

UNIVERSIDAD AUTÓNOMA DE CIUDAD JUÁREZ

Instituto de Ingeniería y Tecnología

Departamento de Ingeniería Eléctrica y Computación



Aplicación móvil de realidad aumentada para apoyar la didáctica de la
geometría en nivel básico escolar

Reporte Técnico de Investigación presentado por:

Alexis Raul Arellano Banda 105894

Berenice Villanueva Sandoval 121182

Requisito para la obtención del título de
INGENIERO EN SISTEMAS COMPUTACIONALES

Asesor

Dra. Vianey Guadalupe Cruz Sánchez

Ciudad Juárez, Chihuahua

Mayo de 2018

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

Asunto: Liberación de Asesoría

Ing. Jesús Armando Gándara Fernández


Jefe del Departamento de Ingeniería

Eléctrica y Computación

Presente.-

Por medio de la presente me permito comunicarle que después de haber realizado las asesorías correspondientes al reporte técnico **Aplicación móvil de realidad aumentada para apoyar la didáctica de la geometría en nivel básico escolar**, los alumnos Alexis Raul Arellano Banda y Berenice Villanueva Sandoval de la Licenciatura en Ingeniería en Sistemas Computacionales, considero que lo han concluído satisfactoriamente, por lo que pueden continuar con los trámites de titulación intracurricular.

Sin más por el momento, reciba un cordial saludo.

Atentamente

Dra. Vianey Guadalupe Cruz Sánchez
Profesor Investigador IIT



Ccp. Coordinador del programa de sistemas computacionales

Alumnos

Archivo

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

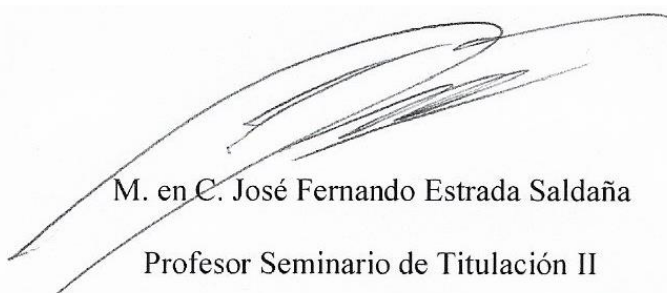
Asunto: Autorización de impresión

C. Alexis Raul Arellano Banda

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del proyecto de **Aplicación móvil de realidad aumentada para apoyar la didáctica de la geometría en nivel básico escolar**, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.



M. en C. José Fernando Estrada Saldaña
Profesor Seminario de Titulación II

Ciudad Juárez, Chihuahua a 19 de mayo de 2018

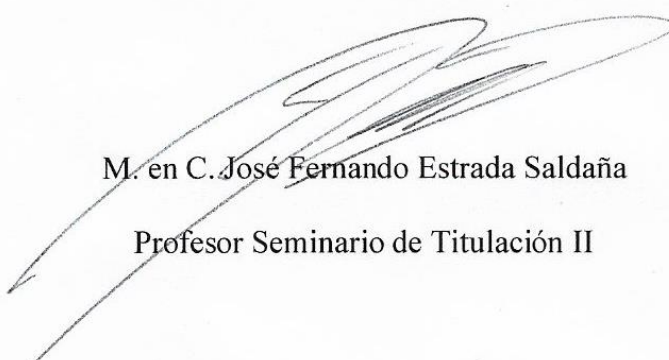
Asunto: Autorización de impresión

C. Berenice Villanueva Sandoval

Presente.-

En virtud de que cumple satisfactoriamente los requisitos solicitados, informo a usted que se autoriza la impresión del proyecto de **Aplicación móvil de realidad aumentada para apoyar la didáctica de la geometría en nivel básico escolar**, para presentar los resultados del proyecto de titulación con el propósito de obtener el título de Licenciado en Ingeniería en Sistemas Computacionales.

Sin otro particular, reciba un cordial saludo.



M. en C. José Fernando Estrada Saldaña

Profesor Seminario de Titulación II

Declaración de Originalidad

Nosotros, Alexis Raul Arellano Banda y Berenice Villanueva Sandoval, declaramos que el material contenido en esta publicación fue generado con la revisión de los documentos que se mencionan en la sección de referencias y que la solución desarrollada es original y no ha sido copiada de ninguna otra fuente, ni ha sido usada para obtener otro título o reconocimiento en otra Institución de Educación Superior.

Alexis Raul Arellano Banda

Berenice Villanueva Sandoval

Índice

Declaración de Originalidad	2
Índice.....	3
Índice de figuras.....	6
Índice de tablas	11
Resumen.....	12
Introducción	13
I. Planteamiento de la necesidad	15
1.1 Antecedentes	15
1.1.1 Aplicaciones de realidad aumentada en la geometría.....	15
1.1.2 Aplicación de realidad aumentada en la educación.....	18
1.2 Definición de la necesidad	21
1.3 Objetivo.....	22
1.3.1 Objetivos particulares	22
1.4 Justificación.....	22
II. Marco Referencial	23
2.1 Marco conceptual	23
2.1.1 Matemática y geometría	23
2.1.2 Aplicación móvil	24
2.1.3 Dispositivo móvil	25
2.2 Marco teórico	26
2.2.1 Realidad aumentada.....	26
2.2.2 Tecnología usada en la realidad aumentada	26
2.2.3 Niveles de realidad aumentada	30
2.2.4 Sistema móvil de realidad aumentada	32
2.2.5 Aplicaciones de la realidad aumentada	32
2.3 Marco tecnológico.....	35
2.3.1 Sistemas operativos para dispositivos móviles.....	35
2.3.2 Android.....	36
2.3.3 Arquitectura del sistema operativo Android.....	36
2.3.4 Componentes de una aplicación Android.....	38
III. Desarrollo del proyecto.....	40

3.1 Herramientas utilizadas	40
3.1.1 Software y lenguaje de programación	40
3.1.2 Dispositivo móvil Android para pruebas.....	41
3.2 Modelado de objetos geométricos 3D	41
3.2.1 Prisma triangular.....	42
3.2.2 Pirámide triangular	45
3.2.3 Cono.....	48
3.3 Desarrollo de la interfaz de usuario.....	50
3.4 Marcadores de realidad aumentada	54
3.5 Integración de modelos 3D y marcadores de realidad aumentada	60
3.5.1 Cubo.....	60
3.5.2 Cono.....	61
3.5.3 Prisma triangular.....	63
3.6 Botones virtuales	64
3.7 Codificación de scripts	67
3.7.1 Botones virtuales cubo.....	67
3.7.2 Habilitar/deshabilitar la visualización de los botones virtuales (ToggleScript).....	69
3.7.3 Cerrar aplicación (ExitScript).....	70
3.7.4 Reestablecer valores de objetos geométricos (ResetScript)	71
3.7.5 Autoenfoco de la cámara (Camera Focus Controller)	72
3.8 Integrar scripts al proyecto en Unity	72
3.9 Inhabilitación de botones virtuales.....	74
3.9.1 Rotación en eje Y (Toggle_rh)	74
3.10 Nombre de la figura geométrica aumentado	76
3.11 Fórmulas de área y volumen aumentados	80
3.12 Escena de ayuda	81
3.13 Validación	82
3.13.1 Pantalla principal	82
3.13.2 Visualización de figuras geométricas en la aplicación	83
3.13.4 Pruebas realizadas.....	85
3.13.5 Encuesta.....	85
IV. Resultados y Discusiones	87
4.1 Resultados	87

4.2 Análisis de resultados.....	93
V. Conclusiones	95
5.1 Objetivos del proyecto	95
5.2 Recomendaciones.....	95
5.3 Aportaciones.....	96
Referencias.....	97
Anexo 1 Instalación y configuración de software.....	100
Anexo 2 Integración de modelos 3D y marcadores de realidad aumentada	106
Anexo 3 Código de scripts.....	113
Anexo 4 Script para habilitar/deshabilitar botones virtuales	144
Anexo 5 Fórmulas de las figuras geométricas	155

Índice de figuras

Figura 1 Dispositivo HMD z800 AR [22]	27
Figura 2 Google Tango AR. [23]	28
Figura 3 Uso de un “Spatial Display” [24]	28
Figura 4 Código QR y de barras. [25].....	30
Figura 5 Marcadores de realidad aumentada. [26].....	31
Figura 6 Aplicación de geolocalización con realidad aumentada. [27]	31
Figura 7 Microsoft Hololens. [28]	32
Figura 8 Aplicación de realidad aumentada de la compañía automotriz Infinity. [29]	33
Figura 9 Aplicación móvil de realidad aumentada de la empresa IKEA. [30]	33
Figura 10 Aplicación de realidad aumentada en la medicina. [31].....	34
Figura 11 Pokemon Go, aplicación de realidad aumentada que utiliza la geolocalización. [32].....	34
Figura 12 Aplicación de realidad aumentada para la educación. [33].....	35
Figura 13 Arquitectura del sistema operativo Android [21]	38
Figura 14 Agregar cilindro.....	42
Figura 15 Modificación de vértices al cilindro	42
Figura 16 Agregar material	43
Figura 17 Agregar textura.....	43
Figura 18 Textura a utilizar.....	43
Figura 19 Vista <i>UV Editing</i>	44
Figura 20 Seleccionar textura	44
Figura 21 Función <i>Unwrap</i>	44
Figura 22 Modelo 3D de prisma triangular.....	45
Figura 23 Agregar cono	45
Figura 24 Cambiar el número de vértices del cono	46
Figura 25 Agregar material	46
Figura 26 Crear nueva textura.....	47
Figura 27 Vista <i>UV Editing</i>	47
Figura 28 Seleccionar textura	47
Figura 29 Función <i>Unwrap</i>	48

Figura 30 Modelo 3D de la pirámide triangular	48
Figura 31 Agregar como	49
Figura 32 Modelo 3D del cono	49
Figura 33 Agregar objeto <i>Canvas</i>	50
Figura 34 Propiedades de imagen del objeto <i>Panel</i>	50
Figura 35 Propiedades <i>Toggle_Formulas</i>	51
Figura 36 Propiedades de imagen del objeto <i>Toggle</i>	51
Figura 37 Valores modificados a los elementos de la interfaz	53
Figura 38 Vista jerárquica de los elementos	54
Figura 39 Generación de marcador de realidad aumentada.....	54
Figura 40 Agregar imagen a la base de datos	55
Figura 41 Proceso para agregar imagen a la base de datos	56
Figura 42 Base de datos de Vuforia	56
Figura 43 Descarga de la base de datos	57
Figura 44 Prefabs de la extensión Vuforia.....	57
Figura 45 Selección de imagen para el objeto <i>Image Target</i>	58
Figura 46 Valores de posición del marcador	58
Figura 47 Propiedades de cada uno de los marcadores creados	59
Figura 48 Agregar cubo	60
Figura 49 Propiedades del cubo 3D	60
Figura 50 Material del cubo 3D	61
Figura 51 Cubo 3D sobre su marcador de realidad aumentada	61
Figura 52 Selección del modelo cono en el explorador	62
Figura 53 Valores del cono 3D	62
Figura 54 Material de cono	62
Figura 55 Cono 3D sobre su marcador de realidad aumentada	63
Figura 56 Valores del prisma triangular 3D	63
Figura 57 Material del prisma triangular	63
Figura 58 Prisma triangular 3D sobre su marcador de realidad aumentada	64
Figura 59 Agregar objeto <i>VirtualButton</i>	65
Figura 60 Vista jerárquica del marcador.....	65
Figura 61 Valores del botón virtual	65
Figura 62 Propiedades de los botones virtuales de cada uno de los marcadores.....	66
Figura 63 Crear C# <i>script</i>	67

Figura 64 Código rotación del cubo en eje Y	68
Figura 65 Código rotación del cubo en eje X	68
Figura 66 Código para aumentar el tamaño del cubo	68
Figura 67 Código para reducir el tamaño del cubo	69
Figura 68 Código para cerrar la aplicación	71
Figura 69 Código para restablecer valores iniciales	71
Figura 70 Código de autoenfoque	72
Figura 71 Integración de <i>scripts</i>	72
Figura 72 <i>Scripts</i> del marcador del cubo	73
Figura 73 Asignación de método	73
Figura 74 Asignación de método	74
Figura 75 Selección de objeto <i>ARCamera</i>	74
Figura 76 Asignación de método	75
Figura 77 Asignación de método	75
Figura 78 Vista de métodos asignados.....	76
Figura 79 Agregar objeto <i>Quad</i>	76
Figura 80 Valores del objeto <i>Quad</i>	77
Figura 81 Material del objeto <i>Quad</i>	77
Figura 82 Valores del objeto <i>3D Text</i>	77
Figura 83 Propiedades <i>Text Mesh</i>	78
Figura 84 Propiedades de los nombres 3D	79
Figura 85 Agregar objeto <i>Plane</i>	80
Figura 86 Valores de objeto <i>Plane</i>	80
Figura 87 Asignación de método	81
Figura 88 Código para cargar escena	81
Figura 89 Código de iniciar video tutorial y regresar a la pantalla principal	82
Figura 90 Escena de ayuda	82
Figura 91 Pantalla principal de la aplicación.....	83
Figura 92 Cubo, cono, cilindro y esfera aumentados.....	83
Figura 93 Prismas: triangular, cuadrangular, pentagonal y hexagonal aumentados....	84
Figura 94 Pirámides: triangular, cuadrangular, pentagonal y hexagonal aumentados.	84
Figura 95 Resultados de la pregunta 1.	87
Figura 96 Resultados de la pregunta 2.	88
Figura 97 Resultados obtenidos de la pregunta 3.	88

Figura 98 Resultados de la pregunta 4	89
Figura 99 Resultados de la pregunta 5	89
Figura 100 Resultados pregunta 6.....	90
Figura 101 Resultados de la pregunta 7	90
Figura 102 Resultados de la pregunta 8	91
Figura 103 Resultados de la pregunta 9	91
Figura 104 Resultados de la pregunta 10	92
Figura 105 Condiciones de uso Unity.....	100
Figura 106 Rutas SDK y JDK en Unity	101
Figura 107 Crear clave de licencia Vuforia	102
Figura 108 Panel de administración Vuforia	102
Figura 109 Clave de licencia Vuforia	103
Figura 110 Importación de la extensión de Vuforia	103
Figura 111 Propiedades del objeto ARCamera.....	104
Figura 112 Agregar clave de licencia de Vuforia a Unity	104
Figura 113 Panel de administración de bases de datos Vuforia.....	105
Figura 114 Propiedades prisma cuadrangular	106
Figura 115 Material prisma cuadrangular	106
Figura 116 Prisma cuadrangular en marcador de realidad aumentada	107
Figura 117 Propiedades prisma pentagonal	107
Figura 118 Material prisma pentagonal	107
Figura 119 Prisma pentagonal en marcador de realidad aumentada.....	108
Figura 120 Propiedades prisma hexagonal	108
Figura 121 Material prisma hexagonal	108
Figura 122 Prisma hexagonal y marcador de realidad aumentada	109
Figura 123 Propiedades pirámide triangular	109
Figura 124 Pirámide triangular en marcador de realidad aumentada	109
Figura 125 Propiedades pirámide cuadrangular	110
Figura 126 Pirámide cuadrangular en marcador de realidad aumentada.....	110
Figura 127 Propiedades pirámide pentagonal.....	111
Figura 128 Pirámide pentagonal en marcador de realidad aumentada	111
Figura 129 Propiedades pirámide hexagonal	111
Figura 130 Pirámide hexagonal en marcador de realidad aumentada	112
Figura 131 Fórmulas cubo	155

Figura 132 Fórmulas esfera	155
Figura 133 Fórmulas cono	155
Figura 134 Fórmulas cilindro.....	156
Figura 135 Fórmulas prisma triangular.....	156
Figura 136 Fórmulas prisma cuadrangular	156
Figura 137 Fórmulas prisma pentagonal.....	157
Figura 138 Fórmulas prisma hexagonal.....	157
Figura 139 Fórmulas pirámide triangular	157
Figura 140 Fórmulas pirámide cuadrangular	158
Figura 141 Fórmulas pirámide pentagonal	158
Figura 142 Fórmulas pirámide hexagonal	158

Índice de tablas

Tabla 1 Resumen de antecedentes	20
Tabla 2 Encuesta	86
Tabla 3 Comentarios obtenidos de la pregunta 11	93

Resumen

La geometría es un área importante de la matemática, su correcto aprendizaje implica que el estudiante debe desarrollar habilidades visuales, analíticas y de razonamiento, por mencionar algunas. Por lo tanto, es importante que su aprendizaje sea correcto desde edades tempranas. Sin embargo, en algunas ocasiones, los métodos de enseñanza tradicionales utilizados por los docentes en el nivel básico escolar suelen ser un tanto tediosos, lo que provoca que los estudiantes pierdan interés en el tema.

El presente documento, detalla el desarrollo de una aplicación móvil de realidad aumentada como herramienta para apoyar los métodos de enseñanza tradicionales en el nivel básico escolar, con la cual los estudiantes pueden visualizar figuras geométricas en 3D y al activar algunas funciones en pantalla, pueden interactuar con las mismas.

Al utilizar la aplicación, los estudiantes experimentaron una manera distinta de ver las figuras geométricas en 3D que regularmente solo suelen ver en imágenes planas de los libros de texto y se demostró que el uso de la tecnología móvil en combinación con la realidad aumentada aplicada al tema de geometría, tiene un impacto positivo en el interés del estudiante por el tema.

Palabras clave: Geometría, aplicación móvil, realidad aumentada

Introducción

La geometría es una rama de la matemática encargada del estudio de las propiedades de figuras planas y tridimensionales, es considerada como una de las áreas con más dificultad en las matemáticas, debido a las diferentes habilidades requeridas, tales como, la imaginación, la cual es necesaria para lograr visualizar y representar en el cerebro cuerpos o figuras geométricas tridimensionales. En México, la mayoría de los docentes encargados de enseñar geometría en nivel básico, frecuentemente hacen cambios a sus métodos de enseñanza tradicionales con el fin de lograr que los estudiantes tengan una buena experiencia y un mejor aprendizaje, sin embargo, no es suficiente, ya que la mayoría de los métodos empleados resultan tediosos para los estudiantes, causando un desinterés en su estudio.

El correcto aprendizaje de la geometría desde edades tempranas es fundamental, debido a que ayudará al estudiante a desarrollar habilidades fundamentales tales como, el razonamiento deductivo, razonamiento analítico y resolución de problemas, por mencionar algunos. Por tal razón, la Secretaría de Educación Pública (SEP), ha motivado el uso de materiales que apoyen la práctica educativa, de tal forma que se mejore la calidad en la enseñanza de conceptos tan complejos como la geometría.

La tecnología, ha evolucionado de manera significativa, al mismo tiempo que los costos se van reduciendo, lo que hace posible el acceso a ellas de manera más simple. Una de estas tecnologías es la realidad aumentada, la cual brinda la posibilidad de visualizar objetos e información virtuales en un entorno físico real. Actualmente, la realidad aumentada es aplicada en una amplia variedad de campos, sin embargo, en el campo de la educación tiene gran potencial de uso.

En la enseñanza de la geometría, la realidad aumentada puede brindar al estudiante, la capacidad de poder visualizar y analizar desde diferentes perspectivas cuerpos geométricos tridimensionales, así como interactuar con ellos en tiempo real. Por tal razón, en el presente documento, se describe el desarrollo de una aplicación móvil de realidad aumentada para el apoyo en la didáctica de la geometría en el nivel básico escolar. Con lo anterior, se busca apoyar los métodos de enseñanza tradicionales de la

geometría, aprovechando los avances tecnológicos y la actual tendencia de uso de dispositivos electrónicos, de manera que se aumente y motive el interés del estudiante.

La aplicación fue enfocada a estudiantes de tercer a sexto grado del nivel básico escolar, desarrollada únicamente para el sistema operativo móvil Android, debido a que para desarrollar en el sistema iOS es necesario realizar un pago de licencia para el desarrollo.

A continuación se describe la estructura del documento.

En el capítulo 1, se planteo la necesidad, se definieron los objetivos y la justificación del proyecto.

En el capítulo 2, se describieron algunos conceptos referentes a la matemática, campo de estudio de la geometria, aplicaciones móviles y dispositivos móviles. Tambien se describieron las herramientas tecnológicas que se utilizaron en el desarrollo del proyecto.

En el capítulo 3, se detallo el procedimiento seguido para el desarrollo de la aplicación, el procedimiento siguió la estructura de la metologia de desarrollo en cascada.

En el capítulo 4, se presentaron los resultados obtenidos por las pruebas realizadas con la aplicación.

En el capítulo 5, se presentaron las conclusiones obtenidas durante el desarrollo del proyecto, algunas recomendaciones para trabajos a futuro y las aportaciones.

I. Planteamiento de la necesidad

1.1 Antecedentes

La geometría es un área interesante de la matemática debido a las diferentes maneras en las que se hace presente en la vida cotidiana, por lo que es necesario su correcta comprensión desde edades tempranas. Con el paso de los años, se han utilizado diversas tecnologías para desarrollar herramientas tales como *Cabri 3D* [1], el cual es un software interactivo de geometría desarrollado en un entorno 3D, que contiene objetos tales como: puntos, líneas, planos y poliedros que son representados en una pantalla 2D. En la actualidad, el uso de la tecnología de realidad aumentada en la geometría, hace posible la visualización de objetos e información virtual en un entorno real, por lo que su aplicación ha dado resultados positivos. A continuación, se describen algunos antecedentes enfocados en la educación de la geometría, haciendo el uso de la realidad aumentada.

1.1.1 Aplicaciones de realidad aumentada en la geometría

- *Sistema de realidad aumentada basado en bocetos para la educación de la geometría [6]*

En el año 2012, en Lodz, Polonia se presentó un prototipo de herramienta llamado *Augmented Geometry 3D Objects*, (*AG3DO*) por sus siglas en inglés. El prototipo fue capaz de generar superficies geométricas tridimensionales, denominadas *AG3DO*, a partir de bocetos lineales bidimensionales. La entrada se representó por dos perspectivas ortogonales de un objeto geométrico 3D. Las proyecciones 2D se obtuvieron a partir de bocetos del objeto 3D hechos a mano que el usuario dibujo previamente en el ordenador. Después de la reconstrucción, el objeto virtual tridimensional se superpuso a la escena real a través de realidad aumentada. La visualización se realizó en una pantalla de computadora.

Los resultados del estudio de la aplicación de realidad aumentada, demostraron que la interpretación de los bocetos manuales y las técnicas de reconstrucción de objetos geométricos en 3D en conjunto con la realidad aumentada, brindan una manera

interactiva de apoyar el aprendizaje del usuario y mejoran su habilidad espacial, la cual puede animar a los estudiantes a experimentar con simulaciones matemáticas y aprender sobre la geometría con métodos diferentes a los tradicionales.

- *Herramienta de aprendizaje de geometría para la escuela primaria utilizando realidad aumentada [2]*

En Bali, Indonesia en 2014, fue presentado un prototipo de herramienta de aprendizaje de geometría con realidad aumentada basado en *OpenCv*, para ayudar a los estudiantes de primaria a aprender mediante el uso del transportador. La herramienta fue desarrollada utilizando *Python* como lenguaje de programación y se compone de dos partes:

- Prototipo de dispositivo. Esta hecho de madera, contiene un proyector en el interior el cual es el encargado de proyectar la imagen del prototipo de aplicación que está en ejecución en la computadora y un espejo que refleja la imagen proyectada por el proyector.
- Prototipo de aplicación. La aplicación consta de cuatro marcadores, el marcador rojo es usado como punto de pivote, el cual no se puede mover ya que ejerce la función de punto central del sistema coordinado, los marcadores azul y verde son utilizados como objetivos mientras que el color lima muestra que tan grande es el ángulo. Los estudiantes deben medir el ángulo primero con un transportador para después utilizar el marcador color lima para visualizar el valor del ángulo y poder comprobar si la medición con transportador fue la correcta.

Los resultados obtenidos después de las pruebas realizadas fueron satisfactorios, los estudiantes descubrieron que el prototipo del sistema les ayudó en el aprendizaje de matemáticas, especialmente en el uso del transportador. Este prototipo también incrementa el interés del estudiante en el aprendizaje de matemáticas, de acuerdo a la respuesta por parte de los estudiantes, el 92% de ellos encontraron que el uso del

prototipo hace que el proceso de aprendizaje sea más ágil en comparación a los métodos tradicionales. [2]

- *Sistema de E-Learning utilizando realidad aumentada [5]*

En Pune, India en el año 2016, se propuso un sistema de *e-learning* para la geometría tridimensional, el cual hace uso de realidad aumentada para que los usuarios aprendan los conceptos de la geometría tridimensional de manera más rápida y eficiente. El sistema consistió en el reconocimiento de imágenes, es decir, reconocimiento de marcadores, reconocimiento de botones virtuales y un motor de realidad aumentada. Cuando la cámara captura la página actual, el sistema de *e-learning* identifica primero las imágenes que se encuentran en la página, las cuales actúan como marcadores, acto seguido, se aumenta el contenido visual correspondiente al monitor, en función de la posición y orientación del marcador. Los marcadores son registrados con el SDK de Vuforia, que asigna una clave de licencia única para el marcador. Cada marcador es reconocido de manera única, lo que permite que el motor de juego Unity procese salidas ampliadas específicas. Se hace uso de botones virtuales para la interacción del usuario y lograr una mejor comprensión.

Los dedos del usuario actúan como cursor para indicar la opción elegida por el usuario. Al hacerlo, se aumenta la sensibilidad del botón virtual, lo que hace que la salida se vuelva más sensible a las sombras. Cuando el usuario pasa el dedo sobre un botón específico, este realiza la función de menú predefinida correspondiente. Las funciones que se realiza son escalar, bajar, girar a la izquierda, girar a la derecha, subir, bajar, mover a la izquierda y mover a la derecha. Las propiedades del objeto son visibles a petición en el segundo módulo de la aplicación. Para la activación de este módulo, el estudiante pasa su mano sobre el objeto. La cámara reconoce la sombra de la mano y muestra las propiedades del objeto en particular. El tercer módulo utiliza tecnología de reconocimiento de texto. Aquí, la palabra en particular es introducida en el motor del juego y se le asigna un video específico. Cuando la cámara reconoce que la palabra en particular está disponible en un estilo de fuente particular, la hace coincidir con el vídeo asignado y aumenta el vídeo.

El último módulo del sistema, es un cuestionario de autoevaluación de varios niveles de dificultad y cada nivel tiene una serie de preguntas que el usuario responde basándose en los botones virtuales disponibles. Pasando el dedo sobre el botón virtual, la respuesta correspondiente es grabada y se visualiza la siguiente pregunta. Una vez finalizado el cuestionario, el sistema muestra las estadísticas del desempeño del usuario, con el fin de monitorear y mejorar su progreso en geometría.

- *Aplicación de realidad aumentada con reconocimiento de gestos de mano para el aprendizaje de Geometría 3D [1]*

En Jeju, Corea del Sur, en el año 2017 se llevó a cabo el desarrollo de una Aplicación de realidad aumentada con reconocimiento de gestos de mano para el aprendizaje de Geometría 3D. La aplicación hace uso de una pantalla, webcam, marcadores de realidad aumentada, librerías del *ARToolkit* y un *Leap Motion controller*, el cual ofrece la tecnología para el reconocimiento de los gestos de mano. Su funcionamiento consiste en que los marcadores son detectados y reconocidos por la webcam, de modo que las librerías del *ARToolkit* muestran el objeto en 3D por encima del marcador. El *Leap Motion controller* se encarga de reconocer de los gestos de mano cuando el estudiante intenta interactuar con el objeto 3D, al mismo tiempo que interpreta el gesto y lo transforma en un comando específico para el sistema, el cual realiza la acción programada para ese gesto.

Considerando los resultados de los estudios realizados para la evaluación de la aplicación, se llegó a la conclusión que es útil para el aprendizaje de geometría en 3D de una manera, más eficiente y conveniente que con los métodos tradicionales que utilizan papel y lápiz, sin embargo, la mayoría de los usuarios no están familiarizados con los gestos de mano en el entorno de realidad aumentada, por tal motivo se requiere que los problemas presentados durante la evaluación sean estudiados y solucionados.[1]

1.1.2 Aplicación de realidad aumentada en la educación

- *Mejora de la experiencia educativa con realidad aumentada [7]*

En Opatija, Croacia en el año 2015 se presentó un proyecto llamado *ARAVET*. En el prototipo de este proyecto se introdujo una capa de información digital mediante

marcadores especiales en libros de texto existentes. Haciendo uso de teléfonos inteligentes o tabletas electrónicas, los estudiantes podían acceder a modelos 3D y realizar acciones pre-animadas para visualizar adicionalmente material educativo.

El prototipo fue aplicado a tres diferentes campos. El primero fue la industria textil, se implementó la realidad aumentada en el modelo de aprendizaje de una máquina de coser, los estudiantes fueron capaces de realizar diferentes acciones tales como cambiar la bobina e insertar un hilo en una aguja.

El segundo campo fue la electrónica, se presentó un ciclo de operación del diodo con realidad aumentada, los estudiantes pudieron revertir la corriente eléctrica y ver por qué dejó de funcionar, el interior del diodo se presentó con un flujo de electrones, por lo que los estudiantes podían ver este concepto de una manera que nunca podrían experimentar si solo estuvieran leyendo sobre el mismo desde el libro de texto ordinario.

El tercer campo fue la informática, la implementación de la realidad aumentada fue utilizada en materiales de aprendizaje sobre conceptos lógicos y puertas electrónicas (AND, OR, XOR y NOR).

Los resultados obtenidos mediante encuestas que posteriormente fueron analizadas estadísticamente, mostraron que mediante el uso de materiales aumentados, los estudiantes fueron capaces de entender el contenido de una manera más rápida y sencilla en comparación con el uso de los métodos de aprendizaje tradicionales.

- *Aplicación de la tecnología de realidad aumentada para promover el aprendizaje interactivo [4]*

En el estudio presentado en Japón en el año 2012, se desarrolló una aplicación experimental biológica interactiva para guiar a los estudiantes en la unidad de aprendizaje, mediante la aplicación de la realidad aumentada en la biología básica. El sistema consiste en cinco unidades de aprendizaje, la estructura del microscopio, la operación del microscopio, la observación de células animales y vegetales, la anatomía de la rana y la estructura ósea de la rana. Los estudiantes, aprendieron la estructura y la función del microscopio a través de un proceso interactivo y con la operación real,

comprendieron la diferencia entre la estructura de las células animales y de las plantas. Además, a través de los medios visuales que mostraron el efecto, se observó la anatomía de la rana y sus diversos órganos. Cada unidad se evaluó con una prueba de estudio, la cual fue respondida por los estudiantes siguiendo una serie de instrucciones.

Considerando los resultados obtenidos con base a las evaluaciones realizadas, se descubrió que el integrar la tecnología de realidad aumentada con el aprendizaje de biología, ayudó a mejorar los resultados de aprendizaje de los estudiantes y su motivación.

La tabla 1, muestra un resumen de los antecedentes presentados con sus características principales.

