PLATAFORMA EDUCATIVA DE REALIDAD AUMENTADA PARA LA ENSEÑANZA DE ROBÓTICA.

Autor
JEFERSON DAYAN SÁNCHEZ JEREZ

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

PROGRAMA DE INGENIERÍA MECATRÓNICA

PAMPLONA, COLOMBIA

2017

PLATAFORMA EDUCATIVA DE REALIDAD AUMENTADA PARA LA ENSEÑANZA DE ROBÓTICA.

Autor

JEFERSON DAYAN SÁNCHEZ JEREZ

Director

PhD. CÉSAR AUGUSTO PEÑA CORTES

Ingeniero Electromecánico

Doctor en Automática y Robótica

UNIVERSIDAD DE PAMPLONA

FACULTAD DE INGENIERÍAS Y ARQUITECTURA

PROGRAMA DE INGENIERÍA MECATRÓNICA

PAMPLONA, COLOMBIA

2017

Nota de Aceptación

_____

_____

Director de trabajo de grado

_____

Jurado

_____

Jurado

Pamplona, 2017

Dedicatoria

A mis padres: Ricardo Sánchez Y Sandra Jerez
Por su constante apoyo y sacrificio, gracias a los cuales me
Forme profesional y personalmente, por darme la vida
E inculcarme los valores y actitudes que me hacen ser quien soy.

A mi familia, por su entrega y aliento incondicional en
Todos los momentos de dificultad en mi vida estudiantil, y por su apoyo
Para superar con esfuerzo e ímpetu todos ellos.

A mis profesores, de primaria a estudios profesionales, que me instruyeron
Durante todo mi ciclo de formación. Y gracias a los cuales el día de hoy
Alcanzare una de las mayores metas de mi vida.

A Dios por darme la fortaleza para seguir adelante
Y por permitirme terminar satisfactoriamente mi ciclo de formación profesional.

CONTENIDO

CONTENIDO V
ÍNDICE DE ILUSTRACIONES VII
ÍNDICE DE TABLAS XIII
RESUMEN DEL PROYECTO XIV
PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN XV
OBJETIVOS XVI
1. MARCO DE REFERENCIA 1
1.1. HISTORIA 1
1.2. ESTADO DEL ARTE 4
1.2.1. Educación 4
1.2.2. Cultura 6
1.2.3. Entretenimiento y Comercio 6
1.3. MARCO CONTEXTUAL 10
1.3.1. Realidad aumentada 10
1.3.2. Tipos de seguimientos 12
1.3.3. Tecnologías de visualización 14
1.3.4. Alcances y generalidades 17
1.3.5. Herramientas de desarrollo 19
1.3.6. Funcionamiento básico 21
1.3.7. Fundamentos de robótica 23
1.3.7.1. Historia de la robótica – Breve resumen 23
1.3.7.2. Tipos de robots 26
1.3.7.3. Estructura de los robots industriales 29
1.3.7.4. Configuraciones morfológicas comunes 31
1.3.7.5. Cinemática 33
2. BLENDER 38
2.1. CARACTERÍSTICAS 38
2.2. MOTORES GRÁFICOS 39
2.2.1. Interno 39
2.2.2. Cycles 40

2.3. INSTALACIÓN 40
2.4. ENTORNO DE TRABAJO 40
2.4.1. Entorno de trabajo: Default 42
3. UNITY – VUFORIA 44
3.1. INSTALACIÓN UNITY 44
3.2. CONFIGURACIONES UNITY 45
3.3. VUFORIA 46
3.3.1. Marcadores Vuforia 46
3.3.2. Creación de marcadores Vuforia 47
3.3.3. Creando licencia para el aplicativo 48
3.3.4. Instalación Vuforia-Unity 49
4. DESARROLLO 51
4.1. CINEMÁTICA 51
4.1.1. Cinemática directa 51
4.1.2. Cinemática inversa 57
4.2. MODELADO Y ANIMACIÓN 62
4.2.1. Modelado Raplim 69
4.2.2. Animación Raplim 71
4.2.3. Algoritmo cinemática inversa Python – Liclipse 75
4.2.4. Configuraciones de exportación y modelado Blender 77
4.3. CREACIÓN DE PROYECTO EN UNITY 77
4.3.1. Primer ensayo 78
4.3.2. Segundo ensayo 82
4.3.3. Tercer ensayo 85
4.3.4. Clases C# 90
4.3.4.1. Clase inversa información 90
4.3.4.2. Clase dh 93
4.3.4.3. Clase Activar animación 97
4.3.4.4. Clase Sonido2 98
RESULTADOS 100
CONCLUSIONES
Bibliografía

