez a los usuarios interactuar con objetos virtuales. Combinaba cámaras de video con proyectores que emitían siluetas sobre una pantalla, envolviendo al usuario en un ambiente interactivo, como se puede apreciar en la Figura 28.

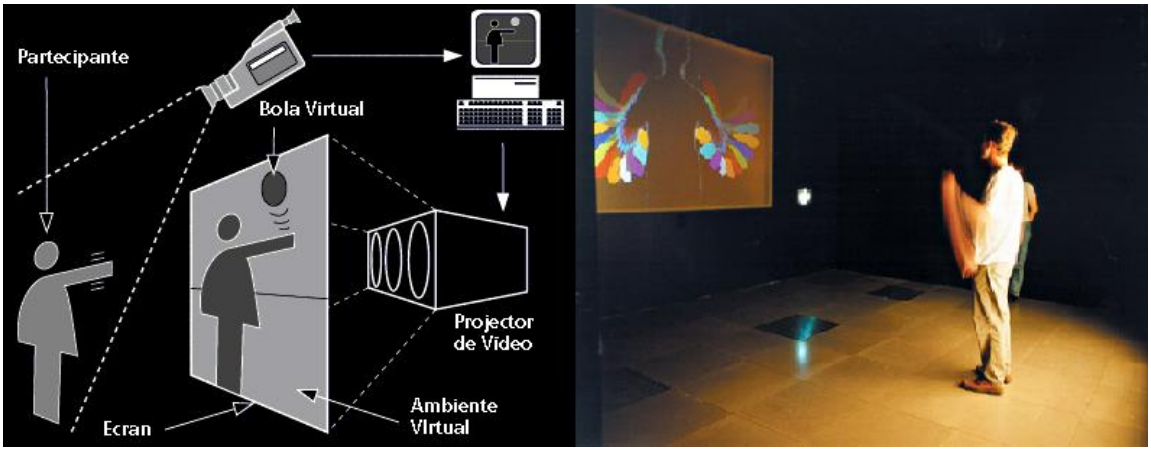


Figura 28: Videoplace - Mayron Kruger

Fuente: http://thedigitalage.pbworks.com/w/page/22039083/Myron%20Krueger

Para los años 90, más específicamente en el año 1992, Louis Rosenberg desarrolla el que es considerado el primer sistema funcional de RA, para el Laboratorio de Investigaciones de la Fuerza Aérea de los Estados Unidos. Este sistema llevaba el nombre de Virtual Fixture. Era un exoesqueleto completo para la parte superior del cuerpo que permitía a los militares controlar maquinaria a través de un entorno virtual guiado desde un espacio de operación remoto. Su funcionamiento no utilizaba los gráficos 3D debido a que en esa época no estaban todavía muy desarrollados, por ello hacía uso de robots (Figura 29).

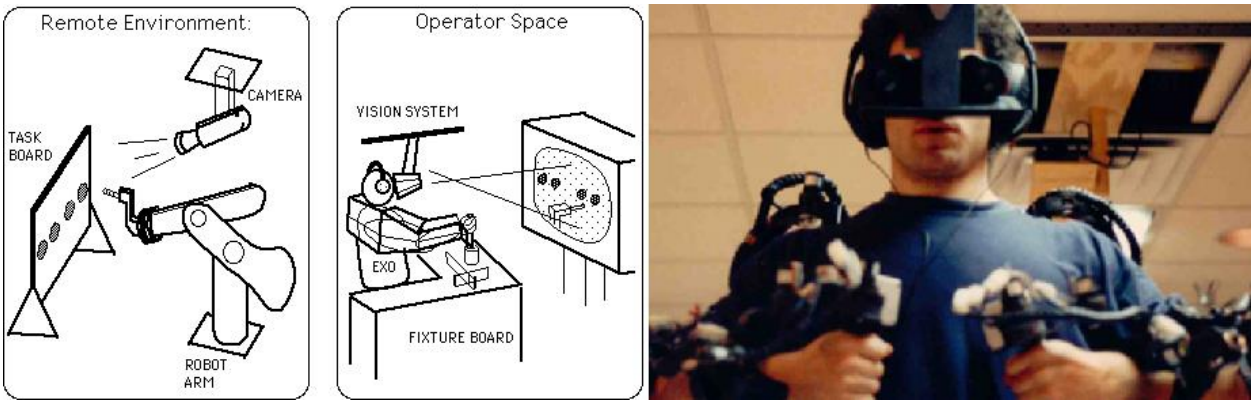


Figura 29: Virtual Fixture - Louis Rosenberg (1992)
Fuente: Izquierda: http://web.stanford.edu/group/rrd/People/vdl/publications/Biorobotics/BioRobotics.f7.gif
Derecha: http://www.newmediabusinessblog.org/images/0/0c/Virtualfixtures.png

El mismo año de la creación de Rosenberg, Steven Feiner, Blair MacIntyre y Doree Seligmann, diseñan el prototipo KARMA (Knowledge-based Augmented Reality for Maintenance Assistance), el cual es considerado la primer utilización importante de un sistema de RA, y que ayudaba en el mantenimiento de impresoras láser. Era un dispositivo que se colocaba en la cabeza junto a tres triángulos que permitían al sistema proyectar una imagen 3D para dar instrucciones al usuario sobre cómo recargar la impresora, en lugar de acudir al manual de uso (Figura 30).

Figura 30: Karma - Feiner, MacIntyre y Seligmann (1992)

Fuente: http://cdn.pocket-lint.com/r/s/970x/assets/images/108881-games-news-feature-achieving-the-ar-dream-through-games-image5-GCS1l8qieh.jpg

Si bien a lo largo de la década del 90 se fueron sucediendo distintos proyectos y aplicaciones sobre RA, no fue hasta el año 1999 que se produjo otro hecho de gran importancia para este campo. Éste vino dado por la creación de ARToolkit (Figura 31), presentado en la HITlab[12] por Hirokazu Kato, un programa informático que en sí mismo era una biblioteca orientada a la creación de aplicaciones de RA. El gran avance que realizó ARToolKit fue el de solucionar, mediante las capacidades de video del dispositivo, dos de los grandes problemas a los que se enfrentaban las aplicaciones de RA en ese tiempo: el seguimiento del punto de vista (mediante el seguimiento en video de unos marcadores[13]) y la interacción con el objeto virtual (superponiendo el modelo 3D en ese marcador real).

Figura 31: Ejemplo de utilización de ARToolKit

Fuente: http://arblog.inglobetechnologies.com/wp-content/uploads/2011/05/ARToolKit-car-demo-Android-300x225.jpg

Finalmente, con la llegada del año 2000 comienza el auge de la tecnología de RA. En concreto, en dicho año, se presenta ARQuake (Figura 32), desarrollado por Bruce H. Thomas, el primer juego con dispositivos de RA al aire libre. Era una versión del clásico juego Quake pero que usaba un dispositivo montado en la cabeza con reconocimiento de la posición, una computadora portátil y la ubicación mediante GPS para proveer los controles y la visión del juego. De esta forma, el usuario podía caminar por el mundo real mientras se enfrentaba a enemigos virtuales.

[12] HITlab (Human Interface Technology Lab): Es un laboratorio multidisciplinario de investigación y desarrollo cuyo trabajo se centra en la tecnología de interfaces humanas. Fuente: http://www.hitl.washington.edu/home/

[13] Marcador: Pieza gráfica que el dispositivo detectará para incorporar la RA. Este marcador ayuda al dispositivo a colocar correctamente un objeto 3D en el espacio real.

Figura 32: ARQuake (2000)

Fuente: https://chopsueyblog.files.wordpress.com/2008/10/arquake.jpg?w=510

Con la llegada de los teléfonos móviles más avanzados se comenzaron a desarrollar aplicaciones de RA dedicadas exclusivamente a estos tipos de dispositivos. Una de las más significativas fue la guía de viajes de Wikitude, denominada AR Wikitude Travel Guide, que fue creada en el año 2008. Era una aplicación que hacía uso de las capacidades del dispositivo para obtener la posición exacta del usuario que estaba utilizando la aplicación y, mediante las imágenes captadas por la cámara de video, se combinaba con datos adicionales tales como el nombre, distancia o cualquier dato de interés sobre accidentes geográficos, edificios u otro tipo de construcciones (Figura 33).

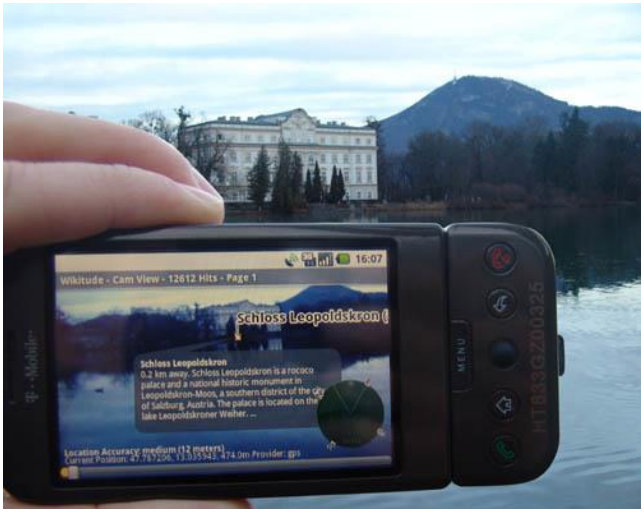


Figura 33: Aplicacion AR Wikitude Travel Guide

Fuente: http://mobilizycom.easycgi.com/images/Wikitude%20-%20500x396%20-%20real.jpg

Otros acontecimientos importantes, dejando de lado las aplicaciones para telefonía móvil, vinieron dados por los lentes Google Glass, creados por Google en el año 2013, y los HoloLens, creados por Microsoft en 2015. Tanto Google Glass como HoloLens son gafas de RA que permiten al usuario ver el mundo que lo rodea transformado, agregando objetos que flotan en el aire, pantalla virtuales sobre paredes o hasta una sala cubierta de personajes virtuales (Figura 34).

Figura 34: HoloLens (izquierda) - Google Glass (derecha)

Fuente Izquierda: https://i.blogs.es/317911/apple-glass/original.jpg
Fuente Derecha: http://e03-elmundo.uecdn.es/assets/multimedia/imagenes/2017/01/03/14834458313775.jpg

Gracias al avance que ha tenido la tecnología de la RA año a año, especialmente en estos últimos, sea mediante algún tipo de dispositivo o bien mediante la utilización de una aplicación para los teléfonos móviles o smartphones, se ha desarrollando software específico de RA para diversas y variadas disciplinas, incluyendo todo tipo de aplicaciones: redes sociales, juegos, comercio, educación, entre otras.

2.3. Modelado 3D

"Vivir no es sólo existir y crear,
es saber gozar y sufrir y no dormir sin soñar.
Descansar es empezar a morir."
Gregorio Marañón.

El modelado 3D se encarga de la creación de modelos de un objeto tridimensional mediante la definición de su geometría para que puedan ser manejados algorítmicamente mediante una computadora.

Un modelo 3D es un objeto representado dentro de un sistema de coordenadas cartesianas, normalmente definidas con las variables *x*, *y* y *z*, que, además, posee diferentes características asociadas al objeto que producen una sensación de realismo.

Para generar un modelo 3D existen distintos tipos de modelado que permiten manipular el objeto. Dentro éstos se encuentran el modelado poligonal, el modelado por curvas parametrizables y el modelado de texturizado, mapeo e iluminación.

En esta sección detallamos los tipos de modelado y los diferentes elementos que componen a los sistemas de modelado 3D, los cuales son fundamentales para manipular los objetos. Es importante aclarar que en esta Tesina no explicamos en profundidad cada uno de los conceptos matemáticos involucrados, ya que estos exceden los límites del presente trabajo.

2.3.1. Modelado poligonal

El modelado poligonal es uno de los métodos más simples para modelar; utiliza polígonos para describir la geometría del objeto. Se basa en mallas[14] poligonales formadas normalmente por triángulos o rectángulos planos. Mientras más polígonos compongan un objeto, se va a obtener una representación más real [26].

El inconveniente que tiene este tipo de modelado reside en la cantidad de vértices que posee el polígono, ya que dependiendo de esto varía el cálculo de las propiedades del objeto.

A los polígonos se lo puede clasificar en dos categorías: los cóncavos y los convexos. Los **polígonos convexos** son aquellos en los cuales al trazar una línea recta entre dos puntos cualquiera del contorno del polígono, todos los puntos que componen el segmento son puntos interiores al polígono. Los polígonos que no cumplan con esa condición son considerados **no convexos o cóncavos**.

Actualmente, las tarjetas gráficas están optimizadas para la utilización de triángulos convexos. Esto se debe a que la utilización de triángulos permite representar con fidelidad cualquier objeto.

Una de las formas más cómoda de manipular los polígonos es mediante la vista **wireframe**, que describimos a continuación, donde cada polígono se puede apreciar con mayor facilidad

2.3.1.1. Wireframe

Wireframe es una presentación visual de un objeto 3D utilizado para el modelado 3D. El algoritmo que utiliza junta las aristas del objeto mediante líneas rectas y curvas generando una malla donde sólo se ven las líneas conectoras, como podemos apreciar en la Figura 35:

[14] Una malla representa una superficie donde vértices y aristas se comparten.

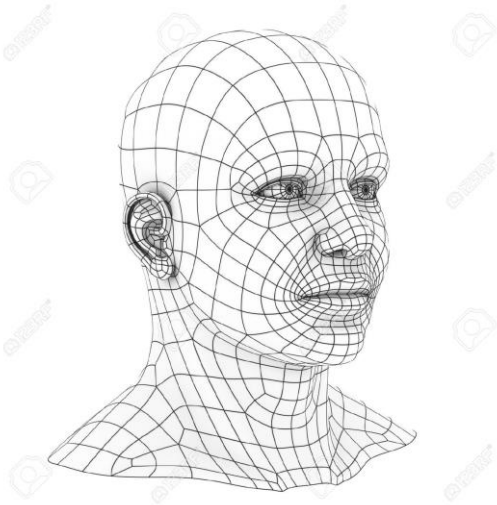


Figura 35: Wireframe de un rostro

Fuente: https://previews.123rf.com/images/koya79/koya791305/koya79130500147/19776128-cabeza-humana-3d-wireframe-Foto-de-archivo.jpg

Por ejemplo, en la Figura 36 vemos cómo es la representación de un cubo:

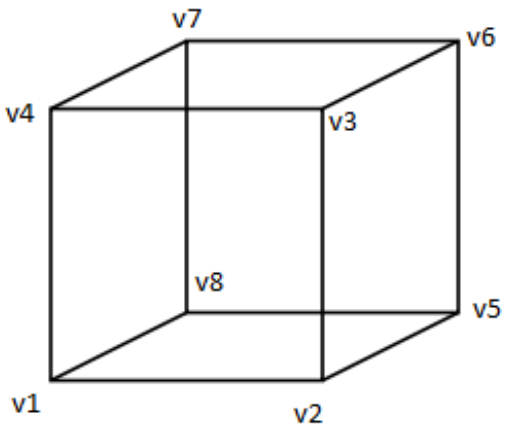


Figura 36: Wireframe de un cubo

Un objeto representado por wireframe necesita una **tabla de vértices** y una **tabla de aristas** para describirlo. Las Tablas 3 y 4 describen los vértices y aristas del cubo de la Figura 36:

Vértices	X	Y	Z
v1	0	0	0
v2	1	0	0
v3	1	1	0
v4	0	1	0
v5	1	0	1
v6	1	1	1
v7	0	1	1
v8	0	0	1

Tabla 3: Tabla de vértices del cubo

Segmento	Vértice Inicial	Vértice Final
1	v1	v2
2	v1	v4
3	v1	v8
4	v2	v3
5	v2	v5
6	v3	v4
7	v3	v6
8	v4	v7
9	v5	v6
10	v5	v8
11	v6	v7
12	v7	v8

Tabla 4: Tabla de aristas del cubo

2.3.2. Modelado por curvas parametrizables

Antes de comenzar a hablar de modelado mediante curvas parametrizables, es necesario que definamos dos conceptos que nos van a servir como base: curvas y superficies

Según Evguenii Kurmyshevn [27], una **curva en 2D** se define de la siguiente manera: "*Una curva en el espacio es un objeto geométrico unidimensional, es decir, se caracteriza por sólo una variable independiente.*"

Dentro de un sistema de coordenadas, se puede representar una curva C en el plano mediante la siguiente función:

□(□) = □(□)□ + □(□)□

Donde a cada valor de la variable real □ (variable independiente), le corresponde un punto C que tiene el vector de posición en las coordenadas (□(□), □(□)). Esta función también es conocida como función paramétrica.

Por otro lado, una **superficie en 3D** se define como un conjunto continuo bidimensional de puntos y se puede expresar mediante las siguientes funciones:

□ = □(□, □) □ = □(□, □) □ = ℎ(□, □)

Con estos conceptos de base podemos hablar de este modelado que se refiere a la obtención de mallas o cuadriláteros curvados que están definidos por superficies curvadas de puntos. Cada conjunto de estas superficies se define por una fórmula matemática que establece el espacio tridimensional, con la que se genera cada uno de los puntos del cuadrilátero. Se pueden realizar modificaciones de la forma o curvatura de la pieza mediante la modificación de algún parámetro de la misma.

Los tipos de curvas más comunes en los software de modelado 3D tridimensionales son: las **curvas de Bézier**, los **B-splines** y las **NURBS**. Cada una de estas curvas permite obtener una representación muy aproximada del modelo que se pretende crear. Por otro lado, cuando se modifica alguno de los parámetros que lo definen, se debe calcular nuevamente la forma generada en su totalidad o, en el mejor de los casos, de los vértices adyacentes.

2.3.2.1. Curvas de Bézier

Se denomina curva de Bézier a un método de definición de una curva en serie de potencias. Consiste en definir algunos puntos de control, a partir de los cuales se calculan los puntos de la curva.

Según Ángel Pareja León [26], los puntos de control se definen en base a la siguiente fórmula:

$$(x(t), y(t)) = \sum_{i=0}^{n} B_{i,n}(t) P_i$$

Donde □□ son los puntos de control, y □□,□ los llamados Polinomios de Bernstein.

Por lo tanto, mientras más puntos de control □□ se agreguen, se va a lograr una curva más definida, como se ve en la Figura 37 [28]:

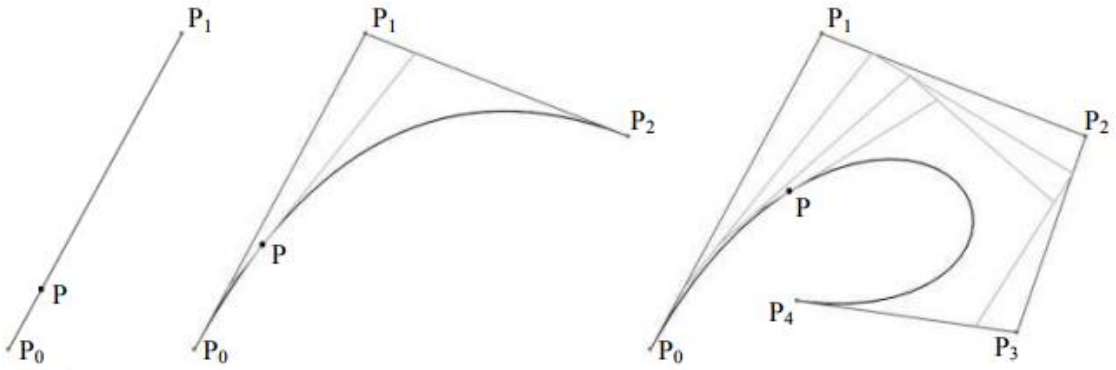


Figura 37: Variación de una curva respecto de los puntos P_i

Fuente: http://www.cimec.org.ar/~ncalvo/curvas_doc.pdf

Cuando se agrega un punto de control, se interpola linealmente en cada uno de los segmentos y luego entre los puntos resultantes, siempre con el mismo valor del parámetro. Este proceso se repite en forma iterativa por cada punto de control que se agrega.

2.3.2.2. B-splines

B-spline es una curva parametrizada por otras funciones spline[15]. Las curvas de Bézier son un tipo de B-splines; en realidad una curva B-spline es simplemente una generalización de una curva de Bézier.

La curva B-spline, según Ángel Pareja León [26], se define de la siguiente manera:

$$(x(t), y(t)) = \sum_{i=0}^{n} N_i^k(t) P_i$$

Donde □□ son los puntos de control y □□□(□) las funciones de base o funciones de Bernstein, las cuales definen la curva de la sección entre dos puntos de control.

De forma similar a las curvas de Bézier, definen la precisión y suavidad de la curva en base a las funciones de grado k.

La ventaja que posee B-spline, comparado con las curvas de Bézier (Figura 38), es que los puntos de control que forman la curva no son tan distintos a los de la curva final que se quiere modelar.

[15] Función spline: es una función que se utiliza en aplicaciones que requieren la interpolación de datos o un suavizado de curvas. Un spline es una curva definida en porciones mediante polinomios.

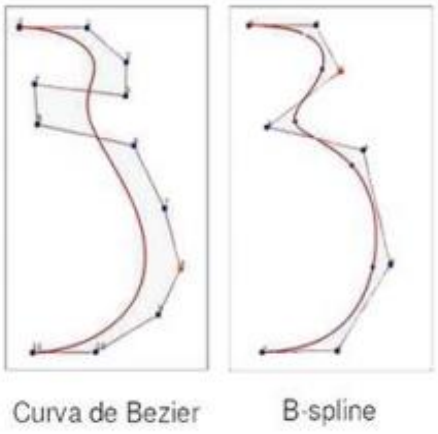


Figura 38: Comparación entre la curva de Bézier y B-spline

Fuente: https://eciencia.urjc.es/bitstream/handle/10115/12069/Memoria%20PFC%20Lamborghini%20Gallardo%20-%20%C3%81ngel%20Pareja%20Le%C3%B3n.pdf?sequence=1&isAllowed=y

2.3.2.3. NURBS

Las Non Uniform Rational B-splines (NURBS) corresponden a un caso más general de las curvas anteriores, en donde los B-splines y las curvas de Bézier son casos particulares de NURBS. No son simples polinomios, sino cocientes de polinomios. Este tipo de curvas surge del problema de que con curvas polinómicas no se pueden cubrir todo tipo de curvas, como sucede, por ejemplo, con una circunferencia.

Según Ángel Pareja León [26], su función de generación está definida de la siguiente forma:

$$B(t) = \sum_{i=0}^{n} \frac{w_i N_i^k(t)}{\sum_{i=0}^{n} w_i N_i^k(t)} b_i$$

Donde $N_i^k(t)$ son las funciones de base y w_i los coeficientes de pesos. Si aumentamos el valor de un peso w_i la función de base $N_i^k(t)$ aumenta, lo que implica que la NURBS "se curva" hacia el punto b_i. Si disminuimos el valor del peso, la curva "se aleja" del punto b_i. Cuando se utiliza un peso $w_i = 0$, la función de base $N_i^k(t) = 0$ y el punto b_i no intervienen en la fórmula de la NURBS.

2.3.2.4. Creación de superficies

Existen muchas variantes útiles para construir superficies. La más sencilla de comprender es el producto cartesiano o tensorial de curvas de Bézier o de NURBS. Para definir una superficie como un producto cartesiano, se la debe asumir como una función de dos parámetros (u,v), como explicamos en la sección 2.3.2.

Para un valor dado del parámetro *v* fijo se obtiene una curva isoparamétrica con *u* variable, y lo mismo sucede si se fija *u* y se varía *v*. La superficie puede pensarse como combinación de "curvas de control", en el mismo sentido en que las curvas son combinaciones de puntos de control.

Los puntos de control de la superficie forman una grilla rectangular de (n+1) por (m+1) puntos que pueden ubicarse libremente en el espacio, pero están "interconectados" como en una cuadrícula [28].

2.3.3. Modelado de texturizado, mapeado e iluminación

Esta técnica de modelado se refiere a la parte estética que le da realismo al modelo 3D. Si bien anteriormente explicamos cómo se modela desde el aspecto morfológico, no hay que quitarle importancia al impacto visual que genera agregarle el texturizado e iluminación a un objeto.

2.3.3.1. Fuentes de luz e iluminación

El modelado por iluminación se refiere al cálculo de intensidad de cada punto de color en la escena. Para ello, intervienen varios factores como la intensidad de la luz, el material del objeto y la orientación del mismo respecto a la dirección de la luz. Se pueden definir seis tipos de fuentes de luz [26]:

1. **Punto de luz**: Un punto de luz es una luz que fluye en todas direcciones, de ahí a que también sea denominada luz omnidireccional (Figura 39.1). Éste es el tipo más simple de fuente de luz y puede ser situado en cualquier punto de la escena.

2. **Luz de cono**: Esta simulación de luz define un tipo de luz que genera un cono de iluminación desde la fuente hasta el plano que se establezca como tope, por lo tanto define un factor de incidencia según la distancia del objeto a ella (Figura 39.4).

3. **Luz direccional**: Esta tipología de luz simula la incidencia de rayos desde una fuente de luz situada en el infinito y de forma paralela al objeto. Se puede ligar al tipo de luz que ofrece, por ejemplo, una estrella en el firmamento o el propio sol. Esta luz se puede emplazar en cualquier lugar de la escena.

4. **Luz de área**: La luz de área se puede entender como una luz que se aplica en un área en concreto y que proviene desde un punto de grandes dimensiones (Figura 39.4). Utilizando un analogismo fotográfico, sería la luz que se visualiza al utilizar una caja de luz.

5. **Luz lineal**: La luz lineal tiene longitud, pero no anchura. Es la luz típica que ofrece un fluorescente en la vida real (Figura 39.2). Debe ser utilizada con sumo cuidado puesto que su costo computacional es muy elevado.

6. **Luz de ambiente**: La luz irradiada desde esta fuente es distribuida por toda la escena completa. Es la luz que, por defecto, tenemos en cualquier escena, determinando el nivel de iluminación y sombreado de ésta.

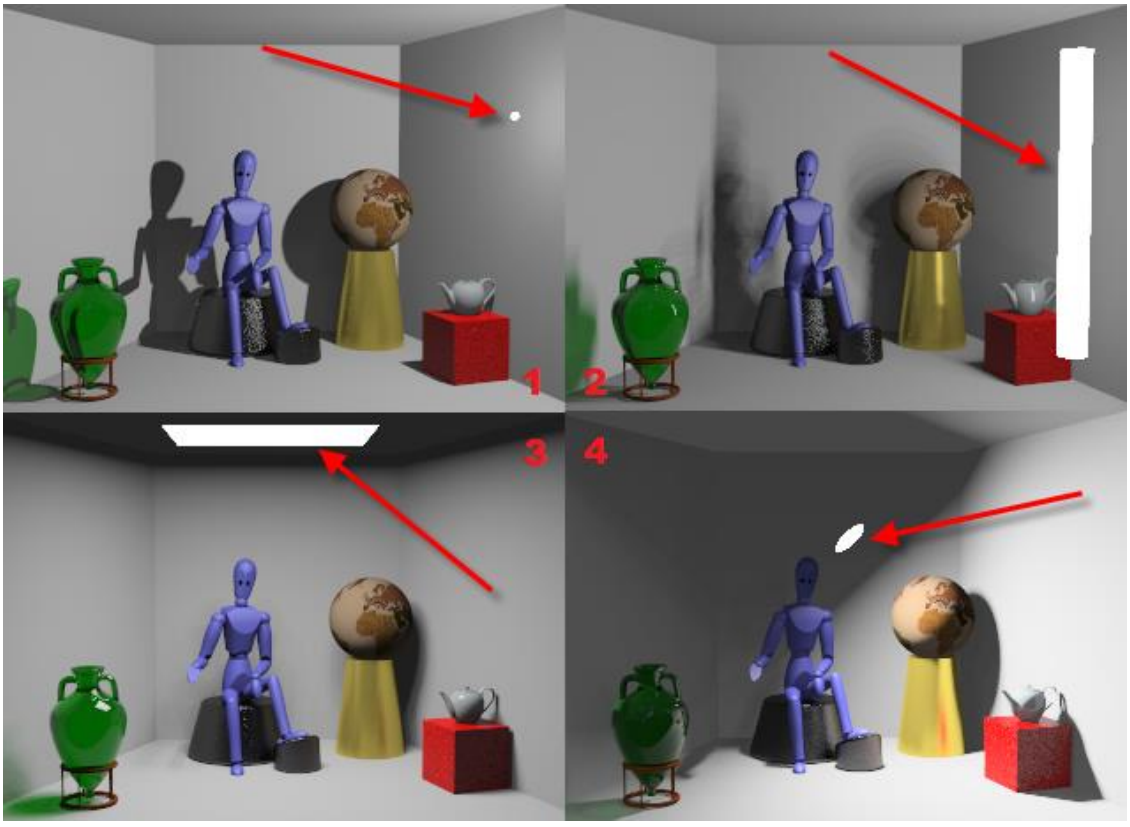


Figura 39: 1 Punto de luz. 2 Luz lineal. 3 Luz de área. 4 Luz de cono

2.3.3.2. Sombreado

Otro aspecto importante a la hora de modelar es el sombreado que poseen las superficies de un objeto. El sombreado se calcula para cada superficie, basándose en su vector normal mediante diversos algoritmos.

La mayoría de las técnicas de sombreado calculan la sombra local, de caras, el suavizado y la sombra o brillo especular en la superficie. Atendiendo a su tipología, se tienen tres tipos de sombreados [26]:

1. **Sombreado de caras (flat shading)**: Este sombreado asigna un único valor constante a cada polígono visible de la superficie según el ángulo respecto a la fuente de luz. En general, para realizar el cálculo del sombreado sobre una superficie dada, se realiza una media en base a los valores de sombreado que posean las esquinas o vértices del polígono.
   También es llamado en algunas ocasiones sombreado poligonal, dado la forma de realizar el cálculo por polígonos.
   Esta forma de cálculo de sombreado es una de las más simples; solamente se debe tener el cuenta la dirección de la luz respecto a cada polígono

2. **Sombreado suavizado (smooth shading)**: En este tipo de sombreado se obtienen valores continuos para toda la superficie del objeto. La idea básica de esta técnica es promediar las normales superficiales de los polígonos con sus adyacentes, creando una transición suavizada entre ambos. Normalmente se calculan, en primera instancia, los valores de sombreado de todos los vectores normales de las caras del objeto y se interpolan los valores entre ambos para el resto de la superficie. Tan sólo cuando un ángulo entre dos normales es mayor al límite establecido, el sombreado genera un nuevo valor y suaviza la superficie.
   Un tipo de sombreado suavizado muy utilizado es el sombreado de Gouraud.
   El sombreado suavizado contempla las propiedades de la luz de ambiente y difusa del objeto y provoca que los objetos con cierta complejidad superficial luzcan realmente bien.

3. **Sombreado especular**: Esta técnica de sombreado genera superficies brillantes y colectivas, logrando un reflejo de la luz que le da mayor realismo. Además de generar superficies suaves y continuas, como se realizan en el sombreado suavizado, calcula el color de cada uno de los vértices del polígono y sus reflejos, a lo largo de varios puntos de la superficie de la malla.

Las bases de este tipo de sombreados son los algoritmos desarrollados por Phong y sus variaciones. La técnica de sombreado desarrollada por Phong realiza interpolaciones de normales en vez de realizarla sobre intensidades. En cada vértice del polígono se calcula la normal como la media de las normales de los polígonos adyacentes, y la normal de los puntos intermedios se calcula por interpolación lineal. Los atributos tenidos en cuenta por este proceso son la luz ambiente, difusa, y la luz especular, ofreciendo resultados de sombreados realmente buenos, pero su costo computacional es bastante más elevado que los métodos anteriores.