Tabla 1 Resumen de antecedentes

Año	Título	Tecnología Utilizada	Temas de geometría abordados	Lugar
2012	Sistema de realidad aumentada basado en bocetos para la educación de la geometría	Realidad Aumentada	Cuerpos geométricos 3D.	Lodz, Polonia
2014	Herramienta de aprendizaje de geometría para la escuela primaria utilizando realidad aumentada	Realidad Aumentada	Cuerpos geométricos Uso del transportador Ángulos	Bali, Indonesia
2016	Sistema de <i>E-learning</i> utilizando realidad aumentada	Realidad Aumentada	Conceptos de geometría tridimensional Traslaciones.	Pune, India
2017	Aplicación de realidad aumentada con reconocimiento de gestos de mano para el aprendizaje de Geometría 3D	Realidad Aumentada	Propiedades de cuerpos geométricos 3D Construcción de cuerpos geométricos 3D.	Jeju, Corea del Sur

2015	Mejora de la experiencia educativa con realidad aumentada	Realidad Aumentada	No se abordan temas de geometría. El artículo se enfoca en temas industria textil, electrónica e informática.	Opatija, Croacia
2017	Aplicación de la tecnología de realidad aumentada para promover el aprendizaje interactivo	Realidad Aumentada	No se abordan temas de geometría. El artículo se enfoca en temas de biología básica	Sapporo, Japón

En el desarrollo de la aplicación descrita en este documento se intenta que la estructura de la misma así como los conceptos y temas presentados estén a la par del actual plan de estudios de geometría que se imparte en el nivel básico escolar en México de manera que pueda ser incorporada para su utilización durante el ciclo escolar como herramienta de apoyo al mismo programa de estudios.

1.2 Definición de la necesidad

Actualmente, el nivel básico escolar de México, se encuentra en un proceso de transformación de su modelo educativo. La necesidad que existe de que los estudiantes se centren en aprendizajes clave es imperante. Uno de los aprendizajes que se espera obtener es el pensamiento matemático, el cual no se ha podido desarrollar en todos y cada uno de los estudiantes. La geometría, es uno de los conceptos que se imparte desde nivel básico y es aplicada en campos tales como, la astronomía, la arquitectura y en general en la vida cotidiana, además de ser utilizada en otros temas de las matemáticas. Sin embargo, los métodos tradicionales de enseñanza empleados por la mayoría de los profesores se basan en la transmisión de los contenidos, lo cual tiene como consecuencia que los estudiantes no consideren el tema interesante, útil y tengan imágenes conceptuales pobres de los objetos geométricos. Diversos esfuerzos se han realizado para facilitar a los profesores de herramientas tecnológicas que apoyen en su quehacer didáctico, sin embargo, no todas las escuelas disponen de una herramienta tecnológica que facilite, tanto al profesor como al estudiante, enriquecer la imagen conceptual de los objetos.

1.3 Objetivo

Desarrollar una aplicación móvil de realidad aumentada, como herramienta de apoyo en la impartición del tema de geometría, en el nivel básico escolar.

1.3.1 Objetivos particulares

- Crear una interfaz basada en los principios de interacción humano-computadora para el estudiante de nivel básico.
- Diseñar los modelos que permitan al estudiante, visualizar la imagen conceptual del objeto geométrico.
- Implementar la aplicación en un dispositivo móvil.
- Desarrollar la aplicación utilizando tecnología de realidad aumentada.

1.4 Justificación

La geometría, es una de las áreas que atiende las matemáticas, donde los estudiantes en la mayoría de los casos deben hacer uso de habilidades tales como, la imaginación, para visualizar cuerpos geométricos en 3D a partir de imágenes 2D. Para impartir el tema, algunos maestros incluyen el uso de métodos prácticos como el armado de figuras geométricas para ayudar a potenciar la imaginación de los estudiantes y poder visualizarlas de mejor forma, con lo cual intentan motivar a los estudiantes. Sin embargo, en la actualidad las nuevas generaciones de estudiantes a nivel básico han crecido bajo el uso de la tecnología, y los métodos tradicionales no son atractivos en la mayoría de los casos. Lo anterior, propicia el uso de aplicaciones tales como la realidad aumentada, la cual ha demostrado que puede influir positivamente en el ámbito de la educación.

Con la creación de una herramienta basada en realidad aumentada se pretende apoyar al profesor durante la impartición del tema de geometría, con el fin de motivar y aumentar el interés del estudiante al ver representados objetos geométricos con los cuales pueda interactuar. Aunado a lo anterior, al ser una aplicación implementada en dispositivos móviles, se pretende que el estudiante no sólo utilice la herramienta en el aula de clases, sino que además, pueda seguir interactuando con ella en su hogar o cualquier otro lugar.

II. Marco Referencial

2.1 Marco conceptual

A continuación. Se describen algunos conceptos referentes a la matemática y el campo de estudio de la geometría, así como los conceptos más generales de aplicaciones móviles y dispositivos móviles en base al desarrollo de la aplicación aquí propuesta.

2.1.1 Matemática y geometría

La matemática se puede definir como la ciencia encargada del estudio de las propiedades y relaciones de los números, cuerpos geométricos, formas, símbolos entre otros, así como de analizar y describir objetos de estudio básicos como la cantidad, la estructura, el espacio y el cambio. Son parte esencial en el desarrollo del pensamiento racional de las personas el cual es necesario para el aprendizaje de las mismas matemáticas y otras disciplinas, así mismo fomenta capacidades tales como la abstracción, generalización y el razonamiento lógico entre otros. [16]

La geometría es una rama de la matemática encargada del estudio de las propiedades de figuras planas y tridimensionales. Existe una gran cantidad de tipos de geometría, sin embargo las cuatro con más relevancia son las siguientes:

- Geometría euclidiana: También conocida como geometría clásica, es la geometría que más se enseña en los niveles básicos de educación. Se basa en los postulados de Euclides, la cual de manera general se puede describir como el estudio de las propiedades de las formas regulares tales como las líneas y planos, círculos y esferas, triángulos y conos por mencionar algunos. Tiene una curvatura plana, por lo que la suma de los tres ángulos interiores es equivalente a cero. [17]
- Geometría no euclidiana: Se le denomina así a toda geometría que difiera con los postulados establecidos por Euclides en la geometría euclidiana y existe una gran variedad de esta geometrías, tales como la geometría hiperbólica, elíptica por mencionar algunas. Todas estas geometrías tienen en común que su curvatura es constante. [18]

- Geometría analítica: Estudia las figuras y construcciones geométricas en un sistema de coordenadas. Las líneas y curvas son representadas como un conjunto de coordenadas, relacionadas por una regla de correspondencia la cual usualmente es una función o relación. Los sistemas de coordenadas más utilizados son los planos cartesianos, polares y paramétricos. [19]
- Geometría diferencial: La geometría diferencial estudia los planos, líneas y superficies en un espacio tridimensional utilizando los principios de cálculo diferencial e integral. Se enfoca en una gran variedad de problemas tales como superficies de contactos. El rango de aplicación va desde problemas de ingeniería hasta los cálculos de campos gravitacionales. [20]

2.1.2 Aplicación móvil

Una aplicación móvil se puede definir como un software diseñado para dispositivos móviles, tales como teléfonos inteligentes y tabletas electrónicas. Regularmente las aplicaciones están enfocadas en brindarle la posibilidad al usuario de utilizar herramientas similares a las que se tienen acceso en una computadora.

Existen 3 tipos de aplicaciones móviles: nativas, web e híbridas. [10]

- Aplicaciones nativas: Son aplicaciones desarrolladas específicamente para alguna plataforma, tomando en cuenta el tipo de dispositivo, sistema operativo y versión del mismo. Consisten en archivos ejecutables los cuales se descargan directamente al dispositivo y se almacenan, se pueden obtener por medio de alguna tienda de aplicaciones. [10]
- Aplicaciones web: Son aplicaciones diseñadas para su ejecución en el navegador web del dispositivo móvil sin tomar en cuenta el tipo de sistema operativo y versión, se desarrollan utilizando lenguajes de programación como HTML, CSS, JavaScript entre otros. [10]
- Aplicaciones híbridas: Las aplicaciones híbridas, utilizan las mejores características de las aplicaciones web y nativas, esto quiere decir que utilizan

lenguajes de programación multiplataforma al mismo tiempo que se puede acceder a muchas de las funciones del dispositivo. [10]

2.1.3 Dispositivo móvil

Un dispositivo móvil puede definirse como una especie de computadora de tamaño pequeño diseñado para llevar a cabo una función, pero tienen la capacidad de desempeñar funciones más generales. Los dispositivos móviles cuentan con cuatro características principales: movilidad, tamaño reducido, comunicación inalámbrica e interacción hombre-computadora. [11]

Tipos de dispositivos móviles más relevantes:

- **Teléfonos inteligentes:** Un teléfono inteligente, también conocido como *smartphone*, puede definirse como la combinación de un asistente personal digital (*PDA*) y una extensión de los servicios web de redes sociales basados en computadoras de escritorio para el mundo móvil. Actualmente, los teléfonos inteligentes generalmente incluyen una cámara de alta resolución para tomar fotos y videos, funcionalidad de GPS y brújula, así como sensores de movimiento y varias interfaces de red, como interfaces celulares de alta velocidad, Bluetooth y Wi-Fi. Generalmente tienen la forma de teléfonos móviles, pero son un poco más grandes para acomodar una pantalla más grande y hardware adicional.[34]
- **Tabletas electrónicas:** Las tabletas electrónicas cumplen muchas funcionalidades que antes solo eran posibles de realizar en un teléfono inteligente o con una computadora de escritorio. Con la conectividad celular y Wi-Fi, las tabletas se han convertido en una herramienta ideal para el consumo multimedia y para mantenerse conectado con amigos a través de correo electrónico, mensajería instantánea y redes sociales basadas en la web sin la necesidad de sentarse en un escritorio.[34]

La arquitectura de procesamiento de un dispositivo móvil consiste en tres bloques principales:

- CPU: Consiste en un procesador de uno o más núcleos basados en arquitectura ARM de 32 y 64 bits.[34]
- GPU: Esta unidad de procesamiento está específicamente diseñada para manejar de manera eficiente operaciones y efectos gráficos 2D y 3D.[34]
- Modem: A veces también denominado procesador de "banda base". Incluye todos los componentes digitales necesarios para comunicarse con una red celular. [34]

Los dispositivos móviles actuales cuentan con las siguientes características principales disponibles: cámara frontal y trasera, sensores de movimiento, conectividad bluetooth y Wi-fi, GPS, pantallas táctiles, alta capacidad de almacenamiento, sistema operativo y baterías de larga duración por mencionar algunas.

2.2 Marco teórico

En el marco teórico, se describe el término de realidad aumentada, el tipo de tecnología utilizada, los tipos de interfaces, clasificación de niveles y sus aplicaciones.

2.2.1 Realidad aumentada

La realidad aumentada se puede definir como la visualización en tiempo real de un entorno físico real de manera directa o indirecta que ha sido ampliado por medio de la adición de elementos virtuales de información creados por un ordenador, esto quiere decir que combina elementos físicos del mundo real con los virtualmente creados, lo cual se hace posible cuando se toma la imagen del mundo real y se le superponen imágenes o modelos en 3D. [13, 14]

2.2.2 Tecnología usada en la realidad aumentada

Para la realidad aumentada se hace uso principalmente de los siguientes dispositivos: monitores o pantallas, dispositivos de entrada, dispositivos de rastreo y computadoras. Los tipos de monitores o pantallas que generalmente se utilizan en la realidad

aumentada son: *Head Mounted Displays* (HMD), dispositivos de mano y los *Spatial Displays*. [13]

2.2.2.1 HMD

Los HMD, son dispositivos de visualización que constan de una o dos pantallas, los cuales van montados sobre la cabeza del usuario por medio de un arnés o como parte de un casco. Colocan imágenes del mundo real y virtual sobre la visión del mundo físico real del usuario.

Se suelen utilizar 2 tipos diferentes de visualización, el primero es la visualización mediante video haciendo uso de dos cámaras y se requiere el procesamiento de ambas para proveer al usuario de la imagen aumentada del mundo real en conjunto con los objetos virtuales. El segundo tipo es la visualización óptica, el usuario puede ver a través de un lente el mundo físico real y superpone los objetos virtuales en el mismo, sin embargo al no procesar una imagen del mundo real, causa que se note cierto retraso en la superposición de la imágenes cuando el usuario se mueve, aunque algunos dispositivos modernos han logrado mejorar este tipo de visión mediante el uso de sensores para evitar el retraso en la alineación de las imágenes superpuestas. [13] (Ver Figura 1)



Figura 1 Dispositivo HMD z800 AR [22]

2.2.2.2 Dispositivos de mano

Los dispositivos de mano son generalmente dispositivos portátiles que cuentan con una pantalla, capacidad de procesamiento computacional y una cámara que el usuario pueda sostener con una o ambas manos. Utilizan el tipo de visión por medio de video para la

creación de la imagen aumentada en la pantalla y hacen uso de diversos sensores como el acelerómetro entre otros. Los dispositivos más comunes son los teléfonos inteligentes y las tabletas electrónicas ya que reúnen todas las características necesarias para el funcionamiento de la realidad aumentada. [13] (Ver Figura 2)



Figura 2 Google Tango AR. [23]

2.2.2.3 Spatial displays

Los *Spatial Displays*, sustituyen los monitores y pantallas por proyectores de video, hologramas y tecnologías de rastreo. Separan la mayor parte de la tecnología usada del usuario para integrarla más al entorno de manera que la imagen aumentada con los elementos virtuales pueda ser vista por varios usuarios y no solo uno. A su vez, pueden proyectar una imagen y detectar la presencia del usuario por medio de una cámara y dispositivos de rastreo, de esa manera se obtiene la información de la posición del usuario y superponer algún objeto o imagen en la posición del usuario o cercano a él, por lo tanto, el usuario puede ver en la imagen proyectada, el objeto creado virtualmente de manera que se percibe como si estuviera interactuando con él. [13] (Ver Figura 3)



Figura 3 Uso de un “Spatial Display” [24]

2.2.2.4 Dispositivos de entrada y rastreo

Existe gran variedad de dispositivos de entrada para realidad aumentada, tales como guantes, pulseras y dispositivos móviles entre otros. Generalmente los dispositivos de entrada son elegidos de acuerdo a los requerimientos de la aplicación o sistema. [13]

Los dispositivos de rastreo consisten en cámaras digitales, sensores ópticos, GPS, acelerómetros, sensores inalámbricos y muchos otros dependiendo de qué tan exacto se requiere que sea el sistema o aplicación que se está desarrollando. [13]

2.2.2.5 Computadoras

Las computadoras utilizadas en la realidad aumentada requieren de una capacidad de procesamiento alta, así como suficiente cantidad de memoria RAM para ser capaces de procesar las imágenes captadas por la cámara. Para poder utilizar la realidad aumentada de manera portátil, los dispositivos móviles de nueva generación como los teléfonos inteligentes y tabletas electrónicas cumplen con estos requisitos, mientras que si se requiere que el equipo sea estacionario se suelen utilizar computadoras con procesador de alto rendimiento en conjunto con una tarjeta de gráficos de alto nivel. [13]

2.2.2.6 Interfaces de realidad aumentada

Uno de los aspectos más importantes de la realidad aumentada es el crear métodos para lograr establecer una interacción intuitiva entre el usuario y los objetos virtuales. Existen cuatro principales tipos de interfaces de realidad aumentada:

- **Interfaz tangible:** Las interfaces tangibles tienen la capacidad de interactuar con el mundo real mediante el uso de objetos y herramientas físicas. [13]
- **Interfaz colaborativa:** Las interfaces compartidas hacen uso de múltiples pantallas para apoyar las actividades remotas tales como la videoconferencia mediante el uso de interfaces 3D en conjunto con múltiples dispositivos para mejorar la interacción en el espacio de trabajo. [13]

- Interfaz híbrida: Las interfaces híbridas hacen uso de diferentes interfaces de manera que estas se complementen para lograr llevar a cabo una gran variedad de actividades de realidad aumentada para situaciones no planeadas o de las cuales no se conozca el tipo de interacción utilizada. [13]
- Interfaz multimodal: Las interfaces multimodales combinan la entrada de objetos reales y formas naturales de lenguaje, tales como el habla, tacto, gestos de mano o la mirada. [13]

2.2.3 Niveles de realidad aumentada

Los niveles dentro de la realidad aumentada representan la complejidad de las aplicaciones y las tecnologías utilizadas, de modo que mientras mayor sea el nivel, las posibilidades de aplicación aumentan. [14]

- Nivel 0: En el nivel 0 se encuentran los códigos de barras y los códigos QR, los cuales contienen principalmente información en HTML para enlazar otros contenidos. En el nivel 0 no se encuentran objetos en 3D. [14] (Ver Figura 4)

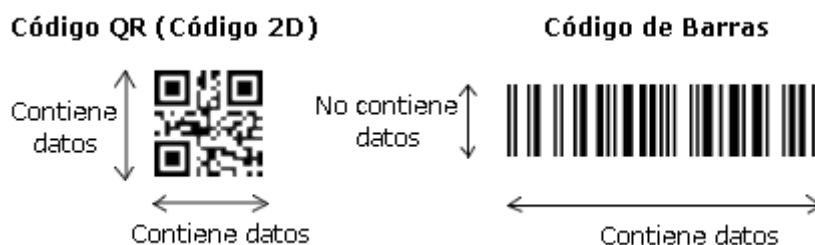


Figura 4 Código QR y de barras. [25]

- Nivel 1: En el nivel 1, la realidad aumentada consiste en el reconocimiento de patrones en 2D y 3D, se utilizan marcadores, los cuales son imágenes en blanco y negro que contienen información dentro de ellas, la cual es interpretada por el sistema de realidad aumentada. [14] (Ver Figura 5)



Figura 5 Marcadores de realidad aumentada. [26]

- Nivel 2: El nivel 2 de realidad aumentada consiste en el uso del GPS y geolocalización sin marcadores, de este modo, las aplicaciones de este nivel utilizan la entrada de información proporcionada por el GPS del dispositivo para mostrar puntos de interés o información basados en la localización del dispositivo. [14] (Ver Figura 6)



Figura 6 Aplicación de geolocalización con realidad aumentada. [27]

- Nivel 3: Consiste en una visión aumentada del entorno real mediante el uso de un dispositivo, con el cual se puede ver el mundo real en conjunto con la información y objetos virtuales directamente desde la vista del usuario. [14] (Ver Figura 7)

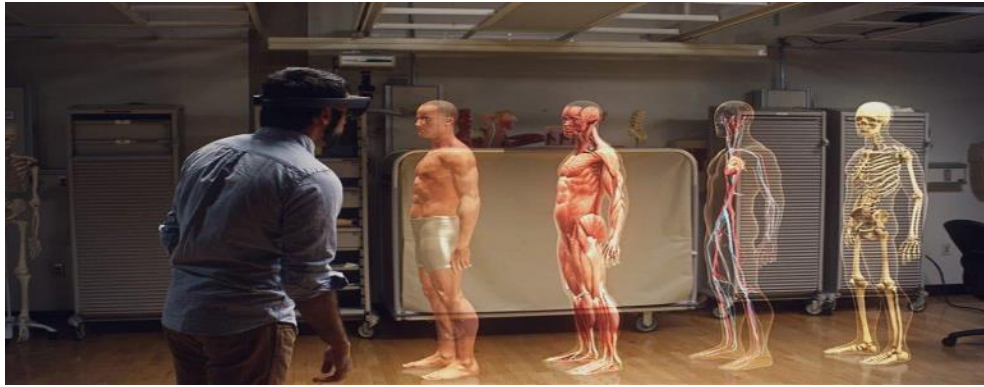


Figura 7 Microsoft Hololens. [28]

2.2.4 Sistema móvil de realidad aumentada

Un sistema móvil de realidad aumentada utiliza aplicaciones móviles en conjunto con sistemas inalámbricos, debe ser capaz de lograr una interacción entre el usuario y la información digital que esta superpuesta sobre los objetos y/o superficies del entorno real mediante el uso de interfaces requeridas por el sistema.

Si el usuario es capaz de enfocarse más en la aplicación o sistema en vez de prestarle más atención a los dispositivos utilizados al mismo tiempo que le brinda privacidad y la capacidad de compartir información cuando se requiera, si esto se cumple, el sistema es calificado como exitoso. [13]

2.2.5 Aplicaciones de la realidad aumentada

Actualmente, la realidad aumentada puede ser aplicada en una gran variedad de áreas, sin embargo, los usos de la realidad aumentada que más han destacado son el uso para publicidad, el uso comercial, la medicina, la educación y el entretenimiento. [13]

2.2.5.1 Realidad aumentada en la publicidad

El uso de la realidad aumentada en la publicidad brinda una gran variedad de posibilidades de promoción de artículos y servicios. Un claro ejemplo es el de las compañías de comercio que hacen uso de la realidad aumentada para promocionar sus artículos en sus sitios de venta en línea, la técnica de promoción más utilizada en este ámbito consiste en el uso de marcadores, los cuales el cliente los coloca frente a una

webcam o la cámara de su dispositivo móvil y mediante un software o aplicación proporcionado por la compañía, el cliente es capaz de visualizar el artículo desde diferentes perspectivas. [13] (Ver Figura 8)



Figura 8 Aplicación de realidad aumentada de la compañía automotriz Infinity. [29]

2.2.5.2 Realidad aumentada en el uso comercial

Para su uso comercial, la realidad aumentada ofrece soluciones a las industrias comerciales que cuentan con problemas tales como la construcción de prototipos de alto costo a manera de poder observar posibles mejoras y errores, la solución brindada por la realidad aumentada es la de la construcción de modelos en 3D de prototipos para la visualización de los mismos sin la necesidad de gastar una gran cantidad de capital. El área del turismo y las compañías dedicadas a la venta de muebles son otro caso del uso de la realidad aumentada. [13] (Ver Figura 9)



Figura 9 Aplicación móvil de realidad aumentada de la empresa IKEA. [30]

2.2.5.3 Medicina

En la medicina la realidad aumentada tiene una infinidad de aplicaciones, sin embargo, la visualización de imágenes en 3D y la cirugía asistida por imagen son las áreas que

mayor uso hacen de esta. Algunos casos son el análisis de imágenes biomédicas, cardiología y anatomía entre otros. [15] (Ver Figura 10)



Figura 10 Aplicación de realidad aumentada en la medicina. [31]

2.2.5.4 Entretenimiento

En el entretenimiento, los videojuegos para móviles es el área que más aplicación de realidad aumentada posee. Un ejemplo es el de los juegos de mesa, el hecho de poder visualizar e interactuar con animaciones mediante un dispositivo móvil, sea un teléfono inteligente o una tableta electrónica, resulta más llamativo para el jugador creando una experiencia de juego agradable. Otro ejemplo es de aplicaciones que utilizan la realidad aumentada en conjunto con la geolocalización, la cual en vez de utilizar marcadores para visualizar los objetos y/o animaciones, se basan en la información proporcionada por el GPS del dispositivo para mostrar los objetos virtuales en el entorno real. [13] (Ver Figura 11)



Figura 11 Pokemon Go, aplicación de realidad aumentada que utiliza la geolocalización. [32]

2.2.5.5 Realidad aumentada en la educación

En el área de la educación, la realidad aumentada ofrece muchas ventajas, se puede utilizar para el apoyo en el aprendizaje en diversas áreas mediante el uso de esta tecnología, así como, a nivel cultural. El uso de dispositivos móviles y realidad aumentada en conjunto con los sistemas de educación tradicionales no garantiza que el

aprendizaje sea más fácil, sin embargo hace más factible la posibilidad de aumentar el interés de la audiencia a la que va dirigida. Actualmente ya existen una gran cantidad de aplicaciones que hacen énfasis en este rubro, un ejemplo es el caso de aplicaciones que apoyan el aprendizaje de las matemáticas o de aplicaciones culturales utilizadas en museos, las cuales consisten en la reconstrucción de imágenes de hechos históricos. [13, 14] (Ver Figura 12)



Figura 12 Aplicación de realidad aumentada para la educación. [33]

2.3 Marco tecnológico

En el marco tecnológico, se describen las herramientas tecnológicas utilizadas para el desarrollo de la aplicación propuesta.

2.3.1 Sistemas operativos para dispositivos móviles

Un sistema operativo móvil es una plataforma de software en la cual otros programas llamados aplicaciones pueden funcionar en dispositivos móviles como teléfonos inteligentes, tabletas electrónicas entre otros. Con el paso del tiempo los sistemas operativos móviles han pasado por un proceso de evolución desde el cual al principio eran orientados a un sistema similar al de una computadora, hasta el actual que es orientado a teléfonos inteligentes, lo cual ha sido posible con los avances tecnológicos tales como el hardware, software y el Internet los cuales han ido ocurriendo con el paso de los años. Algunos de los sistemas operativos móviles más relevantes que existieron y/o existen actualmente son los siguientes [12]:

- Android, propiedad de Google.
- iOS, propiedad de Apple.
- Windows Phone, propiedad de Microsoft.
- Symbian, propiedad de Nokia.
- Blackberry OS, propiedad de RIM.

- Bada, propiedad de Samsung.
- webOS, propiedad de Hewlett-Packard.

2.3.2 Android

Android es un sistema operativo para móviles desarrollado por la *Open Handset Alliance*, la cual es dirigida por Google. Google reveló Android en noviembre del año 2007. [12]

Android utiliza un *kernel* Linux en conjunto con APIs de alto nivel escritas en lenguaje C. Las aplicaciones que se desarrollan para ese sistema, están escritas en lenguaje Java y funcionan bajo la máquina virtual Dalvik utilizando una compilación en tiempo real para traducir el *bytecode* de Java al *dex-code* de Dalvik[12]. Para los dispositivos con versión de Android mayor a 5.0, la máquina virtual Dalvik es reemplazada por la máquina virtual Android. Cada aplicación creada cuenta con un identificador Unix (UIDs) único y con permisos distintos, es decir, que una aplicación no puede leer o modificar el código fuente de otra. Para hacer posible la transferencia de información entre dos aplicaciones los permisos se tienen que declarar de manera estática al momento de iniciar la instalación. [8]

2.3.3 Arquitectura del sistema operativo Android

La arquitectura del sistema operativo Android está conformada de la siguiente manera (ver Figura 13):

- Kernel de Linux: El uso del kernel de Linux hace posible que Android aproveche funciones de seguridad claves al mismo tiempo que permite a los fabricantes de dispositivos móviles desarrollar controladores de hardware para un kernel conocido. [8]
- Capa de abstracción de hardware (HAL): Es la encargada de crear interfaces estándares para exponer las capacidades de hardware del dispositivo al *framework* de la API Java de más alto nivel. Consiste en varios módulos, los cuales cada uno implementa una interfaz para un componente en específico de hardware. [8]

- Tiempo de ejecución de Android: Tiene como finalidad el ejecutar varias máquinas virtuales en dispositivos cuya memoria es baja mediante la ejecución de archivos en formato DEX, el cual es un formato de códigos de bytes creado específicamente para Android y está optimizado para ocupar el menos espacio de memoria posible. El proceso consiste en crear cadenas de herramientas y compilar fuentes de Java en código de bytes DEX. Las compilaciones *ahead-of-time* (AOT) y *just-in-time* (JIT), la recolección optimizada de elementos sin uso, así como mejorar la compatibilidad con la depuración, son tan solo algunas de las funciones principales del tiempo de ejecución de Android. [8]

- Bibliotecas C/C++ nativas: Un gran número de servicios y componentes del sistema Android están basados en código nativo que requiere el uso de bibliotecas nativas escritas en lenguaje C y C++. Android proporciona a los desarrolladores la API del *framework* de Java para mostrar la funcionalidad de las bibliotecas nativas a las aplicaciones. [8]

- *Framework* del API de Java: El conjunto total de funciones del sistema Android está disponible por medio de API escritas en lenguaje Java. Las API son necesarias para la creación de aplicaciones de Android por medio de la simplificación y reutilización de componentes del sistema y servicios tanto centrales como modulares, tales como los siguientes:
 - Sistema de vista para la creación de la interfaz del usuario de una manera gráfica.
 - Administrador de recurso que facilita el acceso a recursos sin la necesidad de utilizar código.
 - Administrador de notificaciones para la personalización de notificaciones creadas por la aplicación las cuales son que se mostradas en la barra de estado del sistema.
 - Administrador de actividad para administrar y monitorear el ciclo de vida de una aplicación.
 - Proveedores de contenido para brindar el acceso a los datos desde una aplicación a otra. [8]

- Aplicaciones del sistema: Las aplicaciones del sistema están incluidas en el sistema Android de manera predeterminada y brindan capacidades esenciales a las cuales los desarrolladores tienen acceso desde sus propias aplicaciones. [8]

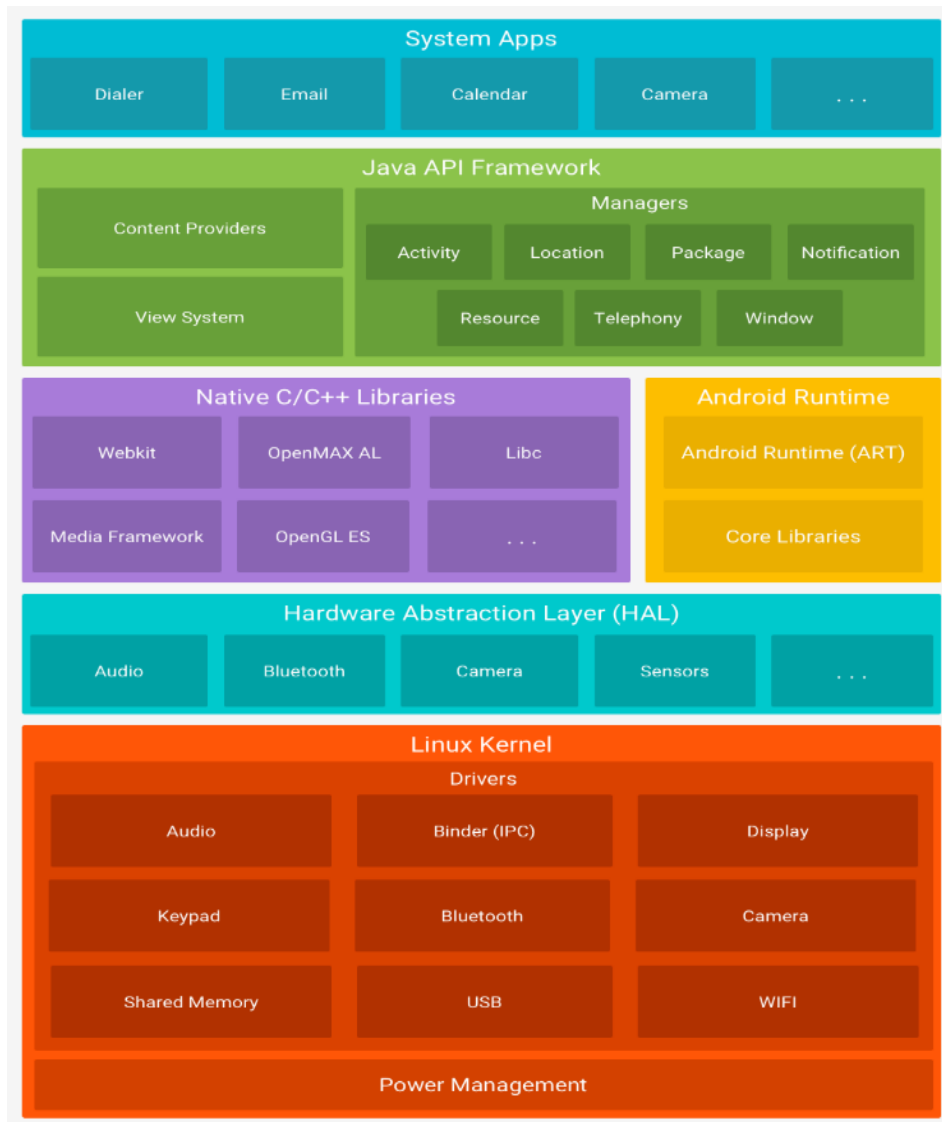


Figura 13 Arquitectura del sistema operativo Android [21]

2.3.4 Componentes de una aplicación Android

Los componentes de una aplicación Android son puntos diferentes desde los cuales el sistema puede ingresar a la aplicación, algunos son puntos de entrada reales para el usuario y algunos de ellos son dependientes entre sí, sin embargo cada componentes es único y tiene una función específica para definir el comportamiento general de la

aplicación. Actualmente existen cuatro tipos diferentes de componentes: actividades, servicios, proveedores de contenido, receptores de mensaje. [9]

- **Actividades:** Las actividades representan una pantalla en la interfaz del usuario, cada una de ellas es independiente de otras trabajando de manera conjunta, ya que el objetivo final de estas actividades es brindarle al usuario una experiencia sólida y satisfactoria. [9]
- **Servicios:** Los servicios realizan operaciones prolongadas o realizan tareas para procesos remotos, por lo tanto, se ejecutan en segundo plano de manera que no interrumpan la interacción del usuario mientras alguna otra actividad o aplicación está en ejecución. [9]
- **Proveedores de contenido:** Los proveedores de contenido se encargan de administrar los datos o información que puede compartir una aplicación es decir que una aplicación puede acceder a los datos de otra aplicación sin importar la ubicación de almacenamiento, esto es posible a través de un proveedor de contenido, siempre y cuando este lo permita. [9]
- **Receptores de mensajes:** La función principal de los receptores de mensajes es responder a los anuncios de mensajes del sistema, estos son originados por el sistema mismo y por las aplicaciones, aunque los receptores no tienen una interfaz de usuario, son capaces de crear notificaciones dentro de la barra de estado del sistema para hacerle saber al usuario sobre el mensaje creado. [9]

III. Desarrollo del proyecto

En el presente capítulo, se describen las herramientas utilizadas y los pasos realizados durante el desarrollo del proyecto. Es un desarrollo tecnológico y el tipo de estudio es descriptivo, el cual consiste en una aplicación móvil de realidad aumentada con la cual se pueden visualizar figuras geométricas 3D aumentadas e interactuar con las mismas.