ÍNDICE DE ILUSTRACIONES

Ilustración 1: Primer sistema de realidad aumentada 1

Ilustración 2: Proyecto Boeing, Caudell y Mizell 2

Ilustración 3: Aplicativo basado en el sistema Studierstube aplicado a la enseñanza de geometría 1996 3

Ilustración 4: Primeros aplicativos de código abierto RA 3

Ilustración 5: HoloLens, Google Glass 4

Ilustración 6: Pruebas al sistema de RA 5

Ilustración 7: Pruebas del sistema Construct3D 5

Ilustración 8: Séptima iteración del trabajo para el museo de américas 6

Ilustración 9: Aplicación para la empresa Duscholux 7

Ilustración 10: Probador Virtual 7

Ilustración 11: Libro interactivo 8

Ilustración 12: Realidad aumentada en medicina 8

Ilustración 13: Realidad Aumentada en medicina 9

Ilustración 14: Realidad aumentada en medicina 2 9

Ilustración 15: Mesa real, y lámpara y sillas virtuales 10

Ilustración 16: Elementos básicos de un sistema de realidad aumentada 11

Ilustración 17: Marcador vuforia 13

Ilustración 18: Marcador cilíndrico vuforia 13

Ilustración 19: Pantallas ópticas transparentes 14

Ilustración 20: Pantalla de superposición de imágenes 15

Ilustración 21: Display Espacial 16

Ilustración 22: gafas de realidad aumentada 16
Ilustración 23: Versión física Raplim 17
Ilustración 24: Dimensiones raplim 18
Ilustración 25. Diagrama Funcionamiento 21
Ilustración 26: Unity - Vuforia 22
Ilustración 27: El gallo de Estrasburgo 24
Ilustración 28: Muñeca dibujante de Henri Maillardert 24
Ilustración 29: Anatomía modelo para el desarrollo de robots industriales 25
Ilustración 30: Robot Da Vinci 26
Ilustración 31: Robot Humanoide 27
Ilustración 32: Robot móvil 27
Ilustración 33: Robot industrial Kuka 28
Ilustración 34: Robot Veebot 28
Ilustración 35: Robots tele-operados 29
Ilustración 36: Componentes brazo robótico 29
Ilustración 37: Eslabones 30
Ilustración 38: Articulación prismática 30
Ilustración 39: Articulaciones rotacionales 30
Ilustración 40: Robot cartesiano 31
Ilustración 41: Robot cilíndrico 31
Ilustración 42: Robot polar 32
Ilustración 43: Robot articulado 32
Ilustración 44: Robot Scara 32
Ilustración 45: Robot paralelo 33

Ilustración 46: Aplicación de algoritmo dh 34
Ilustración 47: Aplicación cinemática inversa 36
Ilustración 48: Elección de posición de codo del brazo robótico 37
Ilustración 49: Logo Blender. 38
Ilustración 50: Motores de pre-renderizado 39
Ilustración 51: Entornos de trabajo Blender 41
Ilustración 52: Selección de entorno de trabajo 41
Ilustración 53: Entorno de trabajo Blender 43
Ilustración 54: Assets Unity 44
Ilustración 55: Asistente de instalación 45
Ilustración 56: Preferencias Unity 46
Ilustración 57. Creando base de datos para marcadores 47
Ilustración 58: Creando y añadiendo marcador a la base de datos 48
Ilustración 59: Creando licencia 49
Ilustración 60: Importar Vuforia a Unity 50
Ilustración 61: Importación de Vuforia 50
Ilustración 62: Diagrama de desarrollo del proyecto. 51
Ilustración 63: Posición inicial Raplim, cinemática directa 51
Ilustración 64: Estructura cinemática Raplim 52
Ilustración 65: Eslabones y articulaciones Raplim 52
Ilustración 66: Ejes de rotación Raplim 53
Ilustración 67: Ejes Zi Raplim 53
Ilustración 68: Sistema S0 Raplim 54
Ilustración 69: Ubicación ejes Z1 a Zn-1 54