En la Figura 40 podemos observar una comparación de las diferentes técnicas y el nivel de detalle que presenta cada una [29]:

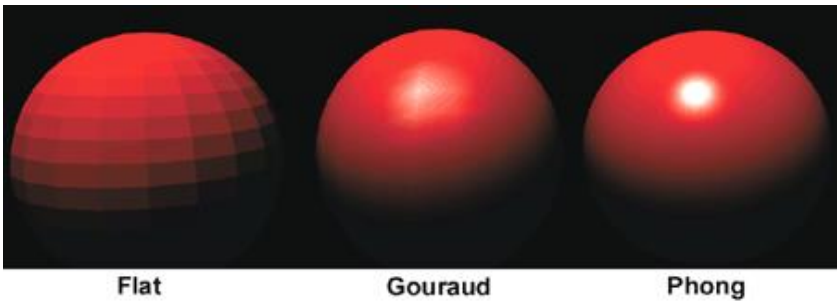


Figura 40: Distintos algoritmos de sombreado

Fuente: http://blenderperu.org/wp-content/uploads/2015/11/SHADING.gif

2.3.3.3. Texturizado

El texturizado es la técnica mediante la cual se agrega “piel” a un objeto 3D; posibilita aumentar el nivel de detalle, cubriendo los diferentes polígonos con una/s imágen/es.

Permite cambiar el color o incluso la sensación de rugosidad a la superficie. La imagen (en general de dimensiones cuadradas y potencia de 2) se aplica mediante una función matemática sobre un objeto. Esto define la correspondencia entre cada uno de los píxeles de la textura y uno o varios puntos del objeto, puesto que un punto del objeto sólo almacena la información de un único punto de la textura, y el mismo punto de la textura puede aplicarse a varios objetos. Como cada punto guarda la información de qué textura utiliza, permite incluir más de una textura en el objeto [26].

Para poder lograr el texturizado se debe mapear una imagen bidimensional en la superficie de un objeto. También se puede emplear una combinación de diferentes imágenes, cada una de ellas definiendo una característica de la superficie. En muchas ocasiones, las imágenes son repetidas a lo largo de una superficie para ahorrar costo computacional. Para la obtención de la/s imágen/es se puede/n utilizar tanto imágenes pintadas, fotografías o incluso patrones sintéticos.

Existen muchas técnicas de mapeado, como la proyección o projecting (Figura 41) y la envoltura o wrapping (Figura 42), y, dependiendo de qué técnica se utilice, cambian algunos atributos (rugosidad, reflectividad, entre otros).

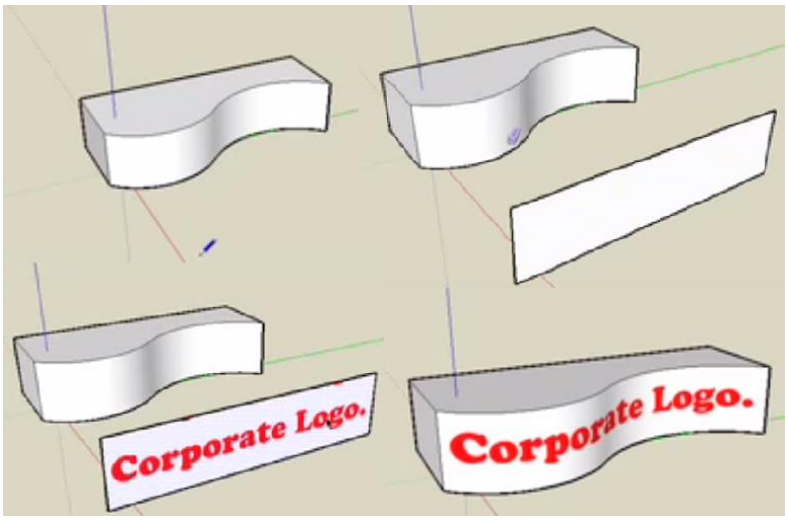


Figura 41: Técnica de proyección

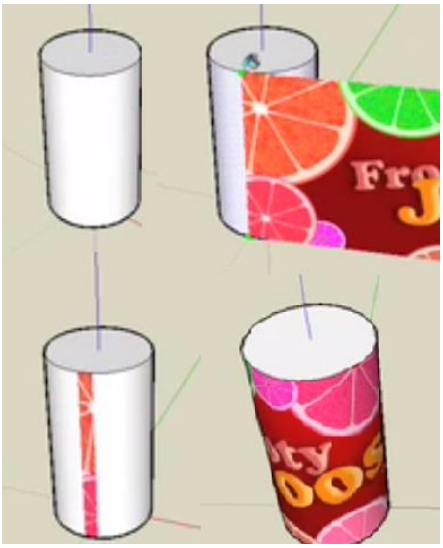


Figura 42: Técnica de envoltura

2.3.4. Renderizado

El término renderizado viene del inglés *render*, y se utiliza para referirse al proceso de generar una imagen o video mediante el cálculo de iluminación partiendo de un modelo en 3D.

Lo que se pretende es obtener una simulación realista del comportamiento tanto de luces, texturas y materiales, así como también de los comportamientos físicos.

Para realizar el proceso final de renderizado, dependiendo de la potencia que tenga el motor de renderizado (de cada aplicación), se utiliza una combinación de las técnicas anteriores: modelado poligonal, modelado por curvas parametrizables y modelado de texturizado, mapeado e iluminación.

2.3.5. Elementos básicos de los sistemas de modelado 3D

En la presente sección explicamos los elementos que componen los sistemas de modelado 3D, que nos permiten visualizar o modificar el objeto de diferentes formas [30].

2.3.5.1. Escena

Es el archivo que contiene toda la información necesaria para identificar y posicionar los modelos, luces y cámaras para su renderización.

Una escena puede identificarse con un sistema de coordenadas en el espacio, en las cuales tiene lugar la renderización. Este espacio a menudo se denomina sistema de coordenadas universal o mundo (como explicamos en la sección 2.2.7.1.). Pero, al operar con los objetos de la escena, se pueden utilizar diferentes sistemas de coordenadas como, por ejemplo, el sistema de coordenadas local del mismo objeto.

Otra característica que posee la escena es la iluminación, que es fundamental para imprimirle realismo.

2.3.5.2. Materiales y texturas

En los sistemas de modelado los objetos tienen por defecto un color uniforme y liso en su superficie. Con los materiales y texturas se puede cambiar la visualización del objeto para que adquiera mayor realismo.

2.3.5.3. Transformaciones (mover, rotar y escalar)

Cuando hablemos de transformación debemos tener muy claro a qué nos referimos. Su concepto es muy sencillo: transformar es mover, rotar, escalar. Estas acciones permiten manipular el objeto 3D.

En la Sección 2.2.7.2 explicamos el fundamento matemático de estas acciones.

2.3.5.4. Visores

Los visores son herramientas imprescindibles para trabajar en cualquier programa de modelado 3D, ya que muestran el entorno, en apariencia tridimensional, desde diferentes puntos de vista. Por defecto, en un programa de modelado 3D se tienen cuatro visores :

- El visor que muestra la escena desde la parte izquierda (sección lateral).
- El visor que muestra la escena desde arriba (planta).
- El visor que muestra la escena desde la parte frontal (sección frontal).
- El visor que muestra la escena desde una perspectiva cualquiera.

En función de la operación de modelado que se esté realizando, convendrá visualizar de un modo u otro la escena. Para ello se utilizan distintos formatos del visor (o modo render del visor). Los principales son:

- **Modo box**: Es muy esquemático. Representa los objetos como cajas que los contienen.
- **Modo wireframe**: En él se ven las aristas de los objetos (como explicamos en la sección 2.3.1.1.).
- **Modo suavizado**: Es una aproximación al render final, donde se ve la superficie del elemento. También se puede conjugar con la wireframe viendo las dos a la vez.

Por otro lado, también existen distintos tipos de vista desde las cuales se puede observar un objeto:

- **Vistas Axonométricas (u ortogonales)**: Estas vistas se refieren a la proyección paralela de un objeto 3D con la pantalla. Los lados del objeto tienen la misma inclinación respecto a la pantalla, con lo que se produce un escorzo[16] uniforme en los bordes del objeto. En esta vista se elimina el efecto de distancia del visor.

- **Vistas Ortográficas**: Son aquellas que ofrecen una superficie plana definida por los ejes de coordenadas universales. Por lo tanto, las únicas posibilidades que brindan los ejes X ,Y, Z son: superior, inferior, anterior, posterior, izquierda y derecha.

- **Vista Usuario**: Es aquella que queda definida por la persona que trabaja con la escena, partiendo de una vista ortográfica y haciéndola rotar. Cada punto de visión que se logra al hacer la rotación es una vista axonométrica, por lo cual las líneas permanecen siempre paralelas, al contrario que en la vista perspectiva.

- **Vista Perspectiva**: Es la que más se asemeja a la vista humana. En ella los objetos dan la sensación de profundidad y espacio.

2.3.6. Historia del modelado 3D

Los pasos iniciales en la historia del modelado 3D fueron dados en 1950 por Ben Laposky, que creó las primeras imágenes gráficas utilizando un osciloscopio, es decir imágenes generadas por una máquina electrónica análoga [31].

Unos años más tarde, en 1955 en el Lincoln Laboratory del Instituto de Tecnología de Massachusetts (MIT), se desarrolló el primer sistema gráfico SAGE (Semi Automatic Ground Environment) de las Fuerzas Aéreas Norteamericanas. Éste procesaba datos de radar y otras informaciones de localizaciones de objetos mostrándolos a través de una pantalla CRT.

Casi una década después, en 1963, Ivan Sutherland (considerado el padre de la computación gráfica) creó el sistema Sketchpad basado en su propia tesis doctoral “A Machines Graphics Communications System”, estableciendo las bases que conocemos hoy en día sobre los gráficos interactivos por computadora. Sutherland propuso la idea de utilizar un teclado y un lápiz óptico para seleccionar, situar y dibujar una imagen representada en la pantalla (Figura 43).

[16] Representación de una figura situada oblicua o perpendicularmente al plano de la pantalla, que se logra acortando sus líneas de acuerdo con las reglas de la perspectiva.

Figura 43: Sketchpad de Sutherland

Fuente:http://design.osu.edu/carlson/history/images/ivan-sutherland.jpg

La mayor innovación fue la estructura de datos utilizada por Sutherland. Estaba basada en la topología del objeto que iba a representar, es decir, describía con toda exactitud las relaciones entre las diferentes partes que lo componían. De esta manera, introdujo lo que se conoce como programación orientada a objetos, un concepto innovador en ese tiempo. Antes de eso, las representaciones visuales de un objeto realizadas en una computadora se basaban en un dibujo y no en el objeto en sí mismo. Con el sistema Sketchpad, se trazaba una clara distinción entre el modelo representado en la estructura de datos y el dibujo que se veía en la pantalla.

En Sketchpad, los gráficos en 3D eran construidos desde polígonos vistos en modo wireframe, lo cual permitía ver el objeto por su parte trasera tan fácilmente como por su parte frontal. Ésto causó un gran revuelo, demostrando que la computadora era capaz de calcular qué líneas eran las que definían la parte observable del objeto mientras eliminaba de la pantalla el resto. Las líneas ocultas se almacenaban en la memoria de la computadora y volvían a aparecer cuando se colocaba el cuerpo en una posición diferente respecto al observador. Las limitaciones del sistema provenían de la capacidad de la computadora.

En ITEK y General Motors se desarrollaron proyectos paralelos al Sketchpad, pero sin tanta trascendencia.

Posteriormente, en 1973, la Universidad de Utah inició un programa financiado por el Departamento de Defensa de Estados Unidos, del cual, en 1974, Sutherland fue la cabeza. Allí se desarrollaron algoritmos para generar sombras en la superficies de los objetos, de acuerdo al impacto de la luz. Bui Tuong Phong observó que el impacto de la luz en los objetos generaba reflejos y desarrolló lo que él llamó el “algoritmo Phong shader”, que sirvió como base para lo que hoy conocemos como sombreado especular. Este algoritmo producía buenos resultados pero

era muy lento para renderizar. Fue entonces cuando Jim Blinn, graduado de la Universidad de Utah, usó el trabajo de Phong para mejorar el renderizado. En esta Universidad, también se desarrollaron muchos avances, entre ellos el mapeado de texturas, sombras y animación facial.

En 1975, también en la Universidad de Utah, Martin Nexwell desarrolló uno de los objetos más conocidos dentro del ámbito del modelado 3D: la tetera de Nexwell (Figura 44). Para lograrlo utilizó curvas de Bezier, introduciendo manualmente los puntos de control [32].

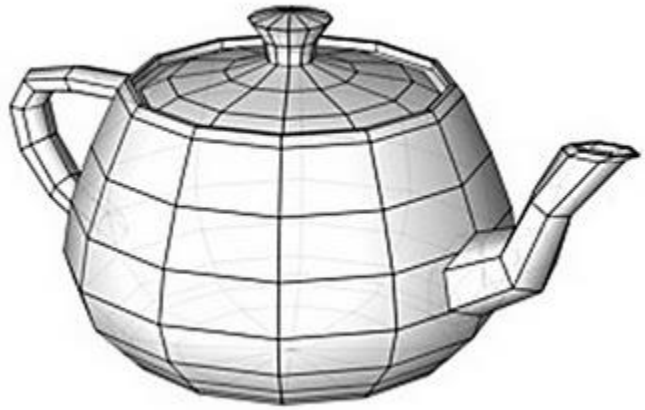


Figura 44: Tetera de Nexwell o Tetera de Utah

Fuente: https://i1.wp.com/designaholic.mx/wp-content/uploads/2013/06/1VisibleAndWire.jpg?w=400

En los años siguientes, con las mejoras del hardware, se fueron utilizando en mayor medida los gráficos en tres dimensiones. En 1982, Steven Lisberger desarrolló Tron, la primera película de Disney que usó masivamente gráficos en 3D (Figura 45).

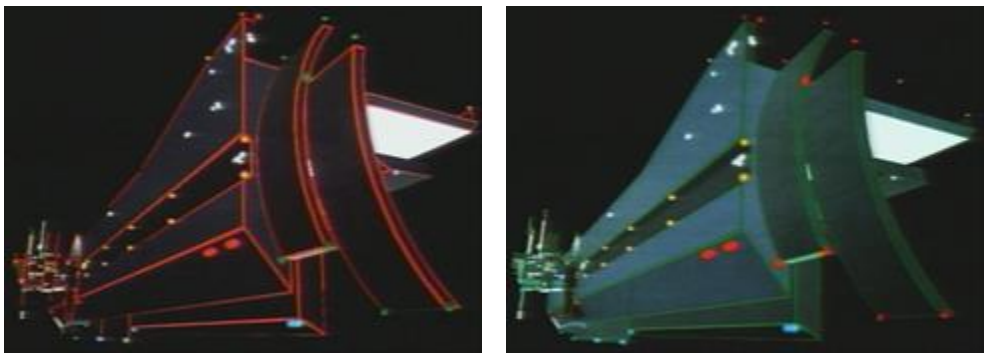


Figura 45: Gráficos 3D de Tron

Fuente: http://tron-sector.com/gallery/default.aspx?a=g&s=1&c=4

Un año más tarde, Jaron Lanier, comenzó con el uso de guantes con interruptores y sensores para detectar movimientos de la mano, con los que grabaron parte de la película de realidad virtual DataGlove.

En 1985, Pixar Animation Studios realizó el cortometraje Luxo Jr. y, en 1989, Tin toy.

Años más tarde, en 1993, Steven Spielberg dirigió Jurassic Park, una exitosa película de ciencia ficción a partir de efectos de computación gráfica.

Y, dos años más tarde, Buena Vista Pictures creó el primer largometraje completamente generado en computadora: Toy Story (Figura 46).

Figura 46: Toy Story

Fuente: http://toystory.disney.com/toy-story-gallery?

3. Herramientas

En este capítulo describimos los entornos de desarrollo y las herramientas investigadas disponibles para el desarrollo de la aplicación, indicando cuáles elegimos y en qué parámetros nos basamos para realizar dicha la elección.

3.1.Entornos para desarrollo

Los Entornos de Desarrollo Integrado (IDE o Integrated Development Environment) son herramientas o plataformas que permiten a los desarrolladores poder llevar a cabo sus tareas con mayor facilidad. Normalmente cuentan con procesadores sintácticos para ayudar a la escritura del código y suelen utilizar colores para sectores de códigos significantes. Muchos de los IDE permiten la inclusión de plugins[17] para extender sus capacidades.

Para el desarrollo de nuestro proyecto decidimos investigar dos de los IDE más conocidos: Unity 3D y Android Studio.

3.1.1. Unity 3D

Figura 47 : Logo Unity 3D

Esta herramienta es un motor (gráfico) de videojuegos en 2D y 3D que permite un desarrollo rápido de aplicaciones gracias a la amplia variedad de extensiones y modelos que presenta a través de su tienda Asset Store. [33]

También es importante destacar que tiene un soporte multiplataforma que permite realizar aplicaciones para Windows, Linux, Android, iOS, Windows Phone, PlayStation, XBox, Wii, Nintendo, entre otras.

[17] Plugin: aplicación o programa informático que se relaciona con otro para agregarle una función nueva y, generalmente, muy específica.

3.1.1.1 Interfaz

La interfaz de Unity 3D es bastante fácil de aprender y contiene elementos visuales presentados en ventanas con los cuales interactúa el usuario para desarrollar los proyectos. Sus componentes principales son: la escena, la ventana de proyecto, la ventana de inspección, la ventana de jerarquía, la ventana de juego y Asset Store (tienda). En la Figura 48 podemos apreciar la interfaz de Unity 3D y sus componentes:

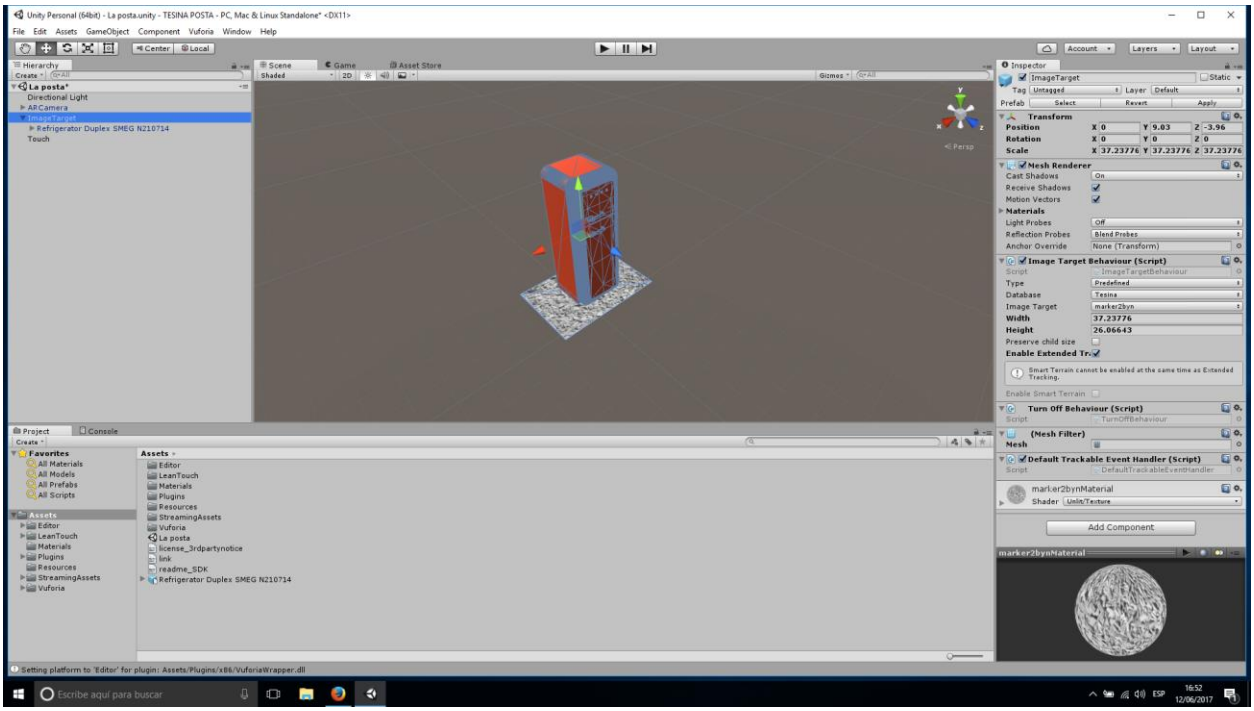


Figura 48: Interfaz de Unity 3D

La **escena** muestra de forma gráfica la ubicación y tamaño de los objetos que van a ser visualizados (o no) en tiempo de ejecución. En la Figura 49 podemos observar la escena con el modelo 3D de una heladera sobre una textura que debe captar la cámara al momento de realizar el tracking de la imagen.

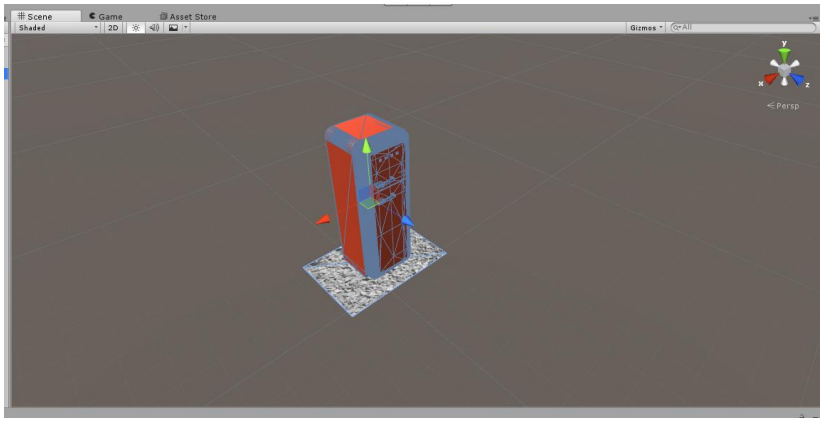


Figura 49: Escena con el modelo 3D relacionado con su marcador

En la **ventana de proyecto** (Figura 50) se presentan todos los elementos que se encuentran disponibles en el proyecto (como materiales, scripts, modelos, etc.), los cuales se podrán agregar a la aplicación en caso de ser necesario.

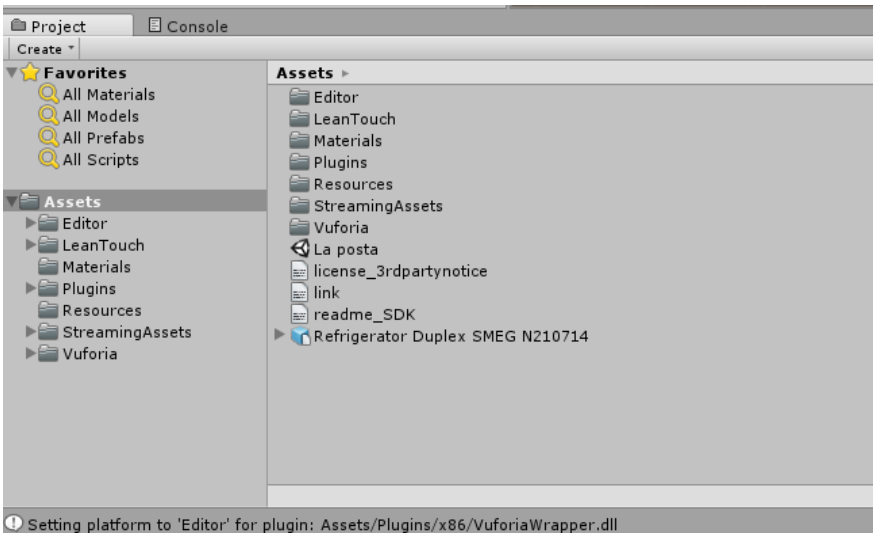


Figura 50: Ventana de proyecto

En la **ventana de inspección** (Figura 51) se encuentran las configuraciones que permiten las API de los objetos. Una de las configuraciones más común es la de los ejes X, Y, Z para la ubicación dentro de la escena.

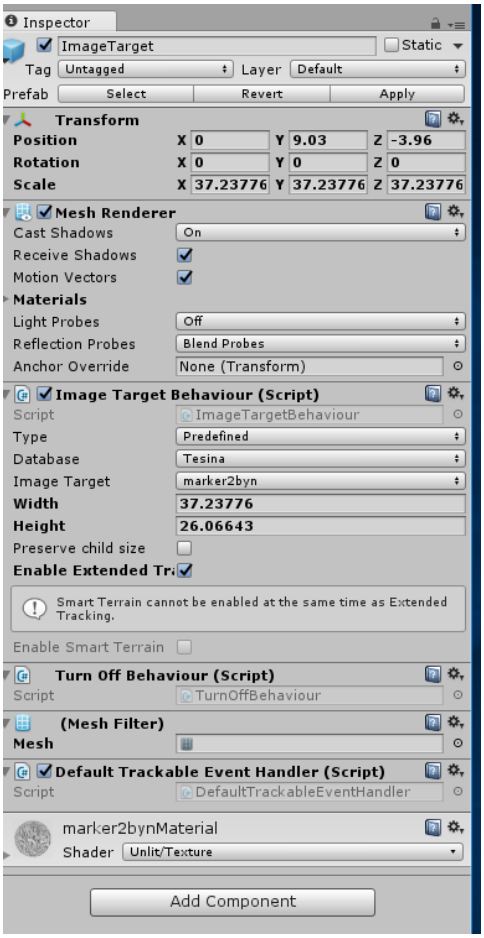


Figura 51: Ventana de inspección

La **ventana de jerarquía** muestra de forma gráfica la dependencia (jerarquía) de los objetos con los que cuenta el proyecto. Por ejemplo, en la Figura 52 vemos que el objeto Refrigerator depende de un ImageTarget, creando la relación del modelo 3D de la heladera con la imagen que luego buscará la cámara para hacer el seguimiento.

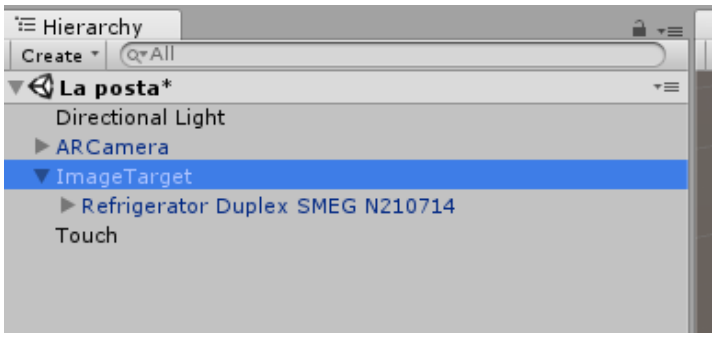


Figura 52: Ventana de jerarquía

La **ventana de juego** (Figura 53) es donde se ejecutan las pruebas de la aplicación.

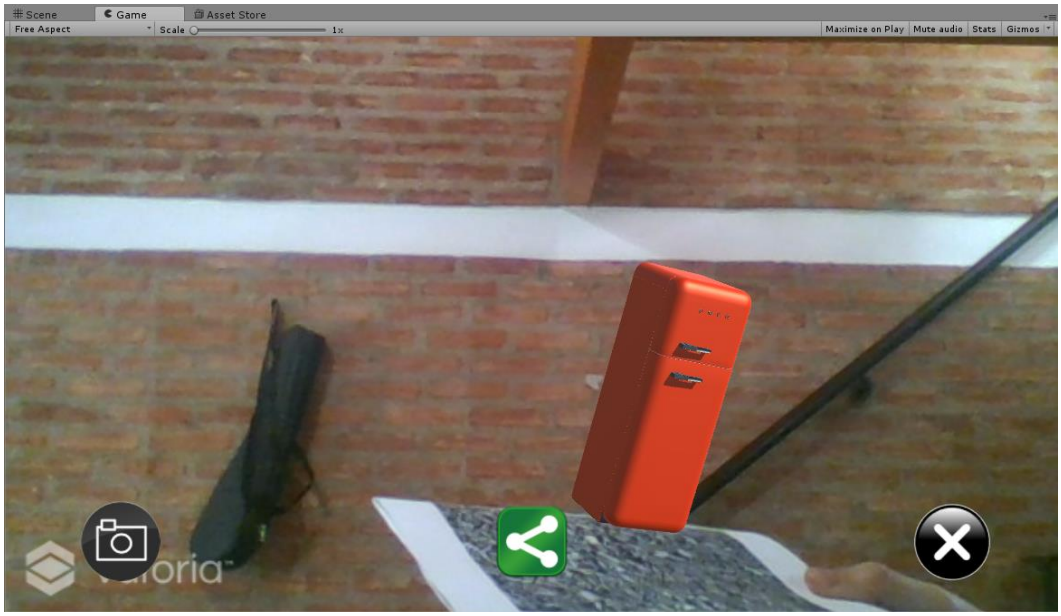


Figura 53: Ventana de juego

Asset Store (Figura 54) es la tienda que posee Unity 3D, donde se pueden encontrar desde modelos 3D hasta scripts para manejar la pantalla táctil de un dispositivo móvil.

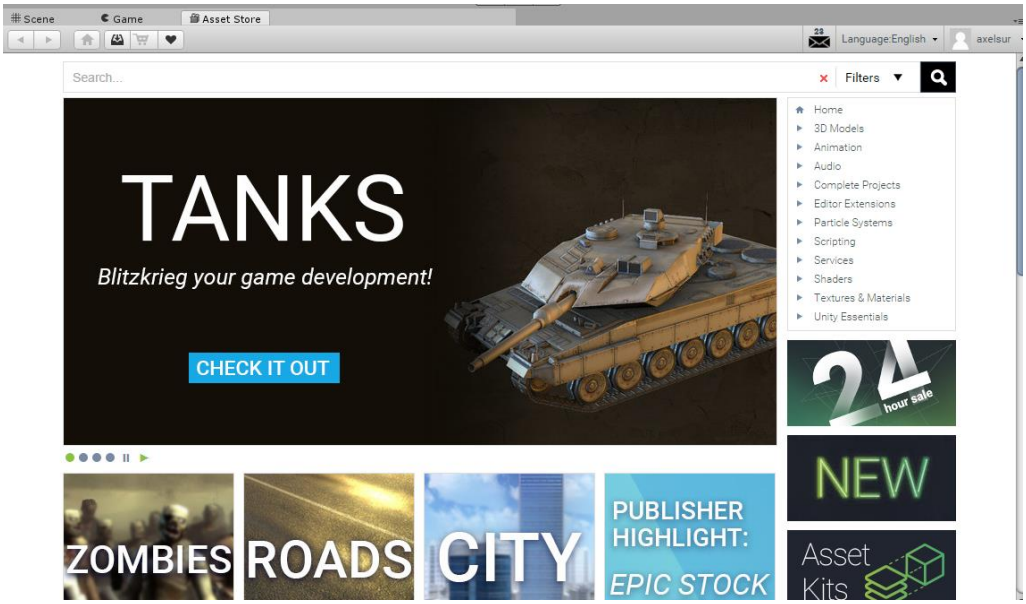


Figura 54: Asset Store

3.1.2. Android Studio

Figura 55: Logo Android Studio

Android Studio es el entorno oficial para el desarrollo de aplicaciones bajo el sistema operativo Android. Es similar a los IDE más conocidos, como ECLIPSE. [34]

Esta herramienta multiplataforma se puede ejecutar bajo Windows, Linux, etc., siempre que se instale previamente el JDK[18] correspondiente al sistema operativo.

Utiliza como lenguaje a Java junto con XML, aunque también se puede incorporar código en C o C++.

3.1.2.1. Estructura del proyecto

Cada proyecto en Android Studio está distribuido en módulos de código, entre ellos: módulos de apps para Android, módulos de bibliotecas y módulos de Google App Engine.

Android Studio muestra los archivos del proyecto en una estructura en árbol, como podemos ver en la Figura 56:

[18] JDK: Java Development Kit.

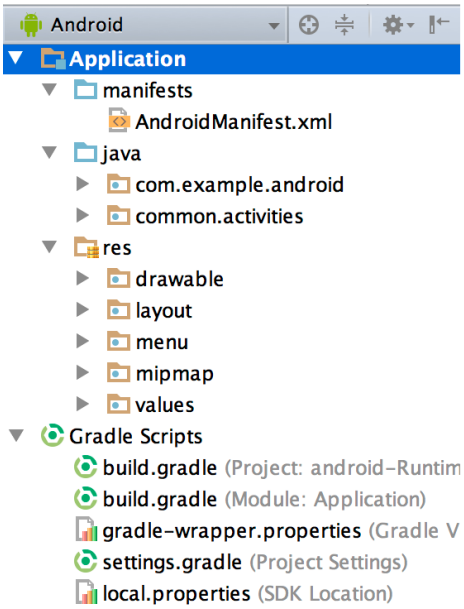


Figura 56: Estructura de Android Studio

Cada módulo de la aplicación contiene las siguientes carpetas:

- **manifests**: Contiene el archivo AndroidManifest.xml.