3.1 Herramientas utilizadas

3.1.1 Software y lenguaje de programación

- Blender. Es un software dedicado a la creación de gráficos tridimensionales. Algunas de las características que ofrece son: modelación, animación, iluminación, renderizado. Durante el desarrollo de la aplicación móvil, Blender fue utilizado para la creación de los modelos geométricos 3D que posteriormente son aumentados.
- Unity 2017.3. Es un motor de juegos multiplataforma para el desarrollo de una gran variedad de aplicación. Se utilizó Unity para el desarrollo completo de la aplicación móvil.
- Vuforia. Vuforia le permite a los desarrolladores en Unity crear aplicaciones de Realidad Virtual y realidad aumentada, se utilizó Vuforia para la creación de las imágenes de destino que actúan como marcadores y los botones virtuales que permiten la interacción del usuario con el objeto geométrico 3D aumentado.
- Microsoft Visual Studio Community 2017. Se utilizó Visual Studio para la codificación en C# de los scripts en los cuales se encuentran las funciones utilizadas por la aplicación móvil.
- C#. Se utilizó C# como lenguaje de programación para los scripts debido a la fácil integración y manejo de las librerías de Unity y Vuforia en el editor de Visual Studio.

- Brosvision Augmented Reality Marker Generator. Es una aplicación web la cual se utilizó para la generación de las imágenes utilizadas como marcadores de realidad aumentada.
- Paint 3D. Fue utilizado para re-escalar las imágenes generadas por la aplicación web de Brosvision.

3.1.2 Dispositivo móvil Android para pruebas

Durante el desarrollo de la aplicación, se utilizó un dispositivo móvil con sistema operativo Android para las distintas pruebas de funcionamiento y rendimiento, las características del dispositivo son las siguientes:

- Samsung Galaxy Note 5
- OS Android 6.0.1
- Chipset Exynos 7420 Octa
- CPU Octa-core (4x2.1 GHz Cortex-A57 & 4x1.5 GHz Cortex-A53)
- GPU Mali-T760MP8
- Memoria RAM: 4GB

3.2 Modelado de objetos geométricos 3D

Para el desarrollo de la aplicación móvil, se utilizó Blender para la creación de los modelos en 3D de las siguientes figuras geométricas: cono, prismas y pirámides. El cubo, cilindro y esfera vienen prefabricados en Unity, por lo tanto, no fue requerido crear los modelos en Blender.

Prisma:

- Triangular
- Cuadrangular
- Pentagonal
- Hexagonal

Pirámide:

- Triangular
- Cuadrangular
- Pentagonal

- Hexagonal

A continuación, se detalla el procedimiento seguido para el modelado de los objetos geométricos en Blender.

3.2.1 Prisma triangular

Para el modelado del prisma triangular se realizaron las siguientes acciones:

En un archivo nuevo de Blender, se eliminó el cubo 3D que por defecto Blender coloca al crear un nuevo archivo.

Se creó un cilindro en 3D por medio de la barra de herramientas colocada en la parte inferior izquierda, se seleccionó la función *Add>Mesh>Cylinder* (ver figura 14).

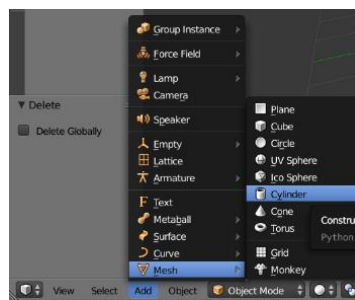


Figura 14 Agregar cilindro

Al crear el cilindro, se mostraron las propiedades del objeto, se modificó el número de vértices asignándole el número 3 en la opción de vértices (ver figura 15). Al realizar dicha acción, la silueta del cilindro cambio y se mostró en forma de prisma triangular.



Figura 15 Modificación de vértices al cilindro

El siguiente paso consistió en agregar un nuevo material al prisma triangular por medio de la opción que se muestra en la figura 16. Dicha vista se encuentra alineada a la parte derecha del espacio de trabajo. Posteriormente se agregó el nuevo material dando clic al botón *New*.

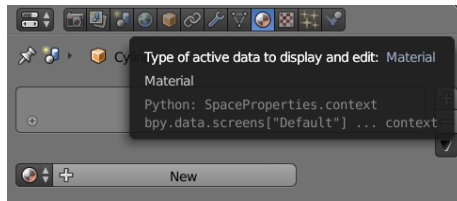


Figura 16 Agregar material

Al material se le agregó una textura por medio de la opción que se encuentra a la derecha inmediata de la opción material. En dicha opción se creó la nueva textura dando clic en el botón *New*. Al crear la nueva textura, se muestran las propiedades, en la propiedad *Image* se dio un clic en el botón *Open* (ver figura 17) y se seleccionó la imagen a utilizar como textura. El agregar un material y textura al modelo 3D es necesario para resaltar los bordes de las figuras.

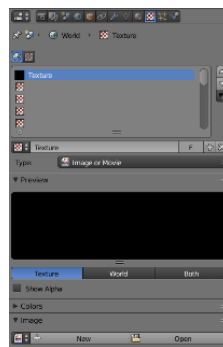


Figura 17 Agregar textura

La textura utilizada para el modelado de los objetos geométricos requeridos para el desarrollo de la aplicación móvil descrita en el documento, consistió en un cuadro de contorno negro y fondo transparente (ver figura 18).



Figura 18 Textura a utilizar

Se cambió la vista *Default* del espacio de trabajo a la vista *UV Editing* como se muestra en la figura 19.

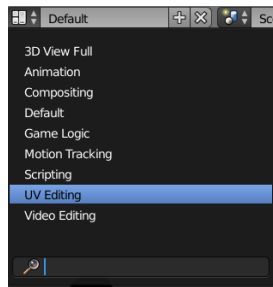


Figura 19 Vista *UV Editing*

La vista *UV Editing* muestra dos espacios de trabajo, el espacio de trabajo del lado izquierdo muestra la textura a utilizar, la cual es seleccionada por medio de la barra de herramientas colocada en la parte inferior izquierda. Para seleccionar la textura se le dio un clic a la opción que se muestra en la figura 20 y se seleccionó la imagen que se agregó al crear la textura.

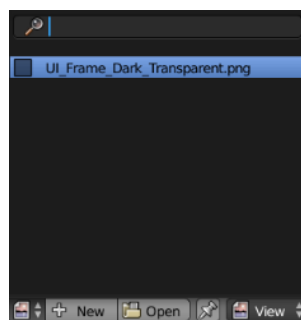


Figura 20 Seleccionar textura

El espacio de trabajo del lado derecho muestra el prisma triangular creado, se cambió el modo *Object Mode* por *Editing Mode*. En dicho modo se seleccionó el prisma triangular pulsando la tecla A del teclado, posteriormente se pulso la tecla U del teclado y se seleccionó la función *Unwrap* (ver figura 21)

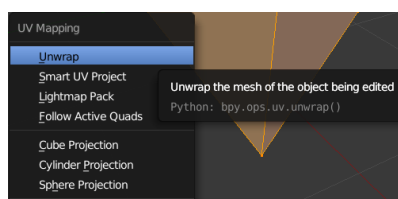


Figura 21 Función *Unwrap*

Se regresó a la vista *Default* de Blender. Se seleccionó la opción *Render>Render Image* para mostrar el prisma triangular 3D creado (ver figura 22). Por último, se guardó el archivo *.blend* del objeto creado en la carpeta *Assets* del proyecto creado anteriormente en Unity.

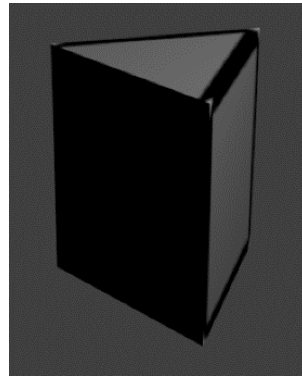


Figura 22 Modelo 3D de prisma triangular

El proceso realizado en el modelado del prisma triangular es el mismo para el resto de los prismas, la única modificación realizada fue el número de vértices.

3.2.2 Pirámide triangular

Para el modelado de la pirámide triangular se realizaron las siguientes acciones:

En un archivo nuevo de Blender, se eliminó el cubo 3D que por defecto Blender coloca al crear un nuevo archivo.

Se creó un cono en 3D por medio de la barra de herramientas colocada en la parte inferior izquierda, se seleccionó la función *Add>Mesh>Cone* (ver figura 23).

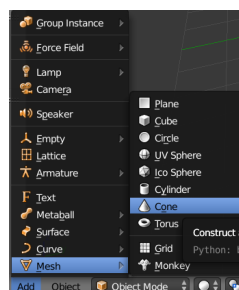


Figura 23 Agregar cono

Al crear el cono, se mostraron las propiedades del objeto, se modificó el número de vértices asignándole el número 3 en la opción de vértices (ver figura 24). Al realizar dicha acción, la silueta del cono cambio y se mostró en forma de pirámide triangular.

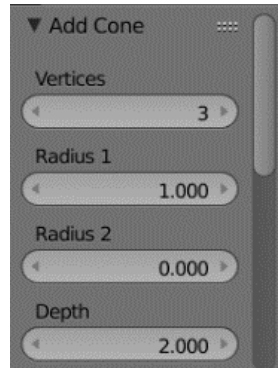


Figura 24 Cambiar el número de vértices del cono

El siguiente paso consistió en agregar un nuevo material a la pirámide triangular por medio de la opción que se muestra en la figura 25. Dicha vista se encuentra alineada a la parte derecha del espacio de trabajo. Posteriormente, se agregó el nuevo material dando clic al botón *New*.

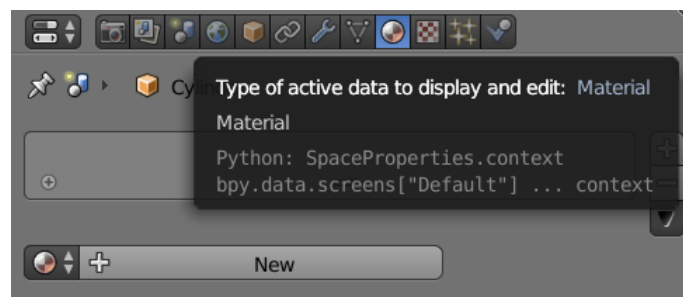


Figura 25 Agregar material

Al material se le agregó una textura por medio de la opción que se encuentra a la derecha inmediata de la opción material. En dicha opción, se creó la nueva textura dando clic en el botón *New*. Al crear la nueva textura, se muestran las propiedades, en la propiedad *Image* se dio un clic en el botón *Open* (ver figura 26) y se seleccionó la imagen a utilizar como textura. El agregar un material y textura al modelo 3D es necesario para resaltar los bordes de las figuras.

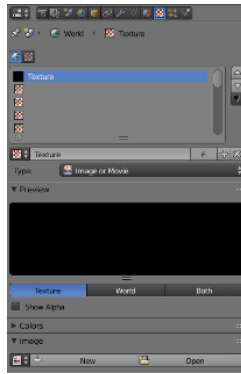


Figura 26 Crear nueva textura

Se cambió la vista *Default* del espacio de trabajo a la vista *UV Editing* como se muestra en la figura 27.

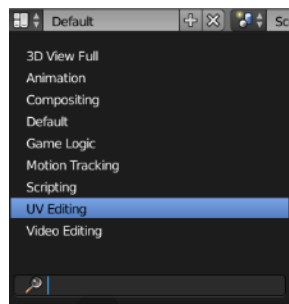


Figura 27 Vista *UV Editing*

La vista *UV Editing* muestra dos espacios de trabajo, el espacio de trabajo del lado izquierdo muestra la textura a utilizar, la cual es seleccionada por medio de la barra de herramientas colocada en la parte inferior izquierda. Para seleccionar la textura se le dio un clic a la opción que se muestra en la figura 71 y se seleccionó la imagen que se agregó al crear la textura.

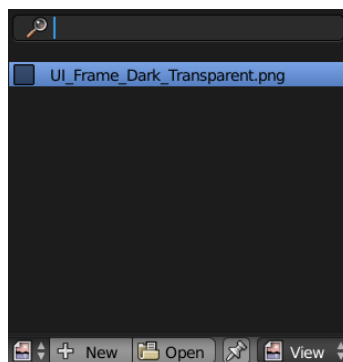


Figura 28 Seleccionar textura

El espacio de trabajo del lado derecho muestra la pirámide triangular creada, se cambió el modo *Object Mode* por *Editing Mode*. En dicho modo se seleccionó el prisma triangular pulsando la tecla A del teclado, posteriormente se pulsó la tecla U del teclado y se seleccionó la función *Unwrap* (ver figura 29)

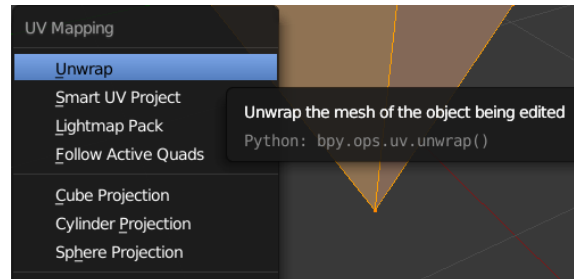


Figura 29 Función *Unwrap*

Se regresó a la vista *Default* de Blender. Se seleccionó la opción *Render>Render Image* para mostrar la pirámide triangular 3D creada (ver figura 30). Por último se guardó el archivo *.blend* del objeto creado en la carpeta *Assets* del proyecto creado anteriormente en Unity.

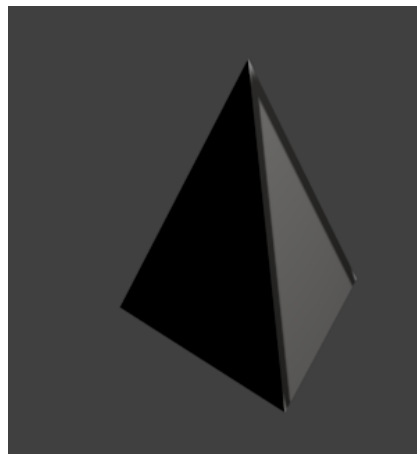


Figura 30 Modelo 3D de la pirámide triangular

El proceso realizado en el modelado de la pirámide triangular es el mismo para el resto de las pirámides modeladas, la única modificación realizada fue el número de vértices.

3.2.3 Cono

Para el modelado del cono se realizaron las siguientes acciones.

En un archivo nuevo de Blender, se eliminó el cubo 3D que por defecto Blender coloca al crear un nuevo archivo.

Se creó un cono en 3D por medio de la barra de herramientas colocada en la parte inferior izquierda, se seleccionó la función *Add>Mesh>Cone* (ver figura 31).

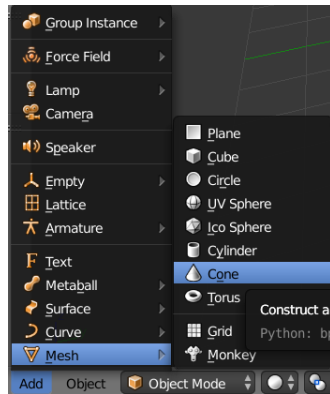


Figura 31 Agregar cono

Al crear el cono, no se modificó el número de vértices, por lo tanto, el número de vértices por defecto son 32.

No se requirió agregar un material y textura al cono dado que no es necesario resaltar el contorno creado por los vértices.

Se seleccionó la opción *Render>Render Image* para mostrar el cono 3D creado (ver figura 32). Por último se guardó el archivo *.blend* del objeto creado en la carpeta *Assets* del proyecto creado anteriormente en Unity.



Figura 32 Modelo 3D del cono

3.3 Desarrollo de la interfaz de usuario

Los recursos gráficos utilizados durante el desarrollo de la aplicación móvil se encuentran en la carpeta *Assets > Simple UI*.

A continuación, se detallan los pasos seguidos en el desarrollo de la interfaz así como la función de cada componente.

En el editor de Unity, se agregó a la escena del proyecto el objeto *Canvas* por medio de la barra de herramientas *GameObject>UI>Canvas* (ver figura 33).

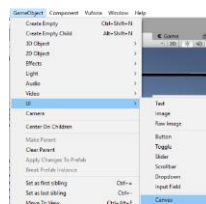


Figura 33 Agregar objeto *Canvas*

Es necesario que todos los objetos y elementos que sean parte de la aplicación de realidad aumentada, sean hijos del elemento *ARCamera*.

El objeto *Canvas* es el área donde todos los elementos de la interfaz de usuario deben estar dentro. Por lo tanto, los elementos que componen la interfaz deben ser hijos del objeto *Canvas*.

Se agregó un objeto *Panel* por medio de *GameObject>UI>Panel*, como hijo del objeto *Canvas*. Se cambió la imagen fuente por defecto del *Panel* por la imagen utilizada en la textura de los modelos 3D (ver figura 34). Se agregó otro objeto *Panel* sin modificar la imagen fuente por defecto.

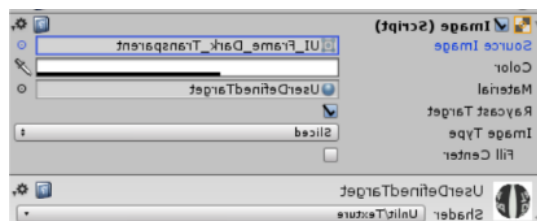


Figura 34 Propiedades de imagen del objeto *Panel*

Se agregó un objeto *Toggle* por medio de *GameObject>UI>Toggle*, como hijo del objeto *Canvas*. El *Toggle* es una casilla de verificación que le permite al usuario cambiar una opción de activo o inactivo. Fue utilizado para habilitar y deshabilitar la visualización de fórmulas. Las propiedades fueron modificadas por medio de la ventana *Rect Transform* (ver figura 35). Se le asignó el nombre de *Toggle_Fomulas* en el editor de Unity.

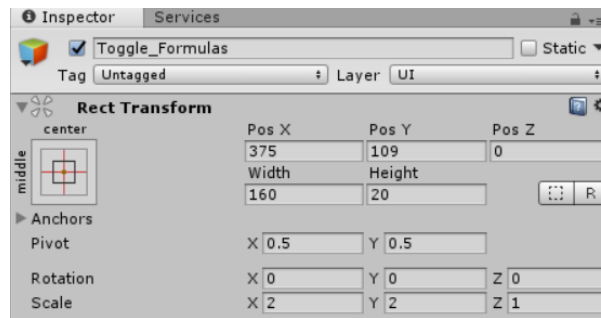


Figura 35 Propiedades *Toggle_Fomulas*

Se agregó otro objeto *Toggle*. Se agregó una imagen por medio de *GameObject>UI>Image* como hijo del objeto *Toggle* y se agregó otra imagen como hijo de la imagen anterior. Fue utilizado para habilitar y deshabilitar la rotación horizontal de las figuras geométricas aumentadas. La modificación de propiedades se realizó de la misma manera que el *Toogle* agregado anteriormente. Se le asignó el nombre de *Toggle_rh* en el editor de Unity.

Se agregó otro objeto *Toggle*. Se agregó una imagen por medio de *GameObject>UI>Image* como hijo del objeto *Toggle* y se agregó otra imagen como hijo de la imagen anterior. Se modificó la imagen fuente del objeto *Image* como se muestra en la figura 36. El procedimiento fue el mismo para el segundo objeto *Image*. Fue utilizado para habilitar y deshabilitar la rotación vertical de las figuras geométricas aumentadas. Se le asignó el nombre de *Toggle_rv* en el editor de Unity.

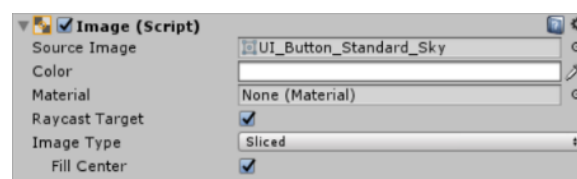


Figura 36 Propiedades de imagen del objeto *Toggle*

Se agregó otro objeto *Toggle*. Se agregó una imagen por medio de *GameObject>UI>Image* como hijo del objeto *Toggle* y se agregó otra imagen como hijo de la imagen anterior. El proceso de cambio de imagen fuente de los objetos *Image* es el mismo para el resto de los *Toggle* agregados. Fue utilizado para habilitar y deshabilitar el aumento de tamaño de las figuras geométricas aumentadas. Se le asignó el nombre de *Toggle_a* en el editor de Unity.

Se agregó otro objeto *Toggle*. Se agregó una imagen por medio de *GameObject>UI>Image* como hijo del objeto *Toggle* y se agregó otra imagen como hijo de la imagen anterior. Fue utilizado para habilitar y deshabilitar la reducción de tamaño de las figuras geométricas aumentadas. Se le asignó el nombre de *Toggle_r* en el editor de Unity.

Se agregó un objeto *Button* por medio de *GameObject>UI>Button*, como hijo del objeto *Canvas*. El *Button* responde a un clic del usuario y es utilizado para ejecutar una acción. Fue utilizado reestablecer los valores establecidos a las figuras geométricas. Se le asignó el nombre de *btnReset* en el editor de Unity.

Se agregó otro objeto *Button*. Fue utilizado para cerrar la aplicación. Se le asignó el nombre de *btnSalir* en el editor de Unity.

Se agregó otro objeto *Button*. Fue utilizado para abrir otra escena en la cual se muestran los créditos, un botón para ver un video tutorial y un botón para regresar a la aplicación. Se le asignó el nombre de *btnAyuda* en el editor de Unity.

En la figura 37, se muestra un diagrama con los valores de las propiedades que fueron modificadas de los elementos que componen la interfaz de usuario.

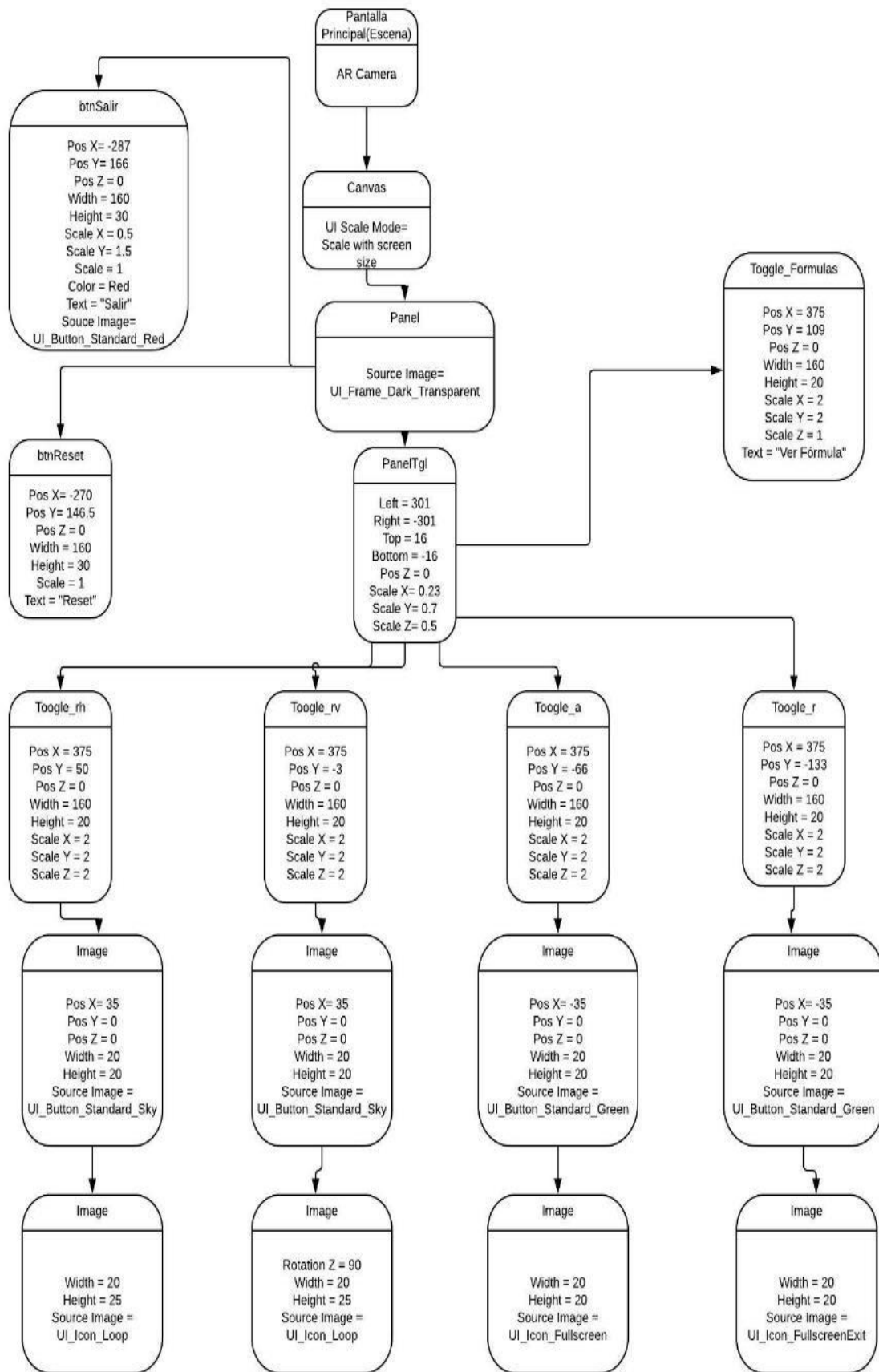


Figura 37 Valores modificados a los elementos de la interfaz

La figura 38, muestra la vista del panel de administración de la escena de los elementos y sus respectivos elementos hijos.

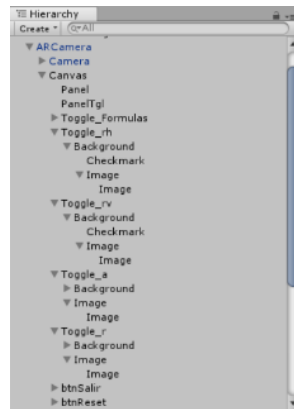


Figura 38 Vista jerárquica de los elementos

3.4 Marcadores de realidad aumentada

En la creación de marcadores de realidad aumentada, se hizo uso de la aplicación web *Augmented Reality Marker Generator* propiedad de Brosvision, la cual genera de manera aleatoria una imagen con líneas, triángulos y cuadrados dando como resultado una gran cantidad de puntos rastreables. Para generar la imagen se accedió al sitio <http://www.brosvision.com/ar-marker-generator/> y se deseleccionó la opción de color, se dejó activa las opciones de línea, triángulos y cuadrados. Se guardó la imagen dando clic derecho sobre la misma y se seleccionó la opción Guardar Imagen como (ver figura 39).

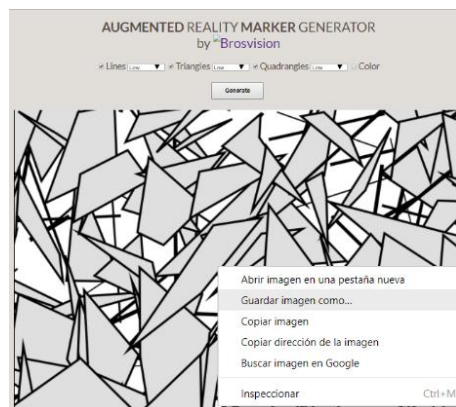


Figura 39 Generación de marcador de realidad aumentada

Se realizó el procedimiento anterior para la generación de las doce imágenes que corresponde al mismo número de figuras geométricas que se aumentan en la aplicación móvil.

Para poder utilizar las imágenes generadas como marcadores, es necesario agregarlas a la base de datos de Vuforia, la cual fue creada en la etapa de configuración de Vuforia. Antes de agregar las imágenes a la base de datos, se procedió a re escalar las imágenes utilizando el programa Paint que viene integrado al sistema operativo Windows. Simplemente se abrió la imagen en formato .PNG y se guardó el formato .JPG, es necesario realizar el procedimiento para que las imágenes sean compatibles con la base de datos de Vuforia.

Se accedió a la base de datos de nombre *target* en el panel de administración de Vuforia y se inició el proceso para agregar la imagen haciendo clic en el botón *Add Target* (ver figura 40).

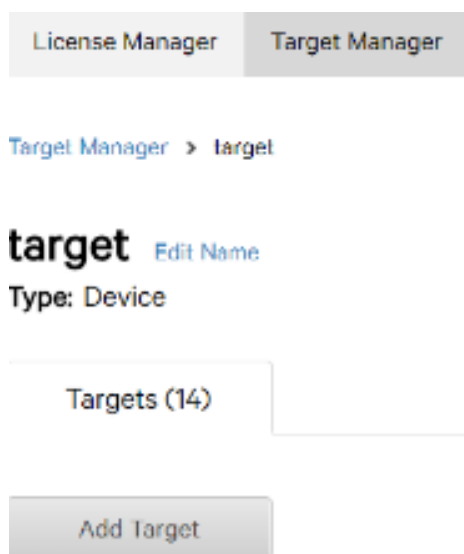


Figura 40 Agregar imagen a la base de datos

Para agregar la imagen, se muestra el formulario de la figura 41. Se seleccionó *Single Image*, en el apartado *File* se buscó y seleccionó la imagen en formato .JPG, en el apartado *Width* se le asignó el valor 10 y en el apartado *Name* se toma de manera automática el nombre del archivo seleccionado. Para finalizar el proceso se le selecciono el botón *Add*.

Add Target

Type:

Single Image Cuboid Cylinder 3D Object

File:

Choose File

(jpg or png (max file 2MB))

Width:

Enter the width of your target in scene units. The size of the target should be on the same scale as your augmented content. Vuforia uses meters as the default unit scale. The target's height will be calculated when you upload your image.

Name:

Name must be unique to a database. When a target is detected in your application, this will be reported in the API.

Figura 41 Proceso para agregar imagen a la base de datos

El procedimiento anterior, se repitió para las doce imágenes guardadas. En el panel de administración de base de datos de Vuforia, se muestran las imágenes que fueron agregadas. El atributo *rating* muestra la calidad rastreable de la imagen siendo 5 estrellas la mejor calificación para una imagen (ver figura 42).