Ilustración 70: Posición X1 a Xn-1 Raplim 55
Ilustración 71: Sistemas S1 a Sn-1 Raplim 55
Ilustración 72: Sistema S4 Raplim 56
Ilustración 73: Vista superior Raplim 57
Ilustración 74: Vista desde el plano r parte 1 58
Ilustración 75: Vista desde plano r parte 2 59
Ilustración 76: Vista desde plano r parte 3 60
Ilustración 77: Operación de revolución 62
Ilustración 78: Agregar Primitivas 63
Ilustración 79: Modelo de avión Blender 63
Ilustración 80. Modelo escudo universidad de Pamplona 64
Ilustración 81: Creando materiales parte 1 64
Ilustración 82: Creando materiales parte 2 65
Ilustración 83: Creando material parte 3 65
Ilustración 84: Pre-renderizado motor gráfico interno 66
Ilustración 85: Pre-renderizado motor gráfico cycles 66
Ilustración 86: Mejora de materiales con nodos de composición 67
Ilustración 87: Emparentar por Huesos 67
Ilustración 88: Emparentar por objeto 68
Ilustración 89: Animación Blender 69
Ilustración 90: Escalado modelo en Blender 70
Ilustración 91: Cambio punto pivote modelos 70
Ilustración 92: Desarrollo de nuevos efectores finales – Blender 71
Ilustración 93: Restricción subordinar parámetros 73

Ilustración 94: Restricción subordinar 74
Ilustración 95: Interpolación curvas 74
Ilustración 96: Diagrama esquemático primer prototipo 78
Ilustración 97: Propiedades (Inspector), FrameMarker 78
Ilustración 98: Propiedades Camera y ARcamera 79
Ilustración 99: Resultados primera prueba 81
Ilustración 100: Resultados segunda prueba 81
Ilustración 101: Diagrama conceptual segundo prototipo 82
Ilustración 102: Motor gáfico Unity Vs Blender 82
Ilustración 103: Biblioteca de assets proyecto - Pestaña project 83
Ilustración 104: Enlace Asset Modelo-Simulación 83
Ilustración 105: Animación Raplim en Unity 84
Ilustración 106: Raplim - Primera prueba RA 84
Ilustración 107: Raplim - Segunda prueba RA 85
Ilustración 108: Diagrama conceptual tercer prototipo 85
Ilustración 109: Primera Interfaz 86
Ilustración 110: Segunda interfaz 86
Ilustración 111: Interfaz final aplicativo 87
Ilustración 112: Aplicativo con interfaz final 88
Ilustración 113: Componente Animation 89
Ilustración 114: Evolución del modelo 3D 100
Ilustración 115: Secuencia cuchara entorno Blender 100
Ilustración 116: Evolución del aplicativo de RA 101
Ilustración 117: Icono RA Raplim 101

Ilustración 118: Evolución interfaces de visualización 101

Ilustración 119: Secuencias en aplicativo RA Raplim 102

ÍNDICE DE TABLAS

Tabla 1: Tipos de tracking 12

Tabla 2: SDK de desarrollo, realidad aumentada parte 1 20

Tabla 3: SDK de desarrollo, realidad aumentada parte 2 21

Tabla 4: Parámetros DH Raplim 56

Tabla 5: Posiciones actuadores Raplim 71

Tabla 6: Dimensiones cuchara 72

Tabla 7: Dimensiones vaso 72

Tabla 8: Puntos intermedios predefinidos 73

RESUMEN DEL PROYECTO

El presente proyecto tiene como fin desarrollar una plataforma educativa de realidad aumentada, que permitirá al usuario visualizar sobre una superficie del mundo físico una proyección tridimensional de los movimientos articulares y desplazamientos de un robot serial antropomórfico, necesarios para tomar o alcanzar un objeto. Así mismo el usuario al ubicarse en diferentes ángulos alrededor de la superficie podrá visualizar el robot desde un punto de vista diferente.