- **java**: Contiene los archivos de código fuente de Java, incluido el código de prueba JUnit.

- **res**: Contiene todos los recursos, como diseños XML, cadenas de IU e imágenes de mapa de bits.

3.2.2.2. Interfaz de usuario

La ventana principal de Android Studio está compuesta por varias áreas, que se muestran en la Figura 57:

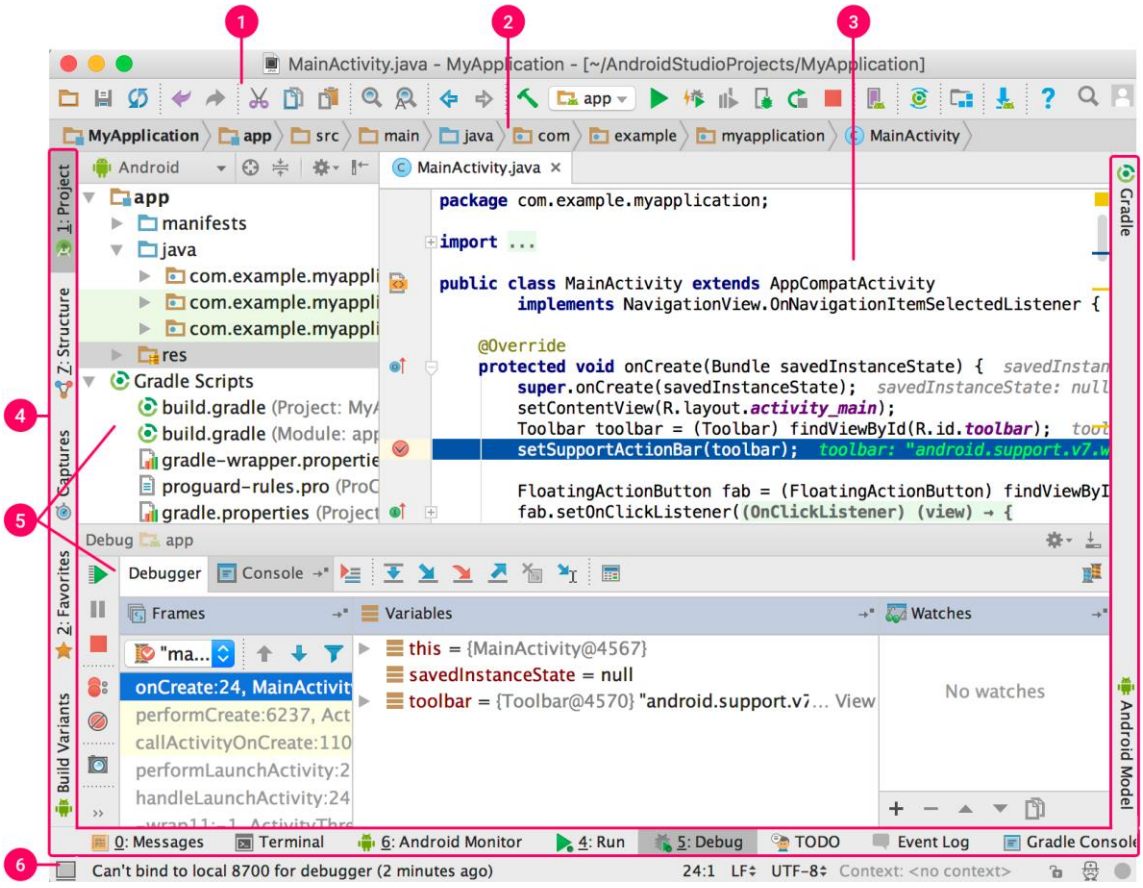


Figura 57: Interfaz de usuario de Android Studio

Fuente: https://developer.android.com/studio/images/intro/main-window_2-2_2x.png

A continuación detallamos las áreas de la interfaz de usuario:

1. **Barra de herramientas**: Permite realizar una gran variedad de acciones, como la ejecución de la app y el inicio de herramientas de Android.

2. **Barra de navegación**: Ayuda a explorar el proyecto y abrir archivos para editar. Proporciona una vista más compacta de la estructura visible en la ventana del proyecto.

3. **Ventana del editor**: Es el área donde se puede crear y modificar código. Según el tipo de archivo actual, el editor puede cambiar. Por ejemplo, cuando se visualiza un archivo de diseño, el editor muestra el editor de diseño.

4. **Barra de la ventana de herramientas**: Se extiende alrededor de la parte externa de la ventana del IDE y contiene los botones que permiten expandir o contraer ventanas de herramientas individuales.

5. **Ventanas de herramientas**: Permiten acceder a tareas específicas, como la administración de proyectos, las búsquedas, los controles de versión, etc. Se pueden expandir y contraer.

6. **Barra de estado**: Muestra el estado del proyecto y del IDE en sí, como también cualquier advertencia o mensaje.

3.1.3. Unity 3D vs Android Studio

En la Tabla 5 mostramos diferentes características que consideramos importantes a la hora de seleccionar la herramienta con la cual desarrollamos nuestra aplicación.

	Andrioid Studio	Unity 3D
Uso de la interfaz	La interfaz para el usuario se basa, en su mayoría, en la edición de archivos de textos.	El sistema de ventanas, el árbol de jerarquías y el menú de propiedades de cada elemento permiten al usuario crear aplicaciones sólo agregando y configurando elementos.
Curva de aprendizaje	Para iniciar una nueva aplicación hay que hay que aprender en detalle a utilizar el lenguaje.	El uso de elementos gráficos y el conjunto de elementos y scripts que trae, permiten hacer aplicaciones simples, incluso con pocos conocimientos de programación.
Precio	Gratuito.	Gratuito en su versión básica y pago en la versión Pro.
Generación del instalador de la aplicación	Es el entorno oficial para desarrollo de aplicaciones. Permite generar archivos de instalación para dispositivos Android, pero no para iOS, por ejemplo.	Es un entorno multiplataforma. Permite generar aplicaciones para una amplia gama de plataformas, como Android, iOS, Windows, Plays Station, etc.

Tabla 5: Comparación de entornos de desarrollo

En base a lo expuesto anteriormente, optamos por utilizar Unity 3D. Los puntos más importantes que consideramos para esta elección fueron el uso de la interfaz y la generación del instalador de la aplicación. Si bien la generación del instalador en ambos casos cumple con lo mínimo requerido, generar un archivo .apk , Unity 3D nos permite que para desarrollos futuros la aplicación pueda implementarse bajo otras plataformas. Además, el sistema de ventanas que posee Unity 3D favorece la manipulación de los objetos 3D que importamos al proyecto.

También, pensando más en detalle en la problemática de esta Tesina, vemos que esta herramienta nos brinda la posibilidad de hacer de nexo entre otro conjunto de herramientas como Vuforia, Blender, Autodesk 3ds Max, entre otras.

3.2. Herramientas de Realidad Aumentada

Las herramientas de RA son aquellas que nos brindan el soporte de funciones y APIs para el reconocimiento de marcadores y/o entorno que se encuentran en el mundo real. De esta forma las aplicaciones tiene una referencia posicional para la ubicación de los objetos (modelos 3D en nuestro caso).

3.2.1. ARToolkit

Figura 58: Logo ARToolkit

ARToolkit es un framework para el desarrollo de aplicaciones de RA de código abierto (open source). Su código fuente está disponible en GitHub, y está compilado para diferentes plataformas (Mac OS X, PC, Linux, Android, iOS); también presenta un plugin para Unity 3D. [16]

El kit de herramientas que trae utiliza técnicas de visión artificial[19] para calcular la posición y la orientación real de la cámara en relación con las formas cuadradas o las superficies planas, permitiendo al programador superponer objetos virtuales (en nuestro caso objetos 3D).

[19] Visión artificial o visión por computadora: es una disciplina científica que incluye métodos para adquirir, procesar, analizar y comprender las imágenes del mundo real con el fin de producir información numérica o simbólica para que puedan ser tratados por una computadora.

ARToolKit actualmente soporta marcadores cuadrados clásicos, códigos de barras 2D, multimarker y seguimiento de características naturales, y cualquier combinación de los anteriores.

3.2.2. Vuforia

Figura 59: Logo Vuforia

Vuforia es un SDK orientado al desarrollo de aplicaciones de RA para dispositivos móviles, que se ejecuta en tiempo real mediante el video capturado a través de las imágenes captadas por la cámara brindada por el hardware del dispositivo. Utiliza la tecnología de visión artificial para reconocer y crear individualmente la ubicación del objeto capturado por la cámara de video en tiempo real. Por otro lado, no todos los objetos pueden ser detectados, dependiendo principalmente de la capacidad de procesamiento del dispositivo móvil. [36]

Las capacidades de Vuforia para el registro de imágenes permiten a los desarrolladores posicionar y orientar en el espacio los objetos virtuales, principalmente objetos 3D u otro tipo de medios, en relación con las imágenes del mundo real cuando éstas se visualizan a través de la cámara de un smartphone. El objeto virtual puede seguir la posición y orientación de la imagen real en tiempo real, de modo que la perspectiva del espectador sobre el objeto se corresponda con su perspectiva sobre el objetivo del mundo real. De esta manera, el objeto u objetos virtuales aparecen como si se tratara de otro objeto del mundo real.

Vuforia ofrece una API 6 en lenguajes Java, C ++ y .Net, a través de una extensión del motor de juego Unity 3D. De esta manera, el SDK soporta el desarrollo nativo para iOS y Android, permitiendo al mismo tiempo el desarrollo de aplicaciones de RA en Unity 3D que son fácilmente portables a ambas plataformas. Las aplicaciones RA desarrolladas con Vuforia, por lo tanto, son compatibles con una amplia gama de dispositivos móviles, incluyendo iPhones, iPad y teléfonos y tabletas Android.

3.2.3. ARToolkit vs. Vuforia

A la hora de evaluar las herramientas para decidir cuál utilizar en el desarrollo de la presente Tesina tuvimos en cuenta las características que expresa la Tabla 6:

	ARToolkit	Vuforia
Precio	Está disponible en forma gratuita.	La versión gratuita tiene limitaciones en algunas funciones, como por ejemplo, la cantidad de imágenes que puede almacenar en la base de datos. También posee versiones paga que mejoran ciertos aspectos de uso.
Compatibilidad	Sirve para Unity y Android Studio.	Sirve para Unity y Android Studio.
Estabilidad de la imagen de RA	Presenta más correciones de posición durante el renderizado.	Tiene mayor estabilidad al presentar los modelos 3D durante la visualización de la escena.

Tabla 6: Comparación de herramientas de Realidad Aumentada

Como conclusión de esta comparativa elegimos utilizar Vuforia, ya que preferimos priorizar la estabilidad de la imagen. Si bien la versión paga es bastante costosa, para los fines prácticos de nuestra aplicación nos alcanza con la licencia para desarrolladores que es gratuita.

3.3. Herramientas de modelado 3D

Para el desarrollo de los modelos 3D necesarios para nuestra aplicación existen varios programas disponibles, pero nuestra idea era seleccionar el óptimo o el que nos brindara mayor comodidad para utilizarlo en nuestra aplicación.

Para ello, analizamos qué software es el más apropiado para realizar nuestro modelado. Uno de los principales problemas que encontramos, estuvo dado por la dificultad a la hora de exportar los modelos al motor gráfico (en nuestro caso Unity 3D), ya que es muy importante contar con una buena relación entre ambos. Existen otras características muy importantes que también tuvimos en cuenta, como, por ejemplo, la facilidad de uso, las interfaces, la compatibilidad, y el precio.

Optamos por probar dos de los programas más conocidos, uno de ellos propietario (Autodesk 3ds Max) y el otro no (Blender), para comparar y de esta forma seleccionar el más cómodo y más efectivo para el desarrollo de los modelos necesarios para nuestra aplicación.

3.3.1. Blender

Figura 60: Logo Blender

Es una aplicación de modelado 3D que, además, incorpora su propio motor de juego mediante el programa Blender Game Engine. Ofrece multitud de herramientas de modelado, con una amplia variedad de objetos y posibilidades, permitiendo el desarrollo de modelos o escenas 3D casi sin limitaciones de creación. [37]

Blender nació en el año 1993, integrando una serie de herramientas para la creación de contenidos en 2D y en 3D. En 2002, la compañía quebró y los acreedores decidieron ofrecerlo como un producto de código abierto y gratuito disponible bajo la licencia GNU GPL. En ese momento, los desarrolladores crearon la Blender Foundation, una asociación sin fines de lucro para recoger donaciones con el objetivo de ofrecer un mantenimiento del producto. Esta decisión dio sus frutos y hoy en día es una de las aplicaciones open source más populares y utilizadas del mundo, por lo que su adquisición es gratuita, así como también su documentación.

Se trata de un conjunto de herramientas que permiten la creación y reproducción lineal y en tiempo real de contenidos 3D. Sus principales características son:

- **Interfaz gráfica de usuario (GUI)**: Posee un entorno de trabajo que utiliza un sistema de ventanas sin solapamiento, que pueden ser configuradas de forma independiente, según lo desee el usuario. Además, posee su propio editor de textos, que se utiliza principalmente para la creación de scripts en Python (que son utilizados en su motor de juegos).

- **Modelado**: El modelado que se puede realizar es de nivel profesional gracias a todas las herramientas que proporciona, compuestas de primitivas geométricas (tales como mallas poligonales, objetos empty, nurbs y metaballs). Además, cuenta con el sistema Ripping que permite crear esqueletos para los modelos de forma sencilla, de tal manera que con sólo agregar una malla, pueden controlarse ambos en conjunto, permitiendo que se le aplique movimiento y que parezcan fluidos y más reales.

- **Animación**: Contiene herramientas para asociar movimientos a los objetos. Mediante IK Chaining se pueden asociar los movimientos de desplazamiento, cambio de tamaño o rotación de un objeto entre dos frames mediante un indicador que los agrupa, facilitando de esta forma el manejo de dichos movimientos.

- **Mapeado de texturas UV**: Permite texturizar los objetos con la imagen que se desee, pudiendo ajustarla a gusto del usuario. Además, proporciona distintas operaciones como dividir o replicar la textura.

- **Renderizado**: Se basa en un sistema de renderizado interno. Realiza el renderizado por secciones, incluye capas y una caché de renderizado. También, puede proporcionar soporte para efectos, como destellos de luz, oclusión ambiental o desenfoque.

- **Shading**: Dispone de una herramienta para agregarle un material y su sombra correspondiente a un objeto, de manera muy sencilla. Distingue dos tipos de sombreado: difuso y especular. Además, tiene un editor de nodos para crear efectos visuales en el material asociado al objeto. Esto es muy útil en el caso de querer representar materiales como la madera o el metal, o incluso la representación de fluidos.

- **Física y partículas**: Se utiliza para simular cabello, humo, entre otros, de una manera muy realista, pero con un alto costo computacional. Para ello se asigna un determinado número de partículas a un objeto, pudiendo controlar su movimiento, su texturizado y su color. Mediante la física, se puede cambiar la percepción del objeto de tal manera que se pueda simular un cuerpo rígido, como una bola de metal, o un cuerpo suave, como una pelota de goma o un fluido. De esta manera se cambian las propiedades físicas de los objetos para que visualmente se perciban como en la realidad. Incluye también detección de colisiones.

- **Gestión de archivos**: Guarda el modelo creado en un archivo .blend, incluyendo la física, las texturas, los materiales y la iluminación. Estos archivos proporcionan compatibilidad hacia delante y hacia atrás, admiten compresión y encriptación, y pueden servir como librerías en otros archivos del mismo tipo. Además, los proyectos en Blender pueden ser exportados a multitud de formatos, entre los que está Autodesk 3ds Max (que es la otra herramienta que investigamos) y DirectX.

- **Soporte multiplataforma**: Está disponible en muchas plataformas, con compatibilidad OpenGL, y mantiene su estructura de ventanas en todas ellas:
    - Windows 2000, XP, Vista.
    - Mac OS X (PPC and Intel).
    - Linux (i386).
    - Linux (PPC).
    - FreeBSD 5.4 (i386).
    - SGI Irix 6.5.
    - Sun Solaris 2.8 (sparc).

3.3.2. Autodesk 3ds Max

Figura 61: Autodesk 3ds Max

Autodesk 3ds Max es una herramienta de creación de animaciones 3D y gráficos. En sus primeras versiones se llamaba 3d Studio. Luego, Kinetix compró los derechos del programa y lanzó tres versiones (desde 1.0 hasta 3.0) bajo el nombre de 3d Studio MAX. Tiempo más tarde, la empresa Discreet compró los derechos y lanzó las versiones 4.0 a 7.0, bajo el nombre 3ds Max. Finalmente, Autodesk retomó el programa (desarrollando desde la versión 8.0) bajo el nombre de Autodesk 3ds Max. [38]

Es una de las herramientas de animación 3D más utilizadas en la actualidad. Tiene una gran variedad de plugins y una capacidad de edición muy sólida. Es muy utilizada por desarrolladores de videojuegos, aunque también se suele usar para anuncios de televisión o animación de películas, entre otras actividades. Sus características principales son:

- **MaxScript**: Es un lenguaje de scripting integrado que ayuda a automatizar tareas repetitivas. El lenguaje utiliza nuevas formas de combinar la funcionalidad existente, desarrollar nuevas interfaces de usuario, herramientas, etc.

- **Character Studio**: Es un plugin que está integrado para ayudar a los usuarios en la animación de personajes virtuales. La herramienta ofrece una edición robusta para la manipulación de objetos, capas y flujos de trabajo. Los objetos que se generan tienen la característica de producir ciclos de caminata, movimiento y movimiento secundario.

- **Explorador de escenas**: Es una herramienta que proporciona una visión jerárquica de la escena de datos, facilitando el trabajo con escenas más complejas. Tiene la capacidad de ordenar, filtrar y buscar escenas por tipo de objeto o propiedad.

- **Asignación / edición de texturas**: Ofrece una variedad de operaciones para trabajar con objetos 3D. La funcionalidad básica es el mapeo de textura.

- **Skinning**: Es el proceso de modificar el cuerpo o la piel para lograr un control preciso de la deformación esquelética, por lo que los movimientos en la animación son suaves.

- **Esqueletos y cinemática inversa (IK)**: los personajes pueden manipularse con esqueletos personalizados usando huesos 3ds Max, solucionadores IK y herramientas de aparejo.

- **Solucionador de paños integrado**: Este es un motor de simulación de tela. Permite al usuario convertir cualquier objeto 3D en ropa y crear prendas desde cero. Esta función se utiliza para mejorar el rendimiento de reproducción.

- **Autodesk Vault**: Se usa para compartir, buscar y reutilizar los activos de 3ds Max. Es algo similar a una "tienda" para Autodesk.

3.3.3. Blender vs Autodesk 3ds Max

En nuestro caso experimentamos con ambas herramientas obteniendo resultados similares, lo que nos permitió concluir que las dos son útiles para nuestro desarrollo en igual medida, diferenciándose sólo en la comodidad de uso. Esto hizo que la selección de la herramienta fuera subjetiva y, por ello, decidimos utilizar las dos.

En la Sección 4.2. mostramos el resultado de modelar una taza con ambas herramientas.

4. Desarrollo de la aplicación

Recordemos que el objetivo que nos planteamos en esta Tesina es el de proveer una aplicación que pueda ser utilizada por los Sistemas de Compra Web (o por catálogos impresos) y sus clientes para visualizar, de forma sencilla e interactiva, los productos ofrecidos de forma virtual en una escala y ubicación acorde al ambiente.

Para ello, desarrollamos una aplicación para dispositivos móviles, bajo el sistema operativo Android, que permite mostrar un producto en el mundo real, representado mediante un modelo 3D a escala real, utilizando tecnología de RA y la cámara de un dispositivo móvil.

También elaboramos los modelos 3D de algunos productos y sus impresiones de códigos, para que los vendedores puedan distribuirlos a los posibles clientes, y creamos una web sencilla para simular a un vendedor, en la etapa de prueba

Para que nuestra aplicación esté disponible para potenciales usuarios, proporcionamos una plataforma de fácil acceso para su descarga.

En el presente capítulo presentamos los diferentes aspectos que analizamos para diseñar, realizar y distribuir la aplicación, las herramientas utilizadas, así como también los inconvenientes que surgieron durante el proceso.

4.1. Diseño de la aplicación

Para el diseño de la aplicación primero identificamos cuál es su función principal y cuáles son los usuarios que la van a utilizar.

Como función principal definimos que la aplicación debe contar con un visualizador en un dispositivo móvil. Este visualizador le mostrará al usuario un producto de tipo mobiliario (modelo 3D) a elección, respetando la escala real, mediante el uso de RA.

Para darle más claridad al uso de la aplicación, y teniendo en cuenta los objetivos planteados, definimos dos tipos de usuarios:

- **Vendedor**: Interviene de forma indirecta, ya que solamente se encarga de poner a disposición el catálogo, página web o algún otro medio, donde el cliente potencial podrá elegir el producto.

- **Cliente potencial**: Es quien utiliza directamente la aplicación.

A continuación detallamos los elementos que componen la aplicación móvil desarrollada:

- **Visualizador**: Es el componente principal, que tiene la tarea de reconocer el entorno, detectar patrones[20] y colocar el producto en pantalla dentro de la escena, simulando su presencia dentro en el mundo real.

- **Base de datos de imágenes**: Se compone de un grupo de imágenes específicas para ser detectadas por el visualizador. Lo más importante para cada imagen es definir un tamaño fijo, ya que el tamaño es la referencia para generar la vista del producto en un tamaño predecible. Esto quiere decir que si no se imprime la imagen en el tamaño establecido, se perderá la referencia real de la escala.

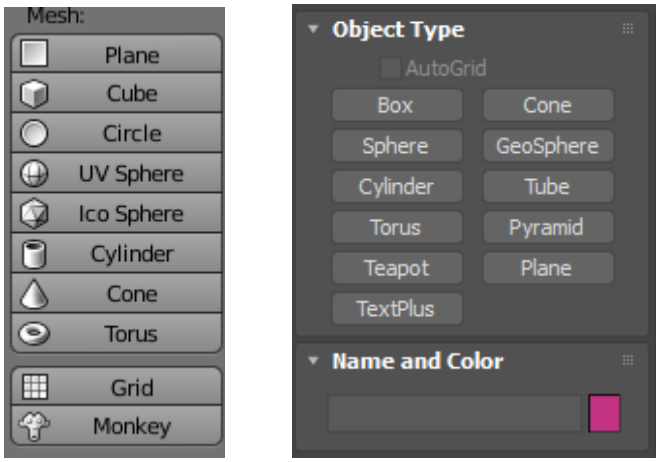
- **Modelos 3D**: Constituyen la representación del producto. Cada modelo está vinculado uno a uno con una imagen de la base de datos.

4.2. Creación de modelos 3D

En esta sección explicamos los pasos que seguimos para generar los modelos 3D de algunos productos que se pueden visualizar con nuestra aplicación.

Como mencionamos anteriormente (en la Sección 3.3.3.), generamos dos modelos 3D utilizando Blender y Autodesk 3ds Max. Con ambas herramientas optamos por crear una taza, para presentar las nociones básicas del modelado 3D y demostrar las similitudes entre ambas herramientas.

En primera instancia seleccionamos una figura geométrica base proporcionada por la herramienta, que se aproxime a la forma de la taza. En ambas herramientas tenemos un Menú con diferentes figuras geométricas básicas (Figura 62):



[20] Patrón: áreas identificables dentro de la imagen.

Figura 62: Objetos básicos en 3D (Izquierda: Blender - Derecha: Autodesk 3ds Max)

De los objetos disponibles que ofrecen, el más “parecido” o más cercano para comenzar a modelar es el cilindro y éste es el que usamos como base, como se puede apreciar en las Figuras 63 y 64:

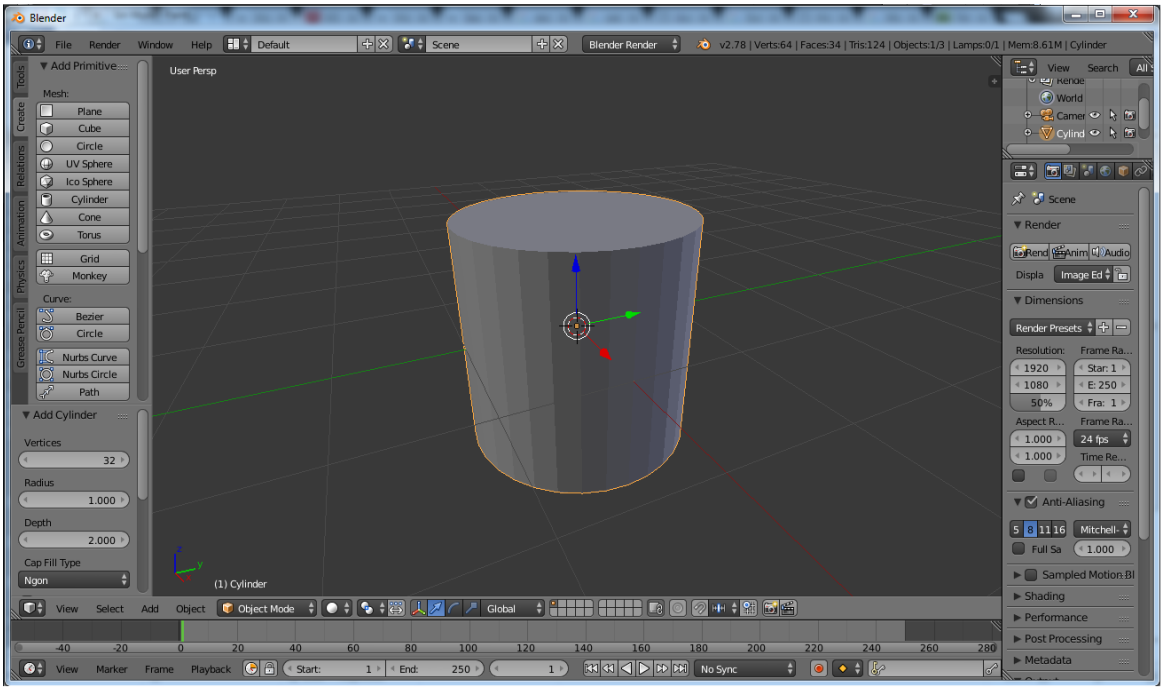


Figura 63: Cilindro Blender

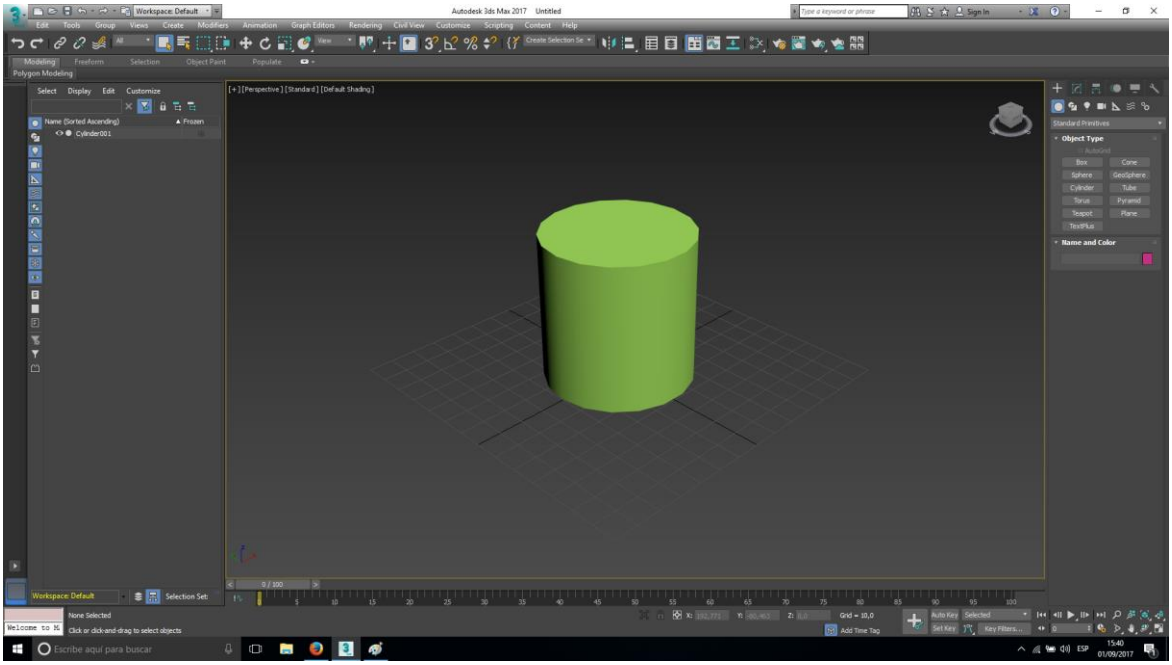


Figura 64: Cilindro Autodesk 3ds Max

El próximo paso es empezar a darle forma. Para ello, en Blender entramos en el modo de edición y cambiamos la vista a Wireframe, como se ve en la Figura 65. En el caso de Autodesk 3ds Max, transformamos el objeto a *Editable Poly*, como se muestra en la Figura 66. Ambos casos de edición se encuadran dentro del modelado poligonal (explicado en la sección 2.3.1.).