<input type="checkbox"/>		conoAR	Single Image	★★★★★	Active
<input type="checkbox"/>		cilindroAR	Single Image	★★★★★	Active
<input type="checkbox"/>		triaAR	Single Image	★★★★★	Active
<input type="checkbox"/>		pira6AR	Single Image	★★★★★	Active
<input type="checkbox"/>		pira5AR	Single Image	★★★★★	Active
<input type="checkbox"/>		pira4AR	Single Image	★★★★★	Active
<input type="checkbox"/>		pira3AR	Single Image	★★★★★	Active
<input type="checkbox"/>		hexaAR	Single Image	★★★★★	Active
<input type="checkbox"/>		pentaAR	Single Image	★★★★★	Active
<input type="checkbox"/>		cuadAR	Single Image	★★★★★	Active
<input type="checkbox"/>		esferaAR	Single Image	★★★★★	Active
<input type="checkbox"/>		cuboAR	Single Image	★★★★★	Active

Figura 42 Base de datos de Vuforia

Al descargar la base de datos de Vuforia se presenta la opción de descargar la base de datos para Android Studio, Xcode, Visual Studio o Unity Editor (ver figura 43), para este proyecto se selección Unity.

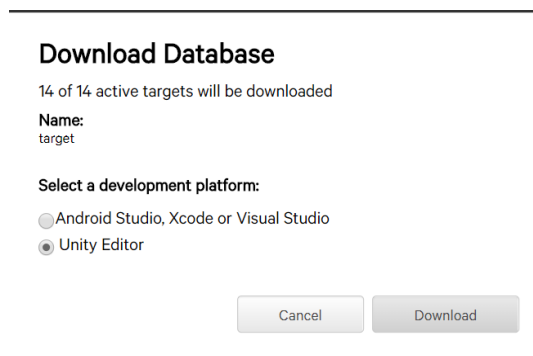


Figura 43 Descarga de la base de datos

Se agregó al proyecto en Unity por medio de *Assets > Import Package > Custom Package*, se buscó el archivo de base de datos descargado y se seleccionó.

En el explorador de archivos de Unity, se seleccionó la carpeta *Assets>Vuforia>Prefabs*. Se seleccionó el objeto *ImageTarget* y se agregó al proyecto arrastrándolo y soltándolo dentro de la escena (ver figura 44). Una vez agregado se colocó como hijo de *ARCamera*.

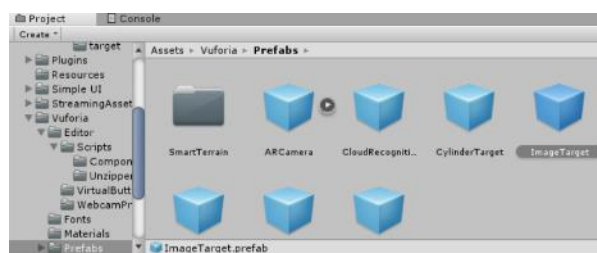


Figura 44 Prefabs de la extensión Vuforia

Se renombró con el nombre de *ImageTargetCubo*. Se procedió a la asignación de la imagen de la base de datos de Vuforia. En el panel *Inspector* se modificaron los valores en *Image Target Behaviour*, seleccionando la base de datos target en el campo *Database*. En el campo *Image Target* se seleccionó la imagen *cuboAR*. (Ver figura 45)

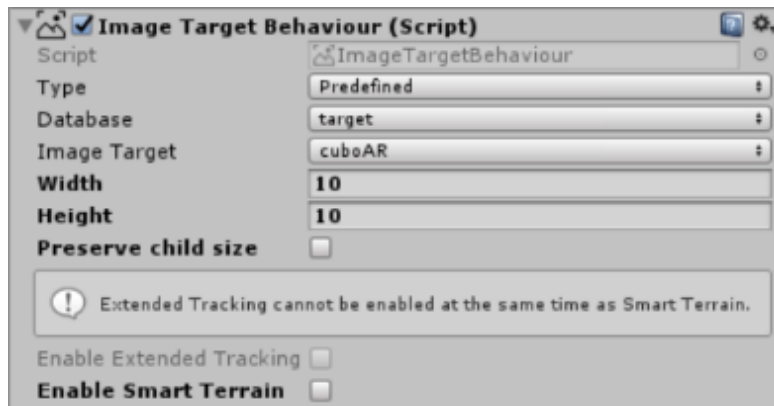


Figura 45 Selección de imagen para el objeto *Image Target*

La posición del marcador dentro de la escena se asignó como se muestra en la figura 46.

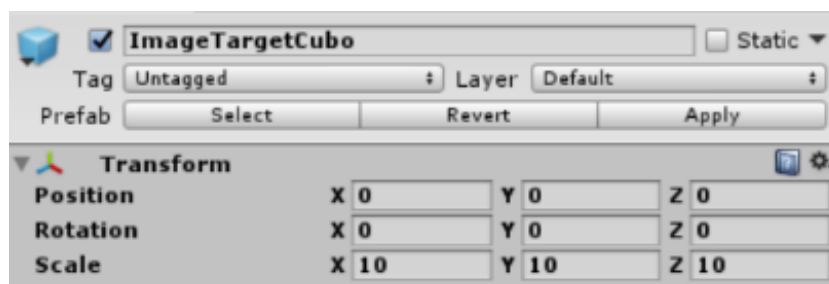


Figura 46 Valores de posición del marcador

El proceso anterior se realizó para los doce marcadores utilizados en la aplicación móvil. En la figura 47, se muestran los nombres de cada marcador, valores de posición y el nombre de la imagen seleccionada de la base de datos.

ImageTargetCubo	<ul style="list-style-type: none"> • Pos X=0, Pos Y= 0, Pos Z= 0 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = cuboAR
ImageTargetEsfera	<ul style="list-style-type: none"> • Pos X=14.79, Pos Y= 0, Pos Z= 0 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = esferaAR
ImageTargetCono	<ul style="list-style-type: none"> • Pos X=-12.47, Pos Y= 0, Pos Z= -34.56 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = conoAR
ImageTargetCilindro	<ul style="list-style-type: none"> • Pos X=-12.56, Pos Y= 0, Pos Z= -17.46 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = cilindroAR
ImageTargetTria	<ul style="list-style-type: none"> • Pos X=0, Pos Y= 0, Pos Z= -17.11 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = triaAR
ImageTargetCuad	<ul style="list-style-type: none"> • Pos X=0, Pos Y= 0, Pos Z= -33.61 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = cuadAR
ImageTargetPenta	<ul style="list-style-type: none"> • Pos X=0, Pos Y= 0, Pos Z= -50.07 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = pentaAR
ImageTargetHexa	<ul style="list-style-type: none"> • Pos X=-12.6, Pos Y= 0, Pos Z= -0.3 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = hexaAR
ImageTargetPira3	<ul style="list-style-type: none"> • Pos X=0, Pos Y= 0, Pos Z= 0 • Rot X=14.94, Rot Y= 0, Rot Z=-16.95 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = pira3AR
ImageTargetPira4	<ul style="list-style-type: none"> • Pos X=15, Pos Y= 0, Pos Z= -32.16 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = pira4AR
ImageTargetPira5	<ul style="list-style-type: none"> • Pos X=15.78, Pos Y= 0, Pos Z= -51.16 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = pira5AR
ImageTargetPira6	<ul style="list-style-type: none"> • Pos X=27.36, Pos Y= 0, Pos Z= 0 • Rot X=0, Rot Y= 0, Rot Z=0 • Scale X= 10, Scale Y= 10, Scale Z = 10 • Image Target = pira6AR

Figura 47 Propiedades de cada uno de los marcadores creados

3.5 Integración de modelos 3D y marcadores de realidad aumentada

A continuación, se detalla el procedimiento realizado para la integración de las figuras geométricas 3D y los marcadores de realidad aumentada creados.

3.5.1 Cubo

En el editor de Unity, se agregó un cubo 3D por medio de *GameObject>3D Object>Cube* (ver figura 48).

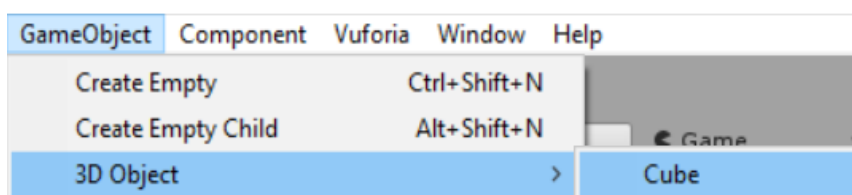


Figura 48 Agregar cubo

Se colocó el cubo 3D como hijo del marcador *ImageTargetCubo*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 49.

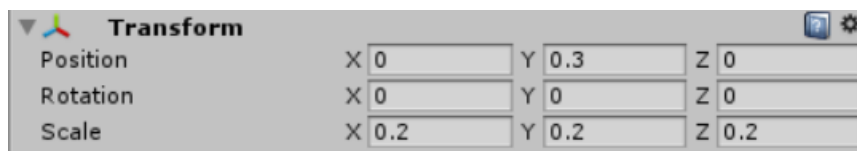


Figura 49 Propiedades del cubo 3D

Al colocar los objetos 3D como hijos de los marcadores de realidad aumentada, los ejes de la posición se colocan en relación a los ejes del marcador y no con los ejes de la escena del proyecto, siendo $X = 0$ el punto central del marcador, Y se refiere a la separación vertical entre el marcador y el objeto 3D, $Z=0$ es la profundidad a la que se encuentra el objeto 3D.

El material utilizado para el cubo 3D fue *DefaultTarget*, el cual se modificó como se muestra en la figura 50. Se seleccionó modificando el campo *Element 0*.

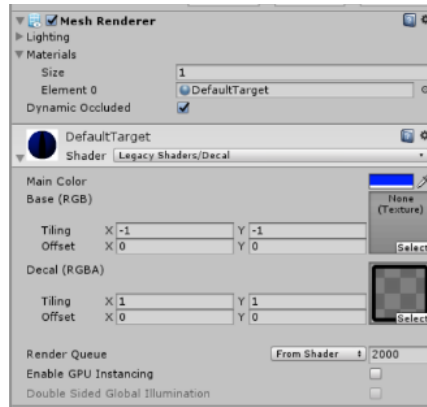


Figura 50 Material del cubo 3D

El resultado del proceso anterior se puede ver en la figura 51.

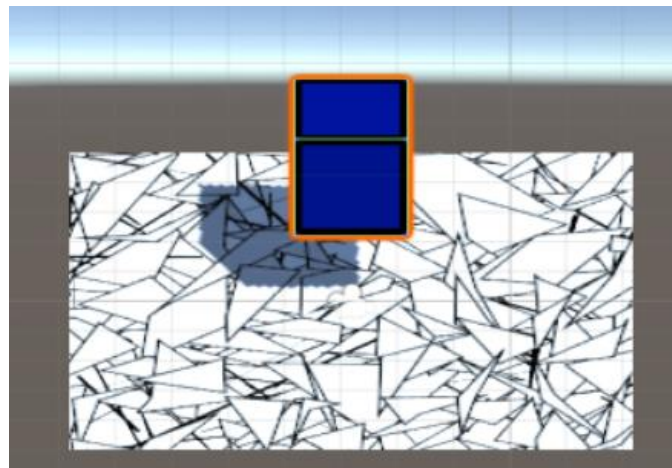


Figura 51 Cubo 3D sobre su marcador de realidad aumentada

Se realizó el mismo procedimiento para la esfera y el cilindro, siendo la selección del objeto por medio de *GameObject>3D Object>Sphere* y *GameObject>3D Object>Cylinder* respectivamente, la única diferencia en el procedimiento.

3.5.2 Cono

El cono 3D no se encuentra prefabricado en Unity, por lo que se modeló en Blender. Los modelos creados en Blender fueron guardados directamente en la carpeta *Assets* del proyecto. Unity reconoce directamente el formato *.blender* por lo que no es necesario realizar la importación de los modelos. En el explorador de archivos del

proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo cono, se arrastró y soltó en la escena del proyecto (ver figura 52).

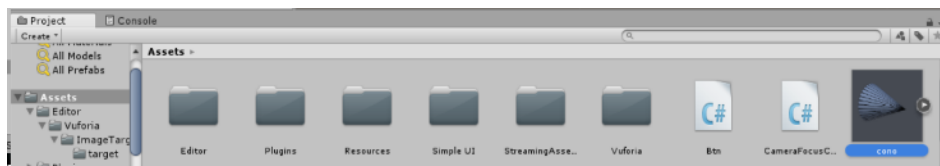


Figura 52 Selección del modelo cono en el explorador

Se colocó como hijo del marcador *ImageTargetCono*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 53.

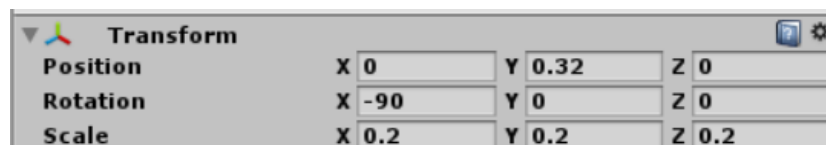


Figura 53 Valores del cono 3D

El valor de rotación en X es modificado automáticamente por Unity asignándole el valor de -90 debido a que Blender tiene los ejes de posición colocados de una manera diferente a Unity, por lo que Unity corrige automáticamente esa diferencia en la rotación. Lo anterior aplica para todas las figuras geométricas que fueron modeladas en Blender.

Se creó un nuevo material por medio de *Assets>Create>Material*. Se renombró como *conomat*. Se modificó el color del material como se muestra en la figura 54 y el resultado se aprecia en la figura 55.

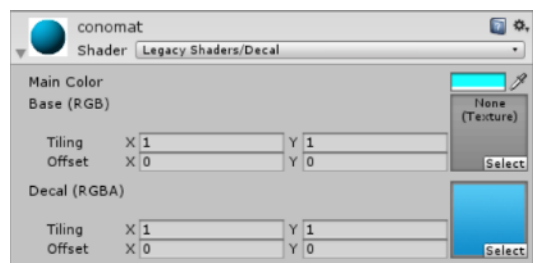


Figura 54 Material de cono

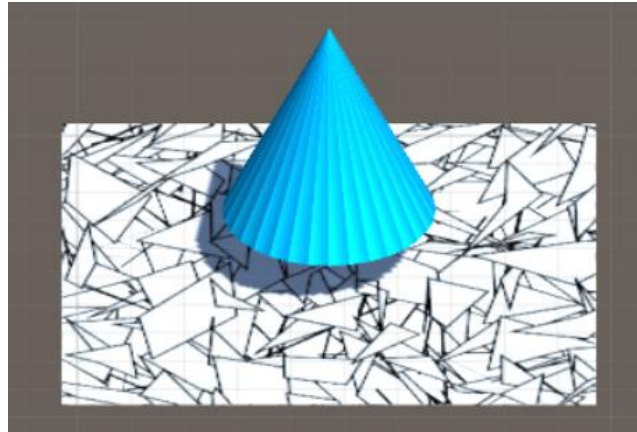


Figura 55 Cono 3D sobre su marcador de realidad aumentada

3.5.3 Prisma triangular

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *tria*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetTria*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 56.

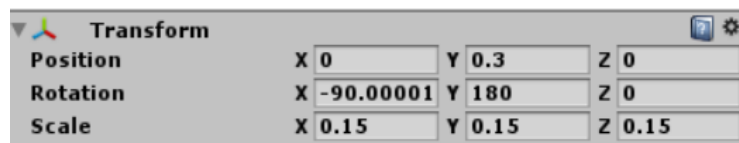


Figura 56 Valores del prisma triangular 3D

Se creó un nuevo material por medio de *Assets>Create>Material*. Se renombró como *triamat*. Se modificó el color del material como se muestra en la figura 57 y el resultado se aprecia en la figura 58.

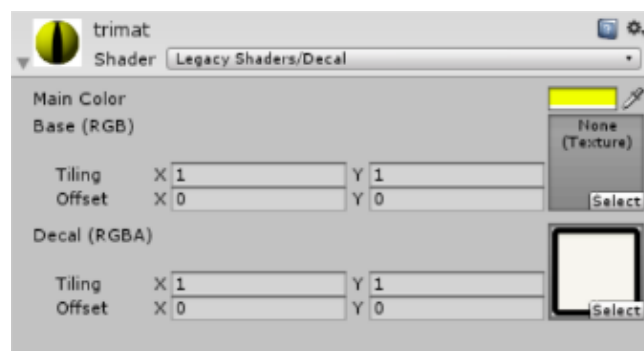


Figura 57 Material del prisma triangular

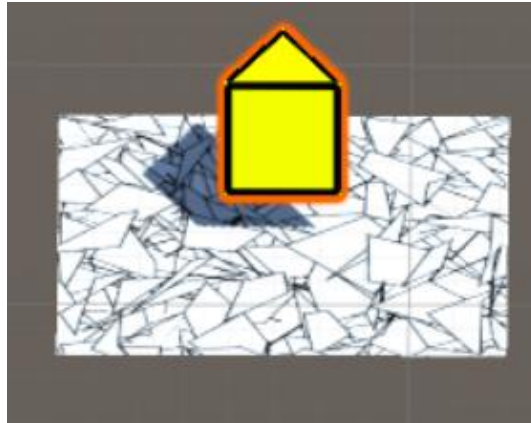


Figura 58 Prisma triangular 3D sobre su marcador de realidad aumentada

En el anexo 2 se detalla el procedimiento para el resto de los prismas y pirámides.

3.6 Botones virtuales

El SDK de Vuforia incluye botones virtuales los cuales proporcionan al usuario una interacción más inversiva que los botones táctiles en pantalla ya que se integran al marcador de realidad aumentada.

Se integraron cuatro botones virtuales a cada marcador de realidad aumentada, cada botón realiza una función al ser presionado. Las funciones que realizan son las siguientes: rotación sobre el eje Y, rotación sobre el eje X, aumento de escala, reducción de escala.

Los botones son habilitados y deshabilitados por medio de las casillas de verificación *Toggle* en la interfaz de usuario.

A continuación, se detalla el proceso de integración de botones virtuales al marcador de realidad aumentada del cubo 3D.

En el explorador de archivos de Unity, se abrió la carpeta *Assets>Vuforia>Prefabs*. Se seleccionó el objeto *VirtualButton*, se arrastró y soltó a la escena (ver figura 59). Se realizó la acción anterior cuatro veces para agregar los cuatro botones que realizan las cuatro funciones mencionadas anteriormente.

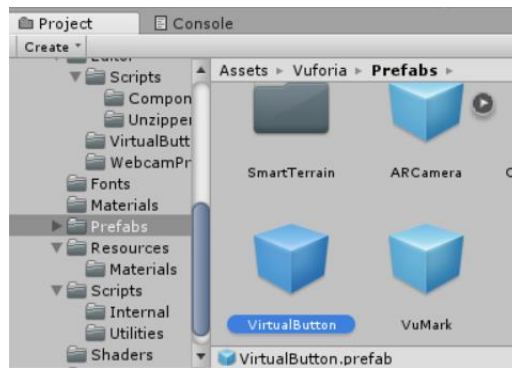


Figura 59 Agregar objeto *VirtualButton*

Se colocaron los botones virtuales agregados como hijos del marcador *ImageTargetCubo*. Se renombraron como: *Rotar*, *RotarV*, *Aumentar* y *Reducir* respectivamente. En la figura 60, se muestra la vista jerárquica del marcador desde el panel de administración de la escena.



Figura 60 Vista jerárquica del marcador

Se modificaron los valores de posición, rotación y escala como se muestra en la figura 61, por lo que los cuatro botones se colocaron uno encima de otro.

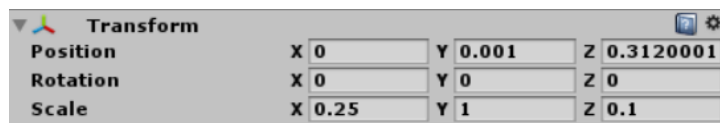


Figura 61 Valores del botón virtual

El eje X= 0 indica que el botón está centralizado, el eje Y=0.001 indica a que el botón se encuentra levemente arriba del marcador, el eje Z= 0.312 indica que el botón se encuentra detrás del objeto 3D aumentado. Se colocó en dicha posición sobre el eje Z y no centralizado para que al momento en que el usuario realice algún gesto o movimiento sobre el objeto el botón virtual sea presionado y soltado al concluir el gesto,

con lo cual se simula una interacción más real con el objeto 3D sin la necesidad de usar un sistema de reconocimiento de gestos.

El procedimiento para agregar los botones al resto de los marcadores es igual al detallado anteriormente, los valores de posición, rotación y escala de igual manera son los mismos dado que los botones son colocados como hijos de sus respectivos marcadores, por lo tanto la posición y rotación va de acuerdo al marcador en el que son colocados como hijos. En la figura 62 se muestran los valores de los botones virtuales con los respectivos nombres asignados.

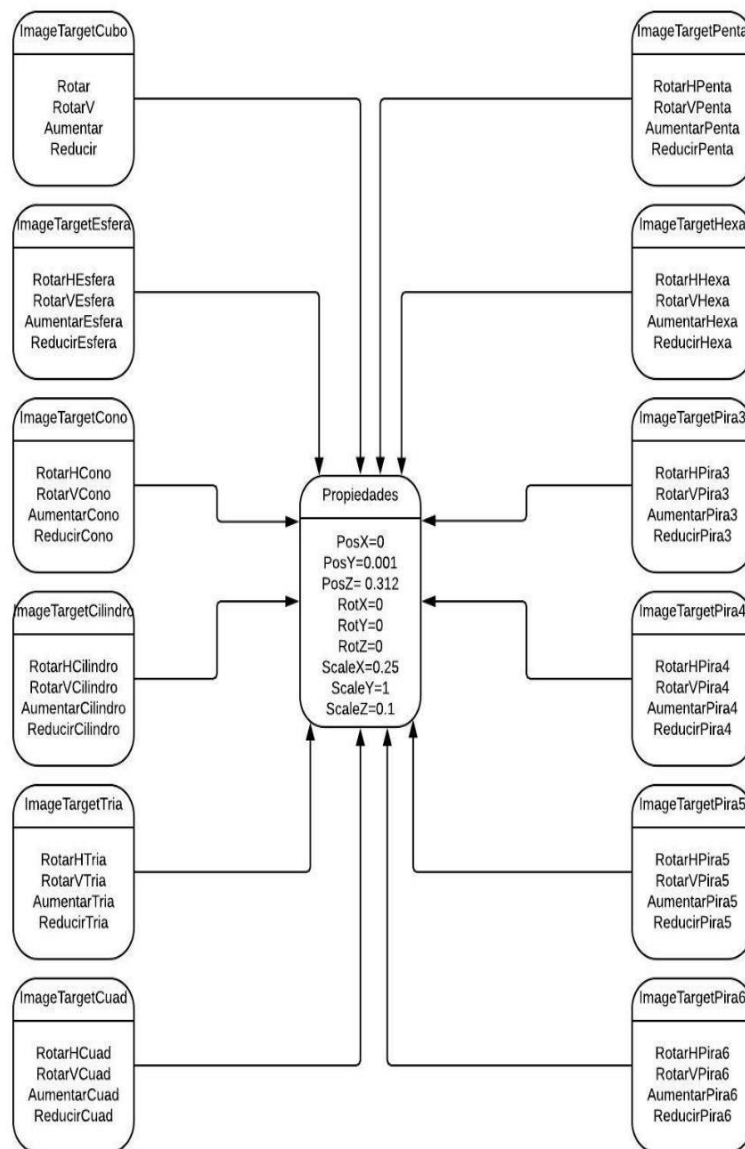


Figura 62 Propiedades de los botones virtuales de cada uno de los marcadores

3.7 Codificación de scripts

Los *scripts* fueron escritos en el lenguaje de programación C# en Microsoft Visual Studio. Se codificaron *scripts* de rotación en eje X, rotación en eje Y, aumento de escala y reducción de escala de las figuras geométricas los cuales son ejecutados por los botones virtuales integrados al marcador. También fueron escritos *scripts* para el autoenfoco de la cámara trasera, habilitar y deshabilitar la visualización de los botones virtuales, cerrar la aplicación, reestablecer los valores de posición, rotación y escala de las figuras.

La creación de un nuevo *script* se realiza por medio de *Assets>Create>C# Script* (ver figura 63).

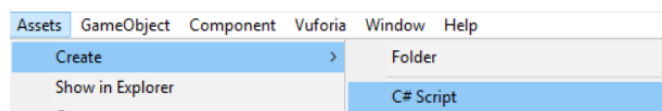


Figura 63 Crear C# *script*

A continuación se presenta y detalla el código fuente de los *scripts*.

3.7.1 Botones virtuales cubo

3.7.1.1 Rotación en eje Y

El *script* de rotación en eje Y consiste en que la figura geométrica gire sobre su eje Y cuando el botón virtual es presionado. En la figura 64 se muestra la parte del código que se encarga de realizar la acción. El método *OnButtonPressed* es el encargado de registrar que el botón fue presionado. La función *transform.Rotate* es utilizada para la rotación del objeto, *Vector3* es utilizado para manejar ángulos y posiciones 3D. *Time.deltaTime* es el tiempo en segundos que tarda en completar el último cuadro por segundo, el valor de *Time.deltaTime* es determinado por los cuadros por segundo a los que funciona la aplicación. La aplicación funciona a 60 cuadros por segundo.

Se realizó la siguiente operación para determinar el valor en grados que rota la figura, $(1/60)*1000$, siendo $(1/60)$ el valor de *Time.deltaTime*, por lo tanto el resultado aproximado es de 16.67, dicho resultado se asignó a la posición del eje Y, por lo que

cada vez que es presionado el botón virtual, la figura geométrica rota aproximadamente 16.67 grados sobre su eje Y.

```
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Cube.transform.Rotate(new Vector3(0, Time.deltaTime * 1000, 0));
}
```

Figura 64 Código rotación del cubo en eje Y

3.7.1.2 Rotación en eje X

El código utilizado para la rotación en eje X de la figura es relativamente igual al de rotación en eje Y, el único cambio es la posición de la instrucción *Time.deltatime * 1000*, la cual se colocó en la posición del eje X, de igual manera rota aproximadamente 16.67 pero sobre el eje X. La figura 65 muestra la parte del código que realiza dicha acción.

```
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Cube.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0, 0));
}
```

Figura 65 Código rotación del cubo en eje X

3.7.1.3 Aumentar escala

El *script* de aumentar escala consiste en que la figura geométrica aumente de tamaño cuando el botón virtual es presionado. En la figura 66 se muestra la parte del código que se encarga de realizar la acción. La función *transform.localScale* es utilizada para cambiar la escala del objeto respecto a su tamaño original. Se asignó el valor *0.05F* a los ejes X, Y, Z para lograr un aumento de escala uniforme. La letra F después del número es para señalar que es de tipo flotante.

```
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Cube.transform.localScale += new Vector3(0.05F, 0.05F, 0.05F);
}
```

Figura 66 Código para aumentar el tamaño del cubo

3.7.1.4 Reducir escala

El código utilizado para reducir la escala de la figura es relativamente igual al de aumentar escala, los cambios realizados son el cambio del operador de adición por el de sustracción y el valor asignado para reducción es de $0.04F$. La figura 67 muestra la parte del código que realiza la acción.

```
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Cube.transform.localScale -= new Vector3(0.04F, 0.04F, 0.04F);
}
```

Figura 67 Código para reducir el tamaño del cubo

Los *scripts* de los botones virtuales de las figuras geométricas restantes son relativamente iguales a los anteriormente descritos, los cambios únicos cambios realizados son: nombre de *script*, nombre de la variable utilizada para el objeto geométrico, nombres de los botones virtuales que se asignan a la variable *VButtonObject* y nombre del objeto geométrico asignado a la variable utilizada para dicho botón. El resto del código fuente de los *scripts* de los botones virtuales se encuentra en el Anexo 3.

3.7.2 Habilitar/deshabilitar la visualización de los botones virtuales (ToggleScript)

Se escribió un *script* en el cual se habilita y deshabilita la visualización de los botones virtuales cuando una casilla de verificación de la interfaz de usuario es activada y desactivada. El código es extenso por lo que solo se detallaran los métodos de habilitación y des habilitación, sin embargo el código fuente completo se encuentra en el anexo 4 con comentarios entre líneas para señalar a que se refiere cada parte del código.

public void T_cambio().- El método *T_cambio()* consiste de una estructura condicional *if*, en el cual se valida si la casilla de verificación *Toggle_rh* de la interfaz de usuario esta activa entonces habilita los botones virtuales de rotación en eje Y, al mismo tiempo que deshabilita el resto de los botones por lo que si se cambia de marcador de realidad aumentada estarán activos los botones de rotación en eje Y sin necesidad de activarlos

cada vez que se cambie de marcador, en caso de que la sea la casilla de verificación sea desactivada, se deshabilitaran todos los botones virtuales.

public void T_cambioV().- El método *T_cambioV()* consiste de una estructura condicional *if*, en el cual se valida si la casilla de verificación *Toggle_rv* de la interfaz de usuario esta activa entonces habilita los botones virtuales de rotación en eje X, al mismo tiempo que deshabilita el resto de los botones por lo que si se cambia de marcador de realidad aumentada estarán activos los botones de rotación en eje X sin necesidad de activarlos cada vez que se cambie de marcador, en caso de que la sea la casilla de verificación sea desactivada, se deshabilitaran todos los botones virtuales.

public void T_cambioA().- El método *T_cambioA()* consiste de una estructura condicional *if*, en el cual se valida si la casilla de verificación *Toggle_a* de la interfaz de usuario esta activa entonces habilita los botones virtuales de aumento de escala, al mismo tiempo que deshabilita el resto de los botones por lo que si se cambia de marcador de realidad aumentada estarán activos los botones de aumento de escala sin necesidad de activarlos cada vez que se cambie de marcador, en caso de que la sea la casilla de verificación sea desactivada, se deshabilitaran todos los botones virtuales.

public void T_cambioR().- El método *T_cambioR()* consiste de una estructura condicional *if*, en el cual se valida si la casilla de verificación *Toggle_r* de la interfaz de usuario esta activa entonces habilita los botones virtuales de reducción de escala, al mismo tiempo que deshabilita el resto de los botones por lo que si se cambia de marcador de realidad aumentada estarán activos los botones de reducción de escala sin necesidad de activarlos cada vez que se cambie de marcador, en caso de que la sea la casilla de verificación sea desactivada, se deshabilitaran todos los botones virtuales.