El sistema de realidad aumentada permitirá al usuario visualizar en tiempo real los ángulos de las articulaciones, la trayectoria y/o posición del efector final determinados en la simulación de la cinemática directa e inversa del robot.

Este proyecto tiene como finalidad proporcionar al estudiantado una herramienta de uso común y relativamente bajo costo, como lo son los celulares inteligentes para implementar sistemas robóticos. Adicionalmente este sistema podría extrapolarse para ilustrar a los usuarios como sería la implementación de un robot en un sitio en particular antes de ejecutarse la misma evitando maquetos o herramientas similares, lo cual será muy atractivo para futuros compradores de un sistema de automatización.

PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

Para enseñar y aprender robótica de manera óptima es indispensable contar con los equipos y herramientas necesarias tales como motores, sensores y robots industriales, los cuales representarían una alta inversión económica para la mayoría de instituciones, lo cual limita el aprendizaje y experiencia que pueden obtener los estudiantes en sus academias.

Una futura solución a esta problemática yace en las nuevas tecnologías emergentes, entre estas la realidad aumentada, la cual está al alcance de todos debido al desarrollo tecnológico de equipos móviles o gafas de bajo costo con las capacidades necesarias para emplear esta tecnología. La versatilidad de la misma permite que esta pueda ser implementada en cualquier campo de conocimiento, entre estos la robótica, ya que al ocuparla de manera efectiva es posible desarrollar laboratorios de robótica de alta tecnología con una baja inversión.

OBJETIVOS

OBJETIVO GENERAL

Desarrollar plataforma educativa de realidad aumentada para enseñar robótica.

OBJETIVOS ESPECÍFICOS

1. Modelar un robot serial tipo antropomórfico para el entorno de realidad aumentada
2. Realizar las animaciones correspondientes a la cinemática inversa y directa del robot serial a emplear.
3. Integración y puesta a punto del sistema de realidad aumentada.

.

1. MARCO DE REFERENCIA

La realidad aumentada (RA) es una tecnología que hoy en día ha tenido gran auge, gracias a los alcances y posibilidades que ésta ofrece, pudiendo ser inmersa en varios campos de aplicación y áreas de conocimiento. Pero solo hasta principios del siglo XXI logra alcanzar su boom, gracias al desarrollo de dispositivos móviles (gafas de realidad aumentada, smartphones, tablets, etc.) con la capacidad de procesamiento necesaria para emplear esta tecnología en cualquier momento y lugar.

1.1. HISTORIA

El primer registro que se tiene de esta tecnología yace en los años 60, cuando el afiliado a la universidad de Harvard, Ivan Sutherland, padre de la realidad aumentada y virtual, desarrolla el primer sistema de RA. [1] Por medio del cual, se podía visualizar a través de un dispositivo montado sobre la cabeza, dibujos básicos, basados en líneas generadas a través de un ordenador. [2]

Ilustración 1: Primer sistema de realidad aumentada, tomado de: [3]

Con el auge tecnológico de la época, los avances en el rendimiento de sistemas de cómputo y el primer contacto con esta tecnología, proyectaron los alcances futuros de la misma, conllevando al desarrollo de nuevos sistemas de realidad aumentada e infraestructuras especializadas.

En 1974, Myron Krueger, construye el primer laboratorio de realidad artificial al cual denomina Videoplace. Él cual combinaba proyectores con videocámaras, para proyectar alrededor de los usuarios un entorno interactivo que podía ser visualizado a través de una pantalla. [2]

Asimismo, otros investigadores de la época, como Dan Sandin, Scott Fisher y otros científicos, experimentan diferentes conceptos, con el propósito de crear una interacción humana-máquina, a través de la superposición de videos en vivo con información generada a través de ordenadores, concibiendo así ambientes y experiencias interactivas. [3]

En 1992, por primera vez aparece el término realidad aumentada, en la obra de Caudell y Mizell (1992) en Boeing, cuyo propósito era enseñar a los trabajadores de una manufacturera de aviones, los esquemas eléctricos de éstos a través de un dispositivo.

Ilustración 2: Proyecto Boeing, Caudell y Mizell, tomado de: [3]

Los años 90 fue una de las décadas de apogeo de esta tecnología en el siglo XX, empresas privadas, organizaciones estatales, y centros de formación universitaria apostaron al desarrollo de sistemas de realidad aumentada, para campos como la medicina, turismo, entre otros.