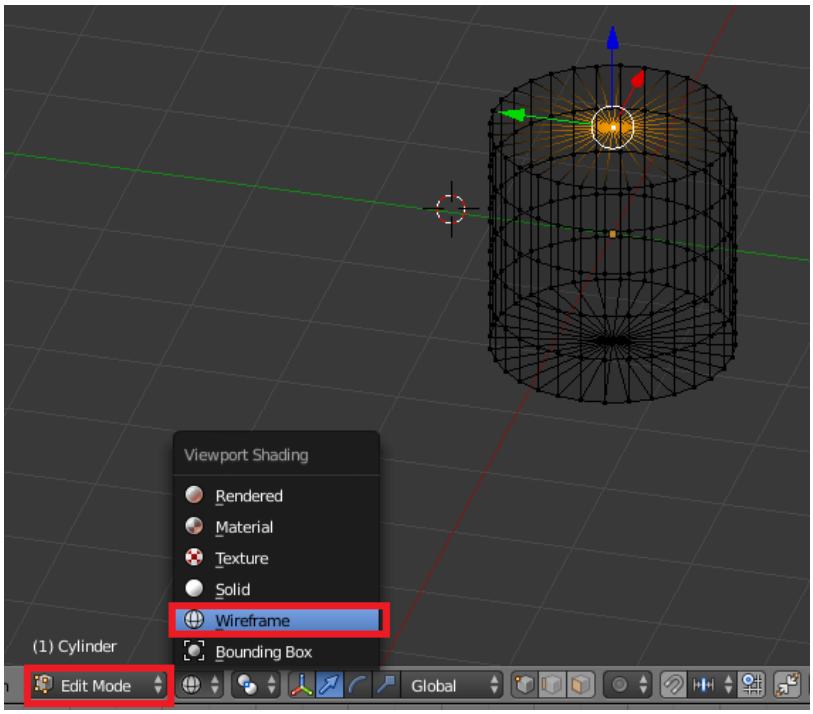


Figura 65: Modo edición - Vista wireframe Blender

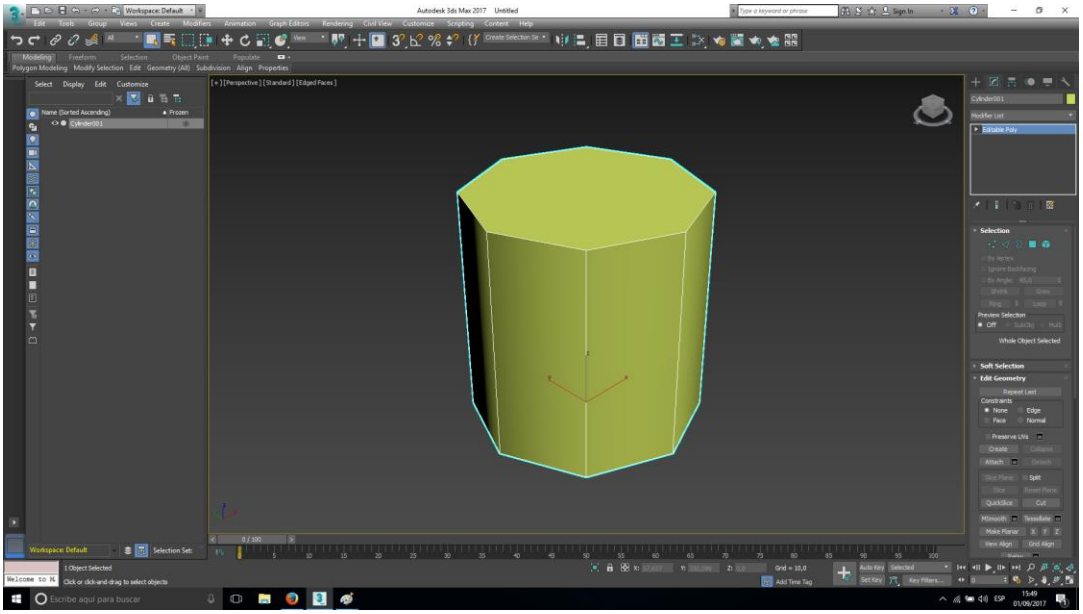


Figura 66: Editable Poly - Autodesk 3ds Max

Luego, le quitamos la parte superior al cilindro para generar un recipiente hueco, como se muestra en las Figuras 67 y 68:

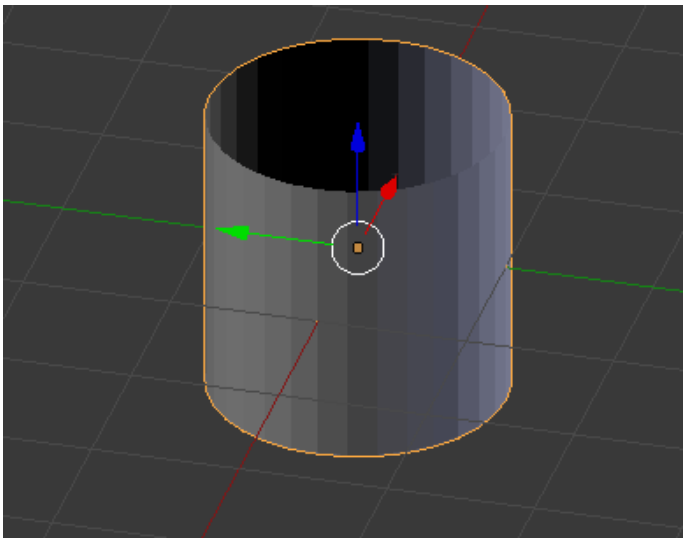


Figura 67: Cilindro sin “tapa” - Blender

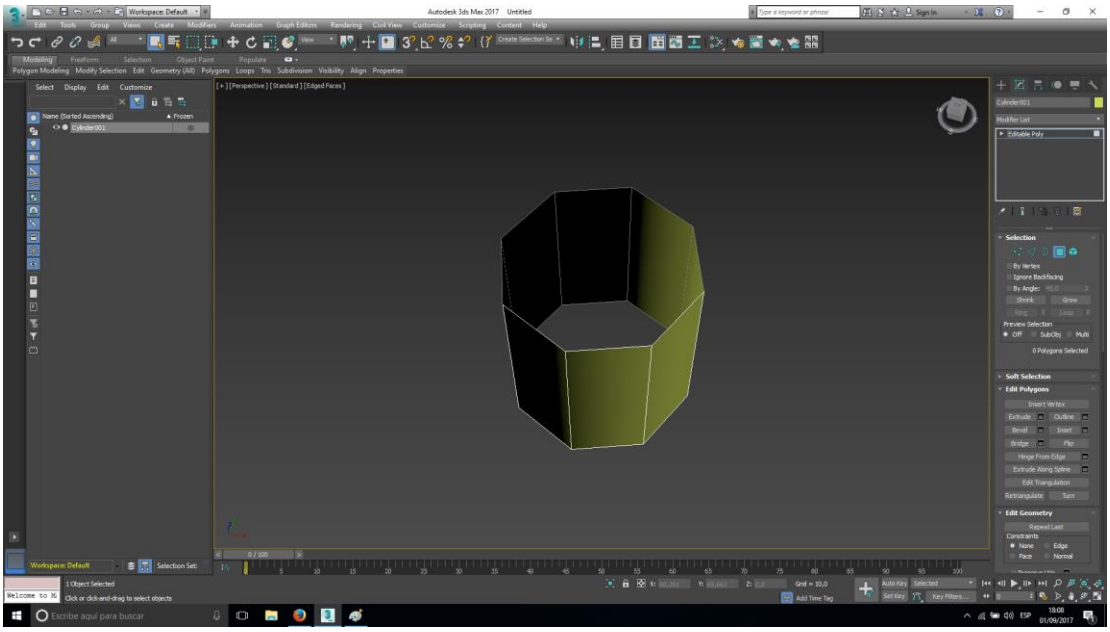


Figura 68: Cilindro sin “tapa” - Autodesk 3ds Max

Para darle más realismo a la taza generamos curvaturas en los laterales utilizando métodos de escalados en las diferentes aristas, como se aprecia en las Figuras 69 y 70:

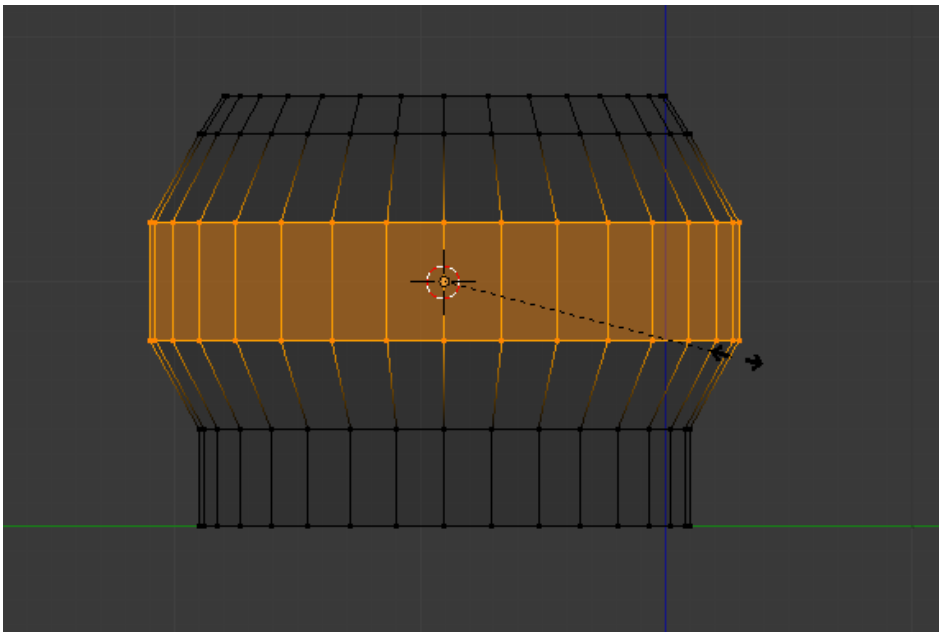


Figura 69: Escalado en Aristas - Blender

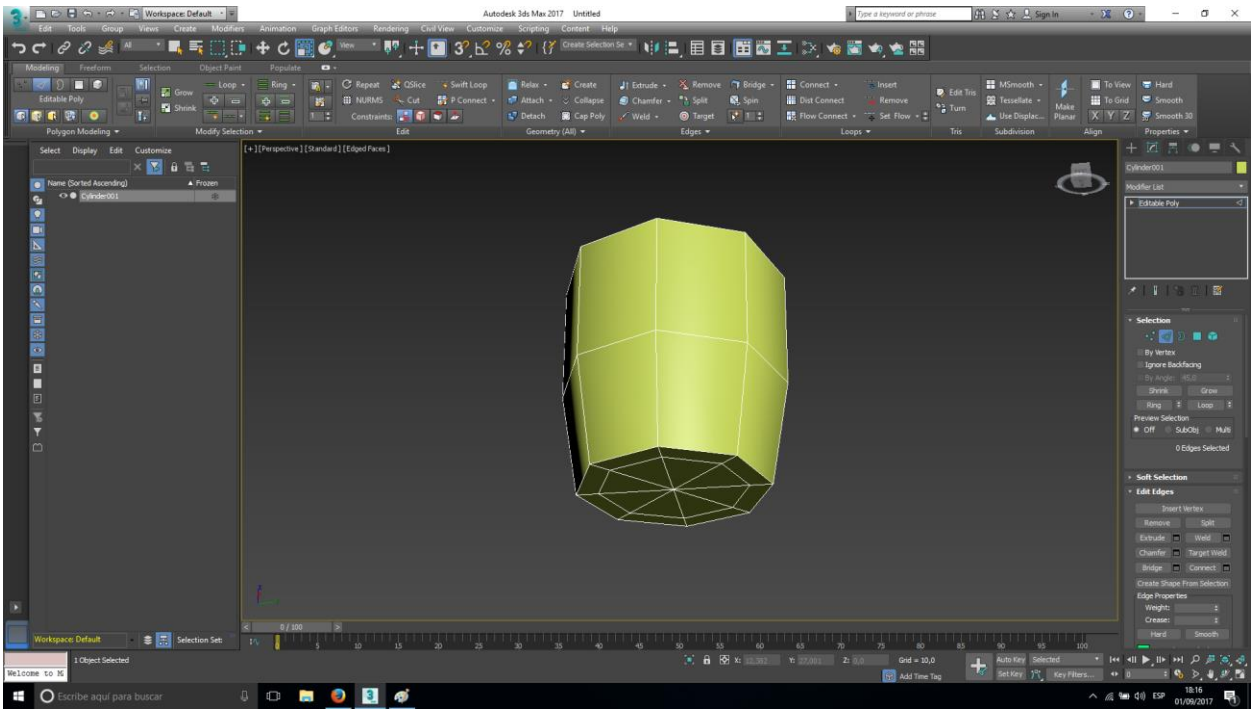


Figura 70: Escalado en Aristas - Autodesk 3ds Max

Después de aplicar diferentes modificaciones en los modelos, obtuvimos como resultado los nuevos modelos mostrados en las Figuras 71 y 72:

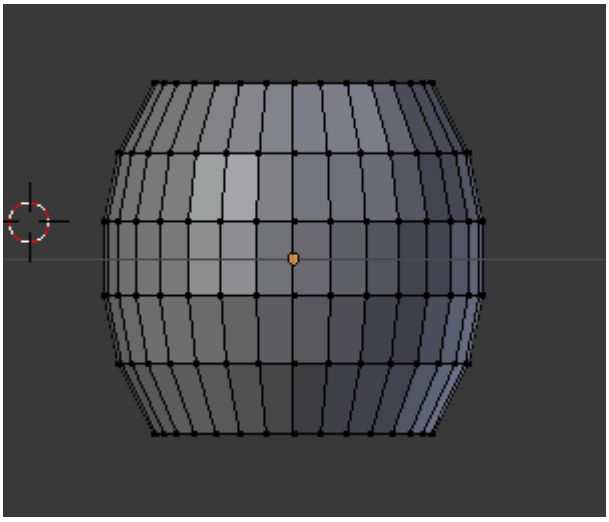


Figura 71: Taza con bordes redondeados - Blender

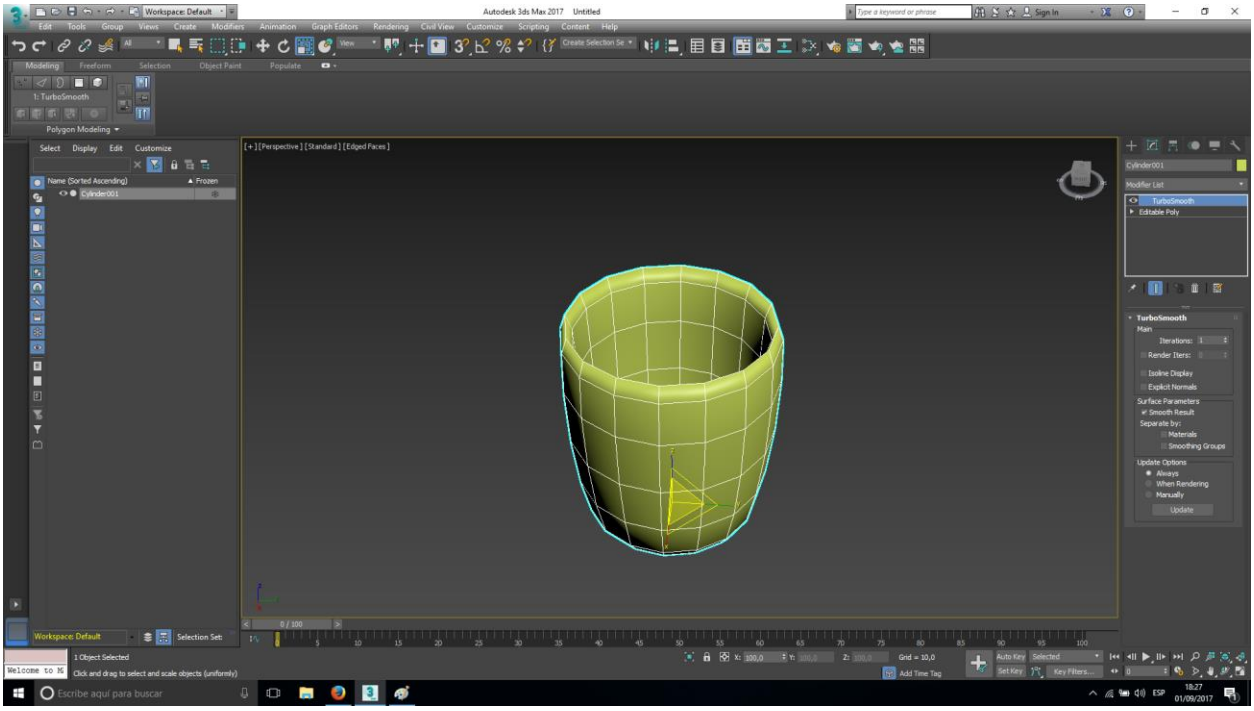


Figura 72: Taza con bordes redondeados - Autodesk 3ds Max

Para el paso final, agregamos el mango de la taza. En el caso de Blender creamos el mango por separado (de la misma forma que creamos la taza) y, luego, lo unimos agregando polígonos. En Autodesk 3ds Max trabajamos los polígonos específicamente, realizando

diferentes modificaciones para establecer la figura del mango. De esta manera logramos la taza completa, como se puede observar en las Figuras 73 y 74:

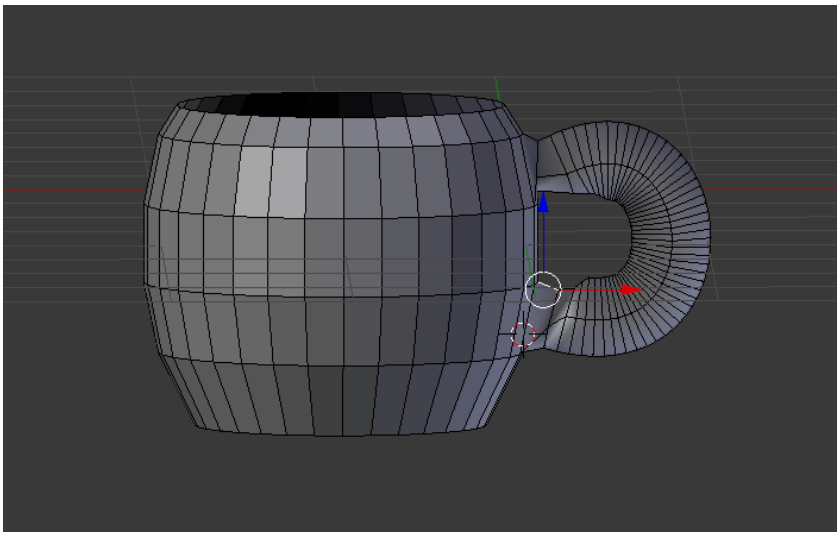


Figura 73: Taza finalizada - Blender

.

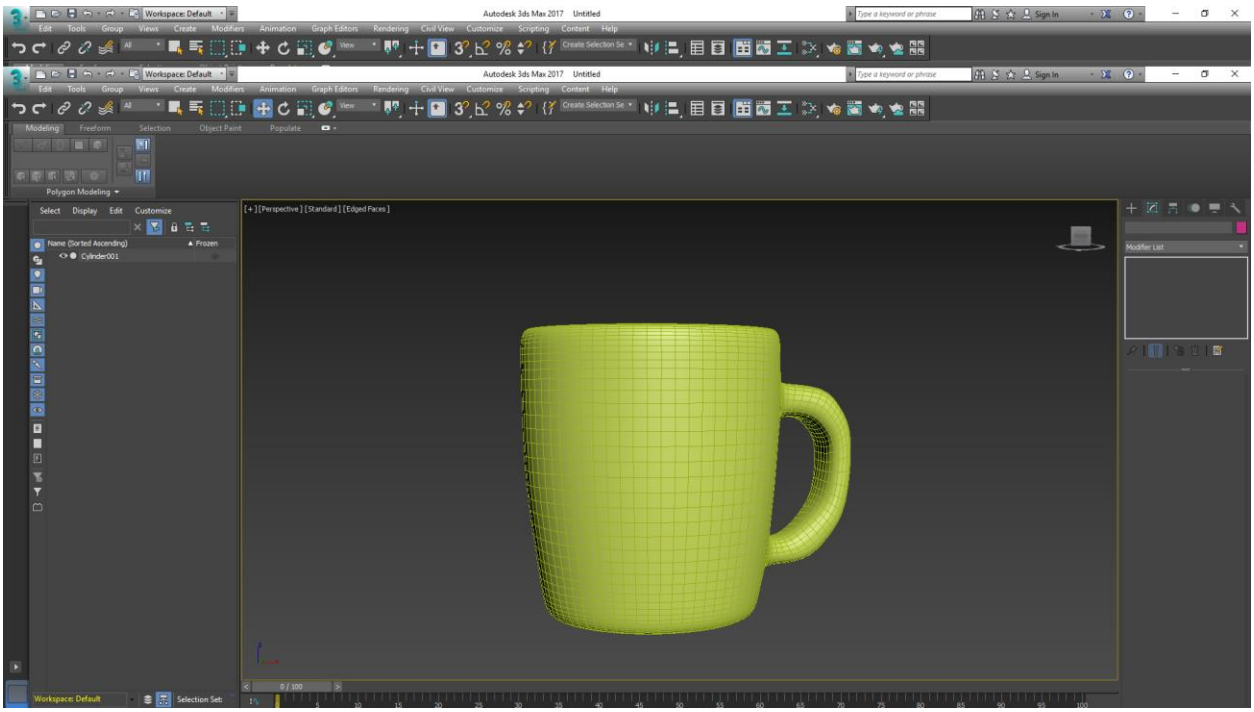


Figura 74: Taza finalizada - Autodesk 3ds Max

De la misma forma que creamos la taza, también generamos, con Autodesk 3ds Max, los modelos que se muestran en la Figura 75:

Figura 75: Modelos 3D creados con Autodesk 3ds Max

El resto de los modelos que agregamos a la aplicación los obtuvimos desde la página https://archive3d.net/, que cuenta con una amplia variedad de modelos 3D gratuitos.

4.3. Creación de marcadores y de la base de datos de imágenes

Habiendo investigado los métodos de seguimiento basados en visión (como explicamos en la Sección 2.2.6.2), decidimos emplear en nuestra aplicación el seguimiento basado en marcas, ya que esta técnica nos facilita la tarea al trabajar con una mejor referencia en las escalas de los modelos 3D.

En primer lugar, analizamos el tipo de marcador a utilizar. Vuforia provee objetos **trackeables** (objetos que permiten el seguimiento) que constituyen la clase principal que representa los objetos del mundo real, mediante seis grados de libertad. Cada objeto, cuando es detectado y se le realiza el seguimiento, tiene un conjunto de parámetros: Tipo, Nombre, ID, Estado e Información de la posición. Vuforia representa tres tipos de objetos **trackeables** que heredan sus propiedades de la clase base:

- **Multi target**: Consisten en múltiples imágenes, con una relación especial, que conforman el objeto de seguimiento.

- **Frame markers**: Son un tipo de marcadores especiales que poseen una identificación que se codifica en un patrón binario a lo largo del borde de la imagen del marcador.

- **Image target**: Son imágenes que el SDK de Vuforia puede detectar y realizar el seguimiento. No necesitan regiones de código, ni regiones en blanco y negro para ser reconocidas (como sucede con los marcadores clásicos).

Todos ellos se generan en línea en base a una imagen JPG o PNG y se almacenan en una base de datos, que luego se descarga e incorpora a la aplicación para realizar las comparaciones con las imágenes físicas utilizadas como marcadores, en tiempo de ejecución. Para generar las bases de datos de imágenes, Vuforia proporciona la herramienta web **Target Manage**r[21].

En un comienzo nos planteamos la idea de utilizar **frame markers**, por el simple hecho de contar con información codificada. Pero, luego de un análisis más profundo, optamos por elegir los **image target**, ya que si bien tienen restricciones a la hora de ser detectados, nos proporcionaron mejores resultados al trabajar cuando el marcador se encuentra a mayor distancia de la cámara que está analizando la escena.

Para poder realizar los marcadores utilizamos las herramientas de edición gráfica **Adobe Photoshop**[22] y **Adobe Illustrator**[23]. En un comienzo decidimos realizar un patrón repetitivo del ícono del objeto que se iba a representar mediante el marcador (Figura 76).

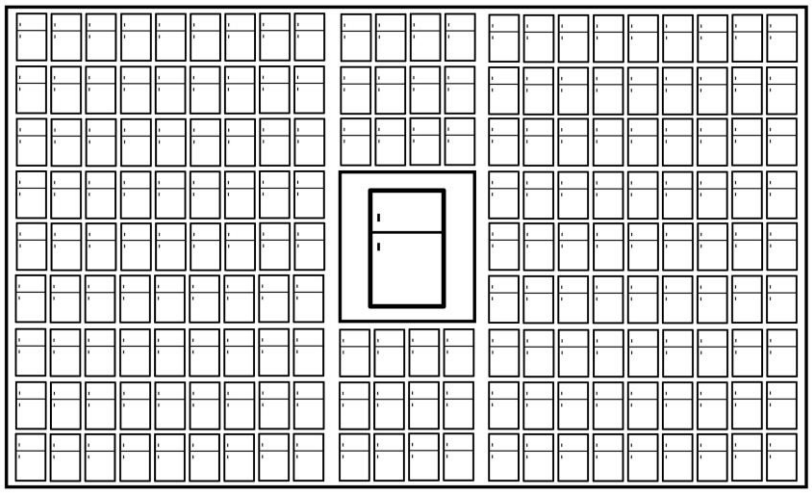


Figura 76: Marcador viejo

Al generar un **image target**, Target Manager analiza la imagen desde el primer píxel superior izquierdo de la imagen continuando fila por fila hasta el último, lo que permite comprender las formas que componen la propia imagen. Este análisis le permite reconocer las características del **image target**, determinando un número de estrellas; a mayor número de estrellas, mejor seguimiento y reconocimiento. Cuando cargamos la imagen que habíamos conformado en Target Manager con un patrón repetitivo, ésta no obtuvo ninguna estrella.

[21] https://developer.vuforia.com/target-manager

[22] Photoshop es un programa informático de edición de imágenes. Está desarrollado por la empresa Adobe Systems Incorporated.

[23] Illustrator es un editor de gráficos vectoriales de la misma empresa que el Photoshop, Adobe Systems Incorporated.

Investigando cómo obtener una mejor calificación, pudimos darnos cuenta que, según el portal de desarrollo de Vuforia, las imágenes deben cumplir ciertas características como tener una distribución de alta densidad y uniformidad con alto contraste local, sin patrones repetitivos. Por ello, decidimos conformar una nueva imagen según las pautas presentadas en la librería de Vuforia[24].

En base a este resultado, optamos por patrones que no sean repetitivos, con un alto nivel de detalle y un buen contraste. De esta forma usamos diferentes texturas alrededor de un ícono representativo, que cumplen la función de diferenciar el objeto a representar, de forma que el nuevo marcador quedó compuesto como se muestra en la Figura 77:

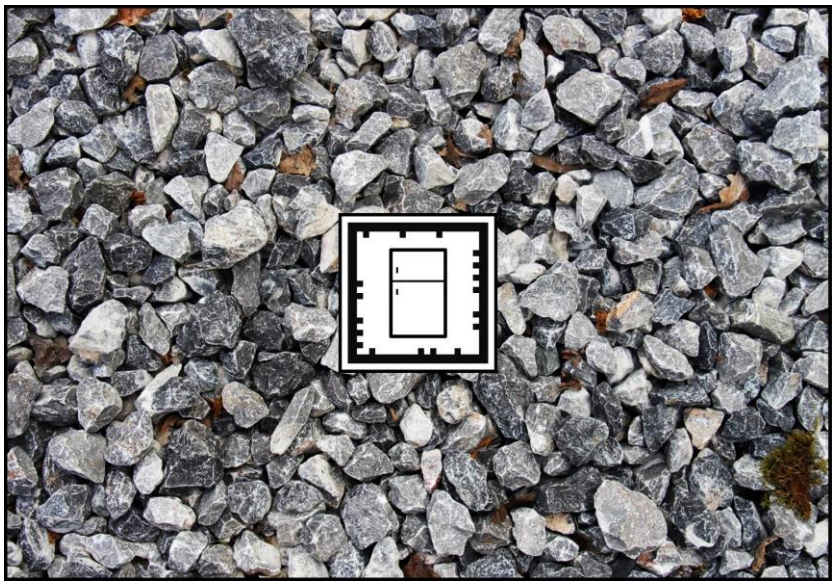


Figura 77: Marcador Nuevo

Con estas nuevas imágenes, obtuvimos una calificación de cinco estrellas, como se puede apreciar resaltado con el cuadro naranja en la Figura 78. Esta nueva calificación mejora la eficiencia a la hora de detectar el marcador y, de esta forma, nos permite trabajar a mayores distancias con la visualización de la cámara.

[24] https://library.vuforia.com/articles/Solution/Optimizing-Target-Detection-and-Tracking-Stability

Target Name	Type	Rating	Status	Date Modified
Marcadorc4-Lamparalava	Single Image	★★★★★	Active	Aug 24, 2017 13:16
Marcadorc3-Florero	Single Image	★★★★★	Active	Aug 24, 2017 13:15
Marcadorc2-Reloj	Single Image	★★★★★	Active	Aug 24, 2017 13:15
Marcadorc1-Taza	Single Image	★★★★★	Active	Aug 24, 2017 13:13
Marcadorm4-Cafetera	Single Image	★★★★★	Active	Aug 24, 2017 13:08

Figura 78: Calidad del marcador

La herramienta Target Manager nos permite observar las características naturales que Vuforia encontró en la imagen (Figura 79) y que va a utilizar la aplicación para detectar la imagen objetivo.

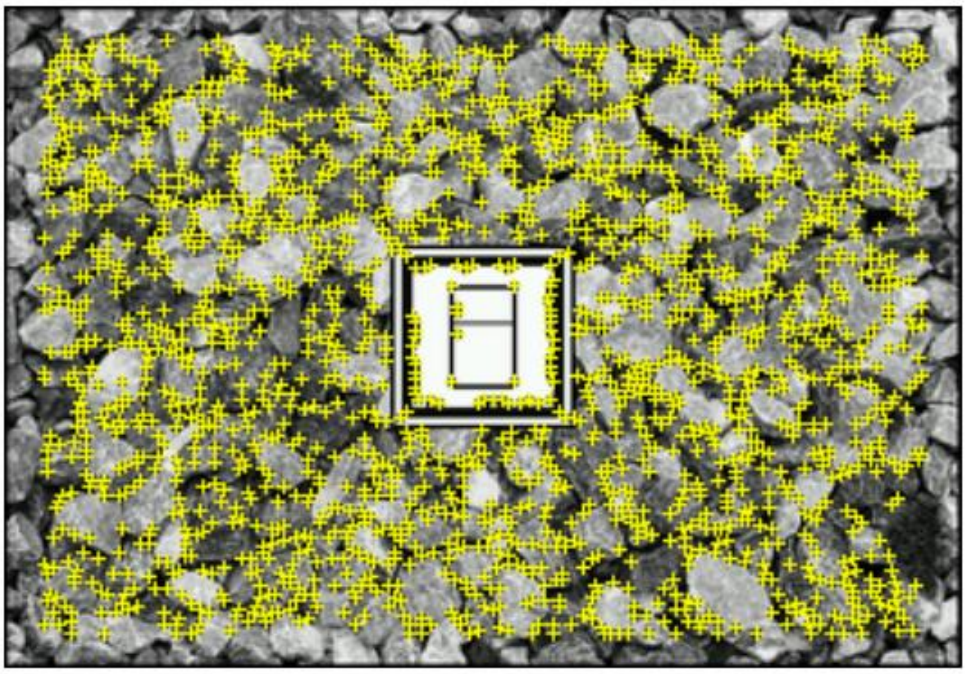


Figura 79: Reconocimiento de las características

En la Figura 80 se muestran otros marcadores que desarrollamos:

Figura 80: Otros marcadores

Cabe destacar que, además de la creación de los marcadores, añadimos tres tamaños distintos para los distintos tamaños de objetos a representar. Esto se debe al rango de visión de las cámaras de los dispositivos móviles, si el marcador está muy alejado no se llegan a distinguir los patrones. En particular, en modelos de gran tamaño, como lo son las heladeras, el marcador queda a una distancia donde es necesario incrementar su tamaño para no perder fácilmente la referencia.

Como comentamos anteriormente, los **image target** generados los incorporamos a nuestra aplicación utilizando una base de datos de imágenes. Ésta puede ser una base de datos en el dispositivo o una base de datos en la nube que funciona a través de Internet. En la Tabla 7 se observa una comparativa que ofrece la librería de Vuforia y que nos sirvió de guía para elegir el tipo de base de datos apropiada para nuestra aplicación. En nuestro caso, optamos por una base de datos alojada en el dispositivo ya la cantidad de imágenes que generamos no es elevada. Para crearla accedimos al Target Manager de Vuforia, seleccionamos *Add Database* y luego especificamos el tipo de base de datos seleccionada y el nombre.

Base de Datos en Dispositivo	Base de Datos en la Nube
Recomendada para base de datos de 1000 objetivos o menos.	Permite hasta 1 millón de objetivos alojados en la nube.
Permite la descarga de objetivos en diferentes combinaciones.	Se compone de una base de datos con todas las imágenes y metadatos.
Los objetivos descargados son sólo para la detección, no hay soporte de metadatos.	Los objetivos recuperados por el reconocimiento de la nube pueden transportar hasta 1MB de metadatos.
La conexión con internet no es necesaria para la detección.	Se requiere una conexión a internet en el dispositivo para poder realizar el reconocimiento.
El tiempo de respuesta de la detección está entre 2 y 3 fotogramas.	El tiempo de respuesta se ve afectado por las condiciones de la red de Internet.
Se pueden tener varias bases de datos activas (cada una con un máximo de 1000 objetivos)	El reconocimiento del millón de objetivos está activo en cualquier momento.
Gratuito.	Gratuito y Pago. Dependiendo del uso.

Tabla 7: Comparación base de datos

Fuente:https://library.vuforia.com/articles/Solution/Comparison-of-Device-and-Cloud-Databases.html

4.4. Creación de la licencia

Para cualquier tipo de aplicación o juego que se desarrolle utilizando Vuforia, se necesita una clave de licencia. De no contar con una, no permite ejecutar el reconocimiento de imágenes. Esto se debe a que Vuforia tiene una versión gratuita y una versión paga, la cual extiende algunas capacidades en cuanto a la cantidad de imágenes que se pueden agregar a la aplicación.

Vuforia proporciona la herramienta **License Manager** (Figura 81), con todos los medios y la información que se necesita para crear y administrar las diferentes licencias.