3.7.3 Cerrar aplicación (ExitScript)

Un *script* simple escrito para cerrar la aplicación y terminar el proceso para evitar que la aplicación siga funcionando en el fondo. El método *ExitApp()* cierra la aplicación ejecutando la instrucción *Application.Quit()*. La figura 68 muestra el código.

```

public void ExitApp()
{
    Debug.Log("Leaving App...");
    Application.Quit();
}

```

Figura 68 Código para cerrar la aplicación

3.7.4 Reestablecer valores de objetos geométricos (ResetScript)

El método *ResetBtn()* ejecuta las instrucciones para regresar los valores de las figuras geométricas a las establecidas inicialmente después de haber sido modificadas por la ejecución de los métodos establecidos en los botones virtuales. La función *localRotation* establece el valor del ángulo de cada eje y Unity guarda las rotaciones en *cuaterniones*. La figura 69 muestra el código con las instrucciones.

```

cubo.transform.localRotation = Quaternion.Euler(0, 0, 0); //reestablece la posicion del cubo
cubo.transform.localScale = new Vector3(0.2f, 0.2f, 0.2f); //reestablece la rotacion del cubo

esfera.transform.localRotation = Quaternion.Euler(0, 0, 0); //reestablece la posicion de la esfera
esfera.transform.localScale = new Vector3(0.2f, 0.2f, 0.2f); //reestablece la rotacion de la esfera

pris3.transform.localRotation = Quaternion.Euler(-90, 180, 0); //reestablece la posicion del prisma triangular
pris3.transform.localScale = new Vector3(0.15f, 0.15f, 0.15f); //reestablece la rotacion del prisma triangular

pris4.transform.localRotation = Quaternion.Euler(-90, 45, 0); //reestablece la posicion del prisma cuadrangular
pris4.transform.localScale = new Vector3(0.15f, 0.15f, 0.15f); //reestablece la rotacion del prisma cuadrangular

pris5.transform.localRotation = Quaternion.Euler(-90, 105, 0); //reestablece la posicion del prisma pentagonal
pris5.transform.localScale = new Vector3(0.15f, 0.15f, 0.15f); //reestablece la rotacion del prisma pentagonal

pris6.transform.localRotation = Quaternion.Euler(-90, 0, 0); //reestablece la posicion del prisma hexagonal
pris6.transform.localScale = new Vector3(0.11f, 0.11f, 0.11f); //reestablece la rotacion del prisma hexagonal

pira3.transform.localRotation = Quaternion.Euler(-90, 60, 0); //reestablece la posicion de la piramide triangular
pira3.transform.localScale = new Vector3(0.17f, 0.17f, 0.17f); //reestablece la rotacion de la piramide triangular

pira4.transform.localRotation = Quaternion.Euler(-90, 45, 0); //reestablece la posicion de la piramide cuadrangular
pira4.transform.localScale = new Vector3(0.17f, 0.17f, 0.17f); //reestablece la rotacion de la piramide cuadrangular

pira5.transform.localRotation = Quaternion.Euler(-90, 35, 0); //reestablece la posicion de la piramide pentagonal
pira5.transform.localScale = new Vector3(0.17f, 0.17f, 0.17f); //reestablece la rotacion de la piramide pentagonal

pira6.transform.localRotation = Quaternion.Euler(-90, 30, 0); //reestablece la posicion de la piramide hexagonal
pira6.transform.localScale = new Vector3(0.17f, 0.17f, 0.17f); //reestablece la rotacion de la piramide hexagonal

cono.transform.localRotation = Quaternion.Euler(-90, 0, 0); //reestablece la posicion del cono
cono.transform.localScale = new Vector3(0.15f, 0.15f, 0.15f); //reestablece la rotacion del cono

cilindro.transform.localRotation = Quaternion.Euler(0, 0, 0); //reestablece la posicion del cilindro
cilindro.transform.localScale = new Vector3(0.2f, 0.2f, 0.2f); //reestablece la rotacion del cilindro

```

Figura 69 Código para restablecer valores iniciales

3.7.5 Autoenfoco de la cámara (Camera Focus Controller)

El método *SetAutofocus()* ejecuta la instrucción para que la cámara del dispositivo móvil Android este continuamente enfocando y realice la detección del marcador de realidad aumentada eficazmente, esto es debido a que algunos dispositivos pierden la capacidad de autoenfoco por lo que la detección del marcador se dificulta y evita que las figuras geométricas sean aumentadas. En la figura 70, se aprecia el código del método.

```
private void SetAutofocus()
{
    if (CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_MODE_CONTINUOUSAUTO))
    {
        Debug.Log("Autofocus set");
    }
}
```

Figura 70 Código de autoenfoco

3.8 Integrar scripts al proyecto en Unity

Para integrar un *script* al proyecto en Unity se selecciona el archivo *.cs*, se arrastra y suelta dentro del objeto en el cual se realiza su ejecución en el proyecto.

Los scripts *ToggleScript*, *ExitScript*, *ResetScript* y *Camera Focus Controller* se agregaron al objeto *ARCamera* (ver figura 71).

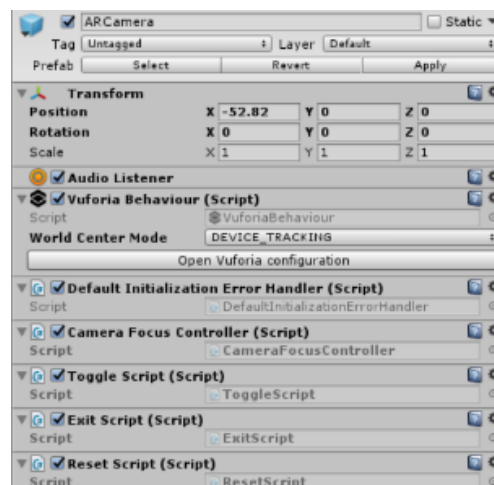


Figura 71 Integración de *scripts*

En el caso de los *scripts* de los botones virtuales, se agregaron los *scripts* de rotación en eje Y, rotación en eje X, aumento de escala y reducción de escala al marcador que corresponde. La figura 72 muestra los *scripts* dentro del marcador *ImageTargetCubo*.

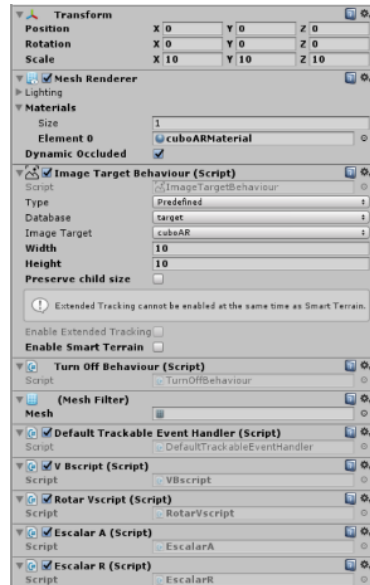


Figura 72 *Scripts* del marcador del cubo

El procedimiento anterior se aplicó para las doce figuras geométricas que aumenta la aplicación móvil.

Para la asignación del *script* de cerrar la aplicación, se seleccionó el botón *btnSalir*, en el panel *Inspector* se agregó un campo a la función *OnClick()*, en dicho campo se seleccionó el objeto *ARCamera* de la escena y se seleccionó el método *ExitScript>ExitApp()* (ver figura 73).

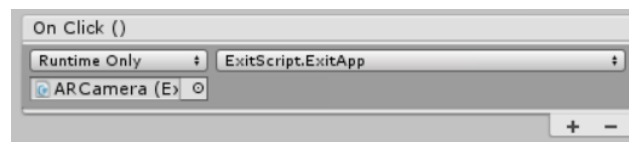


Figura 73 Asignación de método

Para la asignación del *script* de reestablecer, se seleccionó el botón *btnReset*, en el panel *Inspector* se agregó un campo a la función *OnClick()*, en dicho campo se

seleccionó el objeto *ARCamera* de la escena y se seleccionó el método *ResetScript>ResetBtn()* (ver figura 74).

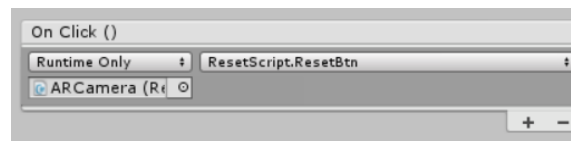


Figura 74 Asignación de método

3.9 Inhabilitación de botones virtuales

El *ToggleScript* se encarga de deshabilitar la visualización de los botones virtuales, sin embargo no desactiva su funcionamiento. Para lograr la completa inhabilitación de los botones virtuales se complementó con la función *OnValueChanged(Boolean)* de la casilla de verificación (*Toggle*). A continuación se muestra el procedimiento seguido para habilitar e inhabilitar los botones virtuales del marcador del cubo 3D.

3.9.1 Rotación en eje Y (*Toggle_rh*)

Se seleccionó el *Toggle_rh* en la escena del proyecto, en el panel *Inspector* se muestra la función *OnValueChanged(Boolean)*, se agregó un campo con el botón + que se encuentra en dicha función. Debajo de la lista desplegable *Runtime* se seleccionó el objeto *ARCamera* (ver figura 75) de la escena del proyecto para tener acceso a los métodos de los scripts integrados a dicho objeto.

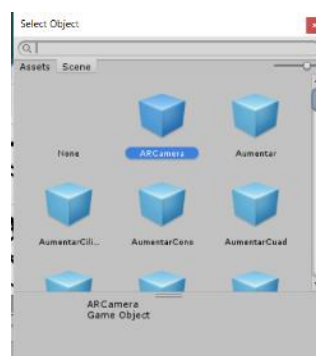


Figura 75 Selección de objeto *ARCamera*

En la lista desplegable colocada a la derecha se muestran los métodos de los *scripts*, se seleccionó *ToggleScript>T_cambio()*. Al asignar dicho método, cuando el *Toggle* sea activado/desactivado cambiará el estado de falso/verdadero con el cual el método

T_Cambio, realiza la habilitación/des habilitación de la visualización del botón virtual de rotación en eje Y.

Para habilitar/inhabilitar el funcionamiento del botón virtual se agregaron cuatro campos más a la función *OnValueChanged(Boolean)*. En cada campo se seleccionó el botón virtual de rotación en eje Y (*Rotar*), rotación en eje X (*RotarV*), aumentar escala (*Aumentar*) y reducir escala (*Reducir*) respectivamente.

Al botón virtual *Rotar* se le asignó la función *VirtualButtonBehaviour>Dynamic bool>enabled* (ver figura 76). Al activar y desactivar el *Toggle* su estado cambia de verdadero a falso y viceversa.

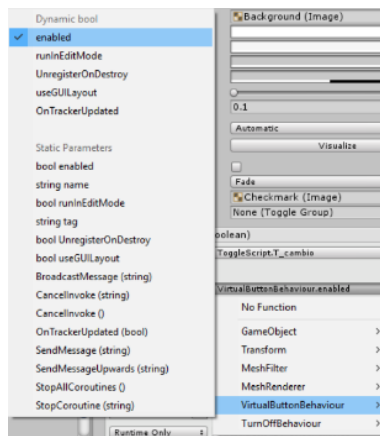


Figura 76 Asignación de método

Al botón virtual *RotarV* se le asignó la función *VirtualButtonBehaviour>Static bool> bool enabled* (ver figura 77).

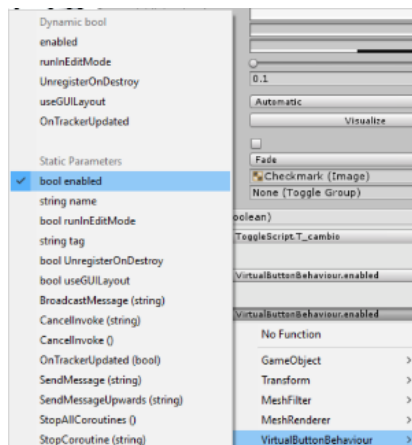


Figura 77 Asignación de método

Se dejó desactivada la casilla de verificación en el campo por lo que el estado del botón es falso logrando que el botón sea completamente inhabilitado. Se realizó el mismo procedimiento del botón *RotarV* para los botones *Aumentar* y *Reducir* (ver figura 78).

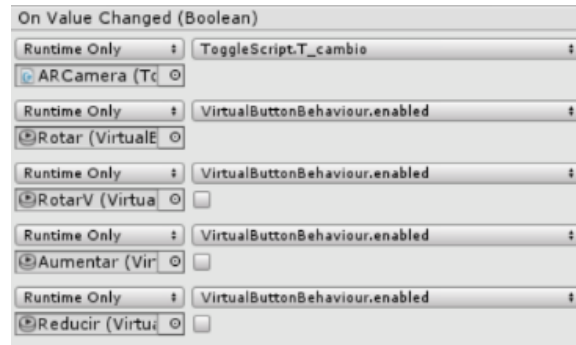


Figura 78 Vista de métodos asignados

El procedimiento descrito se repitió para los botones virtuales *Toggle_rv*, *Toggle_a*, *Toggle_r*, siendo la única diferencia el seleccionar el método asignado a cada botón. Así mismo el procedimiento se aplicó para el resto de las figuras geométricas.

3.10 Nombre de la figura geométrica aumentado

Se agregó el nombre de la figura geométrica al marcador de realidad aumentada, por lo que cuando el marcador es reconocido por la aplicación se aumenta de igual manera que la figura geométrica. A continuación, se detalla el procedimiento seguido para la creación del nombre en 3D del cubo e integrarlo al marcador de realidad aumentada.

Se agregó a la escena del proyecto un objeto 3D *Quad* por medio de *GameObject>3D Object> Quad* (ver figura 79) como hijo de *ImageTargetCubo*.

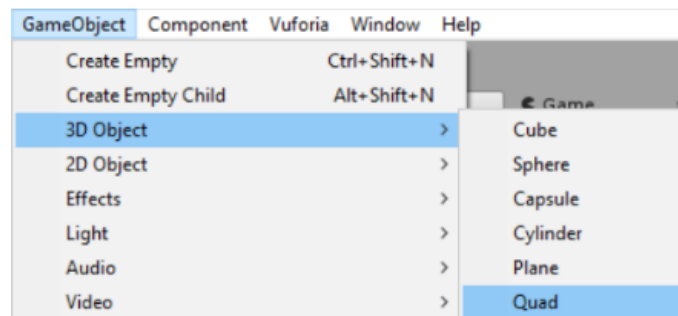


Figura 79 Agregar objeto *Quad*

Se modificaron los valores de posición, rotación y escala como se muestra en la figura 80.

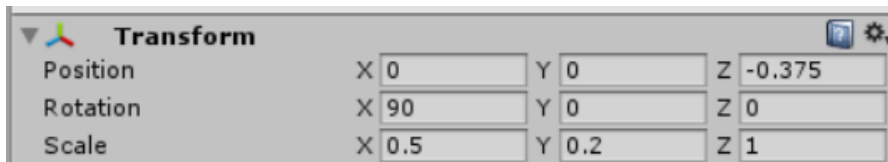


Figura 80 Valores del objeto *Quad*

Se cambió el material por defecto y se seleccionó el material utilizado en el cubo 3D (ver figura 81), por lo que el color es el mismo al de la figura geométrica.

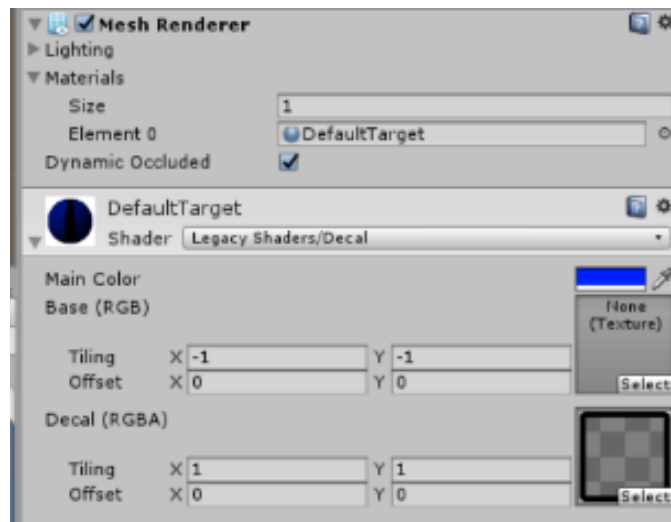


Figura 81 Material del objeto *Quad*

Se agregó un nuevo objeto de texto 3D por medio de *GameObject>3D Object> 3D Text* y se colocó como hijo del objeto *Quad* agregado anteriormente. Se modificaron los valores de posición, rotación y escala como se muestra en la figura 82.

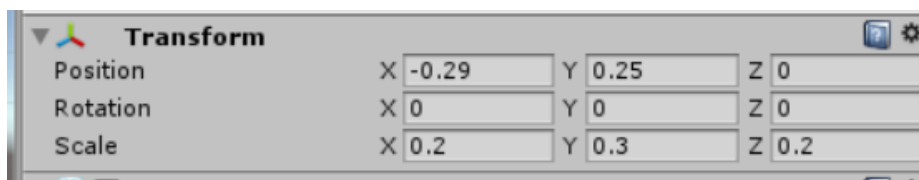


Figura 82 Valores del objeto *3D Text*

En las propiedades *Text Mesh* se establecieron los valores como se muestran en la figura 83.

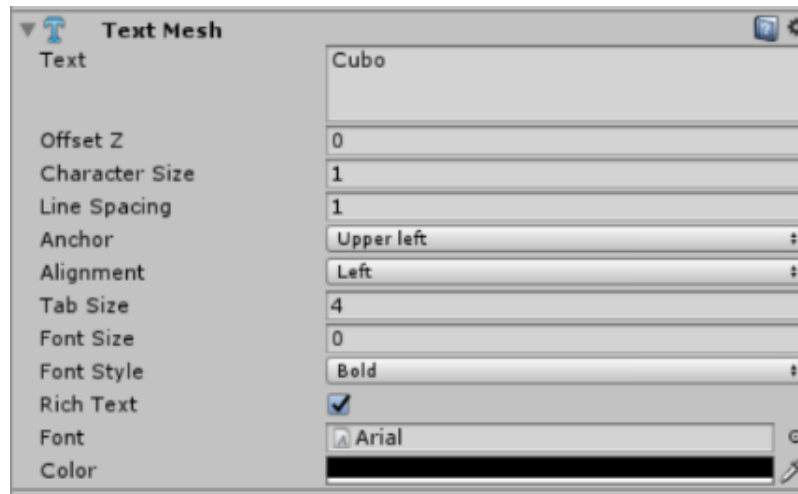


Figura 83 Propiedades *Text Mesh*

El procedimiento para crear e integrar el nombre 3D al marcador de realidad aumentada del cubo es el mismo para las figuras geométricas restantes, sin embargo, los valores de posición, escala, rotación y material son diferentes. En la figura 84 se muestra un diagrama con la jerarquía utilizada y los valores de las propiedades utilizados en cada nombre 3D creado e integrado a los marcadores de realidad aumentada.

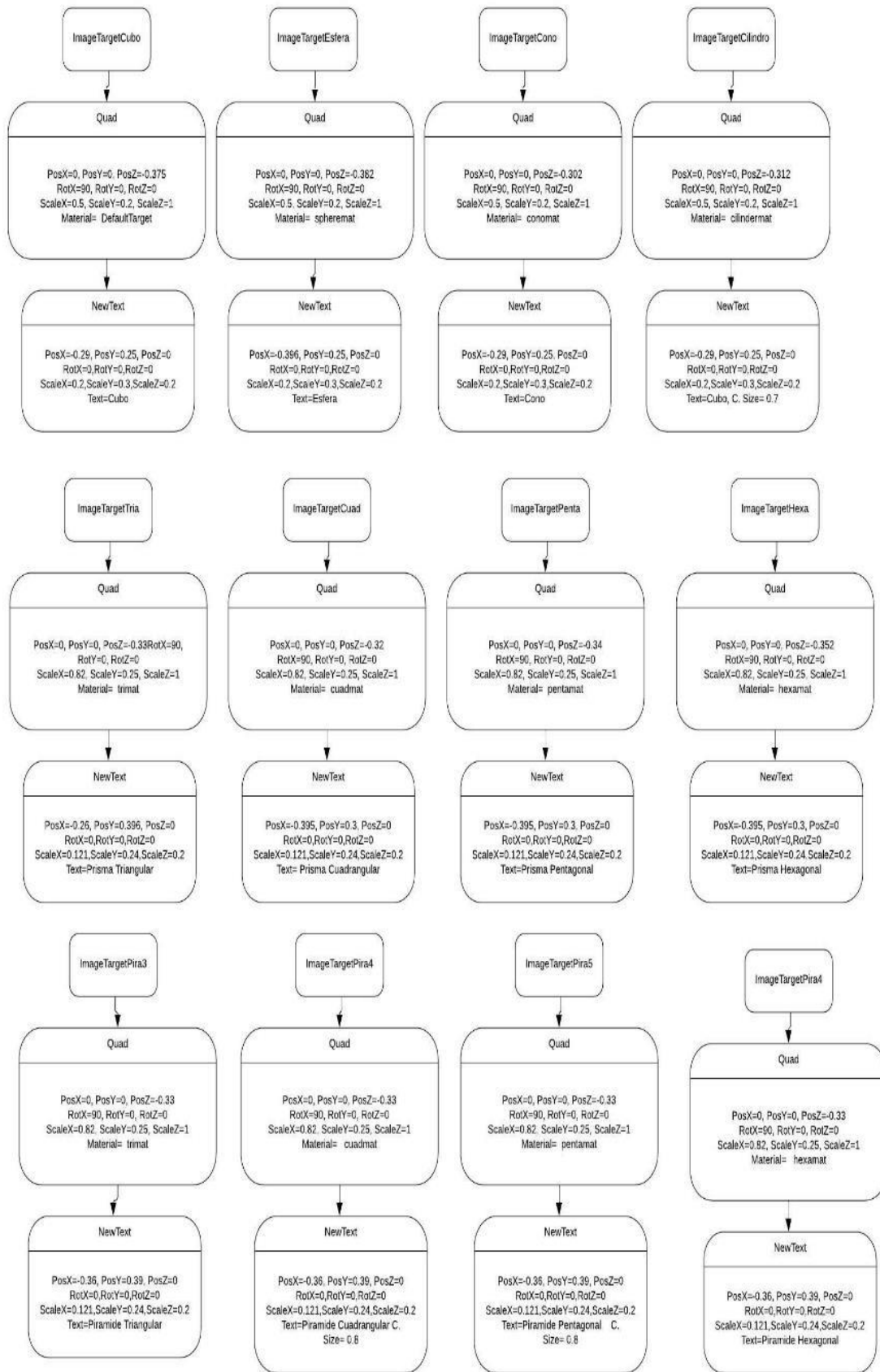


Figura 84 Propiedades de los nombres 3D

3.11 Fórmulas de área y volumen aumentados

Se integraron las fórmulas para calcular el área y volumen de las figuras geométricas dentro del marcador de realidad aumentada.

Se realizó el siguiente procedimiento para la creación del cuadro de texto 3D el cual contiene la fórmula para el cálculo de área y volumen del cubo.

Se agregó a la escena del proyecto un objeto *3D Plane* por medio de *GameObject>3D Object> Plane* (ver figura 85) como hijo de *ImageTargetCubo*. Se renombró como *FormCubo*.

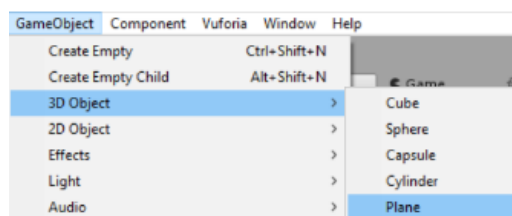


Figura 85 Agregar objeto *Plane*

Se modificaron los valores de posición, rotación y escala como se muestra en la figura 86.

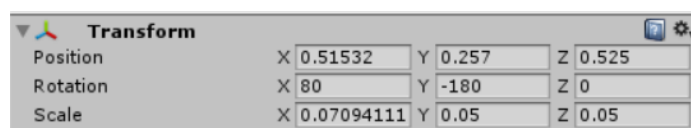


Figura 86 Valores de objeto *Plane*

Se creó una carpeta con el nombre fórmulas en la carpeta *Assets* del proyecto, en la cual se almacenaron las imágenes con las fórmulas de áreas y volúmenes de las figuras geométricas.

Se seleccionó la imagen que contiene las fórmulas del cubo, se arrastró y soltó al objeto *3D Plane* creado anteriormente.

Para habilitar y deshabilitar la visualización de la fórmula en el marcador, se utilizó el método *OnValueChanged(Boolean)* de *Toggle_Formulas* en la interfaz de usuario. Se

agregó un campo a la función y se seleccionó el objeto *FormCubo* se procedió a seleccionar la acción *GameObject>Dynamic bool>SetActive* (ver figura 87).

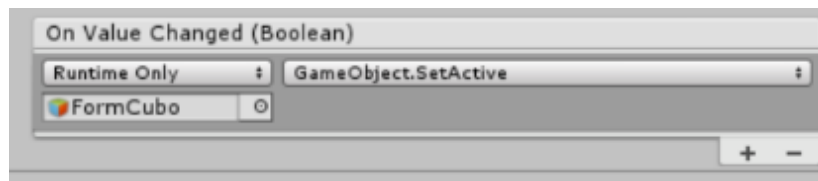


Figura 87 Asignación de método

El procedimiento anterior se realizó para el resto de los marcadores de las figuras geométricas. En el Anexo 5 se encuentran las imágenes con las fórmulas de las figuras.

3.12 Escena de ayuda

Para finalizar la aplicación, se agregó una nueva escena al proyecto por medio de *File>New Scene*, la cual se utilizó para mostrar los créditos y un video tutorial de uso de la aplicación.

Se agregó un nuevo botón a la interfaz principal de la aplicación para poder acceder a la escena recién creada. La figura 88 se muestra el código del método utilizado por el botón de ayuda.

```
public void Loadscenehelp()  
{  
    SceneManager.LoadScene(1);  
}
```

Figura 88 Código para cargar escena

Se agregó un botón a la escena de ayuda el cual al accionarse muestra un video tutorial de uso de la aplicación por medio metodo *PlayTutorial()*. Se agregó un botón a la escena de ayuda el cual al accionarse se regresa a la interfaz principal de la aplicación por medio del método *ReturnApp()*, que carga la escena principal del proyecto. La figura 89 muestra los códigos de los métodos mencionados.

```

public void PlayTutorial()
{
    Handheld.PlayFullScreenMovie("video.mp4", Color.black, FullScreenMovieControlMode.CancelOnInput);
}

public void ReturnApp()
{
    SceneManager.LoadScene(0);
}

```

Figura 89 Código de iniciar video tutorial y regresar a la pantalla principal

Se agregó un objeto *Panel* y un objeto *Text* el cual muestra los nombres de los alumnos que desarrollaron la aplicación. La figura 90 muestra la escena de ayuda.



Figura 90 Escena de ayuda

3.13 Validación

La aplicación se desarrolló con base a las figuras geométricas que comúnmente se ven en los salones de clase de tercer a sexto grado del nivel escolar, las cuales son: cubo, esfera, cono, cilindro, prismas y pirámides. A continuación, se presentan los resultados finales de la aplicación desarrollada.

3.13.1 Pantalla principal

En la figura 91, se muestra la pantalla principal de la aplicación funcionando en un dispositivo móvil con sistema operativo Android, la cual a su vez es la interfaz principal con la cual interactúa el usuario final.

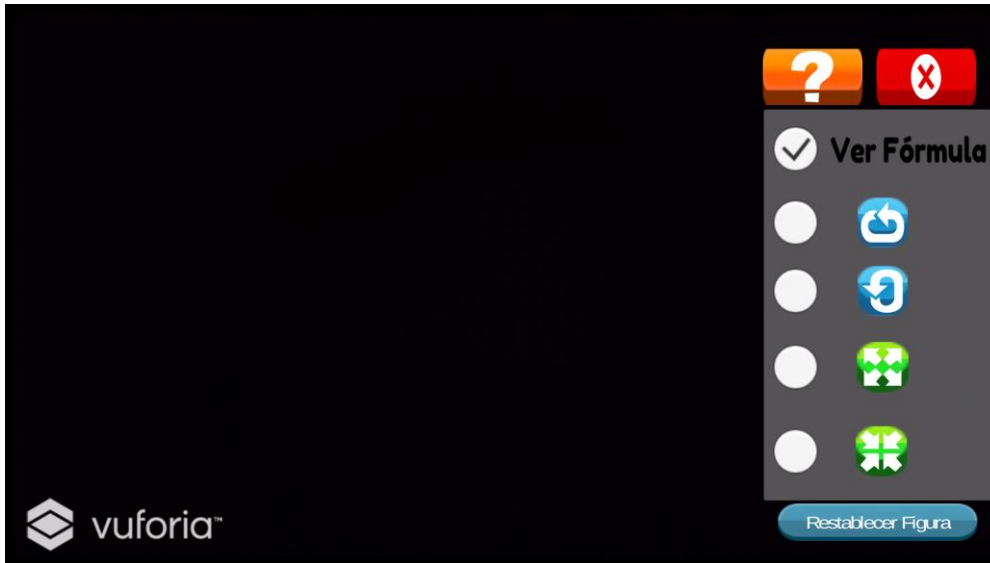


Figura 91 Pantalla principal de la aplicación

3.13.2 Visualización de figuras geométricas en la aplicación

A continuación, se muestran las figuras geométricas aumentadas por la aplicación.

La figura 92 muestra el cubo, cono, cilindro y la esfera aumentados.



Figura 92 Cubo, cono, cilindro y esfera aumentados.

La figura 93, muestra los prismas: triangular, cuadrangular, pentagonal y hexagonal.



Figura 93 Prismas: triangular, cuadrangular, pentagonal y hexagonal aumentados.

La figura 94, muestra las pirámides: triangular, cuadrangular, pentagonal y hexagonal.



Figura 94 Pirámides: triangular, cuadrangular, pentagonal y hexagonal aumentados

3.13.4 Pruebas realizadas

Se realizaron pruebas a una muestra de 44 estudiantes para evaluar la funcionalidad y uso de la aplicación en la escuela primaria “Colegio Hispanoamericano” de los grupos de tercer, cuarto, quinto y sexto grado. Para la realización de las pruebas se contó con el consentimiento de la directora del colegio, a la cual se le realizó una breve demostración de la aplicación y se le mencionó el objetivo de la misma. Se pidió permiso para tomar fotografías para presentarlas como evidencia de la realización de las pruebas, se obtuvo la autorización con la condición que se ocultaran los rostros de los estudiantes.

Las pruebas consistieron en darle una breve explicación del uso de la aplicación, así como, proporcionarles a los estudiantes un dispositivo móvil con la aplicación instalada con la cual visualizaron las figuras geométricas aumentadas. .

Interactuaron con las figuras de manera que las rotaron, aumentaron y redujeron de tamaño utilizando las manos para tocar la figura geométrica que visualizaban en pantalla al activar los botones en pantalla asignados para cada función.

Para finalizar las pruebas, se aplicó una encuesta a los estudiantes que utilizaron la aplicación, dicha encuesta se enfoca en evaluar el funcionamiento y uso de la aplicación, así como la aceptación por parte de los estudiantes.

3.13.5 Encuesta

La siguiente encuesta se aplicó a los estudiantes que utilizaron la aplicación debido a que son el usuario final a la cual se dirige (ver tabla 2).

Pregunta	Escala				
	Totalmente de acuerdo	De acuerdo	Ni de acuerdo, ni en desacuerdo	En desacuerdo	Totalmente en desacuerdo
1.- En general, la aplicación es sencilla de utilizar.					
2.- El color de las figuras geométricas es llamativo.					

3.- Las figuras geométricas se aprecian de manera clara.					
4.- Es fácil saber qué acción realizan los botones en la pantalla.					
5.- Fue agradable la experiencia al interactuar con la figura geométrica.					
6.- Me gustaría utilizar la aplicación para ver las figuras geométricas 3D en clase.					
7.- Considero que la aplicación es un apoyo en el tema de geometría.					
8.- La interacción con la figura geométrica fue sencilla.					
9.- La información mostrada del modelo fue importante.					
10.- Es una aplicación que utilizaría en casa.					
11.- ¿Qué le agregarías a la aplicación?					

Tabla 2 Encuesta

IV. Resultados y Discusiones

En el presente capítulo, se describen los resultados obtenidos de las pruebas realizadas con la aplicación móvil “*GeometriaAR*”, nombre que se le dio a la aplicación debido a su enfoque en la geometría y el uso de la tecnología de realidad aumentada. Con la aplicación, se pretende apoyar la didáctica de la geometría en el nivel básico escolar motivando al estudiante por medio de la visualización e interacción con los cuerpos geométricos 3D.

Debido a que las respuestas especifican un nivel de acuerdo y desacuerdo, se utilizó la “Escala de Likert” para medir el grado de acuerdo y desacuerdo de los usuarios finales. A continuación, se presentan los resultados obtenidos de las encuestas.

4.1 Resultados

La pregunta número 1, recolecta información de la facilidad de uso de la aplicación, 23 estudiantes estuvieron totalmente de acuerdo, 13 estuvieron de acuerdo y 1 en total desacuerdo con el hecho de que la aplicación es fácil de utilizar. En la figura 95, se muestran los resultados obtenidos separados por grupo.