Algunos de los proyectos más relevantes de esta época fueron: El camaleón, desarrollado por Fitzmaurice en 1993, el cual consistía en la visualización de variables meteorológicas, posicionamiento, etc., empleando dispositivos magnéticos de seguimiento; A mediados de los años 90 Steve Mann, afiliado de la MIT desarrolla el primer sistema que permitía al usuario alterar el contenido visualizado por éste; En 1996, la empresa desarrolladora Studierstube desarrolla el primer sistema AR compartido, el cual permitía a múltiples usuarios ver el mismo espacio y realidad aumentada al tiempo. [3]

El impacto de esta tecnología fue tan relevante, que instituciones estatales asignaron grandes recursos, para promocionar el avance y desarrollo de la realidad aumentada. La inversión del ministerio de educación alemán asigna un presupuesto inicial de 21 millones de Euros para un programa de realidad aumentada llamado ARVIKA, en el cual más de 20 grupos de

investigación trabajaron en conjunto con la academia para desarrolla aplicativos de realidad aumentada para la industria.

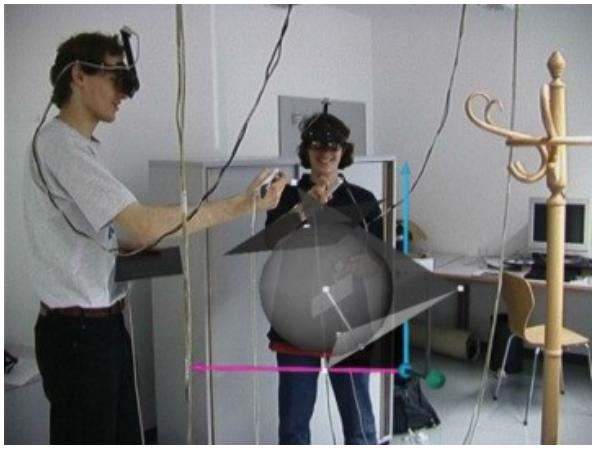


Ilustración 3: Aplicativo basado en el sistema Studierstube aplicado a la enseñanza de geometría 1996, tomado de: [3]

En 1999 se desarrolla la primera plataforma de desarrollo de código abierto para realidad aumentada, desarrolla por Kato y Billinghurst, llamada ARToolKit, la cual se basa en una librería rastreo en 3D, empleándose por primera vez los marcadores como punto de referencia para el sistema. [3]

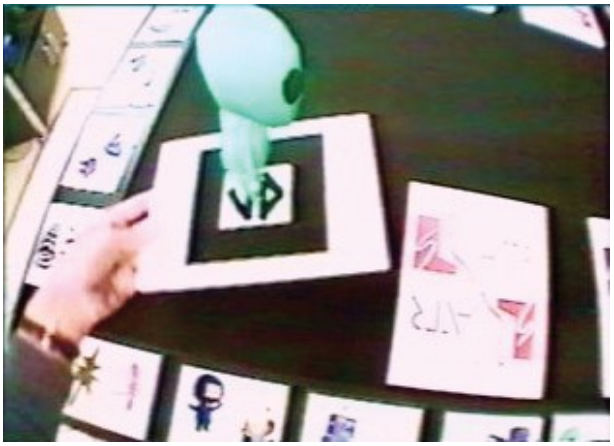


Ilustración 4: Primeros aplicativos de código abierto RA, tomado de: [3]

En 2003, el avance de los teléfonos celulares, permite el desarrollo de sistemas de realidad aumentada autónomas para dispositivos móviles de esa época. Pero es hasta el año 2008 que se desarrolla la primera plataforma RA para teléfonos inteligentes, la cual es el ancestro de la librería VUFORIA, seleccionada para el desarrollo del presente trabajo. [3]

Este mismo año, aparece la primera aplicación comercial de realidad aumentada, desarrollada por una agencia de publicidad en Múnich, Alemania. La cual permitía ver modelos a escala del automóvil BMW Mini, empleando la cámara y pantalla de computadores portátiles, concediendo múltiples vistas y diferentes ángulos del modelo. [1]