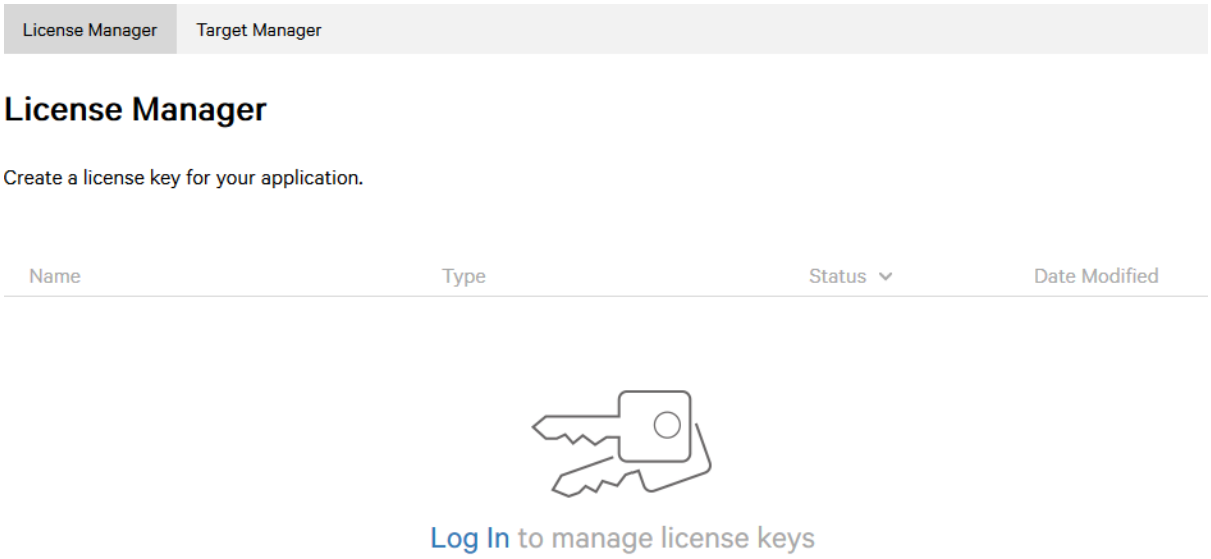


Figura 81: License Manager

Para crear la licencia, en primer lugar, se tiene que elegir el tipo de proyecto que se va a realizar. Vuforia brinda tres opciones de proyecto:

- **Development (Desarrollador)**: Permite desarrollar de forma gratuita antes de actualizar a una licencia comercial.

- **Consumer (Consumidor)**: Permite crear aplicaciones para distribución pública.

- **Enterprise (Empresa)**: Permite crear aplicaciones para distribución empresarial.

En nuestro caso optamos por la de tipo Desarrollador, con la opción de pagar la licencia comercial si se desea.

Una vez establecido el tipo de proyecto, le asignamos el nombre **FARVA** a nuestra aplicación y generamos la clave de licencia. Esta clave es la que utilizamos, luego, en la cámara de RA de Vuforia en Unity 3D.

Hay que destacar que una licencia sólo es válida para utilizar en una sola aplicación. Se debe crear una clave de licencia única por cada aplicación de Vuforia que se desarrolle, pero se puede usar la misma clave de licencia para todas las versiones de los sistemas operativos admitidos por el tipo de licencia.

4.5. Creación de la aplicación

Una vez creados los modelos 3D, los marcadores y la base de datos de imágenes, como explicamos en las secciones anteriores, debemos incorporarlos a las herramientas principales de desarrollo de la aplicación.

Como herramientas utilizamos Unity 3D versión 5.4 y Vuforia versión 6.2 para Unity 3D (por los motivos expuestos en la secciones 3.1.3 y 3.2.3.). Al ser ambos gratuitos se pueden obtener fácilmente desde la página oficial de cada uno:

- **Descarga de Unity 3D**: https://unity3d.com/es/get-unity/download

- **Descarga de Vuforia** (Figura 82): https://developer.vuforia.com/downloads/sdk

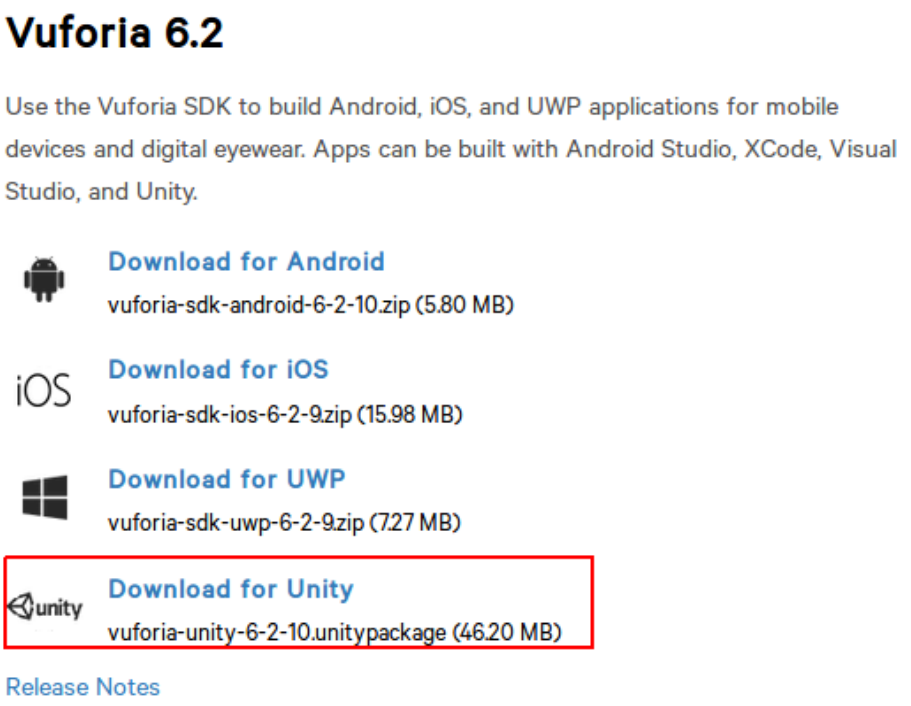


Figura 82: Descarga de Vuforia

Luego de instalado Unity 3D, al ingresar solicita que se cargue una cuenta (Figura 83). En caso de no tener una, se puede crear desde la página oficial de Unity 3D. Esta cuenta se genera para poder vincular los complementos (assets) que utiliza ese usuario, ya sean comprados o gratuitos.

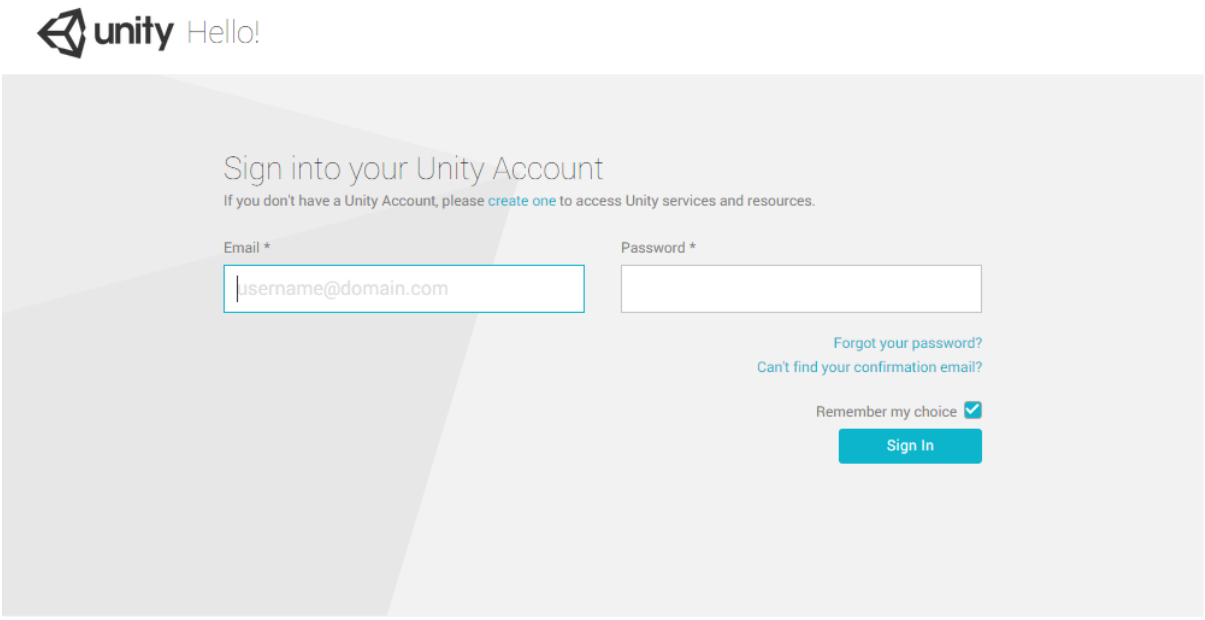


Figura 83: Menú de inicio de sesión de Unity

Con la cuenta configurada, creamos un nuevo proyecto (Figura 84). Unity brinda la opción de elegir entre un proyecto en 2D o uno en 3D; como nuestra aplicación hace uso de modelos 3D, optamos por esta última opción.

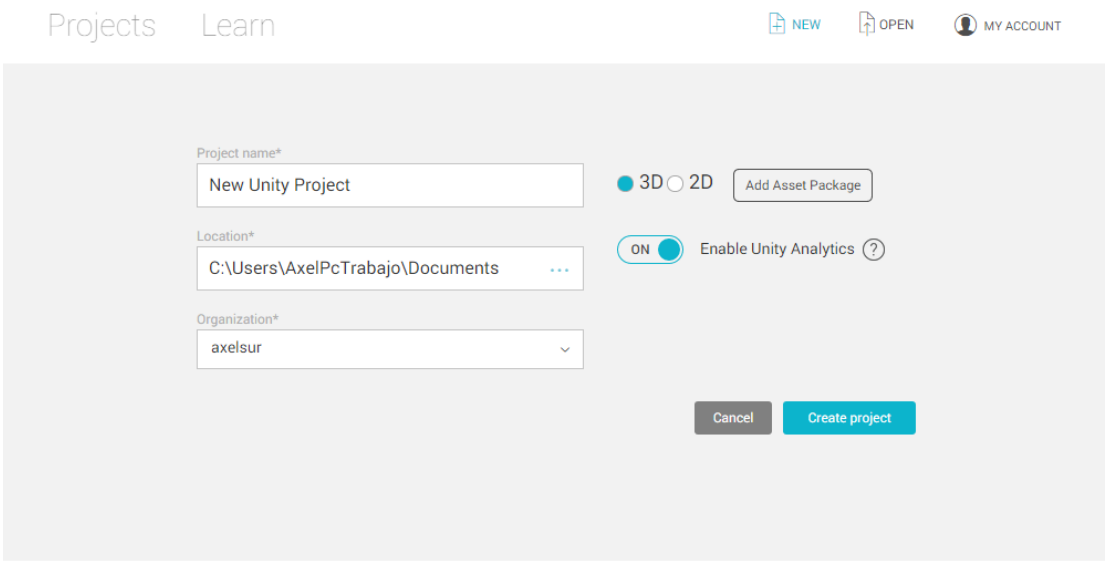


Figura 84: Creación del proyecto

En el proyecto creado incorporamos el SDK de Vuforia, desde la opción *Custom Package* (Figura 85); también se puede importar, realizando doble click sobre el paquete descargado de Vuforia para Unity 3D. Esto agrega al proyecto un conjunto de elementos que incorpora Vuforia para poder trabajar con RA.

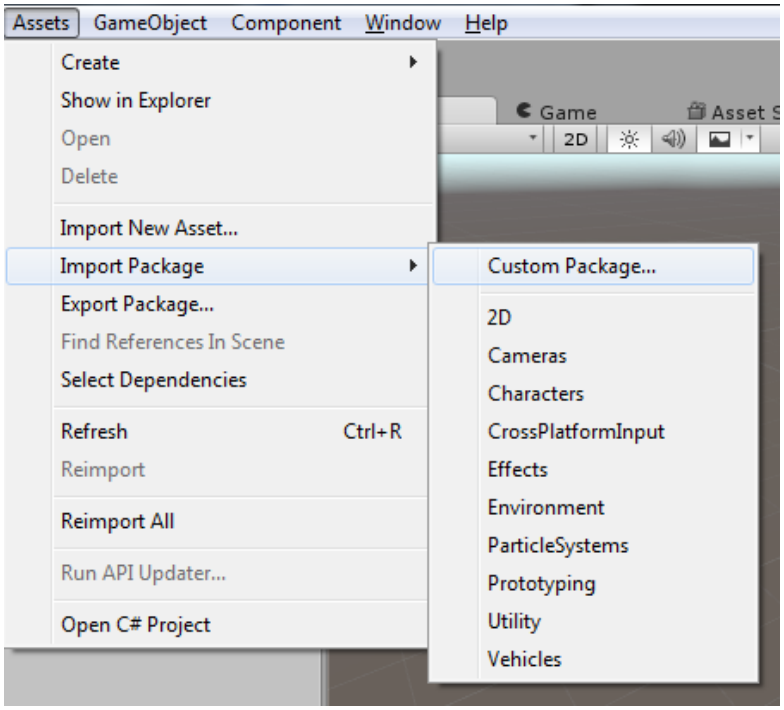


Figura 85: Menú para agregar paquetes externos

Con Vuforia ya incorporado al proyecto, sustituimos la cámara que viene por defecto por una cámara que funcione con RA. Vuforia proporciona un conjunto de **Prefabs** (Figura 86), y uno de ellos es una cámara especial que se denomina **ARCamera** que, luego de eliminar la cámara básica, incorporamos al proyecto.

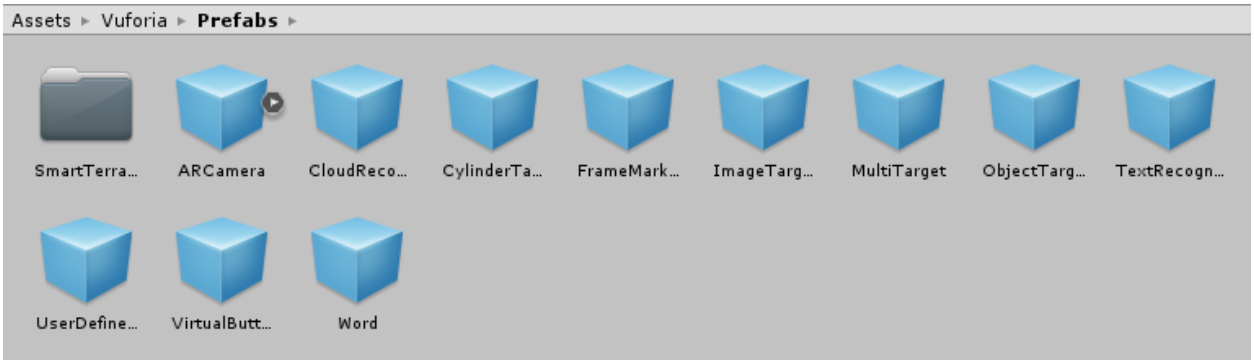


Figura 86: Prefabs de Vuforia

ARCamara es el elemento principal de la aplicación, donde se encuentra la mayor parte de la lógica del SDK. Captura la imagen en vivo de la cámara del dispositivo y se encarga de realizar el reconocimiento de los marcadores que se encuentran en la base de datos de imágenes. Cabe destacar que, dentro de las configuraciones, debemos ingresar la licencia que generamos anteriormente (Figura 87); de no hacerlo, no nos permitirá realizar la detección de las imágenes.

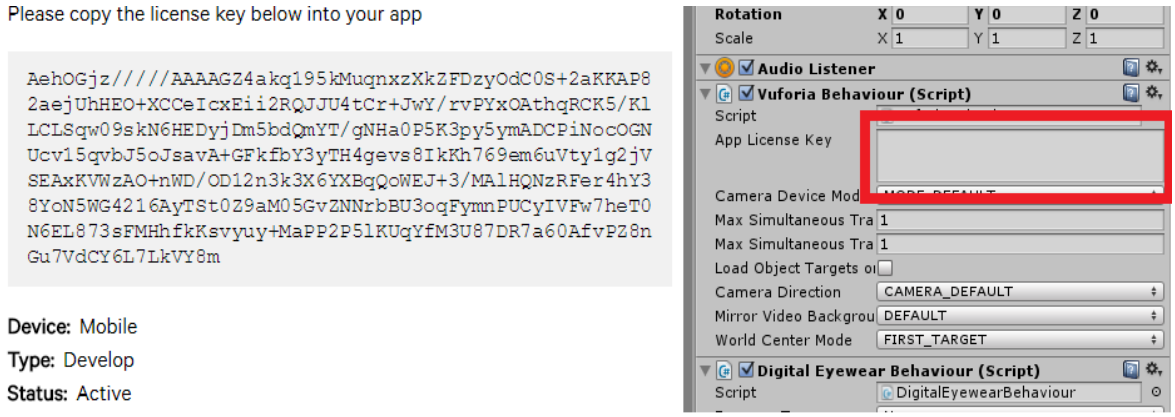


Figura 87: Configuración de la licencia para ARCamera

Con la cámara configurada, agregamos la base de datos de imágenes, obtenida en la Sección 4.3., incorporando el paquete Unity de la base (de igual forma que lo hicimos con Vuforia). El contenido de esta base es lo que la cámara RA va a buscar como patrones de reconocimiento. El SDK de Vuforia brinda la capacidad de utilizar múltiples bases de datos de imágenes simultáneamente (Figura 88). En nuestro caso, sólo utilizamos una.

Para activar esta capacidad y para que se reconozcan las imágenes, marcamos el campo *Datasets* en la configuración de Vuforia del prefab **ARCamera** y, además, importamos los modelos 3D, obtenidos en la Sección 4.2.

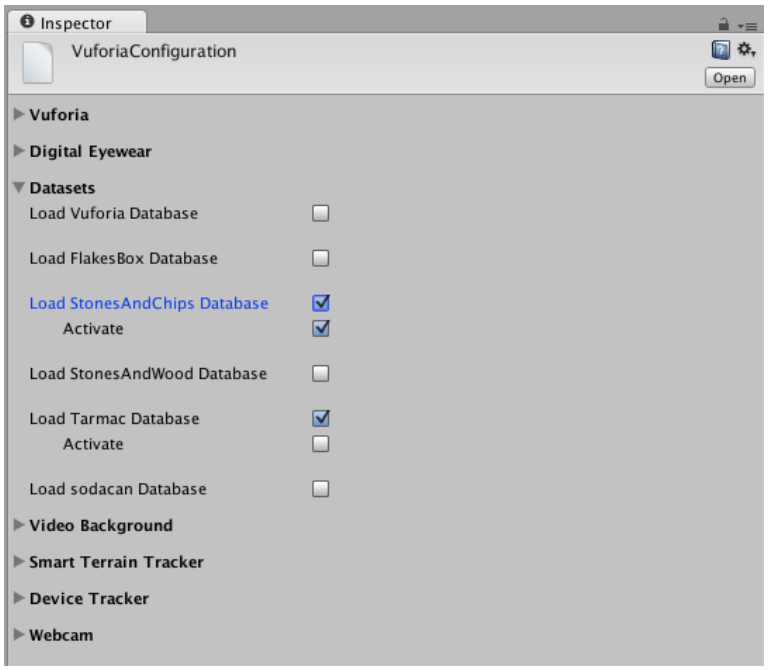


Figura 88: Bases de datos activas

La forma de vincular nuestros modelos a los marcadores es mediante los **Image Target.** Por cada modelo 3D agregamos la imagen que corresponda de la base de datos y se la asignamos a un **Image Target**. Esto se puede hacer de dos formas, la primera arrastrando la imagen arriba del **Imagen Target** y la otra es desde el menú de configuración del **Image Target**, seleccionando la base de datos y la imagen correspondiente. Luego, a cada uno de los modelos 3D los ubicamos jerárquicamente por debajo de los **Image Target**, indicando la relación entre ambos.

En este punto, la aplicación ya permite visualizar los objetos mediante RA, pero todavía éstos no se encuentran a escala real. Para lograrlo, debemos definir cuál es el tamaño de impresión de cada una de las imágenes; en base a esto la aplicación puede determinar la proporción de tamaño que tendrá el modelo 3D.

Una vez definido el tamaño, por ejemplo 12 cm por 12 cm, para la imagen que usamos de marcador, desde Unity 3D ingresamos la escala del **Image Target** que va a servir de referencia para el modelo 3D que tiene relacionado (en este caso 1). A partir de ahí configuramos la escala del modelo 3D, según el tamaño con que se creó en el programa de modelado y en relación a los 12 cm de referencia del **Image Target**.

Con el fin mejorar la utilización de la aplicación, agregamos rotación y movimiento a los objetos mediante gestos con los dedos sobre la pantalla del dispositivo. Para ello, usamos la librería **Lean Touch** que permite simular varios gestos con los dedos. Como tiene su código fuente completo bien comentado, se puede fácilmente modificar o ampliar sus características de

ser necesario. Descargamos esta librería del AppStore de Unity 3D y la importamos para, luego, agregarla a la escena principal de la aplicación.

Por cada modelo 3D agregamos el/los script/s correspondiente/s a la/s acción/es deseada/s. Estos scripts reconocen los gestos realizados sobre la pantalla de forma que nos permiten capturarlos y trabajar con ellos.

En todos los modelos 3D usamos un script para rotación (LeanRotate.cs) y otro para movimiento (LeanTranslate.cs). A continuación, se muestran dichos scripts:

LeanRotate.cs

```
using UnityEngine;

namespace Lean.Touch
{
	public class LeanRotate : MonoBehaviour
	{
		[Tooltip("Ignore fingers with StartedOverGui?")]
		public bool IgnoreGuiFingers;

		[Tooltip("Allows you to force rotation with a specific amount of fingers (0 = any)")]
		public int RequiredFingerCount;

		[Tooltip("Does rotation require an object to be selected?")]
		public LeanSelectable RequiredSelectable;

		[Tooltip("The rotation axis used for non-relative rotations")]
		public Vector3 RotateAxis = Vector3.forward;

		[Tooltip("Should the rotation be performanced relative to the finger center?")]
		public bool Relative;

#if UNITY_EDITOR
		protected virtual void Reset()
		{
			if (RequiredSelectable == null)
			{
				RequiredSelectable = GetComponent<LeanSelectable>();
			}
		}
#endif

		protected virtual void Update()
		{
			if (RequiredSelectable != null && RequiredSelectable.IsSelected == false)
			{
```

```
                        return;
                }

                var fingers = LeanTouch.GetFingers(IgnoreGuiFingers,
RequiredFingerCount);

                var center  = LeanGesture.GetScreenCenter(fingers);
                var degrees = LeanGesture.GetTwistDegrees(fingers);

                Rotate(center, degrees);
        }

        private void Rotate(Vector3 center, float degrees)
        {
                if (Relative == true)
                {
                        var worldReferencePoint =
Camera.main.ScreenToWorldPoint(center);

                        transform.RotateAround(worldReferencePoint,
Camera.main.transform.forward, degrees);
                }
                else
                {
                        transform.rotation *= Quaternion.AngleAxis(degrees,
RotateAxis);
                }
        }
    }
}
```

LeanTranslate.cs

```
using UnityEngine;

namespace Lean.Touch
{
    public class LeanTranslate : MonoBehaviour
    {
        [Tooltip("Ignore fingers with StartedOverGui?")]
        public bool IgnoreGuiFingers = true;

        [Tooltip("Allows you to force rotation with a specific amount of fingers (0 =
any)")]
        public int RequiredFingerCount;

        [Tooltip("Does translation require an object to be selected?")]
        public LeanSelectable RequiredSelectable;
```

```
#if UNITY_EDITOR
		protected virtual void Reset()
		{
			if (RequiredSelectable == null)
			{
				RequiredSelectable = GetComponent<LeanSelectable>();
			}
		}
#endif

		protected virtual void Update()
		{

			if (RequiredSelectable != null && RequiredSelectable.IsSelected ==
false)
			{
				return;
			}

			var fingers = LeanTouch.GetFingers(IgnoreGuiFingers,
RequiredFingerCount);

			var screenDelta = LeanGesture.GetScreenDelta(fingers);

			Translate(screenDelta);
		}

		private void Translate(Vector2 screenDelta)
		{
			var screenPosition =
Camera.main.WorldToScreenPoint(transform.position);

			screenPosition += (Vector3)screenDelta;

			transform.position = Camera.main.ScreenToWorldPoint(screenPosition);
		}
	}
}
```

Realizando pruebas de rotación y movimiento, nos encontramos con algunos problemas. Si el patrón dejaba de ser visible o reconocible para la cámara RA, desaparecía el modelo 3D. Observamos que si teníamos el marcador de frente y lo posicionábamos con una relación del 30% de superficie de la pantalla total, éste empezaba a oscilar de manera involuntaria. Incluso si nos posicionábamos a 20° del marcador, el modelo se desvanecía. También, se necesitaba una buena iluminación y cercanía al marcador para que la cámara pudiera realizar el reconocimiento. Además, nos generaba un posicionamiento constante del modelo. Como resultado de esto, no era posible mover nuestro modelo por el reposicionamiento constante (posición del marcador); la única forma de hacerlo era mover el marcador.

Al investigar más a fondo el tema, concluimos que teníamos estos problemas porque estábamos usando un marcador sin seguimiento extendido (Figura 89), es decir, en cualquier momento en que se deja de captar el patrón, el modelo desaparece.

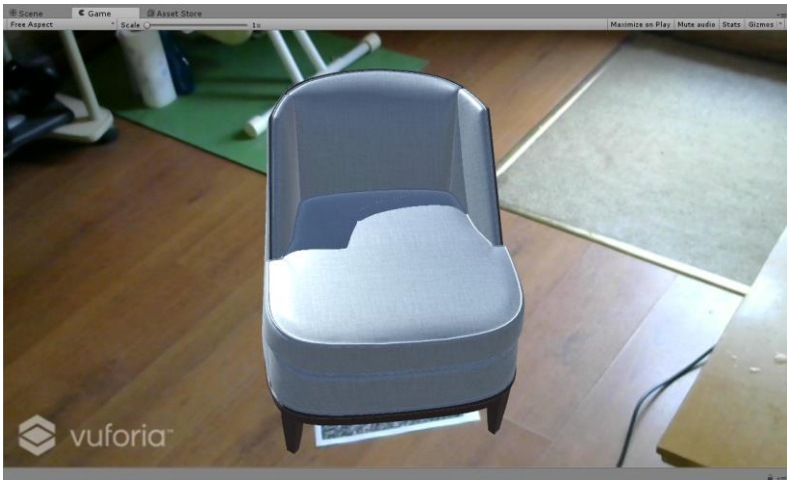


Figura 89: Prueba sin seguimiento extendido

Para solucionarlo, configuramos una opción llamada *Extended Trancking* dentro del modelo 3D (seguimiento extendido, Figura 90). Sin esta opción configurada el modelo 3D queda estrictamente ligado a la posición del marcador.

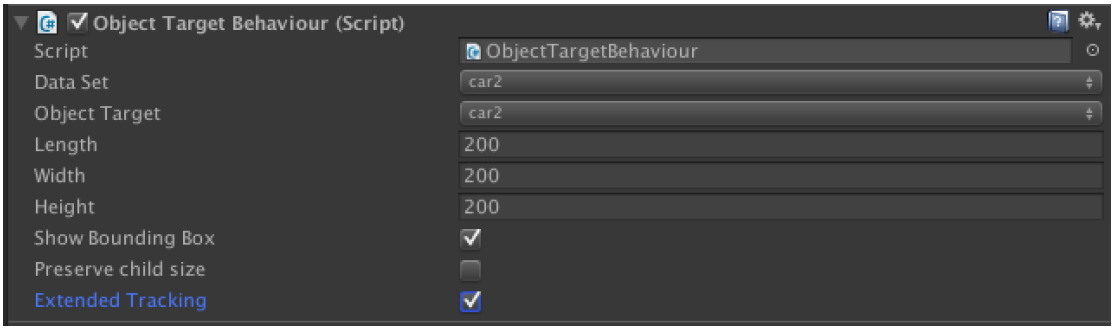


Figura 90: Extended Tracking

De esta forma, resolvimos los problemas de posicionamiento y distancia entre la cámara y el marcador. Como hemos adelantado, el código funciona grabando matices del entorno por lo que si nos encontramos en un entorno uniforme sin objetos que nos ayude a reconocerlo, éste no puede calcular la posición y ángulo en que estamos. La rotación de la cámara se debe hacer a

razón de 25° por segundo, aproximadamente, para no perder el objetivo, siempre que no esté en pantalla el marcador. En la Figura 91 vemos un ejemplo del seguimiento extendido:

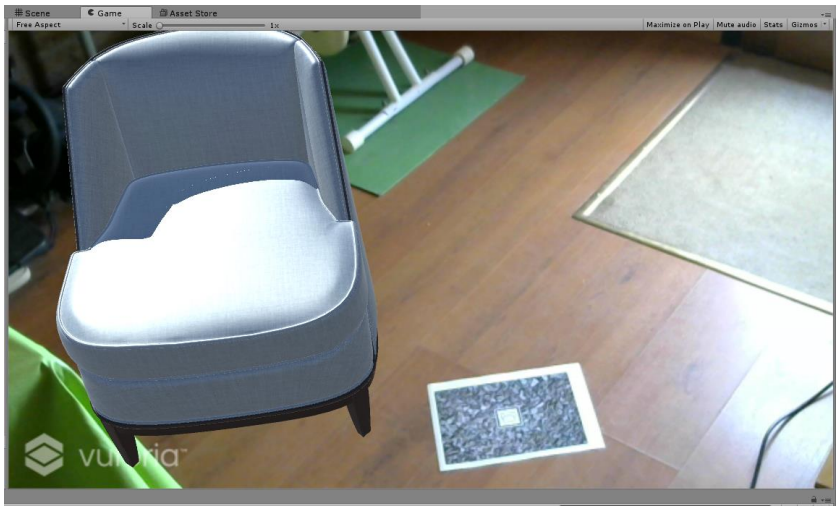


Figura 91: Prueba seguimiento extendido

Por último, incluimos en nuestra aplicación una interfaz de navegación, como se muestra la Figura 92:

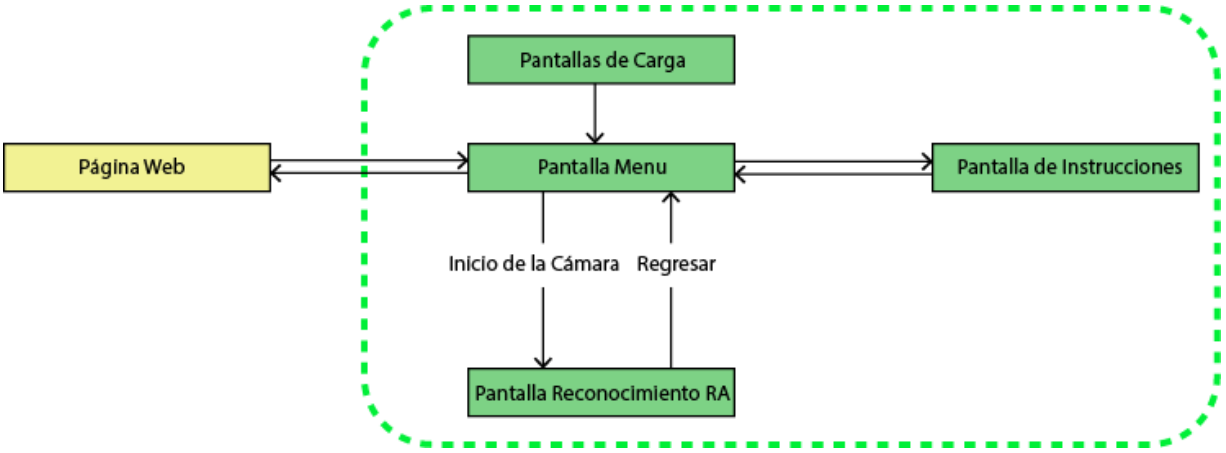


Figura 92: Diagrama de navegación

Para ello, utilizamos el Gameobject que brinda Unity 3D, llamado **Canvas**. El área del **Canvas** se muestra en pantalla como un rectángulo en la Vista de Escena, lo que permite un control simple del posicionamiento de los elementos de la interfaz de usuario (UI). Para crear distintas pantallas de **Canvas** es necesario crear distintas escenas dentro del proyecto (que contiene todos los objetos que serán representados por los distintos juegos o aplicaciones).

Pueden ser usadas para crear UI (escenas), niveles individuales para un juego, o cualquier otra cosa que se quiera representar.