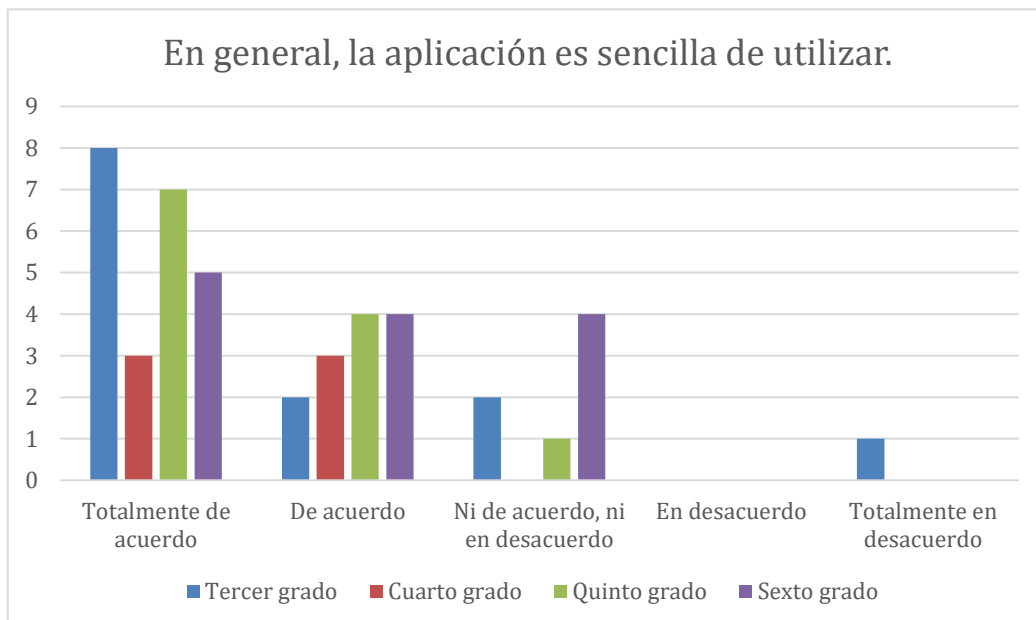


Figura 95 Resultados de la pregunta 1.

En la pregunta número 2, se le preguntó al usuario si el color de los colores aplicados a las figuras geométricas le llama la atención, 40 estuvieron totalmente de acuerdo, 1 de acuerdo, 2 ni de acuerdo ni en desacuerdo y ninguno en total desacuerdo con el hecho de que el color de las figuras geométricas es llamativo. La figura 96, muestra los resultados obtenidos.

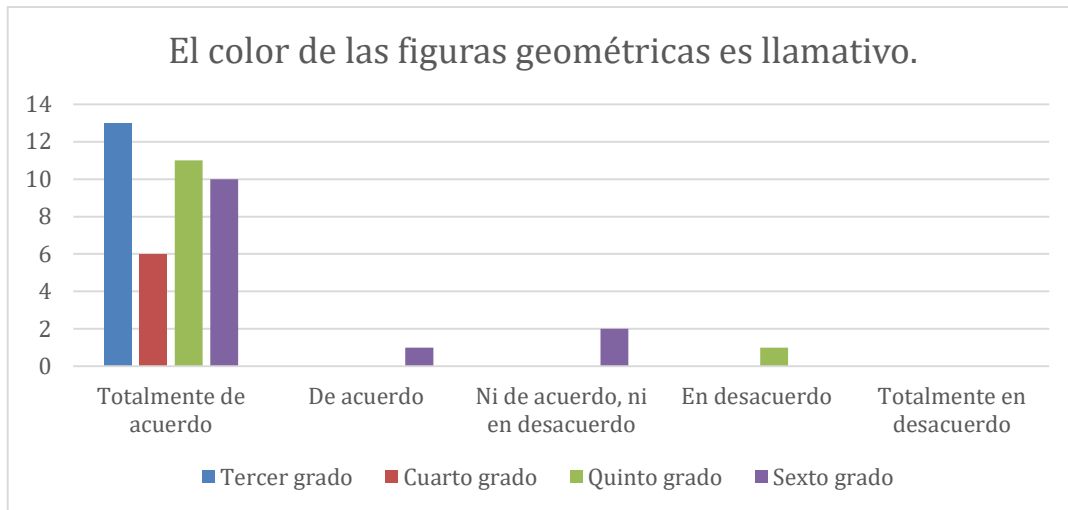


Figura 96 Resultados de la pregunta 2.

La pregunta número 3, se le preguntó al usuario si la apreciación de las figuras geométricas fue clara, 29 estuvieron totalmente de acuerdo, 12 de acuerdo y 3 ni de acuerdo ni en desacuerdo con el hecho de que las figuras geométricas se aprecian de manera clara. La figura 97, muestra los resultados obtenidos.

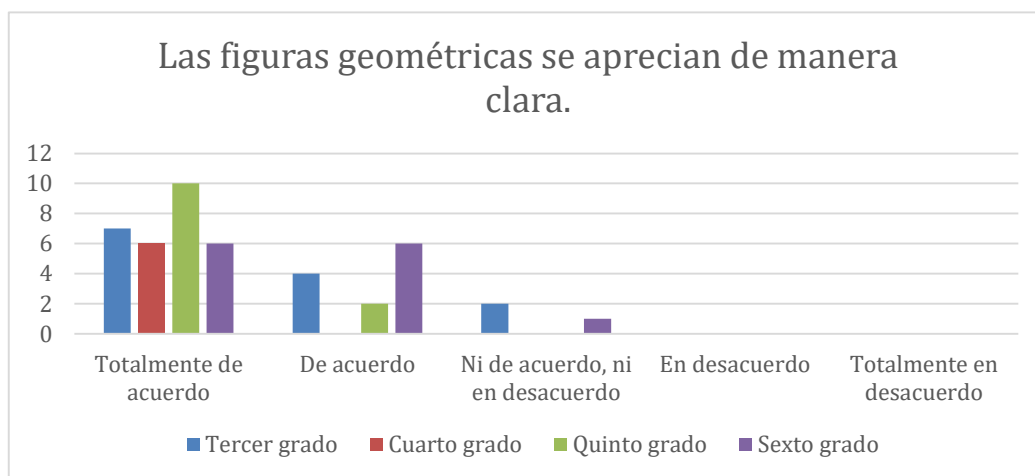


Figura 97 Resultados obtenidos de la pregunta 3.

La pregunta número 4, recolecta información sobre la facilidad de deducir por el usuario que acción realizan los botones en pantalla, 18 estuvieron totalmente de acuerdo, 16 de acuerdo, 6 ni de acuerdo ni en desacuerdo, 3 en desacuerdo y 1 totalmente en desacuerdo con la facilidad de saber que acción realizan los botones en pantalla. Los resultados se muestran en la figura 98.

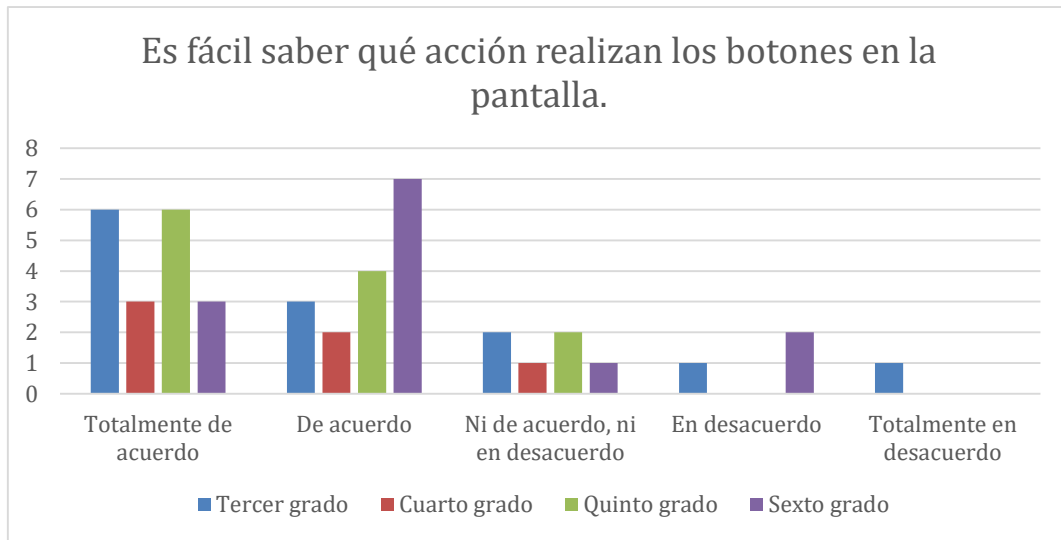


Figura 98 Resultados de la pregunta 4

La pregunta número 5, se le preguntó al usuario si la experiencia al interactuar con la figura geométrica fue agradable, 38 estuvieron totalmente de acuerdo, 5 de acuerdo y 1 ni de acuerdo ni en desacuerdo, con la hecho de que la experiencia al interactuar con la figura geométrica fue agradable. La figura 99 muestra los resultados.

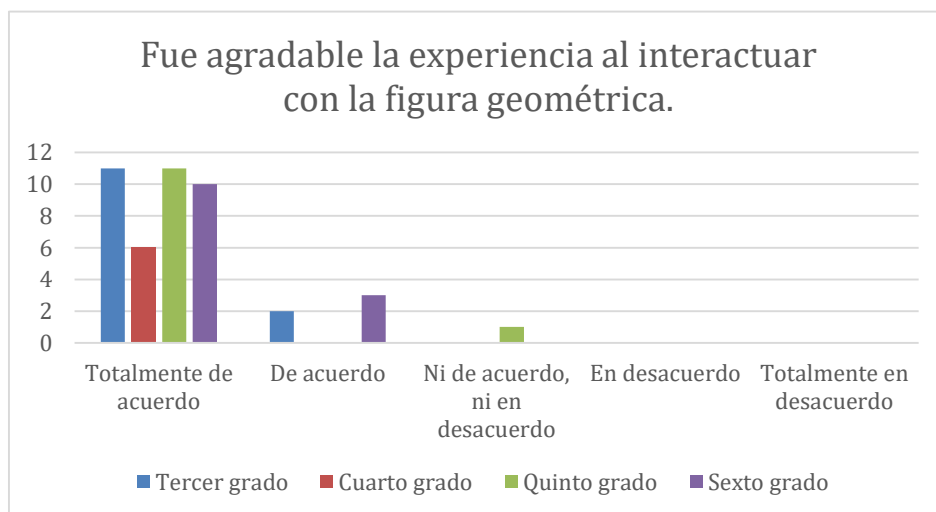


Figura 99 Resultados de la pregunta 5

La pregunta número 6, le preguntó al usuario si le agradaría que la aplicación fuera utilizada en el salón de clase para la visualización de las figuras geométricas 3D, 37 estuvieron totalmente de acuerdo y 7 de acuerdo, no se obtuvieron resultados negativos con el hecho de utilizar la aplicación en clase. La figura 100 muestra los resultados.

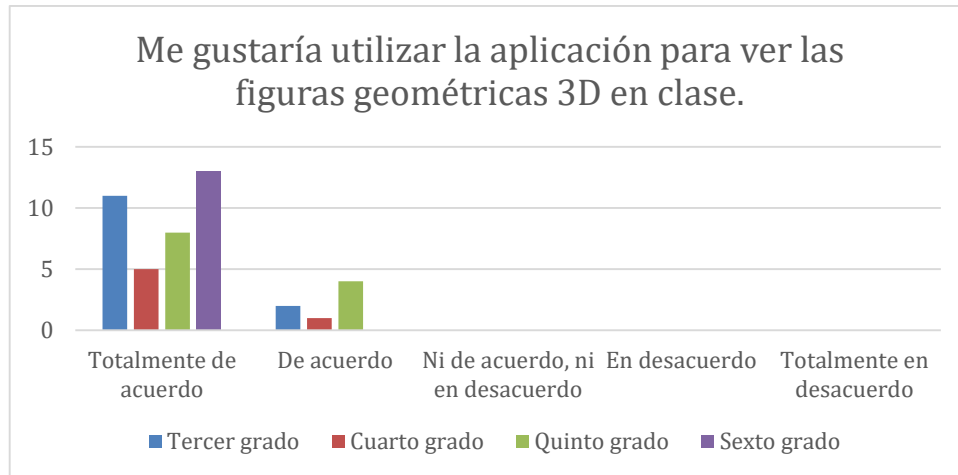


Figura 100 Resultados pregunta 6

La pregunta número 7, le preguntó al usuario si considera que al utilizar la aplicación se apoya el tema de la geometría, 31 estuvieron totalmente de acuerdo, 10 de acuerdo, 2 ni de acuerdo ni es desacuerdo y 1 en desacuerdo al considerar que la aplicación es un apoyo al tema de geometría. La figura 101 muestra los resultados obtenidos.

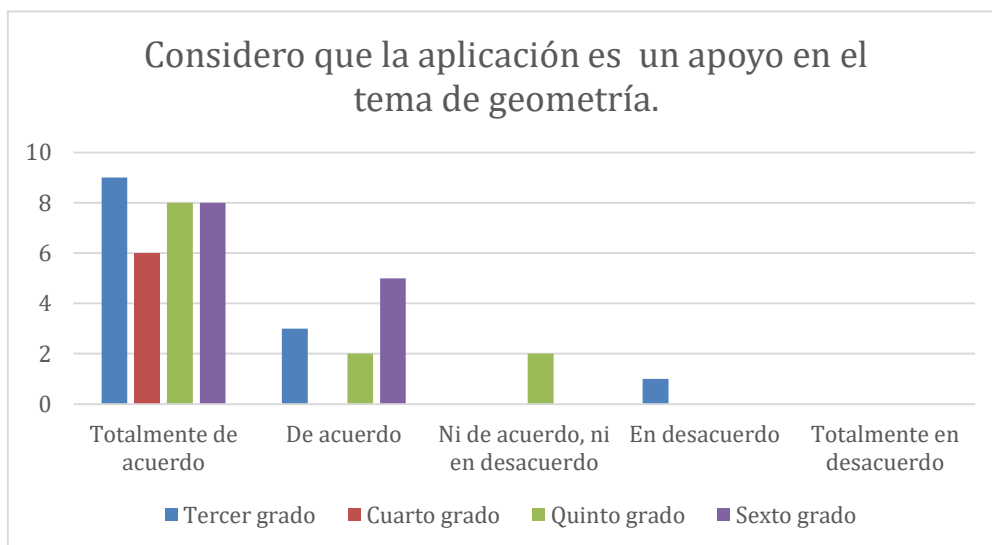


Figura 101 Resultados de la pregunta 7

En la pregunta número 8, se le preguntó al usuario si fue fácil interactuar con la figura geométrica aumentada, 23 estuvieron totalmente de acuerdo, 14 de acuerdo, 6 ni de acuerdo ni en desacuerdo y 1 totalmente en desacuerdo con el hecho la fácil interacción con la figura. Los resultados se muestran en la figura 102.

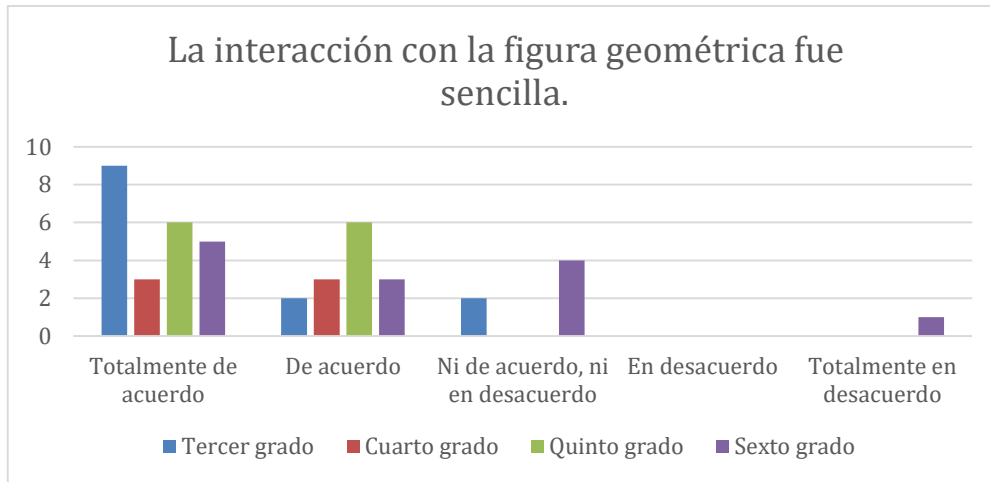


Figura 102 Resultados de la pregunta 8

La pregunta número 9, le preguntó al usuario sobre la importancia de la información mostrada, 29 estuvieron totalmente de acuerdo, 14 de acuerdo y 1 ni de acuerdo ni en desacuerdo con la importancia de la información mostrada. La figura 103 muestra los resultados.

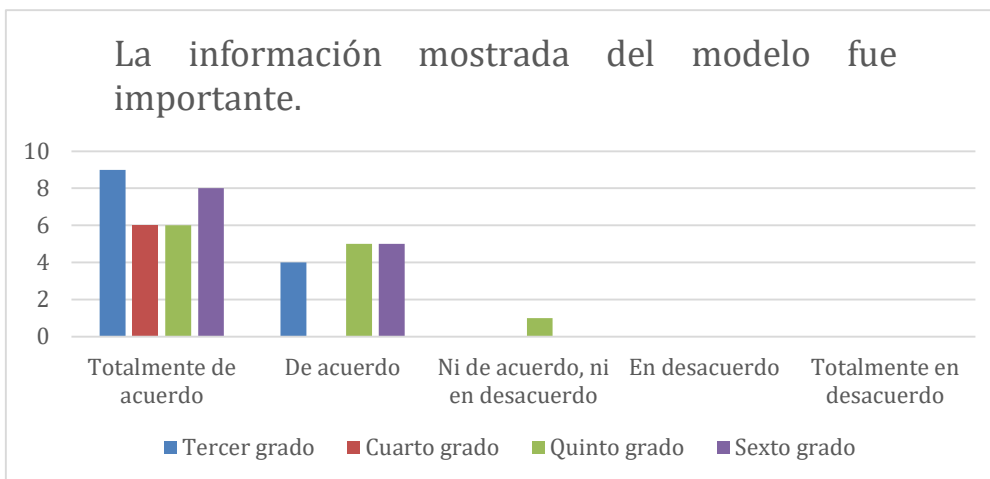


Figura 103 Resultados de la pregunta 9

En la pregunta número 10, se le preguntó al usuario si utilizaría la aplicación en su hogar, 33 estuvieron totalmente de acuerdo, 7 de acuerdo y 4 ni de acuerdo ni en desacuerdo con el hecho de utilizar la aplicación en casa. La figura 104 muestra los resultados obtenidos.

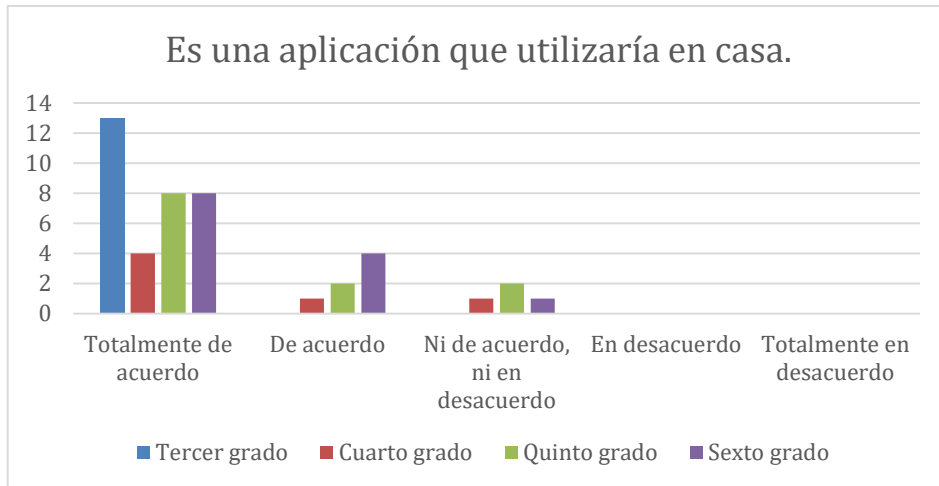


Figura 104 Resultados de la pregunta 10

La pregunta número 11 es de carácter abierto, por lo que se invita al usuario a comentar que le gustaría que se le agregara a la aplicación, hay que considerar que los usuarios finales son niños de entre 7 y 12 de edad por lo que algunas de las respuestas pueden no relacionarse con el tema de la geometría y se un tanto elocuentes. En la tabla 3 se muestran algunos de los comentarios obtenidos por la pregunta 11 clasificados por grado de primaria.

Tercer grado	Cuarto grado	Quinto grado	Sexto grado
Que las cosas las puedas agarrar y que salgan del teléfono	Figura geométrica, mueve figura geométrica	Que se puedan capturar y <i>skins</i>	Más figuras geométricas y que se pueda usar en la escuela para la clase de matemáticas
Agregarle para que tu mano pueda mover la figura	Hacer figuras inventadas	Más matemáticas	Que salgan dinosaurios y que sea más sencillo la interacción con la figura geométrica
Lentes de realidad virtual	Un botón que lo cambie de color	Muchas más figuras	Poner más información de la figura
Videojuegos	Mas colores	Un poco de información de figuras que hallas detectado	Una fórmula más clara

Tabla 3 Comentarios obtenidos de la pregunta 11

4.2 Análisis de resultados

Los resultados obtenidos por las encuestas aplicadas a los 44 estudiantes del nivel básico escolar, indican que la aplicación móvil “*GeometriaAR*” es fácil de utilizar y los colores aplicados a las figuras geométricas son llamativos, además les agradó la interacción con la figura geométrica y consideran que sería un buen apoyo al tema de la geometría, así como que fuera utilizada en el salón de clase para la visualización de la figuras geométricas y también harían uso de dicha aplicación en la casa.

La interacción con la figura geométrica se considera sencilla y fácil de utilizar, sin embargo, con base a los comentarios y los resultados medidos en la pregunta referente a los botones en pantalla, indican que se tiene que trabajar más en la manera que se

presentan los botones. También se considera el hecho de que se agregue más información de la figura geométrica al aumentarse.

Las encuestas fueron aplicadas a usuarios finales de la aplicación, por lo que con base a los resultados, la información recopilada es en su mayoría positiva y demuestra que la aplicación de la tecnología de realidad aumentada al tema de la geometría tiene un impacto positivo en los estudiantes.

V. Conclusiones

En el presente capítulo se presentan las conclusiones que se obtuvieron durante el desarrollo del proyecto. La aplicación “*GeometriaAR*”, es una aplicación móvil de realidad aumentada para apoyar la didáctica de la geometría en el nivel básico escolar por medio de la presentación de las figuras geométricas en 3D y la interacción con las mismas.

Para el desarrollo de la aplicación se utilizaron diversas herramientas, tales como lenguajes de programación, software de modelado 3D, aplicaciones web y motores de juego entre otros.

5.1 Objetivos del proyecto

Respecto a los objetivos del proyecto, se completó el desarrollo de una aplicación móvil de realidad aumentada para el sistema operativo Android, la cual puede utilizarse como herramienta para apoyar la impartición del tema de geometría en el nivel básico escolar.

Se logró la creación de los modelos 3D de las figuras geométricas que el usuario puede visualizar e interactuar con los mismos.

Se implementó una interfaz simple y de acceso directo para facilitar la interpretación del usuario final.

5.2 Recomendaciones

Para la futura implementación de una aplicación móvil de realidad aumentada para apoyar el tema de geometría en el nivel básico escolar, se recomienda el incluir una mayor variedad de figuras geométricas de manera que el catálogo de las mismas sea más extenso, así como, mostrar más información del cuerpo geométrico y no solo las fórmulas para calcular el área y volumen.

Otra recomendación sería el incluir más formas de interacción entre el usuario y la figura geométrica para lograr una mejor experiencia inmersiva. También se recomienda el incluir algunos videojuegos educativos que de igual manera utilicen la tecnología de realidad aumentada y sean enfocados en la geometría de tal manera que la aplicación

sea más completa y tenga un mayor impacto en aumentar el interés de los usuarios finales.

Por último, se recomienda que la aplicación sea implementada en diferentes plataformas móviles de manera que sea mayor el número de usuarios que tengan acceso a la misma.

Las recomendaciones se basan en los comentarios obtenidos en las encuestas que contestaron los usuarios finales al utilizar la aplicación.

5.3 Aportaciones

La aportación al apoyo de la impartición del tema de la geometría en el nivel básico escolar fue bastante interesante, dado que al poder realizar pruebas de uso y funcionalidad con los usuarios finales que en este caso son los estudiantes de primaria, se vio que el uso de la tecnología de la mano de la realidad aumentada tiene un impacto positivo en el interés del estudiante por el tema.

La experiencia vivida durante la realización de las pruebas fue bastante gratificante, el ver que los estudiantes se muestran entusiasmados al utilizar la aplicación, así como las preguntas que realizaban al utilizarla denotaban interés en saber más de la misma. A la experiencia obtenida durante el desarrollo del proyecto, se le suman los conocimientos adquiridos en relación a la tecnología y el desarrollo de software.

Referencias

- [1] Hong-Quan Le and Jee-In Kim, "An augmented reality application with hand gestures for learning 3D geometry," 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, 2017, pp. 34-41.
doi: 10.1109/BIGCOMP.2017.7881712
- [2] J. Purnama, D. Andrew and M. Galinium, "Geometry learning tool for elementary school using augmented reality," 2014 International Conference on Industrial Automation, Information and Communications Technology, Bali, 2014, pp. 145-148.
doi: 10.1109/IAICT.2014.6922112
- [3] B. Kye, & Y. Kim, "Investigation of the relationships between media characteristics, presence, flow, and learning effects in augmented reality based learning," International Journal for Education Media and Technology, vol.2, no.1, pp.4–14, 2008.
- [4] R. C. Chang and Z. S. Yu, "Application of Augmented Reality technology to promote interactive learning," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, 2017, pp. 1673-1674.
doi: 10.1109/ICASI.2017.7988257
- [5] S. Patil, C. Prabhu, O. Neogi, A. R. Joshi and N. Katre, "E-learning system using Augmented Reality," 2016 International Conference on Computing Communication Control and automation (ICCUBEA), Pune, 2016, pp. 1-5.
doi: 10.1109/ICCUBEA.2016.7860038
- [6] S. M. Banu, "Augmented Reality system based on sketches for geometry education," 2012 International Conference on E-learning and E-Technologies in Education (ICEEE), Lodz, 2012, pp. 166-170.
doi: 10.1109/ICeLeTE.2012.6333384
- [7] B. Kraut and J. Jeknić, "Improving education experience with augmented reality (AR)," 2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), Opatija, 2015, pp. 755-760.
doi: 10.1109/MIPRO.2015.7160372
- [8] C. Vanegas, "Desarrollo de aplicaciones sobre Android", Revistas.udistrital.edu.co, 2012. [Online]. Disponible en: <http://revistas.udistrital.edu.co/ojs/index.php/vinculos/article/view/4275/5967>.

- [9] M. Báez, Á. Borrego, L. Cruz, M. González, F. Hernández, D. Palomero, J. Rodríguez de Llera, D. Sanz, M. Saucedo, P. Torralbo and Á. Zapata, *Introducción a Android*. Victoria López y Grupo Tecnología UCM, pp. 15-26.
- [10] P. Delía, "Un análisis experimental de tipo de aplicaciones para dispositivos móviles", Sedici.unlp.edu.ar, 2013. [Online]. Disponible en: <http://sedici.unlp.edu.ar/handle/10915/32397>.
- [11] J. Morillo Pozo, *Introducción a los dispositivos móviles*. pp. 7-25.
- [12] R. Ganiyu, O. Okediran, O. Arulogun and C. Oyeleye, *Mobile Operating Systems and Application Development Platforms: A Survey*. 2014, pp. 2195-2201.
- [13] J. Carmigniani, B. Furht, M. Anisetti, P. Ceravolo, E. Damiani and M. Ivkovic, "Augmented reality technologies, systems and applications", *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 341-377, 2010.
- [14] C. Prendes Espinosa, "realidad aumentada y educación: análisis de experiencias prácticas". *Pixel-Bit. Revista de Medios y Educación*, 2015, pp. 187-203.
- [15] C. Ortiz Rangel, "realidad aumentada en medicina", *Revista Colombiana de Cardiología*, vol. 18, no. 1, pp. 4-7, 2011.
- [16] S. García Peña and O. López Escudero, "La enseñanza de la geometría", 1st ed. Instituto Nacional para la evaluación de la Educación, 2008, pp. 25-46.
- [17] E. Filloy Yagüe, "Didáctica e Historia de la Geometría Euclidiana", 3rd ed. Grupo Editorial Iberoamérica, 2001, pp. 129-132.
- [18] J. Montesinos Amilibia, "La Geometría no Euclidiana", 1st ed. *Rev.R.Acad.Cienc.Exact.Fís.Nat*, 2014, pp. 1-8.
- [19] A. Ramírez Galarza, "Geometría analítica". México: UNAM, Facultad de Ciencias, Coordinación de Servicios Editoriales, 2004, pp. 1-24.
- [20] A. Viña Escalar, "Geometría diferencial". [Oviedo]: Universidad de Oviedo, Servicio de Publicaciones, 1999, pp. 1-22.
- [21] Google, *Android Stack* 2017. Disponible en: https://developer.android.com/guide/platform/images/android-stack_2x.png
- [22] Vrealities, *z800ar3*. 2013. Disponible en: <https://www.vrealities.com/wp-content/uploads/2013/03/z800ar3.jpg>
- [23] Google, *Google Tango AR*. 2016. Disponible en: <https://techcrunch.com/2016/11/01/google-finally-launches-tango/>
- [24] 2012. Disponible en: <https://cchsu73.files.wordpress.com/2012/06/p1150637.jpg>

- [25] Qr vs código de barras. Disponible en: http://www.solutekcolombia.com/images/qr_vs_codigo_de_barras.gif
- [26] 2012. Marcadores ediamsistemas. Disponible en: <http://socialancer.com/wp-content/uploads/2012/08/Marcadores-ediamsistemas.jpg>
- [27] 2015. Disponible en: <https://www.yeeply.com/blog/wp-content/uploads/2015/02/realidad-aumentada-1024x576.jpg>
- [28] Microsoft Hololens. Disponible en: <https://compass-ssl.surface.com/assets/8d/0e/8d0e5ebf-828a-4119-8cf6-483c17cdb131.jpg?n=Education-Edited.jpg>
- [29] 2014. Disponible en: <https://compass-ssl.surface.com/assets/8d/0e/8d0e5ebf-828a-4119-8cf6-483c17cdb131.jpg?n=Education-Edited.jpg>
- [30] Disponible en: <https://i.ytimg.com/vi/vDNzTasuYEW/maxresdefault.jpg>
- [31] Disponible en: <https://media.licdn.com/mpr/mpr/AAEAAQAAAAAAAAAuWAAAAJGRIYmMyOTUwLTIxMDYtNGY0Mi1hOWYyLWRjMDI3MTVjZmE0MQ.jpg>
- [32] Disponible en: <https://i.ytimg.com/vi/F7mcDyBXJOO/maxresdefault.jpg>
- [33] Disponible en: <https://i.ytimg.com/vi/Zn6ZGeDo3ec/maxresdefault.jpg>
- [34] Martin Sauter, "Evolution of Mobile Devices and Operating Systems," in *3G, 4G and Beyond: Bringing Networks, Devices and the Web Together*, 1, Wiley Telecom, 2013, pp.384-
doi: 10.1002/9781118394540.ch5

Anexo 1 Instalación y configuración de software

Blender

El software Blender para modelado 3D es de licencia gratuita y fue descargado desde el enlace <https://www.blender.org/download/>.

El proceso de instalación fue el siguiente:

1. Inicio del asistente instalador.
2. Aceptar los acuerdos de uso de licencia.
3. Elegir la ruta de instalación.
4. Una vez seleccionada la ruta, el asistente comenzara la instalación.