En nuestro caso, creamos tres nuevas escenas: dos para las pantallas de inicio y otra para la creación de un Menú (Figura 93), que posee las opciones de comenzar con el uso del visualizador de RA (botón **Cámara**), mostrar información sobre el uso de la aplicación (botón **Instrucciones**), vincular con el sitio para descargar los marcadores (botón **Ir a la Página**) y cerrar la aplicación (botón **Salir**).

Figura 93: Menú

Para poder implementar estas funciones, tuvimos que crear botones. Éstos están definidos en Unity 3D como un objeto de interacción y reaccionan a un evento cuando se presionan. A cada botón se le asigna un script y una función que se encuentra definida dentro de este script (Figura 94).

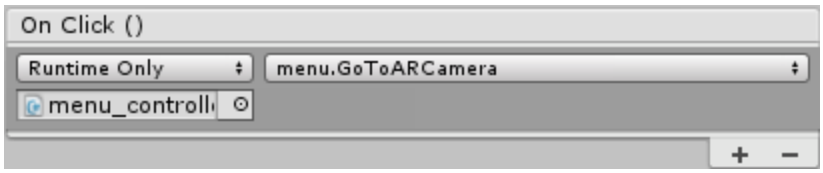


Figura 94: Asignación de scripts a los botones

Además de los botones mencionados anteriormente, agregamos en el visualizador tres botones: el botón **Captura** (que realiza una captura de pantalla que se almacena en "raíz/android/data/com.tesina.farva/files", dentro de la memoria del dispositivo), el botón **Compartir** (que permite compartir la foto capturada por medio de las redes sociales), y el botón **Regresar** (que permite regresar a la escena del menú principal). Para proveer las funcionalidades de estos botones utilizamos **Android Ultimate Plugin**, un complemento de Android para Unity 3D que nos facilitó la tarea de acceder a las características del dispositivo Android.

Al igual que con el Menú, creamos el siguiente script para las funcionalidades particulares de cada botón:

SnapshotShare.cs

```
using UnityEngine;
using System.Collections;
using System.IO;

public class SnapshotShare : MonoBehaviour
{
	private SharePlugin sharePlugin;
	Camera mainCamera;
	RenderTexture renderTex;
	Texture2D screenshot;
	Texture2D LoadScreenshot;
	int width = Screen.width;
	int height = Screen.height;
	string fileName;
	string screenShotName = "PictureShare.png";

	void Start ()
	{
		sharePlugin = SharePlugin.GetInstance();
	}

	public void Snapshot ()
	{
		StartCoroutine (CaptureScreen ());
	}

	public IEnumerator CaptureScreen ()
```

```
	{
		yield return null;

		GameObject.Find ("Canvas").GetComponent<Canvas> ().enabled = false;
		yield return new WaitForEndOfFrame ();
		if (Screen.orientation == ScreenOrientation.Portrait || Screen.orientation == 
ScreenOrientation.PortraitUpsideDown) {
			mainCamera = Camera.main.GetComponent<Camera> ();
			renderTex = new RenderTexture (width, height, 24);
			mainCamera.targetTexture = renderTex;
			RenderTexture.active = renderTex;
			mainCamera.Render ();
			screenshot = new Texture2D (width, height, TextureFormat.RGB24, 
false);
			screenshot.ReadPixels (new Rect (0, 0, width, height), 0, 0);
			screenshot.Apply ();
			RenderTexture.active = null;
			mainCamera.targetTexture = null;
		}
		if (Screen.orientation == ScreenOrientation.LandscapeLeft || 
Screen.orientation == ScreenOrientation.LandscapeRight) {
			mainCamera = Camera.main.GetComponent<Camera> ();
			renderTex = new RenderTexture (height, width, 24);
			mainCamera.targetTexture = renderTex;
			RenderTexture.active = renderTex;
			mainCamera.Render ();
			screenshot = new Texture2D (height, width, TextureFormat.RGB24, 
false);
			screenshot.ReadPixels (new Rect (0, 0, height, width), 0, 0);
			screenshot.Apply ();
			RenderTexture.active = null;
			mainCamera.targetTexture = null;
		}

		File.WriteAllBytes (Application.persistentDataPath + "/" + screenShotName, 
screenshot.EncodeToPNG ());

		GameObject.Find ("Canvas").GetComponent<Canvas> ().enabled = true;
	}


	public void ShareImage ()
	{
		string path = Application.persistentDataPath + "/" + screenShotName;
		sharePlugin.ShareImage ("subject","subjectContent",path);
	}


	public void close ()
	{
		Application.Quit ();
```

```
        }
}
```

Una vez finalizada la interfaz y las funcionalidades de los botones, realizamos la exportación de la aplicación para el sistema operativo Android. Luego, la instalamos en un celular LG G5 y realizamos pruebas exhaustivas. En la Figura 95 se puede apreciar el resultado de la ejecución del visor:

Figura 95: Prueba desde LG G5

4.6. Distribución de la aplicación

Para distribuir nuestra aplicación **FARVA** a los usuarios de Android, utilizamos la plataforma que nos brinda Google, llamada **Google Play Store** (ver Sección 2.1.3.). Es necesario tener un usuario desarrollador para utilizarla; por ello, registramos uno de nuestros correos electrónicos como “usuario desarrollador” [39].

Luego de acceder a nuestra cuenta Google, se nos solicitó que aceptemos el Acuerdo para desarrolladores (Figura 96). Después, pagamos una tarifa de U$S 25 y completamos un formulario de datos.

Figura 96: Creación cuenta desarrollador

Con nuestro usuario desarrollador tenemos acceso a **Google Play Developer** (Figura 97), que es el administrador web que brinda Google para gestionar las aplicaciones que los desarrolladores desean subir.

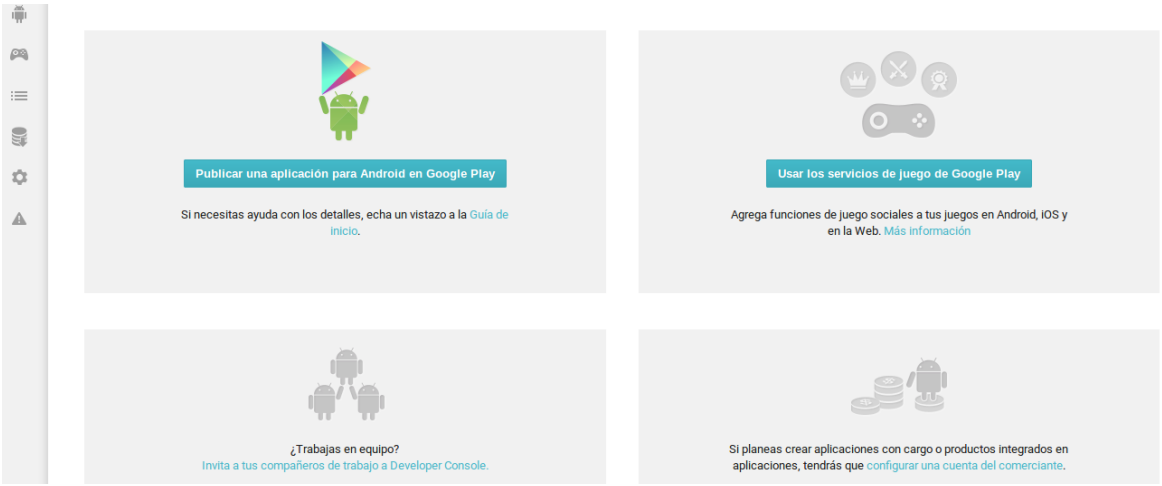


Figura 97: Acceso a Google Play Developer Console

En *Publicar una aplicación para Android en Google Play* cargamos nuestra aplicación indicando el idioma predeterminado y el nombre **FARVA** (Figura 98):

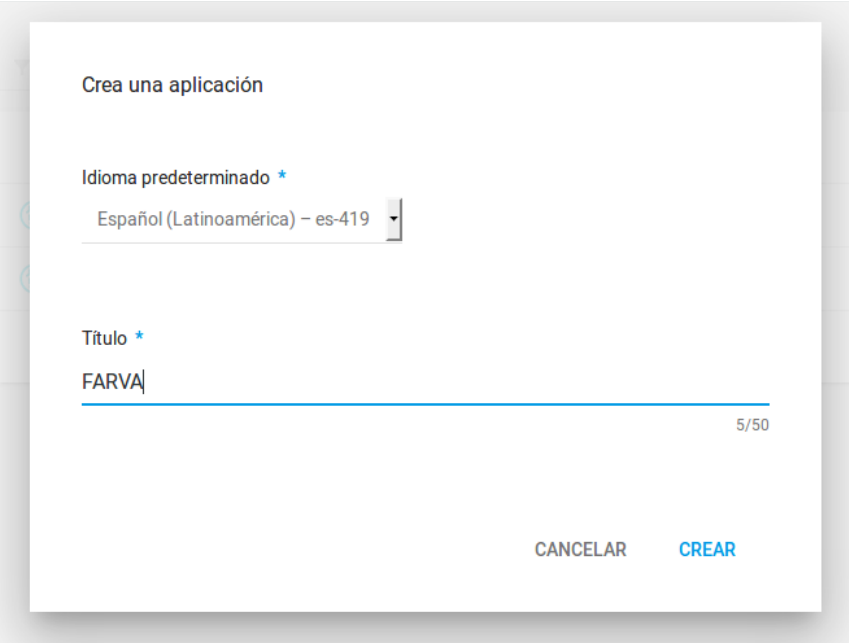


Figura 98: Creación de la aplicación

De esta forma generamos el "repositorio" y, en el menú de configuración de la plataforma, administramos las versiones de nuestra aplicación, su alcance y precio (Figura 99):

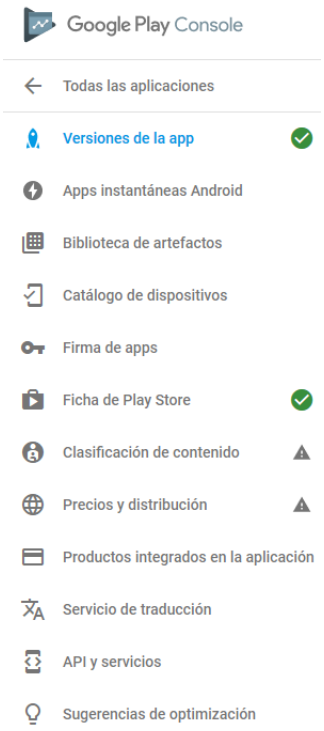


Figura 99: Menú de configuración del Google Play Store

Por último, realizamos la configuración mínima para tener nuestra aplicación en **Google Play Store**. Para esto, completamos los cuatro formularios siguientes que nos exige la plataforma:

- **Versiones de la app**: En este formulario se gestionan las diferentes versiones que subimos a **Google Play Store**; en nuestro caso, subimos la versión 1.0 de **FARVA** (Figura 100).

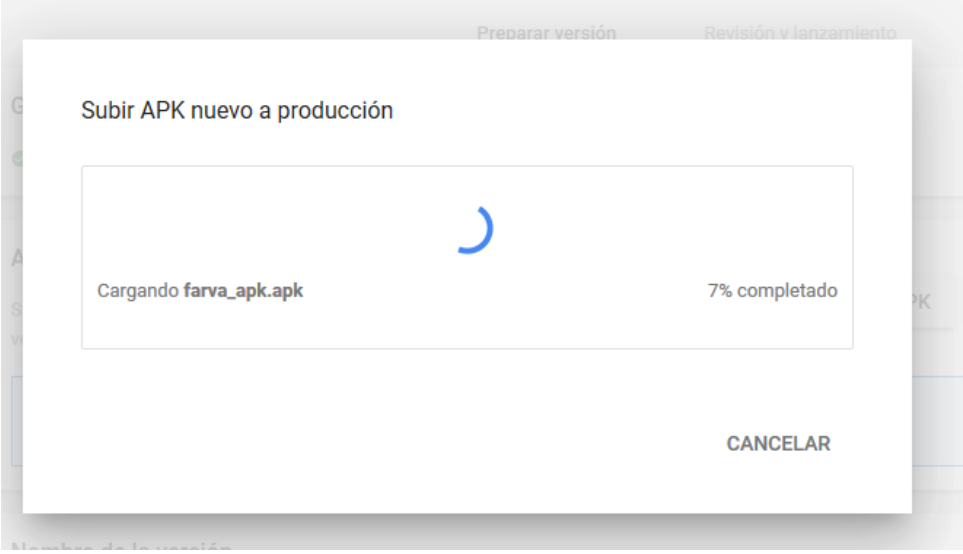


Figura 100: Carga de versión en Google Play Store

- **Ficha de Play Store**: En este formulario cargamos todos los datos referentes a la aplicación que serán visualizados desde la plataforma **Google Play Store**, desde título, descripción e imágenes testigo, hasta la página de políticas de privacidad (Figura 101).

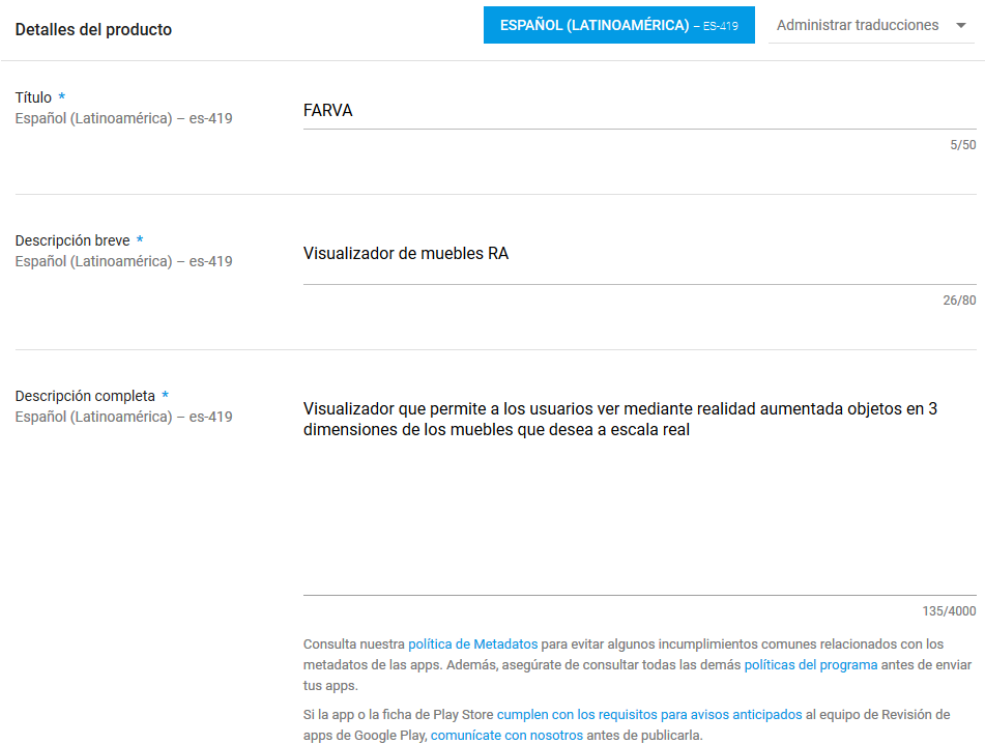


Figura 101: Ficha de Play Store

- **Clasificación del contenido**: En este formulario se indica qué tipo de aplicación se pone disponible. En nuestro caso, seleccionamos la categoría "Agregadores de contenido, servicios de transmisión comercial o tiendas del consumidor" (Figura 102). Además, completamos una serie de preguntas acerca del contenido que posee la aplicación: publicidad, anuncios, violencia, etc. Y realizamos el cálculo de clasificación de contenido.

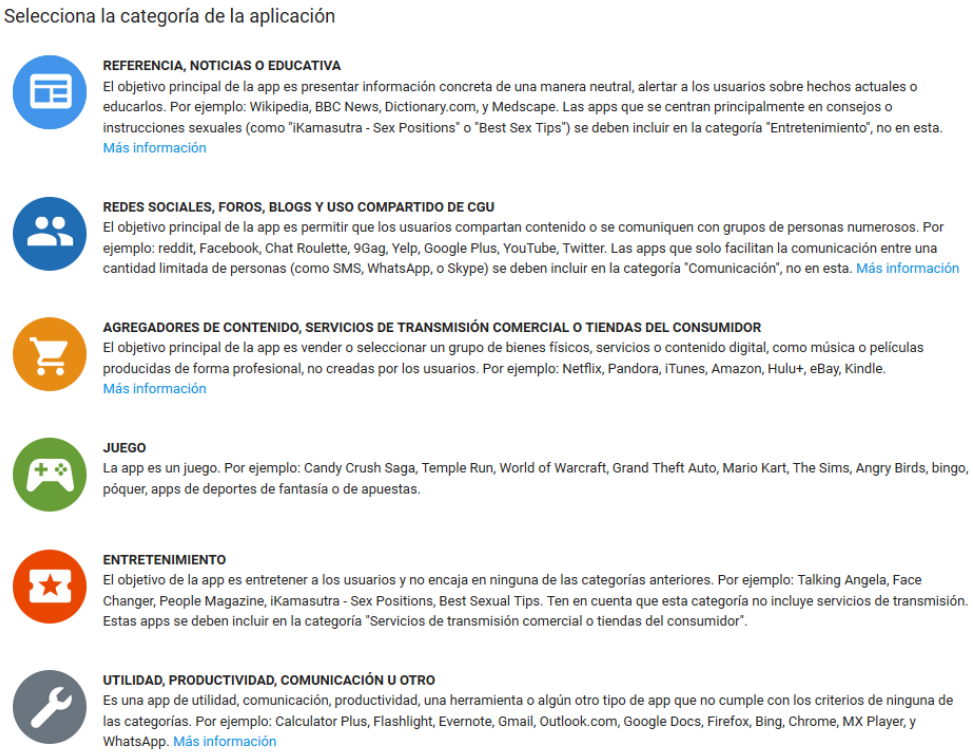


Figura 102: Clasificación del contenido

- **Precios y distribución**: En este último formulario se indica en qué países se permite que sea visible la aplicación y su precio. En nuestro caso, marcamos sólo en Argentina y seleccionamos "Gratis" (Figura 103).

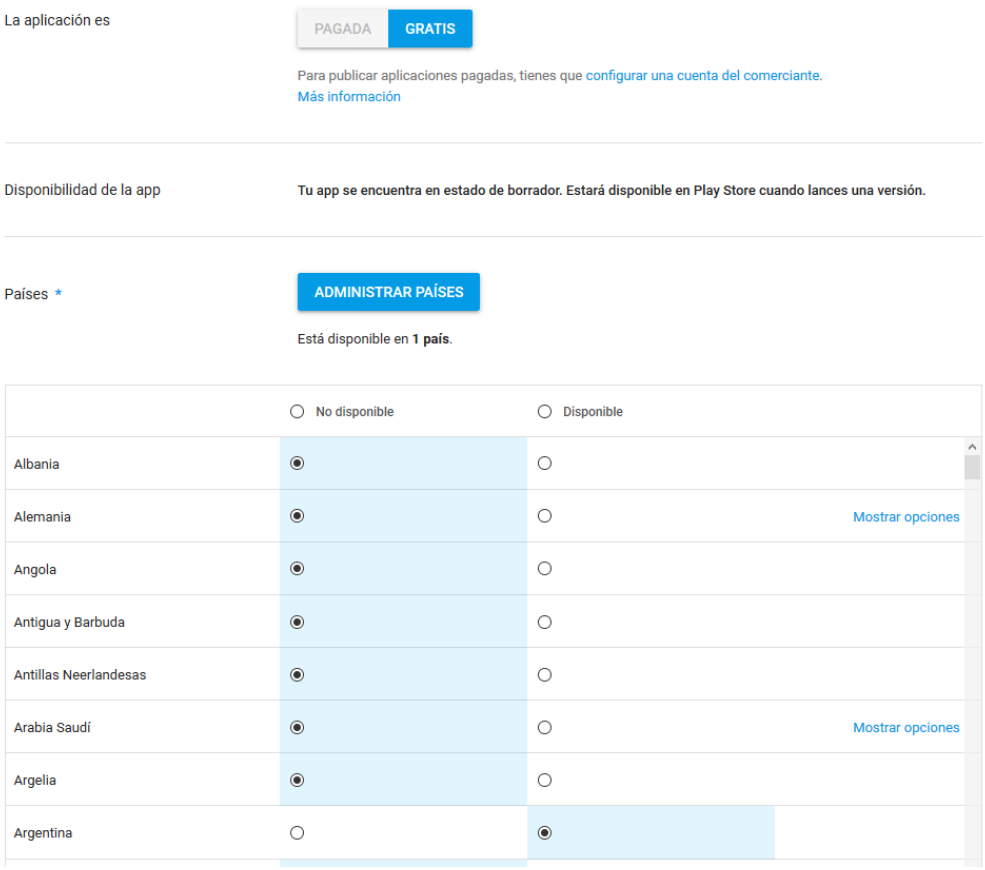


Figura 103: Precios y distribución

Una vez que completamos todos los formularios, ya nos encontramos habilitados para publicar la aplicación.

En la ventana *Administrar Aplicaciones*, al pedir *Revisar*, nos envió a la página de confirmación del lanzamiento de la versión (Figura 104).

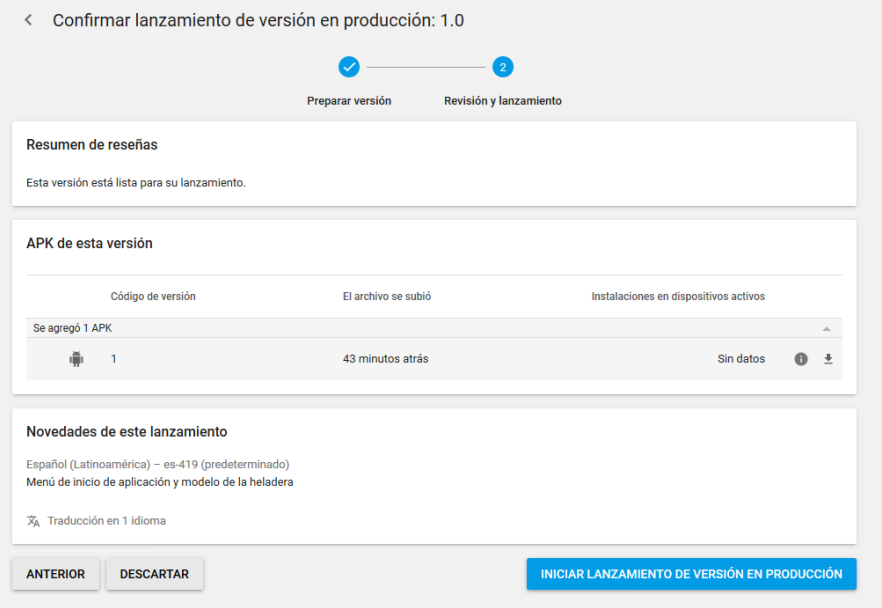


Figura 104: Revisión y lanzamiento

Aquí iniciamos el lanzamiento de la versión. Por último, apareció un menú con un resumen de la versión, en donde seleccionamos *Crear versión*. Ésto envió nuestra aplicación para una revisión final que realiza Google (Figura 105).

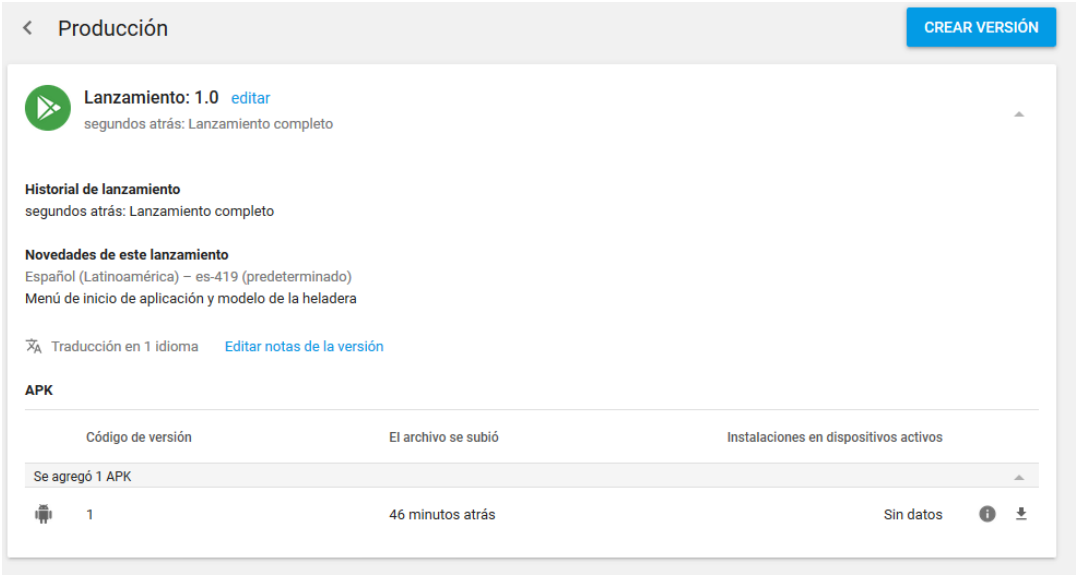


Figura 105: Puesta en producción

Una vez concluida la revisión final (y de no existir ningún inconveniente con los datos ingresados), después de unas horas la aplicación apareció en **Google Play Store** (Figura 106).

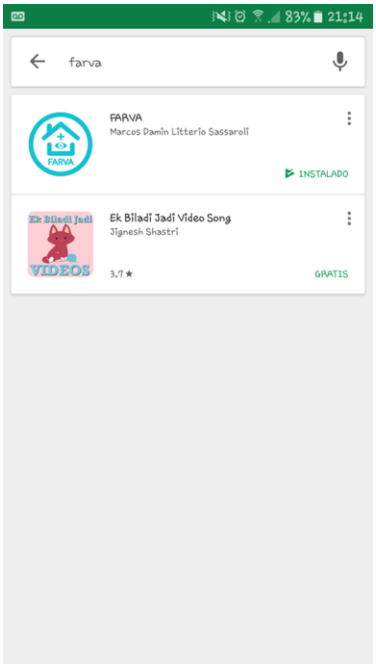


Figura 106: FARVA en Google Play Store

4.7. Uso de la aplicación

El usuario que desee utilizar la aplicación **FARVA**, deberá descargarla desde **Google Play Store** e instalarla en su dispositivo móvil.

Para ejecutarla, debe presionar el ícono de **FARVA**, como se ve en la Figura 107:

Figura 107: Ejecución de FARVA

Primero se visualiza la pantalla de inicio, seguida por la de carga de la aplicación, como se muestra en la Figura 108.

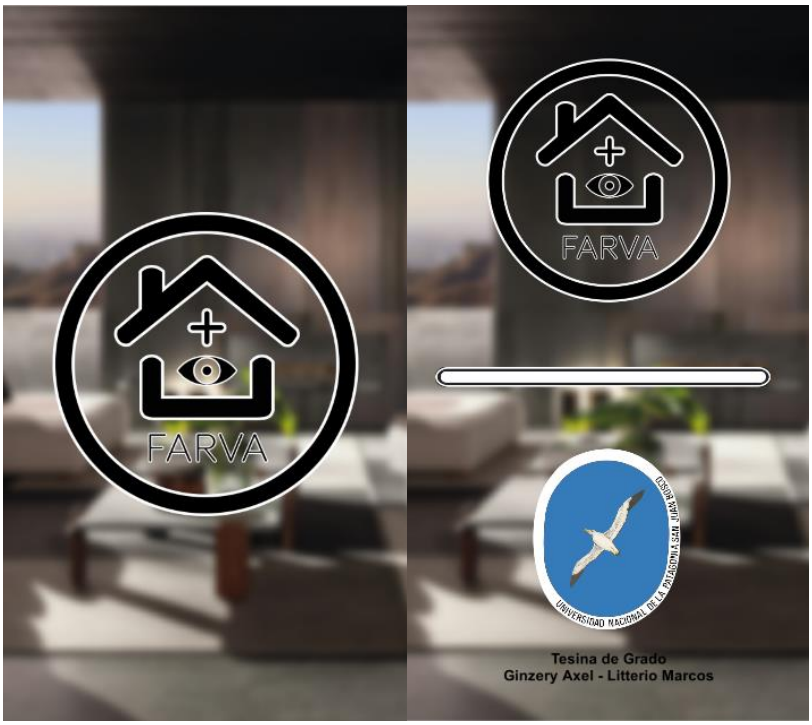


Figura 108: Pantallas de inicio y carga de FARVA

Una vez completada la carga , se presenta el Menú Principal (Figura 109) que cuenta con los siguientes cuatro botones:

Figura 109: Menú principal

- **Salir**: Este botón (Figura 110) cumple la función básica de cerrar la aplicación.

Figura 110: Botón Salir

- **Ir a la Página**: Este botón (Figura 111) tiene un enlace externo a la página **FARVA Store** www.farva.com.ar (Figura 112), que creamos para simular la página de un vendedor. En ella se encuentra información referente a **FARVA** y

están publicados los marcadores y la vista previa de los modelos 3D que desarrollamos en esta Tesina. Esta página funciona como **Catálogo de Productos** del vendedor.

Figura 111: Botón Ir a la Página

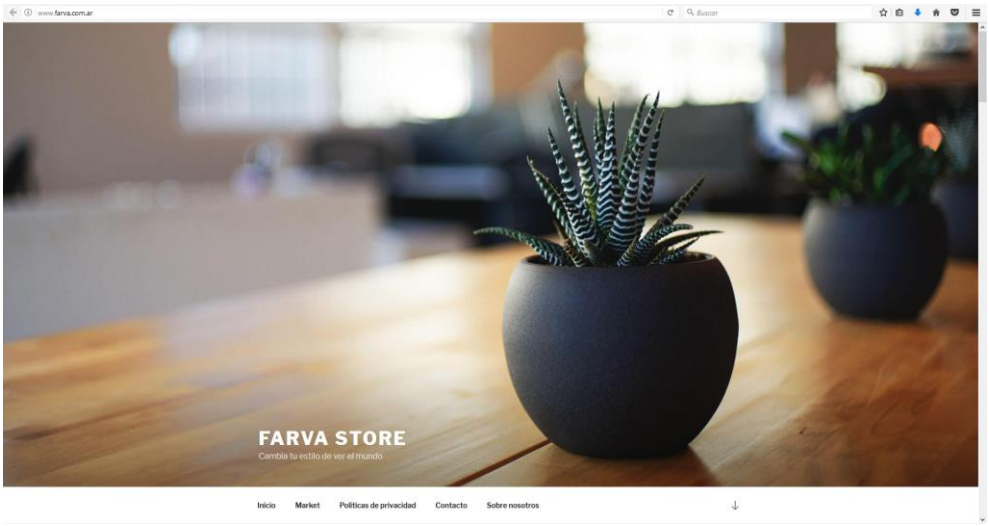


Figura 112: Página del vendedor FARVA Store

- **Instrucciones**: Este botón (Figura 113) brinda una breve descripción sobre cómo utilizar la aplicación (Figura 114).

Figura 113: Botón Instrucciones

Figura 114: Instrucciones de uso

- **Cámara RA**: Este botón (Figura 115) ejecuta la función principal de la aplicación. Permite el acceso al visualizador de la cámara del dispositivo, que es el encargado de realizar la detección del marcador y la representación del modelo 3D.

Figura 115: Botón Cámara Realidad Aumentada

Una vez en el visualizador, sólo es necesario enfocar el marcador impreso para representar el modelo 3D del objeto deseado, como se muestra en la Figura 116:

Figura 116: Visualizador de RA

Cuando se detecta el marcador y aparece el modelo 3D, si se desea se puede girar el dispositivo, sin perder de vista el marcador, para tener otra perspectiva de visualización.
Esta tarea también se realiza girando el marcador impreso y, para facilitarla, la aplicación soporta la interacción sobre la pantalla mediante gestos. Si se desliza un dedo sobre la pantalla del dispositivo se traslada el objeto (Figura 117), mientras que con el uso de dos dedos y el movimiento horario o antihorario de éstos, se rota el objeto representado (Figura 118).

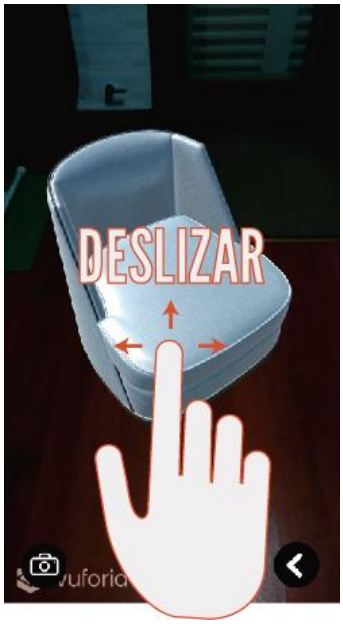


Figura 117: Acción de mover objeto

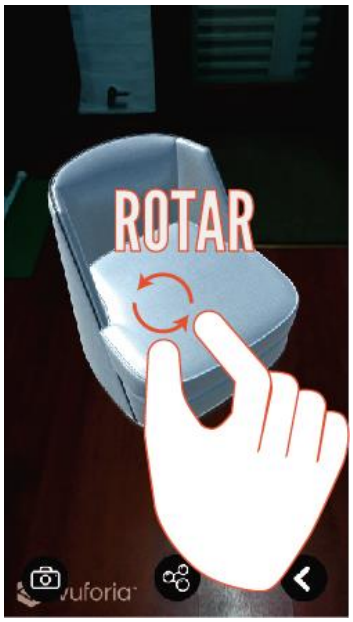


Figura 118: Acción de girar objeto

Además de las acciones con gestos que se pueden realizar sobre la pantalla, en el visualizador se dispone de tres botones, ubicados en la parte inferior (Figura 119):

Figura 119: Botones del visualizador de RA

Estos botones proveen las siguientes funcionalidades:

- **Botón Captura de Pantalla** (Figura 120): Permite realizar una captura de pantalla.

Figura 120: Botón Captura Pantalla

- **Botón Compartir** (Figura 121): Permite compartir la imagen de la última captura de pantalla realizada. En la zona inferior de la pantalla, aparece un menú desplegable que brinda la opción de compartir la imagen a través de Whatsapp, Instagram, Facebook, entre otras (Figura 122).

Figura 121: Botón Compartir

Figura 122: Menú desplegable para Compartir Imagen

- **Botón Regresar** (Figura 123): Permite regresar al menú principal.

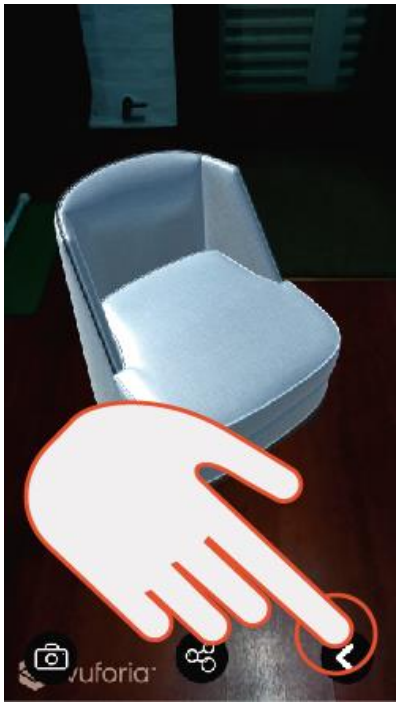


Figura 123: Botón regresar menú principal

5. Conclusiones y trabajos futuros

En este documento presentamos la investigación realizada sobre el sistema operativo Android, y sobre los principios, conceptos y herramientas de RA y Modelado 3D que hemos utilizado para el desarrollo de una aplicación de RA, que pueda ser utilizada por los Sistemas de Compra Web (o por catálogos impresos) y sus clientes para visualizar, de forma sencilla e interactiva, los productos ofrecidos de forma virtual en una escala y ubicación acorde al ambiente.

Como resultado obtuvimos la aplicación móvil **FARVA** para Android, que hace uso de modelos 3D, de una librería de RA novedosa y de uno de los motores gráficos más conocidos. Nuestra aplicación le permite a un potencial cliente de un Sistema de Compra Web visualizar un producto de su agrado en el mundo real, representado mediante un modelo 3D a escala real, utilizando tecnología de RA y la cámara de su dispositivo móvil.

En la página web que desarrollamos para simular un sitio de compras web, proporcionamos un conjunto de impresiones de códigos de los productos que un vendedor podría distribuir a sus posibles clientes. Esta página funciona como un Catálogo de Productos del vendedor.

También distribuimos a **FARVA** mediante la plataforma que nos ofrece Google Play Store, siendo nuestra aplicación gratuita y distribuida para todos los usuarios de Android de Argentina.

Durante el desarrollo de la aplicación encontramos una problemática bien definida, producida por las dificultades de hacer coincidir, de una forma coherente, las dimensiones de los modelos 3D con las imágenes del mundo real. Por esta razón es que utilizamos el reconocimiento mediante marcadores, los cuales deben tener un tamaño específico, obteniendo, de esta forma, un parámetro fijo de las dimensiones del mundo real, para poder escalar nuestros modelos 3D según esa referencia.

Por otra parte, si bien la detección de marcadores aplicada nos permite cumplir con los objetivos planteados, debemos destacar ciertas particularidades. En primer lugar, la obtención de la referencia se puede ver afectada por factores ajenos al correcto funcionamiento de la aplicación como, por ejemplo, la calidad de la cámara del dispositivo, la calidad de impresión del marcador y las características lumínicas de la escena capturada por la cámara.

En segundo lugar, el mayor inconveniente que nos encontramos estuvo dado por la capacidad de detección a la hora de representar modelos 3D de gran tamaño. Al captar marcadores dentro de la escena a una distancia corta no tuvimos mayores complicaciones, pero cuando quisimos representar objetos de mayor tamaño, nos alejábamos más del marcador que estábamos utilizando como referencia, lo que dificultaba su identificación. Entonces, para que la cámara del dispositivo pudiera reconocerlos, tuvimos que utilizar marcadores de mayor tamaño para los objetos 3D más grandes. Todo esto nos permite concluir que para un desarrollo en que

se necesite representar objetos de tamaño muy grande, como edificios o publicidades exteriores, el uso de marcadores impresos no es factible.

Para un un futuro desarrollo, donde las capacidades de detección e identificación de la escena a través del video capturado por la cámara del dispositivo haya mejorado, sería muy interesante utilizar el seguimiento sin marcadores. Esto permitiría que el usuario de la aplicación pueda desligarse de la tarea de imprimir un marcador para representar el objeto que desea.

Además, el uso del diseño 3D, para reproducir virtualmente los objetos deseados, abre un abanico de posibilidades casi ilimitadas de ampliación de la aplicación. Con un conocimiento avanzado de las herramientas de diseño 3D, se podría agregar cualquier tipo de información necesaria a la escena virtual e, incluso, hasta aplicar animaciones a los diferentes modelos 3D.

Si bien la aplicación está pensada actualmente para ser un complemento para la venta, también se podría pensar en extenderla a una aplicación más comercial, que contenga módulos de pago, stock y envío de productos, para que el cliente no sólo visualice los productos de su interés, sino que además los compre en ese mismo momento.

Todo el proceso seguido durante el transcurso de nuestra Tesina nos supuso un desafío, ya que nuestro plan de estudios de la Licenciatura en Informática no incluye materias específicas de programación de dispositivos móviles ni de RA y modelado 3D. El proyecto realizado nos permitió obtener un nuevo enfoque de estas tecnologías y nos brindó la posibilidad de utilizar y experimentar herramientas de las que no poseíamos conocimientos, pero que fueron elegidas por nosotros. Todo esto se constituyó en una experiencia única y gratificante para culminar nuestro ciclo de formación académica.

Bibliografía

[1]. Sirisha Jonnalagadda. *Android Application for Library Resource Access,* San Diego State University, 2012. Consultado 20-03-2017 http://scholarworks.calstate.edu/bitstream/handle/10211.10/3562/Jonnalagadda_Sirisha.pdf?sequence=1

[2]. Augmented reality in: Encyclopædia Britannica 2010. . Consultado 02-03-2017 http://www.britannica.com/EBchecked/topic/1196641/augmented-reality

[3]. Caudell, T.P. & Mizell, D.W. *Augmented reality: an application of heads-up display technology to manual manufacturing processes*, Proceedings of the 25th Hawaii International Conference on System Sciences, 1992.. Consultado 17-10-2016

[4]. Paul Milgram y Fumio Kishino. *A taxonomy of Mixed Reality Visual Displays*, IEICE Transactions on Information Systems E77-D, no.12, Diciembre 1994. Consultado 16-06-2017. https://cs.gmu.edu/~zduric/cs499/Readings/r76JBo-Milgram_IEICE_1994.pdf

[5]. Milgram, P., Takemura, H., Utsumi, A. & Kishino, F. *Augmented Reality: A Class of Displays on the Reality-Virtuality Continuum*, Proceedings of SPIE, Vol. 2351, Telemanipulator and Telepresence Technologies, Hari Das; Ed. 1994. Consultado 22-05-2017 http://etclab.mie.utoronto.ca/publication/1994/Milgram_Takemura_SPIE1994.pdf

[6]. Azuma, Ronald T. *A Survey of augmented reality. Presence: Teleoperators and Virtual Environments*, Malibu, 1997. . Consultado 20-07-2016 https://www.cs.unc.edu/~azuma/ARpresence.pdf

[7]. Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. & MacIntyre, B. *Recent advances in augmented reality*, IEEE Computer Graphics and Applications, 2001.

[8]. Mann, S. *Mediated Reality with implementations for everyday life*, Presence: Teleoperators and Virtual Environments, 2002. . Consultado 18-10-2016 http://wearcam.org/presence_connect/

[9]. Kipper, G., & Rampolla, J. *Augmented Reality: An Emerging Technologies Guide to AR*, Elsevier, 2012.

[10]. Minguell, M. E., Font, J. F., Conellas, P., & Regás. D. C. (2012). *Realidad*

aumentada y códigos QR en educación. En J. Hernández Ortega, M. Pennesi Fruscio, D. Sobrino López, & A. Vázquez Gutiérrez (Eds.), *Tendencias emergentes en Educación con TIC* (pp 135-157). Barcelona, Espiral, 2012. Consultado 20-03-2017 https://ciberespiral.org/tendencias/Tendencias_emergentes_en_educacin_con_TIC.pdf

[11]. Prendes Espinosa, C. *Realidad Aumentada y Educación: Análisis de Experiencias Prácticas*. Píxel-Bit. Revista de Medios y Educación, 46, 2015. Consultado 03-12-2016 http://www.redalyc.org/pdf/368/36832959008.pdf

[12]. Francisco Jurado,Javier. Desarrollo de Videojuegos: Desarrollo de Componentes, Universidad de Castilla la Mancha, 2012.

[13]. Juniper Research. *Mobile Augmented Reality: smartphones, tablets and smart glasses 2013-2018.* Consultado 03-11-2016. https://www.juniperresearch.com/researchstore/enabling-technologies/augmented-reality/augmented-reality-developer-vendor-strategies-2

[14]. Público.es, La Realidad Aumentada: última frontera entre lo físico y lo digital. Consultado 06-04-2017 http://www.publico.es/internacional/exito-realidad-aumentada.html

[15]. Zhou F, Duh HBL, Billinghurst M. Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. In Proceedings of ISMAR 2008. Cambridge, UK, 2008. Consultado 25-04-2017 https://pdfs.semanticscholar.org/d1ba/a7bf1dd81422c86954447b8ad570539f93be.pdf

[16]. David Marimon. *Advances in Top-Down and Bottom-Up Approaches to Video-Based Camera Tracking*. PhD thesis, École Polytechnique Fédérale de Lausanne, 2007.

[17]. Gilles, S., Fitzgibbon, A. W., and Zisserman, A. *Markerless Tracking using Planar Structures in the Scene*. In *Proc. International Symposium on Augmented Reality*, Octubre 2000. . Consultado 30-01-2017 https://www.robots.ox.ac.uk/~vgg/publications/2000/Simon00/simon00.pdf

[18]. Lima, J. Paulo, et al. *Model based markerless 3D tracking applied to augmented*

reality, Journal on 3D Interactive Systems 1, 2010. Consultado 17-02-2017. http://www.seer.ufrgs.br/index.php/jis/article/viewFile/16227/10038

[19]. Hesam Najafi, Nassir Navab, and Gudrun Klinker. *Automated initialization for marker-less tracking: A sensor fusion approach*. Mixed and Augmented Reality, IEEE / ACM International Symposiumon, 2004.

[20]. R. Koch, K. Koeser, and B. Streckel. *Markerless image based 3d tracking for real-time augmented reality applications*. In International Workshop of Image Analysis for Multimedia Interactive Services, 2005.

[21]. Okuma T., Kurata T., Sakaue K. *Fiducial-less 3-D Object Tracking in AR Systems Based on the Integration of Top-down and Bottom-up Approaches and Automatic Database Addition*, Columbia University, Washington, 2003. Consultado 01-03-2017 http://www.cs.columbia.edu/~okuma/pdf/ismar2003-okuma.pdf

[22]. Sanjuán Marimón D. *Advances in Top-down and Bottom-Up Approaches to video-based camera tracking*, Escuela Politécnica Federal de Lausanne, Suiza, 2007. Consultado 28-04-2017 https://infoscience.epfl.ch/record/112719/files/EPFL_TH3970.pdf

[23]. Kiriakos N. Kutulakos, James Vallino. *Affine Object Representations for Calibration-Free Augmented Reality*, Computer Science Department, University of Rochester, Rochester, NY, 1996. Consultado 13-09-2016 http://www.cs.toronto.edu/~kyros/pubs/96.vrais.pdf

[24]. C. Glez-Morcillo, D. Vallejo, J. Albusac, J.J. Castro-Schez. *Realidad Aumentada: Un Enfoque Práctico con ARToolKit y Blender*. Bubok Publishing. 1ª Edición, Julio 2012.

[25]. Sutherland. *A-Head Mounted Display Three Dimensional*, Actas de la Conferencia Fall Joint Computer, 1968.

[26].Ángel Pareja León. *Diseño, animación y visualización inmersiva de un Lamborghini Gallardo en una cueva de realidad virtual*. 2012

[27] Evguenii Kurmyshev. *Fundamentos de métodos matemáticos para física e Ingeniería.* Consultado 08-12-2016

https://books.google.com.ar/books?id=6sqm3nrOgnIC&pg=PA123&dq=definicion+matematica+de+curva&hl=es-419&sa=X&ved=0ahUKEwj3xoGyna_VAhWGIZAKHe2DADEQ6AEITDAH#v=onepage&q&f=false

[28] Curvas parametricas. Consultado 17-04-2017
http://www.cimec.org.ar/~ncalvo/curvas_doc.pdf

[29] Historia de la computación gráfica. Consultado 23-08-2017
http://blenderperu.org/historia-de-la-computacion-grafica-i/

[30] Modelado 3D. Consultado 06-09-2016
http://abc.mitreum.net/wp-content/uploads/clase2-parte1-teoria.pdf

[31] Principales hitos de la computación gráfica. Consultado 23-08-2017
http://fusmcompugrafica.blogspot.com.ar/2014/07/principales-hitos.html

[32] Tetera de Newell. Consultado 23-08-2017
http://www.neoteo.com/tetera-de-newell-hola-mundo-del-modelado-3d/

[33] Unity 3D página oficial. Consultado 04-09-2016
https://unity3d.com

[34] Android Studio Página oficial. Consultado 10-09-2017
https://developer.android.com/studio/intro/index.html?hl=es-419

[35] ARToolkit página oficial. Consultado 27-07-2016
https://artoolkit.org/

[36] Página oficial Vuforia. Consultado 15-10-2016
https://www.vuforia.com/

[37] Página oficial Blender. Consultado 07-11-2016
https://www.blender.org/

[38] Pagina oficial 3Ds Max. Consultado 05-04-2017
https://www.autodesk.com/education/free-software/3ds-max

[39] Android Developer. Consultado 15-01-2017

https://developer.android.com/guide/platform/index.html

Anexo I: Acuerdo de Distribución para Desarrolladores de Google Play

Definiciones

Operador Autorizado: se trata de un operador de red móvil que está autorizado para recibir una comisión por distribuir los Productos que se venden a los usuarios de Dispositivos en su red.
Características de la Marca: hace referencia a los nombres comerciales, las marcas comerciales, las marcas de servicio, los logotipos, los nombres de dominio y otras características de la marca distintivas de cada parte, respectivamente, según sean propiedad de dicha parte (o tengan su licencia) de forma ocasional.
Desarrollador: se trata de cualquier persona o empresa que está registrada en Google Play y tiene su aprobación para distribuir Productos de conformidad con las condiciones de este Acuerdo.
Cuenta de Desarrollador: hace referencia a una cuenta de publicación proporcionada a los Desarrolladores que permite distribuir Productos a través de Google Play.
Dispositivo: hace referencia a cualquier dispositivo que pueda acceder a Google Play, tal como se describe en este Acuerdo.
Google: se trata de Google Inc., corporación de Delaware con sede social en 1600 Amphitheatre Parkway, Mountain View, CA 94043, Estados Unidos; Google Ireland Limited, empresa constituida en Irlanda con sede social en Gordon House, Barrow Street, Dublín 4, Irlanda; Google Commerce Limited, empresa constituida en Irlanda con sede social en Gordon House, Barrow Street, Dublín 4, Irlanda; y Google Asia Pacific Pte. Limited, empresa constituida en Singapur con sede social en 70 Pasir Panjang Road, #03-71, Mapletree Business City, Singapore 117371.
Google Play: se trata del software y los servicios (como Play Console) que Google ha creado y dirige, lo que permite a los Desarrolladores registrados de determinados países distribuir Productos directamente a los usuarios de Dispositivos.
Cuenta de Pago: hace referencia a una cuenta financiera proporcionada por un Procesador de Pagos a un Desarrollador que autoriza al Procesador de Pagos a cobrar y remitir pagos en nombre del Desarrollador por los Productos vendidos a través de Google Play. Los Desarrolladores deben contar con la aprobación de un Procesador de Pagos para tener una Cuenta de Pago y mantener su cuenta en buen estado con el fin de efectuar cargos por los Productos distribuidos a través de Google Play.
Procesadores de Pagos: tal como se especifica y se designa en las Políticas del Programa para Desarrolladores, se trata de una parte autorizada por Google para proporcionar servicios que permitan que los Desarrolladores con Cuentas de Pago efectúen cargos a los usuarios por los Productos distribuidos a través de Google Play.

Play Console: se trata de Google Play Console y de otros servicios o herramientas online que Google proporciona a los Desarrolladores para administrar la distribución de Productos y otras funciones relacionadas.
Productos: hace referencia a software, contenido y material digital distribuidos a través de Google Play.

1. Introducción

1.1 Los Desarrolladores pueden distribuir Productos a los Dispositivos mediante Google Play. Para ello, deben obtener y mantener una Cuenta de Desarrollador válida.
1.2. Si el Desarrollador quiere cobrar una comisión por sus Productos, también debe obtener y mantener una Cuenta de Pago válida proporcionada a través de un Procesador de Pagos autorizado.

2. Aceptación de este Acuerdo

2.1 Este acuerdo (en adelante, el "Acuerdo") constituye un contrato legalmente vinculante entre el Desarrollador y Google en lo que respecta al uso que haga el Desarrollador de Google Play para distribuir Productos. El Desarrollador acepta que Google, únicamente en su nombre y no en nombre de Google, muestre los Productos y los ponga a disposición de los usuarios para que puedan verlos, descargarlos y comprarlos. Para utilizar Google Play y distribuir Productos, el Desarrollador debe aceptar este Acuerdo y proporcionar información completa y exacta en Play Console. El Desarrollador no podrá distribuir Productos a través de Google Play si no acepta este Acuerdo.
2.2 El Desarrollador no podrá utilizar Google Play para distribuir Productos y no podrá aceptar el Acuerdo a menos que se haya verificado como Desarrollador con una cuenta en buen estado. Este Acuerdo finalizará automáticamente si el Desarrollador: a) no tiene una cuenta en buen estado; o b) es una persona o entidad a la que se le haya prohibido utilizar el software de Android en virtud de la legislación de Estados Unidos o de otros países, incluido su país de residencia o el país desde el que utilice el software de Android.
2.3. Si el Desarrollador acepta este Acuerdo en nombre de su empresa o de otra entidad, manifiesta y garantiza que dispone de plena autoridad legal para vincular a dicha empresa o entidad a este Acuerdo. Si no dispone de la autoridad necesaria, no podrá aceptar el Acuerdo ni utilizar Google Play en nombre de su empresa ni de otra entidad.

3. Precios y pagos

3.1. Este Acuerdo cubre tanto los Productos que el Desarrollador quiera distribuir de forma gratuita como aquellos por los que desee cobrar algún importe. Para cobrar por los Productos, el Desarrollador debe disponer de una Cuenta de Pago válida conforme a un acuerdo independiente

con un Procesador de Pagos. Si el Desarrollador ya dispone de una Cuenta de Pago con un Procesador de Pagos antes de registrarse como Desarrollador en Google Play, se aplicarán las condiciones de ese acuerdo salvo que se produzca algún conflicto con el presente documento (en cuyo caso se aplicarán las condiciones de este Acuerdo).
3.2. Los Productos se muestran a los usuarios en nombre del Desarrollador con los precios que este establezca según considere oportuno. Google podrá incluir impuestos aplicables en el precio cobrado a los usuarios en Google Play. El Desarrollador podrá establecer el precio de sus Productos en las monedas permitidas por el Procesador de Pagos. Google podrá mostrar el precio de los Productos a los usuarios en su moneda local, pero no se responsabilizará de la exactitud de los tipos de cambio ni de la conversión de monedas.
3.3 El Desarrollador es el vendedor oficial de los Productos que ofrece a través de Google Play. Para cada transacción, el contrato se establecerá con la entidad de Google aplicable según dónde se vaya a distribuir el Producto (de acuerdo con lo establecido en esta página). El precio que establezca para los Productos determinará la cantidad que recibirá en el pago. Se añadirá una Comisión de Transacción (tal como se define a continuación) al precio de venta, que se repartirá proporcionalmente con el Procesador de Pagos y con el Operador Autorizado, si lo hay. Si la legislación aplicable local requiere que Google, el Procesador de Pagos o el Operador Autorizado retengan impuestos (en adelante, "Retención de Impuestos") en los pagos realizados o recibidos por alguno de ellos, Google también deducirá una cantidad igual a la de la Retención de Impuestos del precio de venta. A efectos de clarificación, la Retención de Impuestos incluye, entre otros, la obligación fiscal de retener impuestos en los pagos transfronterizos o las obligaciones que imponen los impuestos sobre telecomunicaciones. El Desarrollador recibirá el importe restante (es decir, el precio de venta menos la Comisión de Transacción y menos el importe equivalente a la Retención de Impuestos). La "Comisión de Transacción" se indica en esta página y Google puede revisarla cada cierto tiempo. El Desarrollador es responsable de proporcionar a Google cualquier certificado de residencia fiscal aplicable. Si Google o su proveedor de servicios no reciben dicha documentación, Google aplicará la retención según el tipo impositivo nacional.
3.4. El Desarrollador es responsable de determinar si un Producto está sujeto a impuestos y debe conocer el tipo impositivo aplicable que el Procesador de Pagos debe retener en cada jurisdicción fiscal en la que se vendan los Productos. El Desarrollador debe abonar estos impuestos a la autoridad fiscal correspondiente. Si la legislación local aplicable requiere que Google, el Procesador de Pagos o el Operador Autorizado determinen, apliquen y paguen el tipo impositivo aplicable, Google, el Procesador de Pagos o el Operador Autorizado (no el Desarrollador) serán responsables de aplicar y retener los impuestos, además de abonarlos a la autoridad fiscal pertinente. Si Google retiene y abona impuestos sobre el valor añadido en los pagos de los clientes (si la legislación local aplicable así lo requiere) y dicho abono cumple los requisitos aplicables para los impuestos sobre el valor añadido en los pagos de los clientes, Google no aplicará esos impuestos al Desarrollador. Si Google debe retener y abonar impuestos tal como se indica en esta sección, el Desarrollador y Google aceptarán una cantidad del Desarrollador para

Google con fines fiscales. Asimismo, el Desarrollador cumplirá las obligaciones fiscales pertinentes que se deriven de esta cantidad.

3.5. El Desarrollador podrá distribuir los Productos de forma gratuita. Si el Producto es gratuito, quedará exento del pago de la Comisión de Transacción. El Desarrollador no podrá empezar a aplicar cargos a un usuario por un Producto que inicialmente era gratuito a menos que el cargo esté relacionado con una versión alternativa del Producto. El Procesador de Pagos debe procesar todas las comisiones que reciba un Desarrollador por cualquier versión de un Producto distribuida a través de Google Play.

3.6 **El Desarrollador ebe proporcionar asistencia relacionada con su Producto**. Se indica a los compradores que se pongan en contacto con el Desarrollador si detectan defectos o problemas de rendimiento en las aplicaciones descargadas, instaladas o a las que se ha accedido desde Google Play, a las que se puede acceder a través de esta plataforma. El Desarrollador será el único responsable y Google estará exento de cualquier responsabilidad, de llevar a cabo o administrar la asistencia y el mantenimiento de sus Productos, así como de las quejas sobre estos. El Desarrollador debe proporcionar y mantener información de contacto válida y exacta, que se mostrará en la página de detalles de cada aplicación en Google Play, y ponerla a disposición de los usuarios con fines legales y de atención al cliente. En el caso de los Productos de pago o de las transacciones en aplicaciones, el Desarrollador deberá responder a las consultas de los clientes sobre asistencia en un plazo de tres (3) días laborables, que se reducirá a 24 horas si Google considera que los problemas están relacionados con la asistencia o un Producto y tienen carácter urgente. Si no se proporciona información o asistencia adecuadas en relación con los Productos del Desarrollador, esto puede dar lugar a valoraciones bajas del Producto, una visibilidad menos destacada de este, un descenso de sus ventas, conflictos de facturación o la retirada del Producto de Google Play.

3.7. **Autoridad para emitir reembolsos**. El Desarrollador autoriza a Google a proporcionar al comprador un reembolso completo del precio de un Producto o de una transacción en la aplicación en su nombre si el comprador solicita el reembolso en cualquier momento después de realizar la compra. En todos los demás aspectos relacionados con los reembolsos, se aplicarán los términos y condiciones de uso estándar del Procesador de Pagos. Los reembolsos de los usuarios pueden no incluir los impuestos que se les hayan cobrado anteriormente por la compra del Producto. Salvo en aquellos casos en los que el usuario haya iniciado varios procesos de reclamación, se podrá realizar automáticamente en la cuenta del Desarrollador el cargo correspondiente a las reclamaciones relacionadas con la facturación de Productos vendidos por menos de 10 USD, así como a cualquier comisión de procesamiento cobrada por el Procesador de Pagos (a menos que Google determine que el usuario que ha presentado la reclamación tiene un historial de reclamaciones anormal). Las solicitudes de contracargo relacionadas con Productos de precio igual o superior a 10 USD se gestionarán de acuerdo con la política estándar del Procesador de Pagos.

3.8 **Reinstalaciones**. Los usuarios pueden volver a instalar de forma ilimitada cada Producto instalado previamente siempre que, en caso de que el Desarrollador retire un Producto de Google

Play en virtud de lo establecido en las cláusulas i), ii), iii) o iv) de la sección 7.1, dicho Producto se retire por completo de Google Play y los usuarios pierdan el derecho o la posibilidad de volver a instalarlo.

4. Uso de Google Play por parte del Desarrollador

4.1. A excepción de los derechos de licencia otorgados por el Desarrollador en virtud de la sección 5 del presente Acuerdo, Google acepta que no obtiene ningún derecho, título ni interés por parte del Desarrollador (o de sus proveedores de licencias) en virtud de este Acuerdo en relación con los Productos, incluido cualquier derecho de propiedad intelectual que tengan dichos Productos.

4.2 El Desarrollador acepta utilizar Google Play solo para los fines permitidos por a) este Acuerdo; y b) cualquier ley, normativa, directriz o práctica generalmente aceptada que sea aplicable y que se incluya en las jurisdicciones correspondientes (incluida cualquier ley relacionada con la exportación de datos o de software desde o hacia Estados Unidos u otros países pertinentes).

4.3 El Desarrollador acepta que, si utiliza Google Play para distribuir Productos, protegerá los derechos legales y la privacidad de los usuarios. Si los usuarios proporcionan al Desarrollador nombres de usuario, contraseñas u otra información personal o de inicio de sesión, o bien su Producto accede a estos datos o los utiliza, el Desarrollador debe informar a los usuarios de que esa información estará disponible para su Producto y proporcionarles un aviso de privacidad legalmente válido, así como la protección correspondiente. Asimismo, el Producto del Desarrollador solo podrá utilizar esta información para los fines específicos para los que el usuario haya concedido su permiso. Si el Producto del Desarrollador almacena información personal o confidencial proporcionada por los usuarios, debe hacerlo de un modo suficientemente seguro y únicamente durante el tiempo necesario. No obstante, si el usuario ha suscrito un acuerdo independiente con el Desarrollador que permite que el Desarrollador o su Producto almacenen o utilicen información personal o confidencial directamente relacionada con su Producto (sin incluir otros productos o aplicaciones), el uso de dicha información se regirá por las condiciones de ese acuerdo independiente. Si el usuario proporciona al Producto del Desarrollador información sobre cuentas de Google, dicho Producto solo podrá utilizar esa información para acceder a la cuenta de Google del usuario en el momento para el que el usuario haya concedido su permiso y con los fines limitados para los que haya otorgado dicha autorización.

4.4 **Acciones prohibidas**. El Desarrollador acepta no participar en ninguna actividad con Google Play, incluido el desarrollo o la distribución de Productos, que pueda suponer una interferencia, una alteración, un daño o un acceso no autorizado a dispositivos, servidores, redes u otro tipo de propiedades o servicios de terceros, incluidos los usuarios de Android, de Google o de cualquier operador de redes móviles, entre otros. El Desarrollador no puede utilizar la información del cliente obtenida de Google Play para vender o distribuir productos fuera de Google Play.

4.5 **Tiendas alternativas**. El Desarrollador no podrá utilizar Google Play para distribuir o poner a disposición de los usuarios ningún Producto cuya finalidad principal sea facilitar la distribución de juegos y aplicaciones de software para dispositivos Android fuera de Google Play.
4.6. El Desarrollador acepta ser el único responsable de cualquier Producto que distribuya a través de Google Play, incluido el uso de cualquier API de Google Play, y que Google no tiene ninguna responsabilidad ante él ni ante terceros con respecto a dicho Producto y de las consecuencias de sus acciones, incluidos cualquier daño o pérdida que Google pueda sufrir. Estas consecuencias incluyen, entre otras, la responsabilidad del producto, la protección del consumidor o las reclamaciones basadas en la propiedad intelectual relacionados con los Productos.
4.7. El Desarrollador acepta ser el único responsable de cualquier infracción de las obligaciones establecidas en este Acuerdo, de cualquier condición del servicio o contrato con un tercero aplicable o de cualquier normativa o ley aplicable, así como de las consecuencias derivadas de dicha infracción, incluidos cualquier pérdida o daño que pueda sufrir Google, y que Google no tiene ninguna responsabilidad ante él ni ante terceros relacionada con dicha infracción.
4.8 **Valoraciones de Productos**. Google Play permitirá que los usuarios valoren determinados Productos y den su opinión sobre ellos. Solo los usuarios que descarguen el Producto aplicable podrán valorarlo y dar su opinión sobre él en Google Play. Las valoraciones de Productos podrán utilizarse para determinar la ubicación de los Productos en Google Play, en función de la capacidad de Google de cambiar la posición según considere oportuno. Google Play podrá asignar al Desarrollador una puntuación combinada para cualquier Producto que no haya recibido valoraciones de usuarios. La puntuación combinada de un Desarrollador será una representación de la calidad de su Producto en función de su historial y la determinará Google según considere oportuno. En el caso de nuevos Desarrolladores sin historial de Productos, Google podrá utilizar o publicar cifras de rendimiento como, por ejemplo, el porcentaje de reembolsos o desinstalaciones, para identificar o eliminar Productos que no reúnan los estándares aceptables, según su criterio. Google se reserva el derecho de mostrar Productos a los usuarios del modo que considere oportuno.
Los Productos podrán estar sujetos a valoraciones de usuarios con las que el Desarrollador pueda no estar de acuerdo. El Desarrollador podrá ponerse en contacto con Google si tiene preguntas o dudas relacionadas con esas valoraciones.
4.9 **Marketing del Producto**. El Desarrollador será responsable de subir sus Productos a Google Play, proporcionar la información necesaria sobre el Producto y la asistencia a los usuarios, y comunicar con exactitud los permisos de seguridad necesarios para que el Producto funcione en los Dispositivos de los usuarios. No se publicarán los Productos que no se suban según lo establecido en esta cláusula.
4.10 **Contenido restringido**. Cualquier Producto que el Desarrollador distribuya a través de Google Play debe cumplir las Políticas del Programa para Desarrolladores.