Unity

Unity se descargó desde el siguiente enlace <https://store.Unity.com/es/download?ref=personal> del cual se obtiene una licencia gratuita, sin embargo hay que aceptar condiciones de uso dado que la licencia se ofrece para estudiantes, principiantes y aficionados (ver figura 105).



Figura 105 Condiciones de uso Unity

Proceso de instalación:

1. Se inició el asistente de descarga e instalación de Unity.
2. Se aceptó el acuerdo de licencia de uso.
3. Se seleccionaron los componentes a descargar para posteriormente instalar.
4. Se eligió la ruta de descarga de los componentes y la ruta de instalación de Unity.

Durante la instalación de Unity, el asistente de instalación ofreció la opción de descarga e instalación de Microsoft Visual Studio 2017 para ser utilizado como el editor de scripts externo.

Una vez terminada la instalación, se procedió a la configuración de Unity para desarrollar aplicaciones móviles para sistema operativo Android. Primero se descargó el SDK de Android y el JDK de java los cuales fueron descargados desde los siguientes enlaces <https://developer.android.com/studio/index.html>, <http://www.oracle.com/technetwork/java/javase/downloads/index.html>. Para facilitar el proceso de instalación del SDK, se descargó Android Studio el cual incluye el Android SDK. Una vez instalados Android Studio y descargado el JDK de java, en el editor de Unity se seleccionó la opción *Edit>Preferences*. En la ventana de preferencias, se seleccionó la opción *Browse* en la opción de SDK y se buscó la ruta en la cual se almaceno el Android SDK. El mismo procedimiento fue realizado para la opción JDK (ver figura 106).

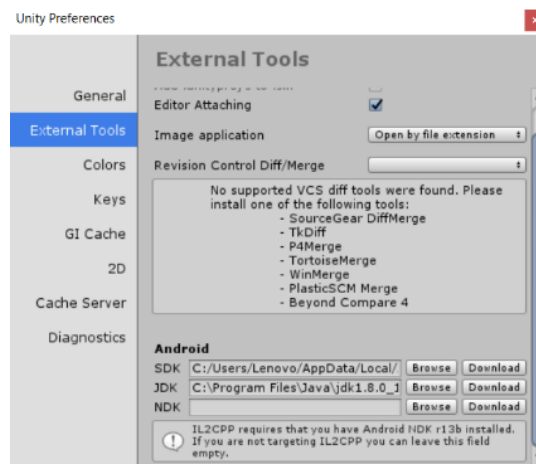


Figura 106 Rutas SDK y JDK en Unity

Los procedimientos de configuración del SDK y JDK son necesarios para poder compilar y crear la aplicación móvil desde el editor de Unity.

Vuforia

Vuforia no requiere de un proceso de instalación, ya que es un SDK (*Software Development Kit*) sin embargo, se requiere seguir un proceso de configuración antes de poder empezar a utilizar sus funciones.

El SDK de Vuforia se obtuvo desde el siguiente enlace <https://developer.vuforia.com/downloads/sdk>, para el proyecto desarrollado en este documento se descargó la extensión de Unity. Al intentar descargar la extensión, el sitio web requiere de una cuenta de desarrollador la cual puede ser creada desde el sitio web de Vuforia. Se ofrecen diferentes tipos de licencia, para el desarrollo de la aplicación móvil se creó una cuenta con licencia gratuita. Una vez creada la cuenta de desarrollador, se inició sesión en el sitio web de Vuforia y se obtuvo acceso al panel de administración de licencia, una vez aquí, se creó una clave de licencia para el proyecto en la opción *Get Deployment Key* (ver figura 107).

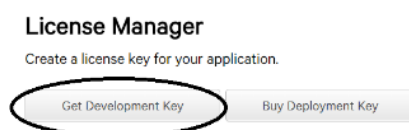


Figura 107 Crear clave de licencia Vuforia

Para crear la clave de licencia, solo se necesitó el nombre de aplicación, el cual puede ser cambiado cuando el usuario lo desee y aceptar el acuerdo de uso de licencia. Una vez hecho este proceso, en el panel de administración se muestra el nombre dado a la aplicación (ver figura 108).

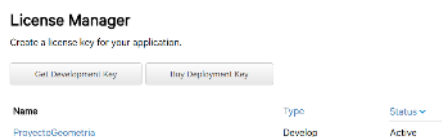


Figura 108 Panel de administración Vuforia

Al seleccionar *ProyectoGeometria*, se muestra la clave de licencia creada (ver figura 109).

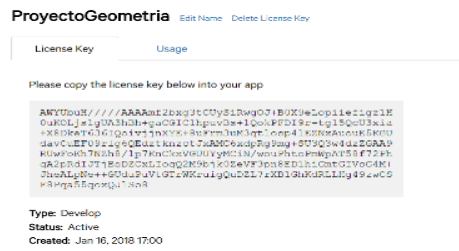


Figura 109 Clave de licencia Vuforia

La clave de licencia creada se copió para ser agregada a la aplicación creada en el editor de Unity. En el editor de Unity se creó un nuevo proyecto para el desarrollo de la aplicación, una vez creado se importó la extensión de Vuforia para Unity descargada anteriormente por medio del menú *Assets>Import Package> Custom Package* (ver figura 110).

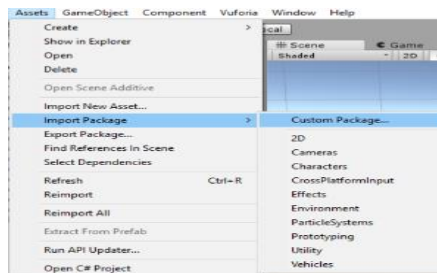


Figura 110 Importación de la extensión de Vuforia

Al crear el proyecto en Unity, se creó de manera automática un proyecto de Visual Studio. Una vez importada la extensión se buscó la carpeta *Assets > Vuforia > Prefabs* por medio del explorador de archivos del proyecto. Se agregó el elemento *ARCamera* seleccionándolo para posteriormente arrástralo y soltarlo en la escena del proyecto, se seleccionó *ARCamera* en el panel de administración de objetos de Unity y en el inspector se muestran las propiedades del objeto *ARCamera*, en el inspector se seleccionó la opción *Open Vuforia Configuration* (ver figura 111).

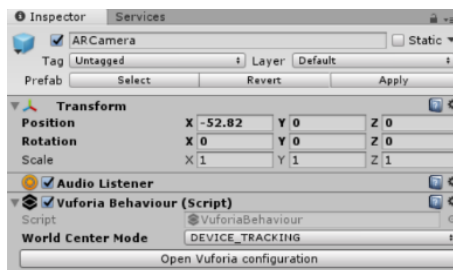


Figura 111 Propiedades del objeto ARCamera

Al seleccionar se muestra una serie de opciones y propiedades de configuración, en la caja de texto con nombre *App License Key* se pegó la llave generada al crear la clave de licencia en el panel de administración de Vuforia (ver figura 112).

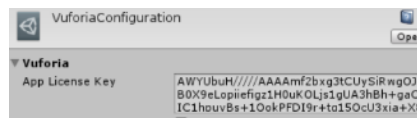


Figura 112 Agregar clave de licencia de Vuforia a Unity

Se procedió a la creación de la base de datos de las imágenes de destino en el panel de administración de Vuforia. Para la creación de la base de datos, se seleccionó la opción *Target Manager > Add Database*. El nombre asignado para la base de datos del proyecto es target (ver figura 113).

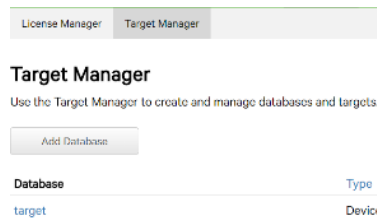


Figura 113 Panel de administración de bases de datos Vuforia

Anexo 2 Integración de modelos 3D y marcadores de realidad aumentada

aumentada

Prisma cuadrangular

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *cuad*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetCuad*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 114.

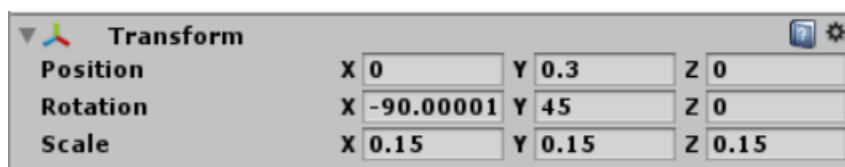


Figura 114 Propiedades prisma cuadrangular

Se creó un nuevo material por medio de *Assets>Create>Material*. Se renombró como *cuadmat*. Se modificó el color del material como se muestra en la figura 115 y el resultado se aprecia en la figura 116.

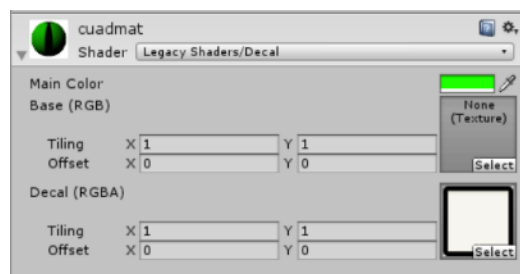


Figura 115 Material prisma cuadrangular

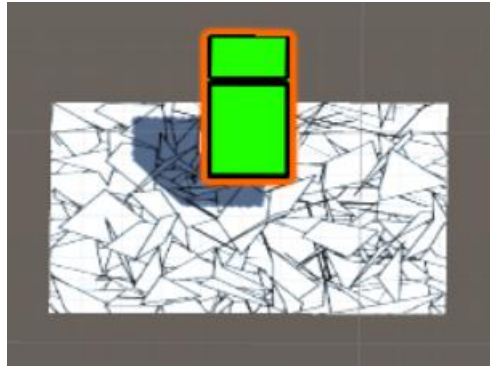


Figura 116 Prisma cuadrangular en marcador de realidad aumentada

Prisma pentagonal

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *pentágono*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetPenta*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 117.

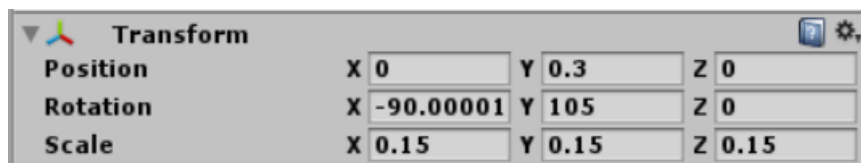


Figura 117 Propiedades prisma pentagonal

Se creó un nuevo material por medio de *Assets>Create>Material*. Se renombró como *pentamat*. Se modificó el color del material como se muestra en la figura 118 y el resultado se aprecia en la figura 119.

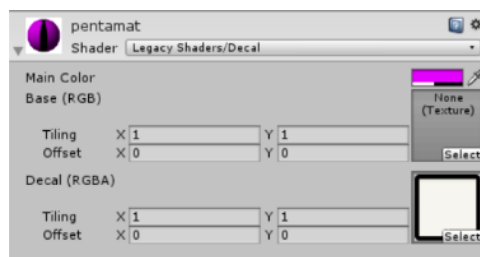


Figura 118 Material prisma pentagonal

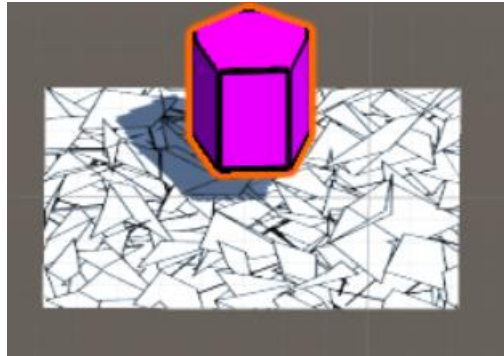


Figura 119 Prisma pentagonal en marcador de realidad aumentada

Prisma hexagonal

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *hexa*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetHexa*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 120.

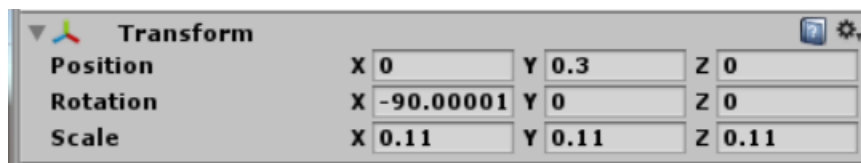


Figura 120 Propiedades prisma hexagonal

Se creó un nuevo material por medio de *Assets>Create>Material*. Se renombró como *hexamat*. Se modificó el color del material como se muestra en la figura 121 y el resultado se aprecia en la figura 122.

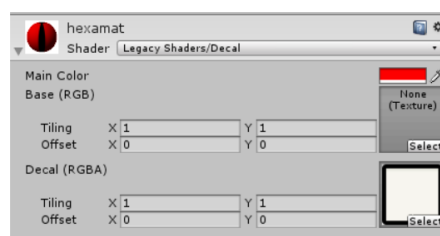


Figura 121 Material prisma hexagonal

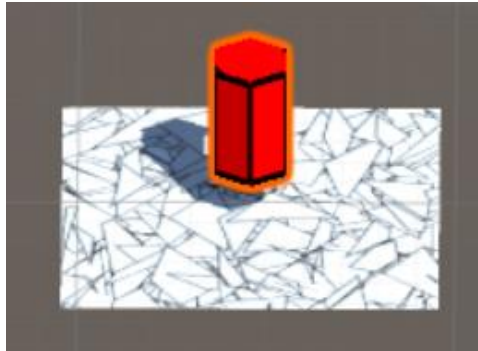


Figura 122 Prisma hexagonal y marcador de realidad aumentada

Pirámide triangular

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *pira3*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetPira3*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 123.

Transform						
Position	X	0	Y	0.3	Z	0
Rotation	X	-90.00001	Y	60	Z	0
Scale	X	0.17	Y	0.17	Z	0.17

Figura 123 Propiedades pirámide triangular

Se utilizó el mismo material del prisma triangular. El resultado se aprecia en la figura 124.



Figura 124 Pirámide triangular en marcador de realidad aumentada

Pirámide cuadrangular

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *pira4*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetPira4*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 125.

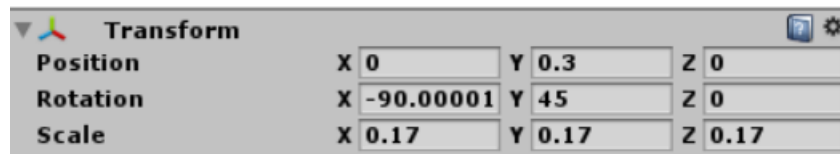


Figura 125 Propiedades pirámide cuadrangular

Se utilizó el mismo material del prisma cuadrangular. El resultado se aprecia en la figura 126.

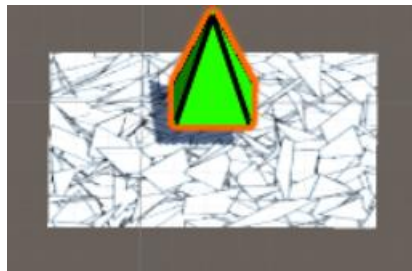


Figura 126 Pirámide cuadrangular en marcador de realidad aumentada

Pirámide pentagonal

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *pira5*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetPira5*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 127.

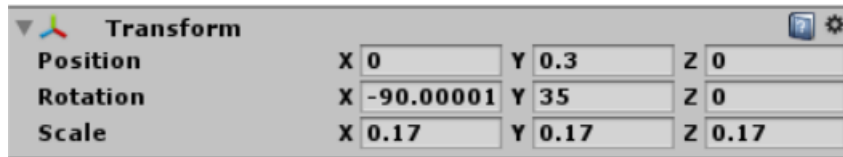


Figura 127 Propiedades pirámide pentagonal

Se utilizó el mismo material del prisma pentagonal. El resultado se aprecia en la figura 128.

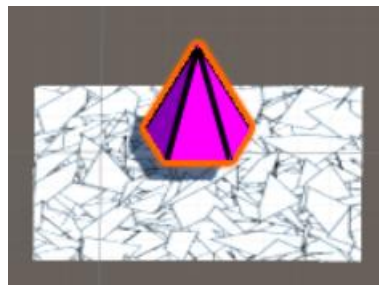


Figura 128 Pirámide pentagonal en marcador de realidad aumentada

Pirámide hexagonal

En el explorador de archivos del proyecto en Unity, se abrió la carpeta *Assets*. Se seleccionó el modelo *pira6*, se arrastró y soltó en la escena del proyecto.

Se colocó como hijo del marcador *ImageTargetPira6*. Se modificó la posición, rotación y escala del objeto como se muestra en la figura 129.

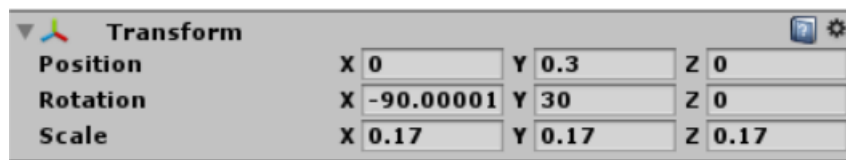


Figura 129 Propiedades pirámide hexagonal

Se utilizó el mismo material del prisma hexagonal. El resultado se aprecia en la figura 130.

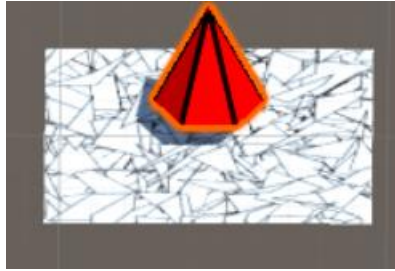


Figura 130 Pirámide hexagonal en marcador de realidad aumentada

Anexo 3 Código de scripts

Esfera

RotarHEsfera

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhesfera : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VButtonObject;
    private GameObject Sphere;
    public float rotateSpeed = 50f;
    // Use this for initialization
    void Start()
    {
        VButtonObject = GameObject.Find("RotarHEsfera");

        VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        Sphere = GameObject.Find("Sphere");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
            Debug.Log("prisionado");
            Sphere.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

            //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
            //} while (i == 0)
        }

        public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
        {
            //i = 1;
            Debug.Log("no");
        }
    }
}
```

RotarVEsfera

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvesfera : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VButtonObject;
    private GameObject Sphere;
    // Use this for initialization
    void Start()
    {
        VButtonObject = GameObject.Find("RotarVEsfera");
    }
}
```

```

VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    //
VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    Sphere = GameObject.Find("Sphere");
}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    Sphere.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

AumentarEsfera

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarEsfera : MonoBehaviour,
IVirtualButtonEventHandler {
    private GameObject VButtonObject;
    //private GameObject Cube;
    private GameObject Sphere;
    // Use this for initialization
    void Start () {
        VButtonObject = GameObject.Find("AumentarEsfera");

VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        Sphere = GameObject.Find("Sphere");
    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    Sphere.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

ReducirEsfera

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class ReducirEsfera : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject Sphere;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("ReducirEsfera");

        //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        Sphere = GameObject.Find("Sphere");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        Sphere.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }

}
}
```

Cono

RotarHCono

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhcono : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cono;
```

```

// Use this for initialization
void Start()
{
    VBButtonObject = GameObject.Find("RotarHCono");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    cono = GameObject.Find("cono");
}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    //do
    //{
    Debug.Log("prisionado");
    cono.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

    //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
    //} while (i == 0)
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    //i = 1;
    Debug.Log("no");
}
}

```

RotarVCono

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvcono : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cono;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVCono");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cono = GameObject.Find("cono");
    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
}
}

```

```

        cono.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarCono

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarCono : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject cono;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("AumentarCono");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cono = GameObject.Find("cono");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        cono.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

ReducirCono

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducircono : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cono;
    // Use this for initialization

```

```

void Start ()
{
    VBButtonObject = GameObject.Find("ReducirCono");

    //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
    Handler(this);

    VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
    ndler(this);
    cono = GameObject.Find("cono");

}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
    cono.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);

}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}

}

```

Cilindro

RotarHCilindro

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhcilindro : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cylinder;
    public float speed = 10f;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarHCilindro");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        cylinder = GameObject.Find("Cylinder");
    }
}

```

```

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    //do
    //{
    Debug.Log("prisionado");
    cylinder.transform.Rotate(new Vector3(0, Time.deltaTime *
1000, 0));
    //cylinder.transform.Rotate(Vector3.up, speed *
Time.deltaTime);
    //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
    //} while (i == 0)
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    //i = 1;
    Debug.Log("no");
}
}

```

RotarVCilidro

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvcilindro : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cylinder;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVCilindro");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cylinder = GameObject.Find("Cylinder");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        cylinder.transform.Rotate(new Vector3(Time.deltaTime * 1000,
0, 0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarCilindro

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarCilindro : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject cylinder;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("AumentarCilindro");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cylinder = GameObject.Find("Cylinder");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        cylinder.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}
```

ReducirCilindro

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducircilindro : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cylinder;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("ReducirCilindro");

//VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
}
```

```

        cylinder = GameObject.Find("Cylinder");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        cylinder.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

Prisma Triangular RotarHTria

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhtria : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject tria;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHTria");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        tria = GameObject.Find("tria");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        tria.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }
}

```

```

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

```

RotarVTria

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvtria : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject tria;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVTria");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        tria = GameObject.Find("tria");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prsionado");
        tria.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarTria

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarTria : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject tria;
    // Use this for initialization

```

```

void Start()
{
    VBButtonObject = GameObject.Find("AumentarTria");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    tria = GameObject.Find("tria");
}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    tria.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

ReducirTria

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;
public class ReducirTria : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject tria;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirTria");

//VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        tria = GameObject.Find("tria");

    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
    tria.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)

```

```

    {
        Debug.Log("no");
    }

}

```

Prisma Cuadrangular RotarHCuad

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhcua : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cuad;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarHCuad");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        cuad = GameObject.Find("cuad");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        cuad.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

```

RotarVCuad

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvcua : MonoBehaviour, IVirtualButtonEventHandler

```

```

{
    private GameObject VBButtonObject;
    private GameObject cuad;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarVCuad");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cuad = GameObject.Find("cuad");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        cuad.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarCuad

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Aumentarcuad : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject cuad;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("AumentarCuad");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        cuad = GameObject.Find("cuad");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        cuad.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }
}

```

```

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

ReducirCuad

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducircuad : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject cuad;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirCuad");

        //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
        Handler(this);

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        cuad = GameObject.Find("cuad");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
        1000));
        cuad.transform.localScale -= new Vector3(0.04F, 0.04F,
        0.04F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

Prisma Pentagonal

RotarHPenta

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

```

```

public class Rotarhpenta : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject penta;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHPenta");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        penta = GameObject.Find("pentagono");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        penta.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

```

RotarVPenta

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvpenta : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject penta;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVPenta");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        penta = GameObject.Find("pentagono");
    }
}

```

```

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    penta.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

AumentarPenta

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Aumentarpenta : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject penta;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("AumentarPenta");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        penta = GameObject.Find("pentagono");
    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    penta.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

ReducirPenta

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

```

```

public class Reducirpenta : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject penta;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirPenta");

//VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        penta = GameObject.Find("pentagono");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        penta.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }

}

```

Prisma Hexagonal RotarHHexa

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhhexa : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject hexa;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHHexa");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        hexa = GameObject.Find("hexa");
    }
}

```

```

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        hexa.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

```

RotarVHexa

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvhexa : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject hexa;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVHexa");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        hexa = GameObject.Find("hexa");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        hexa.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarHexa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarHexa : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject hexa;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("AumentarHexa");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        hexa = GameObject.Find("hexa");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        hexa.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}
```

ReducirHexa

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducirhexa : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject hexa;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("ReducirHexa");

        //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    }
}
```

```

        hexa = GameObject.Find("hexa");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        hexa.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}
}

```

Pirámide Triangular

RotarHPira3

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhpira3 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira3;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHPira3");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira3 = GameObject.Find("pira3");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        pira3.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }
}

```

```

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

RotarVPira3
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvpira3 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira3;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarVPira3");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira3 = GameObject.Find("pira3");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        pira3.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarPira3

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarPira3 : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject pira3;

```

```

// Use this for initialization
void Start()
{
    VBButtonObject = GameObject.Find("AumentarPira3");

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    pira3 = GameObject.Find("pira3");
}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    pira3.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

ReducirPira3

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducirpira3 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira3;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirPira3");

//VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira3 = GameObject.Find("pira3");
    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
    pira3.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
}
}

```

```

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }

}

```

Pirámide Cuadrangular

RotarHPira4

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhpira4 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira4;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHPira4");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        pira4 = GameObject.Find("pira4");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        pira4.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

RotarVPira4
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

```

```

public class Rotarvpira4 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira4;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarVPira4");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        pira4 = GameObject.Find("pira4");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        pira4.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

AumentarPira4

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarPira4 : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject pira4;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("AumentarPira4");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        pira4 = GameObject.Find("pira4");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
    }
}

```

```

        pira4.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

ReducirPira4

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducirpira4 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira4;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirPira4");

        //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira4 = GameObject.Find("pira4");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        pira4.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

Pirámide Pentagonal RotarHPira5

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhpira5 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira5;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarHPira5");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira5 = GameObject.Find("pira5");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        //do
        //{
        Debug.Log("prisionado");
        pira5.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

        //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
        //} while (i == 0)
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        //i = 1;
        Debug.Log("no");
    }
}

```

RotarVPira5

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvpira5 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira5;
    // Use this for initialization
    void Start ()
    {
        VBButtonObject = GameObject.Find("RotarVPira5");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira5 = GameObject.Find("pira5");
    }
}

```

```

}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    pira5.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("no");
}
}

```

AumentarPira5

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarPira5 : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject pira5;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("AumentarPira5");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira5 = GameObject.Find("pira5");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        pira5.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

ReducirPira5

```

using System.Collections;

```

```

using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class ReducirPira5 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira5;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirPira5");

        //VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
        Handler(this);

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
        ndler(this);
        pira5 = GameObject.Find("pira5");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
        1000));
        pira5.transform.localScale -= new Vector3(0.04F, 0.04F,
        0.04F);

    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }

}

```

Pirámide Hexagonal RotarHPira6

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarhpira6 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira6;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("RotarHPira6");
    }
}

```

```

VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    pira6 = GameObject.Find("pira6");
}

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    //do
    //{
    Debug.Log("prisionado");
    pira6.transform.Rotate(new Vector3(0, Time.deltaTime * 1000,
0));

    //Cube.transform.Rotate(new Vector3 (0, rotateSpeed *
Time.deltaTime, 0));
    //} while (i == 0)
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
{
    //i = 1;
    Debug.Log("no");
}
}

```

RotarVPira6

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Rotarvpira6 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VButtonObject;
    private GameObject pira6;
    // Use this for initialization
    void Start()
    {
        VButtonObject = GameObject.Find("RotarVPira6");

VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        //
VButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira6 = GameObject.Find("pira6");
    }

// Update is called once per frame
public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
{
    Debug.Log("prisionado");
    pira6.transform.Rotate(new Vector3(Time.deltaTime * 1000, 0,
0));
}

public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)

```

```

    {
        Debug.Log("no");
    }
}

```

AumentarPira6

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class AumentarPira6 : MonoBehaviour,
IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    //private GameObject Cube;
    private GameObject pira6;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("AumentarPira6");

        VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
        pira6 = GameObject.Find("pira6");
    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prisionado");
        pira6.transform.localScale += new Vector3(0.05F, 0.05F,
0.05F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }
}

```

ReducirPira6

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;

public class Reducirpira6 : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VBButtonObject;
    private GameObject pira6;
    // Use this for initialization
    void Start()
    {
        VBButtonObject = GameObject.Find("ReducirPira6");
    }
}

```

```

//VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEvent
Handler(this);

VBButtonObject.GetComponent<VirtualButtonBehaviour>().RegisterEventHa
ndler(this);
    pira6 = GameObject.Find("pira6");

    }

    // Update is called once per frame
    public void OnButtonPressed(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("prsionado");
        //ube.transform.Rotate(new Vector3(0, 0, Time.deltaTime *
1000));
        pira6.transform.localScale -= new Vector3(0.04F, 0.04F,
0.04F);
    }

    public void OnButtonReleased(VirtualButtonAbstractBehaviour vb)
    {
        Debug.Log("no");
    }

}

```

Anexo 4 Script para habilitar/deshabilitar botones virtuales

ToggleScript

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Vuforia;
using UnityEngine.UI; // libreria de interfaz de usuario de Unity

public class ToggleScript : MonoBehaviour {
    //declaracion botones cubo
    private GameObject btnRotarh;
    private GameObject btnRotarv;
    private GameObject btnAumentar;
    private GameObject btnReducir;

    //declaracion botones esfera
    private GameObject btnRotarhEsfera;
    private GameObject btnRotarvEsfera;
    private GameObject btnAumentarEsfera;
    private GameObject btnReducirEsfera;

    //declaracion botones prisma trinagular
    private GameObject btnRotarhTria;
    private GameObject btnRotarvTria;
    private GameObject btnAumentarTria;
    private GameObject btnReducirTria;

    //declaracion botones prisma cuadrangular
    private GameObject btnRotarhCuad;
    private GameObject btnRotarvCuad;
    private GameObject btnAumentarCuad;
    private GameObject btnReducirCuad;

    //declaracion botones prisma pentagonal
    private GameObject btnRotarhPenta;
    private GameObject btnRotarvPenta;
    private GameObject btnAumentarPenta;
    private GameObject btnReducirPenta;

    //declaracion botones prisma hexagonal
    private GameObject btnRotarhHexa;
    private GameObject btnRotarvHexa;
    private GameObject btnAumentarHexa;
    private GameObject btnReducirHexa;

    //declaracion botones piramide triangular

    private GameObject btnRotarhPira3;
    private GameObject btnRotarvPira3;
    private GameObject btnAumentarPira3;
    private GameObject btnReducirPira3;

    //declaracion botones piramide cuadrangular

    private GameObject btnRotarhPira4;
    private GameObject btnRotarvPira4;
    private GameObject btnAumentarPira4;
    private GameObject btnReducirPira4;
```

```

//botones piramide pentagonal

private GameObject btnRotarhPira5;
private GameObject btnRotarvPira5;
private GameObject btnAumentarPira5;
private GameObject btnReducirPira5;

//declaracion botones piramide hexagonal

private GameObject btnRotarhPira6;
private GameObject btnRotarvPira6;
private GameObject btnAumentarPira6;
private GameObject btnReducirPira6;

//declaracion botones cono

private GameObject btnRotarhCono;
private GameObject btnRotarvCono;
private GameObject btnAumentarCono;
private GameObject btnReducirCono;

//declaracion botones cilindro

private GameObject btnRotarhCilindro;
private GameObject btnRotarvCilindro;
private GameObject btnAumentarCilindro;
private GameObject btnReducirCilindro;

// declaracion botones piramide octagonal

private GameObject btnRotarhPira8;
private GameObject btnRotarvPira8;
private GameObject btnAumentarPira8;
private GameObject btnReducirPira8;

// casillas de verificacion de interfaz de usuario
Toggle t_rh;
Toggle t_rv;
Toggle t_a;
Toggle t_r;

// Use this for initialization
void Start () {

    // asignacion de botones virtuales cubo
    btnRotarh = GameObject.Find("Rotar");
    btnRotarv = GameObject.Find("RotarV");
    btnAumentar = GameObject.Find("Aumentar");
    btnReducir = GameObject.Find("Reducir");
    // asignacion de botones virtuales esfera
    btnRotarhEsfera = GameObject.Find("RotarHEsfera");
    btnRotarvEsfera = GameObject.Find("RotarVESfera");
    btnAumentarEsfera = GameObject.Find("AumentarEsfera");
    btnReducirEsfera = GameObject.Find("ReducirEsfera");
    // asignacion de botones virtuales prisma triangular
    btnRotarhTria = GameObject.Find("RotarHTria");
    btnRotarvTria = GameObject.Find("RotarVTria");
    btnAumentarTria = GameObject.Find("AumentarTria");
    btnReducirTria = GameObject.Find("ReducirTria");
    // asignacion de botones virtuales prisma cuadrangular