5. Concesiones de licencias

5.1 El desarrollador concede a Google una licencia no exclusiva, mundial y libre de derechos de autor para reproducir, ejecutar, mostrar, analizar y utilizar los Productos en relación con i) el funcionamiento y el marketing de Google Play; ii) el marketing de dispositivos y servicios que admitan el uso de los Productos; iii) la realización de mejoras en Google Play y en la plataforma Android; y iv) la comprobación del cumplimiento de este Acuerdo y de las Políticas del Programa para Desarrolladores.
5.2 El Desarrollador concede a Google una licencia no exclusiva y libre de derechos de autor para distribuir los Productos de la forma indicada en Play Console.
5.3. Google podrá recurrir a consultores y a otros contratistas con relación a la aplicación de las obligaciones y al ejercicio de los derechos establecidos en este Acuerdo siempre que dichos consultores y contratistas estén sujetos a las mismas obligaciones que Google. Tras la resolución de este Acuerdo, Google no distribuirá el Producto del Desarrollador, pero podrá conservar y utilizar copias de este para proporcionar servicios de asistencia relacionados con Google Play y la plataforma Android.
5.4. El Desarrollador concede al usuario una licencia no exclusiva, mundial y perpetua para ejecutar, mostrar y utilizar el Producto en el Dispositivo. El usuario puede incluir, entre otros, un grupo familiar con un administrador y miembros de la familia cuyas cuentas estén unidas con el propósito de crear dicho grupo. Los grupos familiares de Google Play están sujetos a límites razonables para evitar el uso inadecuado de las funciones para compartir con la familia. Los usuarios que pertenezcan a un grupo familiar pueden comprar una sola copia del Producto y compartirla con otros miembros del grupo (excepto en el caso de las suscripciones y los Productos en la aplicación, que no se pueden compartir). Si el Desarrollador habilita en Play Console la opción para que los usuarios puedan compartir los Productos comprados anteriormente, esta autorización estará sujeta al presente Acuerdo. Si el Desarrollador lo desea, puede incluir un acuerdo de licencia de usuario final (EULA) independiente en su Producto para que rija los derechos de sus usuarios. No obstante, si se produce algún conflicto con el EULA, este Acuerdo lo sustituirá.
5.5. El Desarrollador manifiesta y garantiza que dispone de todos los derechos de propiedad intelectual, incluyendo las patentes, las marcas, los secretos comerciales, los derechos de autor u otros derechos de propiedad necesarios relacionados con el Producto. Si utiliza materiales de terceros, el Desarrollador manifiesta y garantiza que dispone del derecho para distribuir dicho material en el Producto. El Desarrollador acepta que no enviará material a Google Play que tenga derechos de autor, esté protegido por secretos comerciales o esté sujeto de cualquier otra forma a derechos de propiedad de terceros, como patentes o derechos de publicidad y de privacidad, a menos que sea el propietario de dichos derechos o tenga permiso del propietario legítimo para enviar el material.

6. Publicidad y Características de la Marca

6.1. Cada tercero debe ser propietario de todos los derechos, títulos e intereses, incluidos, sin limitarse a ello, todos los derechos de propiedad intelectual relacionados con sus Características de Marca. Excepto en la medida en que quede expresamente estipulado en este Acuerdo, ningún tercero concede ni debe adquirir ningún derecho, título o interés (incluida, sin limitarse a ello, cualquier licencia implícita) relacionado con cualquier Característica de la Marca de otro tercero. Según las condiciones de este Acuerdo, el Desarrollador concede a Google y a sus afiliados una licencia no exclusiva y libre de derechos de autor durante el periodo de vigencia de este Acuerdo para mostrar las Características de Marca del Desarrollador enviadas por este a Google exclusivamente para su uso online o en dispositivos móviles y, en cualquier caso, únicamente en relación con la distribución y venta del Producto del Desarrollador a través de Google Play o para cumplir de otra forma sus obligaciones en virtud de este Acuerdo. Si el Desarrollador interrumpe la distribución de Productos específicos a través de Google Play, Google dejará de utilizar las Características de Marca de dichos Productos según lo establecido en esta Sección 6.1 en la medida necesaria para poder ejecutar las disposiciones de la Sección 3.8. Ninguna de las disposiciones de este Acuerdo otorga al Desarrollador el derecho a utilizar los nombres comerciales, las marcas, las marcas comerciales del servicio, los logotipos, los nombres de dominio u otras características de marca distintivas de Google.

6.2 **Derechos de publicidad**. Sin perjuicio de la licencia concedida en la sección 6.1, Google y sus afiliados pueden incluir Características de Marca del Desarrollador que este haya enviado a Google con fines comerciales relacionados con la presencia, la distribución y la venta del Producto del Desarrollador a través de Google Play y con su disponibilidad para utilizarlo en dispositivos y a través de otros servicios de Google en: i) Google Play y en cualquier propiedad móvil u online de Google; ii) en publicidad en formato impreso, televisión, lugares públicos (p. ej., vallas publicitarias), online o para móviles fuera de Google Play cuando se mencionen junto con otros Productos de Google Play; iii) en anuncios relacionados con la disponibilidad del Producto; iv) en presentaciones y v) en listas de clientes que aparezcan online o en dispositivos móviles (incluidas, sin limitarse a ello, las listas de clientes publicadas en los sitios web de Google). Si el Desarrollador interrumpe la distribución de Productos específicos a través de Google Play, Google dejará de utilizar las Características de Marca de dichos Productos con fines comerciales. Google concede al Desarrollador una licencia limitada, no exclusiva, mundial y libre de derechos de autor para utilizar las Características de Marca de Android durante el periodo de vigencia de este Acuerdo únicamente con fines comerciales y solo de acuerdo con las directrices de marca de Android.

7. Retirada de productos

7.1 **Retiradas del Desarrollador**. El Desarrollador puede retirar sus Productos en cualquier momento para que no vuelvan a distribuirse a través de Google Play. Para ello, debe cumplir las disposiciones de este Acuerdo y las condiciones de servicio de la Cuenta de Pago del Procesador de Pagos en relación con cualquier Producto distribuido a través de Google Play, incluidos los

requisitos de reembolso, entre otros. La retirada de los Productos por parte del Desarrollador para que no vuelvan a distribuirse a través de Google Play: a) no afecta a los derechos de licencia de los usuarios que hayan adquirido o descargado sus Productos con anterioridad; b) no elimina sus Productos de los Dispositivos ni de ninguna otra ubicación de Google Play en la que estuvieran almacenadas en nombre de los usuarios las aplicaciones adquiridas o descargadas previamente; y c) no modifica la obligación del Desarrollador de suministrar los Productos o servicios descargados o adquiridos previamente por los usuarios y de proporcionar asistencia relacionada con estos. Sin perjuicio de lo anterior, Google no mantendrá en ninguna sección de Google Play (incluida, sin limitarse a ella, la sección en la que se almacenan las aplicaciones adquiridas o descargadas con anterioridad en nombre de los usuarios) ningún Producto que el Desarrollador haya retirado de Google Play enviando una notificación por escrito a Google para comunicarle que la retirada se ha debido a i) una acusación de infracción o una infracción real de derechos de autor, marca, secreto comercial, imagen comercial, patente u otro derecho de propiedad intelectual de cualquier persona, ii) una acusación de difamación o difamación real, iii) una acusación de infracción o infracción real de los derechos de privacidad o de publicidad de terceros o iv) una acusación o resolución de que dicho Producto no cumple la legislación aplicable.

Si el Desarrollador retira un Producto de Google Play según lo estipulado en las cláusulas i), ii), iii) o iv) de esta Sección 7.1 y un usuario final ha comprado dicho Producto en el último año antes de la fecha de la retirada y Google se lo solicita, el Desarrollador deberá reembolsar a dicho usuario final la cantidad abonada por ese Producto menos la parte de la Comisión de Transacción destinada específicamente al procesamiento del pago o de la tarjeta de crédito de la transacción asociada.

7.2 **Retiradas de Google**. Aunque Google no asume la obligación de supervisar los Productos ni su contenido, sí puede retirar un Producto de Google Play o volver a clasificarlo en cualquier momento si recibe un aviso por parte del Desarrollador o llega a su conocimiento de cualquier otro modo y determina según su criterio que un Producto o cualquier parte de este o de las Características de Marca: a) infringe los derechos de propiedad intelectual o cualquier otro derecho de un tercero; b) infringe cualquier ley aplicable o está sujeto a algún mandato judicial; c) incluye contenido pornográfico u obsceno o infringe de cualquier otro modo las políticas de alojamiento de Google o cualquier otra condición de servicio de acuerdo con las actualizaciones ocasionales que pueda realizar Google; d) se está distribuyendo de forma inadecuada por parte del Desarrollador; e) puede generar alguna responsabilidad para Google o los Operadores Autorizados; f) contiene virus, software malicioso o software espía o afecta a la red de Google o de un Operador Autorizado en opinión de Google; g) infringe las condiciones de este Acuerdo o las Políticas del programa para Desarrolladores; o que h) la presentación del Producto afecta a la integridad de los servidores de Google (es decir, los usuarios no pueden acceder a ese contenido o experimentan otro tipo de dificultades). Google se reserva el derecho de suspender o prohibir el uso de Google Play por parte de cualquier Desarrollador si lo considera oportuno. Si el Producto del Desarrollador contiene elementos que podrían causar daños graves en los datos o en los

dispositivos de los usuarios, Google puede inhabilitar el Producto o retirarlo de los dispositivos en los que se haya instalado si lo considera oportuno. Google puede suspender o cancelar la distribución de los Productos del Desarrollador si este incurre en un incumplimiento sustancial de las condiciones de algún acuerdo de confidencialidad o de cualquier otro acuerdo referente a Google Play o a la plataforma Android.
Google dispone de acuerdos de distribución con fabricantes de dispositivos y Operadores Autorizados para incluir las aplicaciones de software cliente de Google Play en los Dispositivos. Es posible que estos acuerdos de distribución soliciten la retirada involuntaria de Productos que infrinjan las condiciones de servicio del Operador Autorizado o del fabricante del Dispositivo.
Si el Producto de un Desarrollador se retira de forma involuntaria por ser defectuoso o dañino, infringir los derechos de propiedad intelectual de otra persona, ser difamatorio, infringir los derechos de publicidad o de privacidad de terceros o incumplir la legislación aplicable y un usuario final ha comprado ese Producto dentro de un plazo de un año antes de la fecha de retirada, i) el Desarrollador deberá reembolsar a Google todas las cantidades percibidas, junto con las comisiones asociadas (es decir, las comisiones de transacción de pagos y devoluciones) y ii) Google podrá deducir de sus ventas futuras la cantidad indicada en la subsección i) anterior si lo considera oportuno.

8. Tus credenciales de Desarrollador

8.1. El Desarrollador acepta ser responsable del mantenimiento de la confidencialidad de las credenciales de Desarrollador que Google pueda proporcionarle o que él mismo haya elegido y que será el único responsable de todos los Productos desarrollados con esas credenciales de Desarrollador. Google puede limitar el número de Cuentas de Desarrollador que se enviarán al Desarrollador o a la empresa u organización para la que trabaje.

9. Privacidad e información

9.1 Para introducir innovaciones y mejoras continuas en Google Play, Google puede recopilar determinadas estadísticas de uso de Google Play y de los Dispositivos, entre los que se incluyen, sin limitarse a ello, datos sobre la forma de utilizar Google Play y los Dispositivos.
9.2 Los datos recogidos se examinarán en su conjunto para realizar las mejoras necesarias en Google Play para los usuarios y los Desarrolladores, y se conservarán de acuerdo con la política de privacidad de Google. Con el fin de garantizar la mejora de los Productos, el Desarrollador puede solicitar por escrito que se le faciliten determinados datos totales.

10. Resolución de este Acuerdo

10.1. Este Acuerdo se aplicará hasta su resolución por parte de Google o del Desarrollador tal como se establece a continuación.

10.2. Si el Desarrollador desea rescindir este Acuerdo, deberá proporcionar a Google una notificación por escrito con treinta (30) días de antelación (salvo que la rescisión del Acuerdo se produzca en virtud de lo estipulado en la Sección 14.1) y dejar de utilizar las credenciales de Desarrollador pertinentes.
10.3. Google puede resolver este Acuerdo en cualquier momento si:
A) el Desarrollador infringe alguna disposición del Acuerdo,
B) la ley se lo exige,
C) el Desarrollador deja de ser un Desarrollador autorizado,
D) Google decide dejar de proporcionar el servicio de Google Play.

11. RENUNCIA DE GARANTÍAS

11.1 EL DESARROLLADOR COMPRENDE Y ACEPTA EXPRESAMENTE QUE ES EL ÚNICO RESPONSABLE DEL USO QUE HAGA DE GOOGLE PLAY Y QUE GOOGLE PAY SE PROPORCIONA "TAL CUAL" Y "SEGÚN DISPONIBILIDAD" SIN GARANTÍA DE NINGÚN TIPO.
11.2 EL DESARROLLADOR UTILIZARÁ GOOGLE PLAY Y CUALQUIER MATERIAL DESCARGADO U OBTENIDO DE OTRO MODO A TRAVÉS DEL USO DE GOOGLE PLAY POR SU CUENTA Y RIESGO, Y SERÁ EL ÚNICO RESPONSABLE DE CUALQUIER DAÑO QUE SE PRODUZCA EN SU SISTEMA INFORMÁTICO O EN CUALQUIER OTRO DISPOSITIVO Y DE CUALQUIER PÉRDIDA DE DATOS QUE SE DERIVE DE DICHO USO.
11.3. GOOGLE RENUNCIA EXPRESAMENTE A TODA GARANTÍA Y CONDICIÓN DE CUALQUIER TIPO, IMPLÍCITA O EXPLÍCITA, INCLUIDAS, ENTRE OTRAS, LAS GARANTÍAS Y CONDICIONES IMPLÍCITAS DE COMERCIABILIDAD, IDONEIDAD PARA UN FIN DETERMINADO Y NO VULNERACIÓN.

12. LIMITACIÓN DE RESPONSABILIDAD

12.1. EL DESARROLLADOR COMPRENDE Y ACEPTA EXPRESAMENTE QUE GOOGLE Y SUS EMPRESAS AFILIADAS Y SUBSIDIARIAS NO SERÁN RESPONSABLES ANTE ÉL DE ACUERDO CON NINGÚN PRINCIPIO DE RESPONSABILIDAD DE NINGÚN DAÑO DIRECTO, INDIRECTO, IMPREVISTO, ESPECIAL, DERIVADO O EJEMPLAR QUE EL DESARROLLADOR PUEDA OCASIONAR, INCLUIDA LA PÉRDIDA DE DATOS, INDEPENDIENTEMENTE DE QUE GOOGLE O SUS REPRESENTANTES DEBAN HABER TENIDO CONOCIMIENTO O HAYAN SIDO INFORMADOS DE LA POSIBILIDAD DE QUE SE PRODUJERAN TALES PÉRDIDAS.

13. Indemnización

13.1 En el sentido más amplio permitido por la ley, el Desarrollador acepta indemnizar, proteger y eximir de toda responsabilidad a Google, a sus empresas afiliadas y a sus respectivos directores, ejecutivos, empleados, agentes y Operadores Autorizados ante cualquier reclamación, acción, demanda o proceso judicial de terceros, o cualquier pérdida, obligación, daño, coste o gasto (incluidos los honorarios de abogados considerados razonables) que se derive de a) un uso por parte del Desarrollador de Google Play que constituya una infracción de este Acuerdo; o de b) la existencia de un Producto del Desarrollador que infrinja cualquier derecho de autor, marca, secreto comercial, imagen comercial, patente u otro derecho de propiedad intelectual de cualquier persona o que difame a cualquier persona o infrinja sus derechos de publicidad o privacidad.
13.2 En el sentido más amplio permitido por la ley, el Desarrollador acepta defender, indemnizar y eximir de toda responsabilidad a los Procesadores de Pagos correspondientes (incluidos Google y terceros) y a sus afiliados, directores, ejecutivos, empleados y agentes ante cualquier reclamación, acción, demanda o proceso judicial de terceros, o cualquier pérdida, obligación, daño, coste o gasto (incluidos los honorarios de abogados considerados razonables) que se derive de la aplicación de impuestos relacionados con la distribución de Productos a través de Google Play por parte del Desarrollador.

14. Cambios en el Acuerdo

14.1. Google puede realizar cambios en este Acuerdo en cualquier momento enviando al Desarrollador una notificación por correo electrónico en la que se describan las modificaciones realizadas. Además, Google publicará una notificación en esta página o en Play Console con una descripción de las modificaciones realizadas. El Desarrollador deberá consultar el Acuerdo y buscar avisos de cambios regularmente. Los cambios no se aplicarán de forma retroactiva. Tales cambios entrarán en vigor y se considerarán aceptados por el Desarrollador: a) de forma inmediata en el caso de quienes se hayan convertido en Desarrolladores después de la publicación de la notificación; o b) para aquellos que ya sean Desarrolladores, en la fecha especificada en la notificación, teniendo en cuenta que deberán haber transcurrido como mínimo 30 días tras la publicación de los cambios (a menos que se trate de cambios realizados por imperativo legal, que entrarán en vigor de forma inmediata). Si el Desarrollador no acepta las modificaciones del Acuerdo, podrá ejercer su legítimo derecho a dejar de utilizar Google Play. El Desarrollador acepta que el uso continuado de Google Play constituye la aceptación de las condiciones modificadas de este Acuerdo.

15. Condiciones legales generales

15.1 Este Acuerdo constituye la totalidad del acuerdo legal suscrito entre el Desarrollador y Google, rige el uso de Google Play y sustituye completamente cualquier acuerdo anterior relacionado con Google Play entre las dos partes.

15.2. El Desarrollador acuerda que, en caso de que Google no ejerza ni ejecute ningún derecho legal o medida que se incluya en este Acuerdo (o que Google haya disfrutado según las leyes aplicables), el no ejercicio de tales derechos o medidas no se considerará una renuncia formal de los derechos de Google y que Google podrán seguir disfrutando de tales derechos o medidas.
15.3. Si cualquier tribunal que tenga la jurisdicción de decidir sobre este asunto dictamina que alguna disposición de estas Condiciones no es válida, esta se eliminará del Acuerdo sin que ello afecte al resto del Acuerdo. Las restantes disposiciones del Acuerdo seguirán considerándose válidas y de obligado cumplimiento.
15.4. El Desarrollador reconoce y acepta que cada uno de los miembros del grupo de empresas filiales de Google será beneficiario externo de este Acuerdo y que dichas empresas estarán plenamente autorizadas a ejecutar o reclamar cualquier disposición de este Acuerdo que les confiera algún beneficio o derecho. A excepción de estos miembros, ninguna otra persona ni empresa será beneficiaria externa de este Acuerdo.
15.5 RESTRICCIONES DE EXPORTACIÓN. LOS PRODUCTOS DISTRIBUIDOS A TRAVÉS DE GOOGLE PLAY PUEDEN ESTAR SUJETOS A LAS REGULACIONES Y LEYES DE EXPORTACIÓN DE ESTADOS UNIDOS. EL DESARROLLADOR DEBE CUMPLIR TODAS LAS REGULACIONES Y LEYES DE EXPORTACIÓN NACIONALES E INTERNACIONALES APLICABLES A LA DISTRIBUCIÓN O AL USO DE LOS PRODUCTOS. ESTAS LEYES INCLUYEN RESTRICCIONES SOBRE DESTINOS, USUARIOS Y USO FINAL.
15.6. Ni el Desarrollador ni Google pueden ceder ni transferir los derechos concedidos en virtud de este Acuerdo sin aprobación previa por escrito de la otra parte. Ni el Desarrollador ni Google podrán delegar las responsabilidades y obligaciones contraídas en virtud de este Acuerdo sin aprobación previa por escrito de la otra parte. Cualquier otro intento de cesión se considerará nulo. Si se produce un cambio de control (por ejemplo, mediante una operación de compra o venta de acciones, una fusión u otra forma de transacción empresarial): a) el Desarrollador se lo notificará a Google por escrito dentro de los 30 días siguientes al cambio de control y b) Google podrá rescindir inmediatamente el Acuerdo en cualquier momento dentro de los 30 días siguientes a la recepción de la notificación por escrito de cambio de control.
15.7. Todas las reclamaciones derivadas de este Acuerdo o de la relación del Desarrollador con Google en virtud de este o relacionadas con el mencionado Acuerdo o con dicha relación se regirán por las leyes del Estado de California excluyendo las disposiciones en materia de conflicto de leyes de California. El Desarrollador y Google aceptan además remitir a la jurisdicción exclusiva de los tribunales federales o estatales ubicados en el condado de Santa Clara (California) la resolución de cualquier cuestión legal derivada de este Acuerdo o de la relación del Desarrollador con Google en virtud de este o relacionada con el mencionado Acuerdo o con dicha relación a menos que el Desarrollador acepte que Google pueda solicitar la aplicación de medidas cautelares en cualquier jurisdicción.

15.8. Las obligaciones recogidas en las Secciones 5, 6.1 (únicamente en la medida necesaria para permitir que Google ejecute las disposiciones de la Sección 3.8), 7, 11, 12, 13 y 15 permanecerán vigentes tras la rescisión o resolución de este Acuerdo.

Anexo II: Ubicaciones admitidas para el registro de comerciantes y desarrolladores

Utiliza la siguiente tabla como referencia cuando te registres como comerciante y desarrollador de Google Play.

Leyenda

Título de la columna	Definiciones
Admite el registro de programadores de Google Play	✔ La ubicación admite el registro de programadores de Google Play. ✘ La ubicación no admite el registro de programadores de Google Play.
Admite el registro de comerciantes	✔ La ubicación admite el registro de comerciantes. ✘ La ubicación no admite el registro de comerciantes. Nota: Si tu ubicación admite el registro de comerciantes, los clientes de Google Play deben encontrarse en una ubicación admitida para poder comprar apps.
Moneda predeterminada del programador	La moneda predeterminada se basa en la ubicación del perfil de pagos del programador. ☆ La ubicación no admite comerciantes.
Distribuidor	Es la entidad de Google que distribuye apps a usuarios de Google Play en tu nombre.

Location	Admite el registro de programadores de Google Play	Admite el registro de comerciantes	Moneda predeterminada del programador	Distribuidor de Google Play
Albania	✔	✘	☆	Google Commerce Limited
Alemania	✔	✔	EUR	Google Commerce Limited
Angola	✔	✘	☆	Google Commerce Limited
Antigua y Barbuda	✔	✘	☆	Google Inc.
Antillas Neerlandesas	✔	✘	☆	Google Inc.
Arabia Saudita	✔	✔	SAR[1]	Google Commerce Limited
Argelia	✔	✘	☆	Google Commerce Limited
Argentina	✔	✔	USD	Google Inc.
Armenia	✔	✘	☆	Google Commerce Limited
Aruba	✔	✘	☆	Google Inc.
Australia	✔	✔	AUD	Google Asia Pacific Limited

Austria	✔	✔	EUR	Google Commerce Limited
Azerbaiyán	✔	✔	USD	Google Commerce Limited
Bahamas	✔	✘	☆	Google Inc.
Bahréin	✔	✔	USD	Google Commerce Limited
Bangladesh	✔	✘	☆	Google Asia Pacific Limited
Barbados	✔	✘	☆	Google Inc.
Bélgica	✔	✔	EUR	Google Commerce Limited
Belice	✔	✘	☆	Google Inc.
Benín	✔	✘	☆	Google Commerce Limited
Bermudas	✔	✘	☆	Google Inc.
Bielorrusia	✔	✔	USD	Google Commerce Limited
Bolivia	✔	✔	USD	Google Inc.
Bosnia-Herzegovina	✔	✘	☆	Google Commerce Limited

Botsuana	✔	✘	☆	Google Commerce Limited
Brasil	✔	✔	BRL	Google Inc.
Brunéi	✔	✘	☆	Google Asia Pacific Limited
Bulgaria	✔	✔	EUR	Google Commerce Limited
Burkina Faso	✔	✘	☆	Google Commerce Limited
Cabo Verde	✔	✘	☆	Google Commerce Limited
Camboya	✔	✘	☆	Google Asia Pacific Limited
Camerún	✔	✘	☆	Google Commerce Limited
Canadá	✔	✔	CAD	Google Inc.
Chile	✔	✔	CLP[1]	Google Inc.
China	✔	✔	USD	Google Asia Pacific Limited
Chipre	✔	✔	EUR	Google Commerce Limited

Ciudad del Vaticano	✔	✘	☆	Google Commerce Limited
Colombia	✔	✔	COP[1]	Google Inc.
Corea del Sur	✔	✔	KRW	Google Asia Pacific Limited
Costa de Marfil	✔	✘	☆	Google Commerce Limited
Costa Rica	✔	✔	CRC[1]	Google Inc.
Croacia	✔	✘	☆	Google Commerce Limited
Dinamarca	✔	✔	DKK	Google Commerce Limited
Ecuador	✔	✘	☆	Google Inc.
Egipto	✔	✔	EGP[1]	Google Commerce Limited
El Salvador	✔	✘	☆	Google Inc.
Emiratos Árabes Unidos	✔	✔	AED[1]	Google Commerce Limited
Eslovaquia	✔	✔	EUR	Google Commerce Limited
Eslovenia	✔	✘	☆	Google Commerce Limited

España	✔	✔	EUR	Google Commerce Limited
Estados Unidos	✔	✔	USD	Google Inc.
Estonia	✔	✔	EUR	Google Commerce Limited
Filipinas	✔	✔	PHP[1]	Google Asia Pacific Limited
Finlandia	✔	✔	EUR	Google Commerce Limited
Fiyi	✔	✘	☆	Google Asia Pacific Limited
Francia	✔	✔	EUR	Google Commerce Limited
Gabón	✔	✘	☆	Google Commerce Limited
Georgia	✔	✘	☆	Google Commerce Limited
Ghana	✔	✘	☆	Google Commerce Limited
Gibraltar	✔	✘	☆	Google Commerce Limited
Grecia	✔	✔	EUR	Google Commerce Limited

Groenlandia	✔	✘	☆	Google Commerce Limited
Guam	✔	✘	☆	Google Inc.
Guatemala	✔	✘	☆	Google Inc.
Guinea Ecuatorial	✔	✘	☆	Google Commerce Limited
Haití	✔	✘	☆	Google Inc.
Honduras	✔	✔	USD	Google Inc.
Hong Kong	✔	✔	HKD	Google Asia Pacific Limited
Hungría	✔	✔	HUF	Google Commerce Limited
India	✔	✔	INR	Google Asia Pacific Limited
Indonesia	✔	✔	IDR[1]	Google Asia Pacific Limited
Irlanda	✔	✔	EUR	Google Commerce Limited
Islandia	✔	✔	EUR	Google Commerce Limited
Islas Feroe	✔	✘	☆	Google Commerce Limited

Islas Malvinas	✔	✘	☆	Google Inc.
Islas Vírgenes Británicas	✔	✘	☆	Google Inc.
Israel	✔	✔	ILS	Google Commerce Limited
Italia	✔	✔	EUR	Google Commerce Limited
Jamaica	✔	✔	USD	Google Inc.
Japón	✔	✔	JPY	Google Asia Pacific Limited
Jordania	✔	✔	USD	Google Commerce Limited
Kazajistán	✔	✔	USD	Google Commerce Limited
Kenia	✔	✘	☆	Google Commerce Limited
Kuwait	✔	✔	USD	Google Commerce Limited
Laos	✔	✘	☆	Google Asia Pacific Limited
Letonia	✔	✔	USD[1]	Google Commerce Limited

Líbano	✔	✔	LBP[1]	Google Commerce Limited
Liechtenstein	✔	✘	☆	Google Commerce Limited
Lituania	✔	✔	EUR	Google Commerce Limited
Luxemburgo	✔	✔	EUR	Google Commerce Limited
Macedonia	✔	✘	☆	Google Commerce Limited
Malasia	✔	✔	MYR[1]	Google Asia Pacific Limited
Maldivas	✔	✘	☆	Google Asia Pacific Limited
Mali	✔	✘	☆	Google Commerce Limited
Malta	✔	✔	USD[1]	Google Commerce Limited
Marruecos	✔	✘	☆	Google Commerce Limited
Mauricio	✔	✘	☆	Google Commerce Limited
México	✔	✔	MXN	Google Inc.

Moldavia	✔	✘	☆	Google Commerce Limited
Mónaco	✔	✘	☆	Google Commerce Limited
Montenegro	✔	✘	☆	Google Commerce Limited
Mozambique	✔	✘	☆	Google Commerce Limited
Namibia	✔	✘	☆	Google Commerce Limited
Nepal	✔	✘	☆	Google Asia Pacific Limited
Nicaragua	✔	✘	☆	Google Inc.
Nigeria	✔	✔	USD	Google Commerce Limited
Noruega	✔	✔	NOK	Google Commerce Limited
Nueva Zelanda	✔	✔	NZD	Google Asia Pacific Limited
Omán	✔	✔	USD	Google Commerce Limited
Países Bajos	✔	✔	EUR	Google Commerce Limited

Pakistán	✔	✔	PKR[1]	Google Asia Pacific Limited
Panamá	✔	✔	USD	Google Inc.
Paraguay	✔	✘	☆	Google Inc.
Perú	✔	✔	PEN	Google Inc.
Polonia	✔	✔	PLN	Google Commerce Limited
Portugal	✔	✔	EUR	Google Commerce Limited
Puerto Rico	✔	✔	USD	Google Inc.
Qatar	✔	✔	USD	Google Commerce Limited
Reino Unido	✔	✔	GBP	Google Commerce Limited
República Checa	✔	✔	CZK	Google Commerce Limited
República Dominicana	✔	✔	USD	Google Inc.
Ruanda	✔	✘	☆	Google Commerce Limited
Rumania	✔	✔	RON[1]	Google Commerce Limited

Rusia	✔	✔	USD	Google Commerce Limited
San Marino	✔	✘	☆	Google Commerce Limited
Senegal	✔	✘	☆	Google Commerce Limited
Serbia	✔	✘	☆	Google Commerce Limited
Seychelles	✔	✘	☆	Google Commerce Limited
Singapur	✔	✔	SGD	Google Asia Pacific Limited
Sri Lanka	✔	✘	☆	Google Asia Pacific Limited
Sudáfrica	✔	✘	☆	Google Commerce Limited
Suecia	✔	✔	SEK	Google Commerce Limited
Suiza	✔	✔	CHF	Google Commerce Limited
Tailandia	✔	✔	THB[1]	Google Asia Pacific Limited
Taiwán	✔	✔	USD	Google Asia Pacific Limited

Tanzania	✔	✘	☆	Google Commerce Limited
Tayikistán	✔	✘	☆	Google Commerce Limited
Territorio Británico del Océano Índico	✔	✘	☆	Google Commerce Limited
Togo	✔	✘	☆	Google Commerce Limited
Trinidad y Tobago	✔	✘	☆	Google Inc.
Túnez	✔	✘	☆	Google Commerce Limited
Turkmenistán	✔	✘	☆	Google Commerce Limited
Turquía	✔	✔	TRY	Google Commerce Limited
Ucrania	✔	✔	UAH[1]	Google Commerce Limited
Uganda	✔	✘	☆	Google Commerce Limited
Uruguay	✔	✘	☆	Google Inc.
Uzbekistán	✔	✘	☆	Google Commerce Limited

Venezuela	✔	✔	USD	Google Inc.
Vietnam	✔	✔	VND[1]	Google Asia Pacific Limited
Yemen	✔	✔	USD	Google Commerce Limited
Zambia	✔	✘	☆	Google Commerce Limited
Zimbabue	✔	✘	☆	Google Commerce Limited