```

```

btnRotarhCuad = GameObject.Find("RotarHCuad");
btnRotarvCuad = GameObject.Find("RotarVCuad");
btnAumentarCuad = GameObject.Find("AumentarCuad");
btnReducirCuad = GameObject.Find("ReducirCuad");
// asignacion de botones virtuales prisma pentagonal
btnRotarhPenta = GameObject.Find("RotarHPenta");
btnRotarvPenta = GameObject.Find("RotarVPenta");
btnAumentarPenta = GameObject.Find("AumentarPenta");
btnReducirPenta = GameObject.Find("ReducirPenta");
// asignacion de botones virtuales prisma hexagonal
btnRotarhHexa = GameObject.Find("RotarHHexa");
btnRotarvHexa = GameObject.Find("RotarVHexa");
btnAumentarHexa = GameObject.Find("AumentarHexa");
btnReducirHexa = GameObject.Find("ReducirHexa");
// asignacion de botones virtuales piramide triangular
btnRotarhPira3 = GameObject.Find("RotarHPira3");
btnRotarvPira3 = GameObject.Find("RotarVPira3");
btnAumentarPira3 = GameObject.Find("AumentarPira3");
btnReducirPira3 = GameObject.Find("ReducirPira3");
// asignacion de botones virtuales piramide cuadrangular
btnRotarhPira4 = GameObject.Find("RotarHPira4");
btnRotarvPira4 = GameObject.Find("RotarVPira4");
btnAumentarPira4 = GameObject.Find("AumentarPira4");
btnReducirPira4 = GameObject.Find("ReducirPira4");
// asignacion de botones virtuales piramide pentagonal
btnRotarhPira5 = GameObject.Find("RotarHPira5");
btnRotarvPira5 = GameObject.Find("RotarVPira5");
btnAumentarPira5 = GameObject.Find("AumentarPira5");
btnReducirPira5 = GameObject.Find("ReducirPira5");
// asignacion de botones virtuales piramide hexagonal
btnRotarhPira6 = GameObject.Find("RotarHPira6");
btnRotarvPira6 = GameObject.Find("RotarVPira6");
btnAumentarPira6 = GameObject.Find("AumentarPira6");
btnReducirPira6 = GameObject.Find("ReducirPira6");
// // asignacion de botones virtuales cono
btnRotarhCono = GameObject.Find("RotarHCono");
btnRotarvCono = GameObject.Find("RotarVCono");
btnAumentarCono = GameObject.Find("AumentarCono");
btnReducirCono = GameObject.Find("ReducirCono");
// asignacion de botones virtuales cilindro
btnRotarhCilindro = GameObject.Find("RotarHCilindro");
btnRotarvCilindro = GameObject.Find("RotarVCilindro");
btnAumentarCilindro = GameObject.Find("AumentarCilindro");
btnReducirCilindro = GameObject.Find("ReducirCilindro");

btnRotarhPira8 = GameObject.Find("RotarHPira8");
btnRotarvPira8 = GameObject.Find("RotarVPira8");
btnAumentarPira8 = GameObject.Find("AumentarPira8");
btnReducirPira8 = GameObject.Find("ReducirPira8");

// asignacion de las casillas de verificacion(Toggle)
GameObject tglRH = GameObject.Find("Toggle_rh");
GameObject tglRV = GameObject.Find("Toggle_rv");
GameObject tglA = GameObject.Find("Toggle_a");
GameObject tglR = GameObject.Find("Toggle_r");
t_rh = tglRH.GetComponent<Toggle>();
t_rv = tglRV.GetComponent<Toggle>();
t_a = tglA.GetComponent<Toggle>();
t_r = tglR.GetComponent<Toggle>();

```

```
//habilitacion inicial de todos los botones virtuales para  
evitar conflictos de variables nulas
```

```
btnRotarh.SetActive(true);  
btnRotarv.SetActive(true);  
btnAumentar.SetActive(true);  
btnReducir.SetActive(true);
```

```
btnRotarhEsfera.SetActive(true);  
btnRotarvEsfera.SetActive(true);  
btnAumentarEsfera.SetActive(true);  
btnReducirEsfera.SetActive(true);
```

```
btnRotarhTria.SetActive(true);  
btnRotarvTria.SetActive(true);  
btnAumentarTria.SetActive(true);  
btnReducirTria.SetActive(true);
```

```
btnRotarhCuad.SetActive(true);  
btnRotarvCuad.SetActive(true);  
btnAumentarCuad.SetActive(true);  
btnReducirCuad.SetActive(true);
```

```
btnRotarhPenta.SetActive(true);  
btnRotarvPenta.SetActive(true);  
btnAumentarPenta.SetActive(true);  
btnReducirPenta.SetActive(true);
```

```
btnRotarhHexa.SetActive(true);  
btnRotarvHexa.SetActive(true);  
btnAumentarHexa.SetActive(true);  
btnReducirHexa.SetActive(true);
```

```
btnRotarhPira3.SetActive(true);  
btnRotarvPira3.SetActive(true);  
btnAumentarPira3.SetActive(true);  
btnReducirPira3.SetActive(true);
```

```
btnRotarhPira4.SetActive(true);  
btnRotarvPira4.SetActive(true);  
btnAumentarPira4.SetActive(true);  
btnReducirPira4.SetActive(true);
```

```
btnRotarhPira5.SetActive(true);  
btnRotarvPira5.SetActive(true);  
btnAumentarPira5.SetActive(true);  
btnReducirPira5.SetActive(true);
```

```
btnRotarhPira6.SetActive(true);  
btnRotarvPira6.SetActive(true);  
btnAumentarPira6.SetActive(true);  
btnReducirPira6.SetActive(true);
```

```
btnRotarhCono.SetActive(true);  
btnRotarvCono.SetActive(true);  
btnAumentarCono.SetActive(true);  
btnReducirCono.SetActive(true);
```

```
btnRotarhCilindro.SetActive(true);  
btnRotarvCilindro.SetActive(true);  
btnAumentarCilindro.SetActive(true);  
btnReducirCilindro.SetActive(true);
```

```

        btnRotarhPira8.SetActive(true);
        btnRotarvPira8.SetActive(true);
        btnAumentarPira8.SetActive(true);
        btnReducirPira8.SetActive(true);

        //desactivacion de todos los botones virtuales para
deshabilitar la visualiacion al iniciar la aplicacion
        Btnoffinicio();

    }

    public void T_cambio()// metodo que habilita los botones de
rotacion en eje Y y deshabilita el resto
    {

        if (t_rh.isOn == true)
        {
            Debug.Log("PRESSrh!!");
            t_rv.isOn = false;
            t_a.isOn = false;
            t_r.isOn = false;

            btnRotarh.SetActive(true);
            btnRotarv.SetActive(false);
            btnAumentar.SetActive(false);
            btnReducir.SetActive(false);

            btnRotarhEsfera.SetActive(true);
            btnRotarvEsfera.SetActive(false);
            btnAumentarEsfera.SetActive(false);
            btnReducirEsfera.SetActive(false);

            btnRotarhTria.SetActive(true);
            btnRotarvTria.SetActive(false);
            btnAumentarTria.SetActive(false);
            btnReducirTria.SetActive(false);

            btnRotarhCuad.SetActive(true);
            btnRotarvCuad.SetActive(false);
            btnAumentarCuad.SetActive(false);
            btnReducirCuad.SetActive(false);

            btnRotarhPenta.SetActive(true);
            btnRotarvPenta.SetActive(false);
            btnAumentarPenta.SetActive(false);
            btnReducirPenta.SetActive(false);

            btnRotarhHexa.SetActive(true);
            btnRotarvHexa.SetActive(false);
            btnAumentarHexa.SetActive(false);
            btnReducirHexa.SetActive(false);

            btnRotarhPira3.SetActive(true);
            btnRotarvPira3.SetActive(false);
            btnAumentarPira3.SetActive(false);
            btnReducirPira3.SetActive(false);

            btnRotarhPira4.SetActive(true);
            btnRotarvPira4.SetActive(false);

```

```

        btnAumentarPira4.SetActive(false);
        btnReducirPira4.SetActive(false);

        btnRotarhPira5.SetActive(true);
        btnRotarvPira5.SetActive(false);
        btnAumentarPira5.SetActive(false);
        btnReducirPira5.SetActive(false);

        btnRotarhPira6.SetActive(true);
        btnRotarvPira6.SetActive(false);
        btnAumentarPira6.SetActive(false);
        btnReducirPira6.SetActive(false);

        btnRotarhCono.SetActive(true);
        btnRotarvCono.SetActive(false);
        btnAumentarCono.SetActive(false);
        btnReducirCono.SetActive(false);

        btnRotarhCilindro.SetActive(true);
        btnRotarvCilindro.SetActive(false);
        btnAumentarCilindro.SetActive(false);
        btnReducirCilindro.SetActive(false);

        btnRotarhPira8.SetActive(true);
        btnRotarvPira8.SetActive(false);
        btnAumentarPira8.SetActive(false);
        btnReducirPira8.SetActive(false);
    }
    else
    {
        Btnoffinicio();
    }
}

public void T_cambioV()// metodo que habilita los botones de
rotacion en eje X y deshabilita el resto
{
    if (t_rv.isOn == true)
    {
        t_rh.isOn = false;
        t_a.isOn = false;
        t_r.isOn = false;

        btnRotarh.SetActive(false);
        btnRotarv.SetActive(true);
        btnAumentar.SetActive(false);
        btnReducir.SetActive(false);

        btnRotarhEsfera.SetActive(false);
        btnRotarvEsfera.SetActive(true);
        btnAumentarEsfera.SetActive(false);
        btnReducirEsfera.SetActive(false);

        btnRotarhTria.SetActive(false);
        btnRotarvTria.SetActive(true);
        btnAumentarTria.SetActive(false);
        btnReducirTria.SetActive(false);

        btnRotarhCuad.SetActive(false);
        btnRotarvCuad.SetActive(true);
    }
}

```

```

        btnAumentarCuad.SetActive(false);
        btnReducirCuad.SetActive(false);

        btnRotarhPenta.SetActive(false);
        btnRotarvPenta.SetActive(true);
        btnAumentarPenta.SetActive(false);
        btnReducirPenta.SetActive(false);

        btnRotarhHexa.SetActive(false);
        btnRotarvHexa.SetActive(true);
        btnAumentarHexa.SetActive(false);
        btnReducirHexa.SetActive(false);

        btnRotarhPira3.SetActive(false);
        btnRotarvPira3.SetActive(true);
        btnAumentarPira3.SetActive(false);
        btnReducirPira3.SetActive(false);

        btnRotarhPira4.SetActive(false);
        btnRotarvPira4.SetActive(true);
        btnAumentarPira4.SetActive(false);
        btnReducirPira4.SetActive(false);

        btnRotarhPira5.SetActive(false);
        btnRotarvPira5.SetActive(true);
        btnAumentarPira5.SetActive(false);
        btnReducirPira5.SetActive(false);

        btnRotarhPira6.SetActive(false);
        btnRotarvPira6.SetActive(true);
        btnAumentarPira6.SetActive(false);
        btnReducirPira6.SetActive(false);

        btnRotarhCono.SetActive(false);
        btnRotarvCono.SetActive(true);
        btnAumentarCono.SetActive(false);
        btnReducirCono.SetActive(false);

        btnRotarhCilindro.SetActive(false);
        btnRotarvCilindro.SetActive(true);
        btnAumentarCilindro.SetActive(false);
        btnReducirCilindro.SetActive(false);

        btnRotarhPira8.SetActive(false);
        btnRotarvPira8.SetActive(true);
        btnAumentarPira8.SetActive(false);
        btnReducirPira8.SetActive(false);

    }
    else
    {
        Btnoffinicio();
    }
}

public void T_cambioA()// metodo que habilita los botones de
aumento de escala y deshabilita el resto
{
    if (t_a.isOn == true)

```

```

{
    t_rv.isOn = false;
    t_rh.isOn = false;
    t_r.isOn = false;

    btnRotarh.SetActive(false);
    btnRotarv.SetActive(false);
    btnAumentar.SetActive(true);
    btnReducir.SetActive(false);

    btnRotarhEsfera.SetActive(false);
    btnRotarvEsfera.SetActive(false);
    btnAumentarEsfera.SetActive(true);
    btnReducirEsfera.SetActive(false);

    btnRotarhTria.SetActive(false);
    btnRotarvTria.SetActive(false);
    btnAumentarTria.SetActive(true);
    btnReducirTria.SetActive(false);

    btnRotarhCuad.SetActive(false);
    btnRotarvCuad.SetActive(false);
    btnAumentarCuad.SetActive(true);
    btnReducirCuad.SetActive(false);

    btnRotarhPenta.SetActive(false);
    btnRotarvPenta.SetActive(false);
    btnAumentarPenta.SetActive(true);
    btnReducirPenta.SetActive(false);

    btnRotarhHexa.SetActive(false);
    btnRotarvHexa.SetActive(false);
    btnAumentarHexa.SetActive(true);
    btnReducirHexa.SetActive(false);

    btnRotarhPira3.SetActive(false);
    btnRotarvPira3.SetActive(false);
    btnAumentarPira3.SetActive(true);
    btnReducirPira3.SetActive(false);

    btnRotarhPira4.SetActive(false);
    btnRotarvPira4.SetActive(false);
    btnAumentarPira4.SetActive(true);
    btnReducirPira4.SetActive(false);

    btnRotarhPira5.SetActive(false);
    btnRotarvPira5.SetActive(false);
    btnAumentarPira5.SetActive(true);
    btnReducirPira5.SetActive(false);

    btnRotarhPira6.SetActive(false);
    btnRotarvPira6.SetActive(false);
    btnAumentarPira6.SetActive(true);
    btnReducirPira6.SetActive(false);

    btnRotarhCono.SetActive(false);
    btnRotarvCono.SetActive(false);
    btnAumentarCono.SetActive(true);
    btnReducirCono.SetActive(false);

    btnRotarhCilindro.SetActive(false);

```

```

        btnRotarvCilindro.SetActive(false);
        btnAumentarCilindro.SetActive(true);
        btnReducirCilindro.SetActive(false);

        btnRotarhPira8.SetActive(false);
        btnRotarvPira8.SetActive(false);
        btnAumentarPira8.SetActive(true);
        btnReducirPira8.SetActive(false);
    }
    else
    {
        Btnoffinicio();
    }
}

public void T_CambioR()// metodo que habilita los botones de
reduccion de escala y deshabilita el resto
{
    if (t_r.isOn == true)
    {
        t_rv.isOn = false;
        t_rh.isOn = false;
        t_a.isOn = false;

        btnRotarh.SetActive(false);
        btnRotarv.SetActive(false);
        btnAumentar.SetActive(false);
        btnReducir.SetActive(true);

        btnRotarhEsfera.SetActive(false);
        btnRotarvEsfera.SetActive(false);
        btnAumentarEsfera.SetActive(false);
        btnReducirEsfera.SetActive(true);

        btnRotarhTria.SetActive(false);
        btnRotarvTria.SetActive(false);
        btnAumentarTria.SetActive(false);
        btnReducirTria.SetActive(true);

        btnRotarhCuad.SetActive(false);
        btnRotarvCuad.SetActive(false);
        btnAumentarCuad.SetActive(false);
        btnReducirCuad.SetActive(true);

        btnRotarhPenta.SetActive(false);
        btnRotarvPenta.SetActive(false);
        btnAumentarPenta.SetActive(false);
        btnReducirPenta.SetActive(true);

        btnRotarhHexa.SetActive(false);
        btnRotarvHexa.SetActive(false);
        btnAumentarHexa.SetActive(false);
        btnReducirHexa.SetActive(true);

        btnRotarhPira3.SetActive(false);
        btnRotarvPira3.SetActive(false);
        btnAumentarPira3.SetActive(false);
        btnReducirPira3.SetActive(true);

        btnRotarhPira4.SetActive(false);
    }
}

```

```

        btnRotarvPira4.SetActive(false);
        btnAumentarPira4.SetActive(false);
        btnReducirPira4.SetActive(true);

        btnRotarhPira5.SetActive(false);
        btnRotarvPira5.SetActive(false);
        btnAumentarPira5.SetActive(false);
        btnReducirPira5.SetActive(true);

        btnRotarhPira6.SetActive(false);
        btnRotarvPira6.SetActive(false);
        btnAumentarPira6.SetActive(false);
        btnReducirPira6.SetActive(true);

        btnRotarhCono.SetActive(false);
        btnRotarvCono.SetActive(false);
        btnAumentarCono.SetActive(false);
        btnReducirCono.SetActive(true);

        btnRotarhCilindro.SetActive(false);
        btnRotarvCilindro.SetActive(false);
        btnAumentarCilindro.SetActive(false);
        btnReducirCilindro.SetActive(true);

        btnRotarhPira8.SetActive(false);
        btnRotarvPira8.SetActive(false);
        btnAumentarPira8.SetActive(false);
        btnReducirPira8.SetActive(true);

    }
    else
    {
        Btnoffinicio();
    }
}

public void Btnoffinicio()// metodo que deshabilita todos los
botones virtuales
{
    btnRotarh.SetActive(false);
    btnRotarv.SetActive(false);
    btnAumentar.SetActive(false);
    btnReducir.SetActive(false);

    btnRotarhEsfera.SetActive(false);
    btnRotarvEsfera.SetActive(false);
    btnAumentarEsfera.SetActive(false);
    btnReducirEsfera.SetActive(false);

    btnRotarhTria.SetActive(false);
    btnRotarvTria.SetActive(false);
    btnAumentarTria.SetActive(false);
    btnReducirTria.SetActive(false);

    btnRotarhCuad.SetActive(false);
    btnRotarvCuad.SetActive(false);
    btnAumentarCuad.SetActive(false);
    btnReducirCuad.SetActive(false);

    btnRotarhPenta.SetActive(false);

```

```

    btnRotarvPenta.SetActive(false);
    btnAumentarPenta.SetActive(false);
    btnReducirPenta.SetActive(false);

    btnRotarhHexa.SetActive(false);
    btnRotarvHexa.SetActive(false);
    btnAumentarHexa.SetActive(false);
    btnReducirHexa.SetActive(false);

    btnRotarhPira3.SetActive(false);
    btnRotarvPira3.SetActive(false);
    btnAumentarPira3.SetActive(false);
    btnReducirPira3.SetActive(false);

    btnRotarhPira4.SetActive(false);
    btnRotarvPira4.SetActive(false);
    btnAumentarPira4.SetActive(false);
    btnReducirPira4.SetActive(false);

    btnRotarhPira5.SetActive(false);
    btnRotarvPira5.SetActive(false);
    btnAumentarPira5.SetActive(false);
    btnReducirPira5.SetActive(false);

    btnRotarhPira6.SetActive(false);
    btnRotarvPira6.SetActive(false);
    btnAumentarPira6.SetActive(false);
    btnReducirPira6.SetActive(false);

    btnRotarhCono.SetActive(false);
    btnRotarvCono.SetActive(false);
    btnAumentarCono.SetActive(false);
    btnReducirCono.SetActive(false);

    btnRotarhCilindro.SetActive(false);
    btnRotarvCilindro.SetActive(false);
    btnAumentarCilindro.SetActive(false);
    btnReducirCilindro.SetActive(false);

    btnRotarhPira8.SetActive(false);
    btnRotarvPira8.SetActive(false);
    btnAumentarPira8.SetActive(false);
    btnReducirPira8.SetActive(false);
}

// Update is called once per frame
void Update () {
}
}

```

Anexo 5 Fórmulas de las figuras geométricas

Cubo

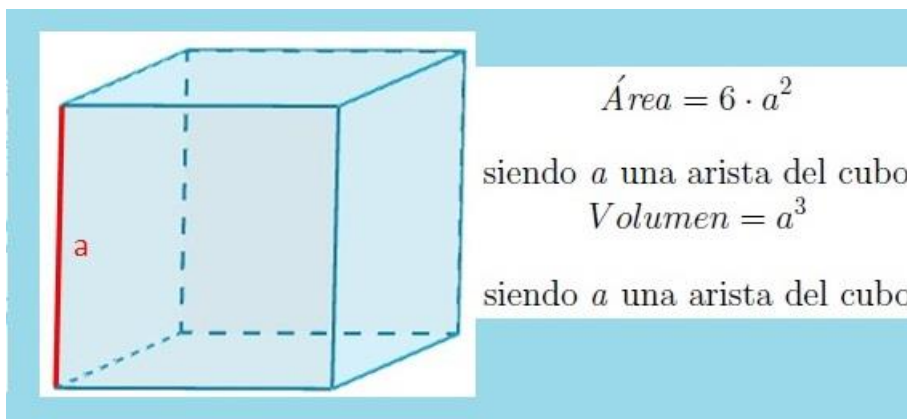


Figura 131 Fórmulas cubo

Esfera

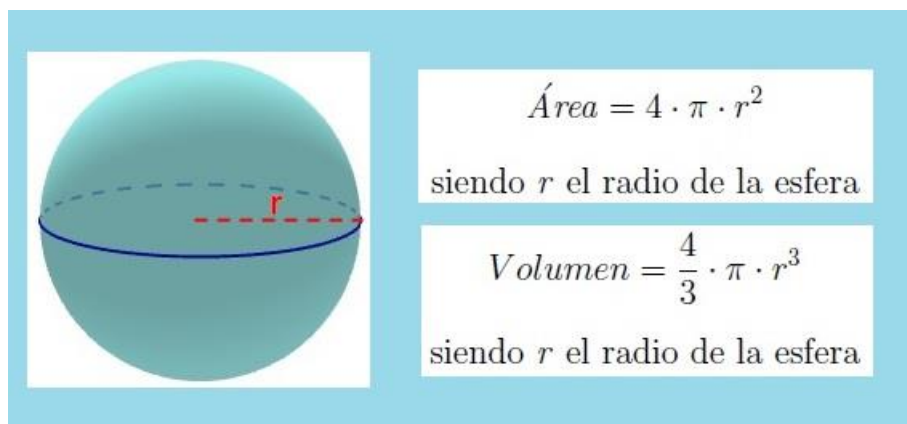


Figura 132 Fórmulas esfera

Cono

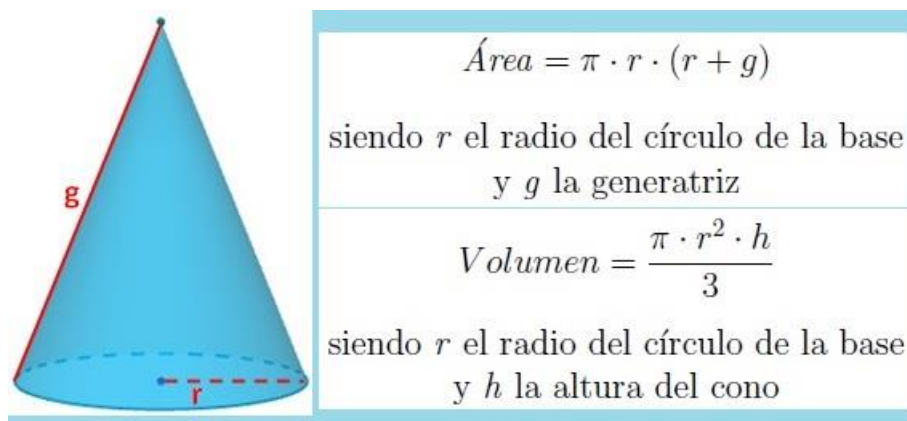


Figura 133 Fórmulas cono

Cilindro

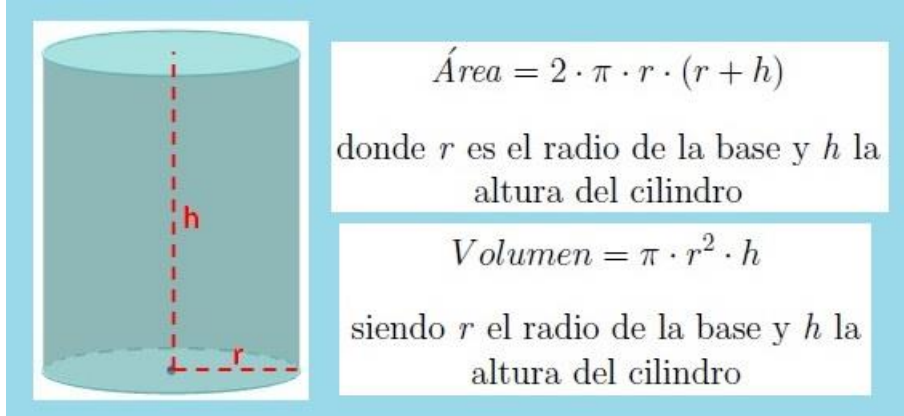


Figura 134 Fórmulas cilindro

Prisma Triangular

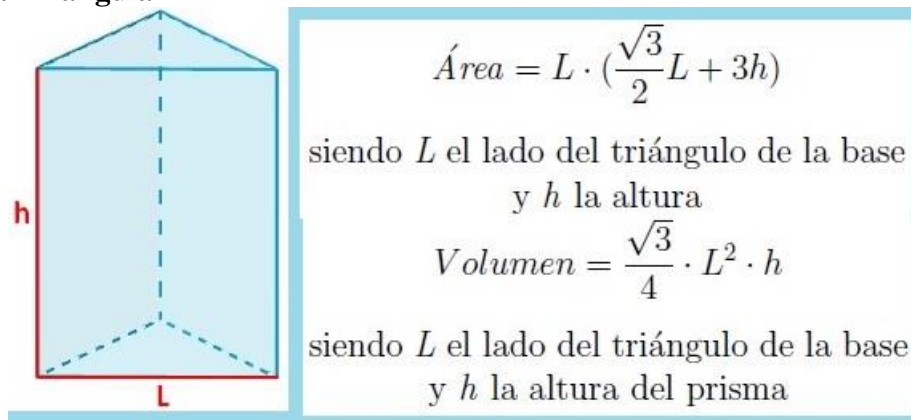


Figura 135 Fórmulas prisma triangular

Prisma Cuadrangular

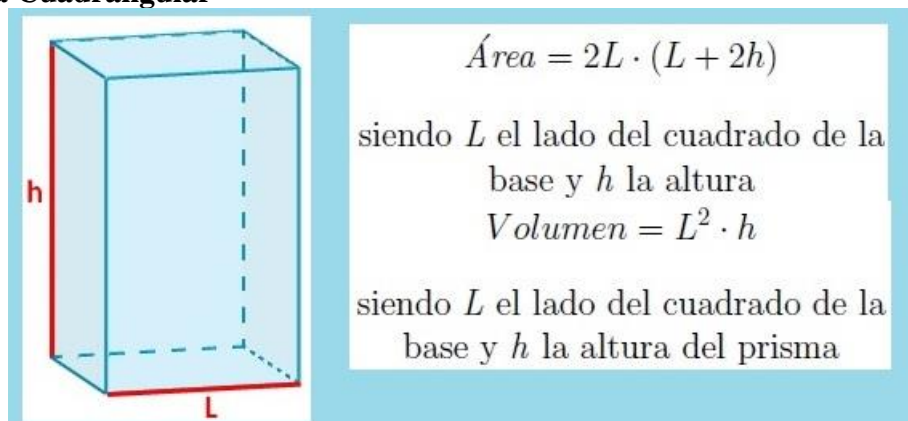
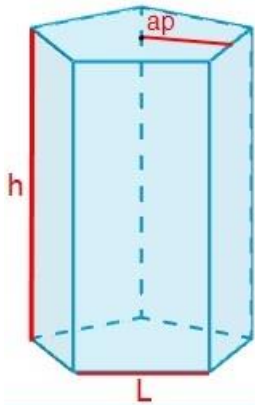


Figura 136 Fórmulas prisma cuadrangular

Prisma Pentagonal



$$\text{Área} = 5 \cdot L \cdot (ap + h)$$

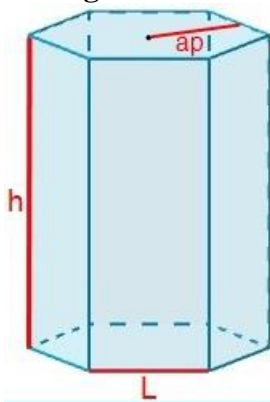
donde L es un lado del pentágono, ap su apotema y h la altura del prisma

$$\text{Volumen} = \frac{5 \cdot L \cdot ap}{2} \cdot h$$

donde L es la longitud del pentágono, ap su apotema y h la altura del prisma

Figura 137 Fórmulas prisma pentagonal

Prisma Hexagonal



$$\text{Área} = 6 \cdot L \cdot (ap + h)$$

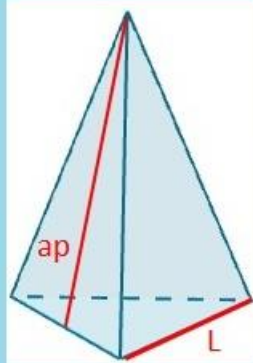
donde L es un lado del hexágono, ap su apotema y h la altura del prisma

$$\text{Volumen} = 3 \cdot L \cdot ap \cdot h$$

donde L es la longitud del hexágono, ap su apotema y h la altura del prisma

Figura 138 Fórmulas prisma hexagonal

Pirámide Triangular



$$\text{Área} = \frac{\sqrt{3}}{2} \cdot L \cdot \left(\frac{1}{2}L + \sqrt{3} \cdot ap\right)$$

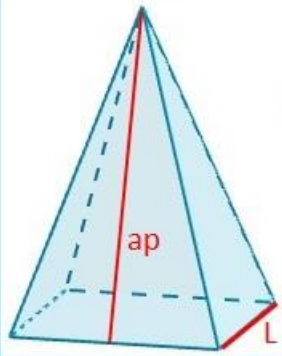
donde L es una arista de la base y ap la apotema de la pirámide

$$\text{Volumen} = \frac{\sqrt{3}}{12} \cdot L^2 \cdot h$$

donde L es una arista de la base y h la altura de la pirámide

Figura 139 Fórmulas pirámide triangular

Pirámide Cuadrangular


$$\text{Área} = L \cdot (2 \cdot ap + L)$$

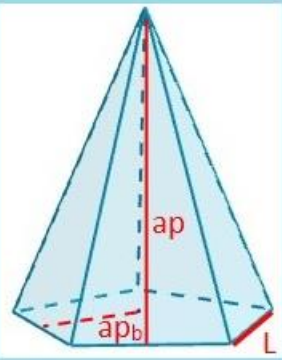
donde L es una arista de la base y ap la apotema de la pirámide

$$\text{Volumen} = \frac{1}{3} \cdot L^2 \cdot h$$

donde L es una arista de la base y h la altura de la pirámide

Figura 140 Fórmulas pirámide cuadrangular

Pirámide Pentagonal


$$\text{Área} = \frac{5 \cdot L}{2} \cdot (ap + ap_b)$$

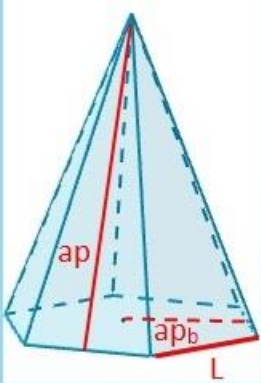
donde L y ap_b son una arista y la apotema de la base y ap la apotema de la pirámide

$$\text{Volumen} = \frac{5}{6} \cdot L \cdot ap_b \cdot h$$

donde L y ap_b son una arista y la apotema de la base y h la altura de la pirámide

Figura 141 Fórmulas pirámide pentagonal

Pirámide Hexagonal


$$\text{Área} = 3 \cdot L \cdot (ap + ap_b)$$

donde L y ap_b son una arista y la apotema de la base y ap la apotema de la pirámide

$$\text{Volumen} = L \cdot ap_b \cdot h$$

donde L y ap_b son una arista y la apotema de la base y h la altura de la pirámide

Figura 142 Fórmulas pirámide hexagonal