En los primeros años del siglo XXI, con el desarrollo tecnológico para dispositivos móviles, librerías de realidad aumentada, sensores y aplicativos como google maps, abrieron campo para el desarrollo de aplicaciones cada vez más sofisticadas y enfocadas a la vida cotidiana del hombre. Esto se ve claro cuando en 2014 google anuncia el desarrollo de sus gafas de realidad aumentada (Google Glass), con multifunciones como visualizaciones de videos, mapas, calendarios, etc. Así mismo en el 2016 Microsoft termina el desarrollo de su programa HoloLens. [2]

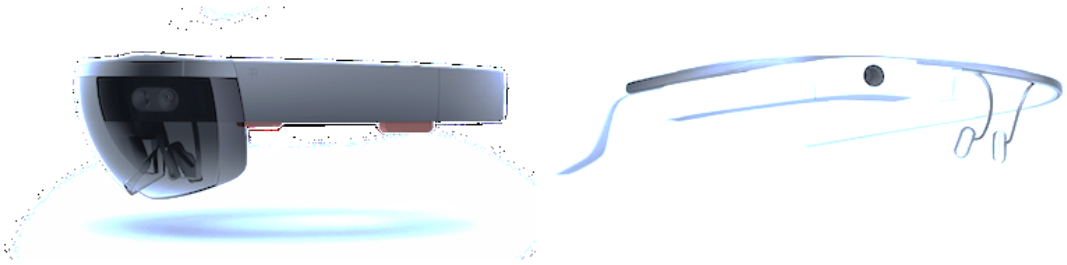


Ilustración 5: HoloLens, Google Glass, tomado de: http://www.mollejuo.com/2015/01/27/hololens-vs-google-glass/

1.2. ESTADO DEL ARTE

El desarrollo de plataformas y librerías de código abierto para realidad aumentada, han abierto una infinidad de posibilidades, para que aficionados o profesionales desarrollen aplicaciones de RA con múltiples contenidos, llegando a ámbitos educativos hasta industriales y de entretenimiento, aunque, el desconocimiento sobre esta tecnología no ha permitido que esta sea trabajada en gran escala, en la actualidad son pocos los proyectos de carácter institucional los que han sido desarrollados. En el presente se mencionaran algunos centrados en aplicaciones didácticas, culturales, entre otras.

1.2.1. Educación

En la educación, la implementación de la realidad aumenta conllevaría múltiples beneficios tanto para el educador como el estudiante, ya que ésta permite crear entornos de aprendizaje interactivos y enriquecedores, por medio de los cuales se asimilaría ágilmente conceptos y temáticas de asignaturas como biología, química, física, etc., que por medios tradicionales al estudiante se le dificultaría comprender. En la última década se han estado desarrollando sistemas RA, para temáticas pedagógicas específicas, como las siguientes:

- "El entorno de aprendizaje ubicuo de realidad aumentada y Tablet para estimular la compresión del espacio tridimensional", desarrollado en la universidad de la Laguna

España, cuyo propósito era comprobar la factibilidad de las nuevas tecnologías en la compresión del espacio tridimensional, como reemplazo de los modelos físicos, teniendo como muestra para este, un grupo de estudiantes universitarios, de secundaria y un grupo de selecto de profesores.
En ésta se desarrollaron una versión física y digital (RA), de seis modelos o figuras, con el propósito que los participantes interactuaran con estos y comprendieran la similitud entre la manipulación del objeto real y virtual. [4]

Ilustración 6: Pruebas al sistema de RA, tomado de: [4]

- Construct3D: Sistema de realidad aumentada desarrollado entre 1999 y 2008 por Hannes Kaufmann en colaboración con los pioneros del sistema Studierstube. Cuyo propósito era suministrar una herramienta educativa, para diferentes niveles de educación, secundaria hasta universitaria, con un enfoque en el área de las matemáticas y la geometría. Éste permite a través del HMD (Head Mounted Display) sistema de visualización montado en la cabeza, contemplar e interactuar con modelos 3D de diferentes figuras geométricas. [5]

Ilustración 7: Pruebas del sistema Construct3D, tomado de: [5]

1.2.2. Cultura

Igualmente que en la educación, la realidad aumentada ha sido empleada por instituciones tales como museos, galerías, o exposiciones, con el propósito de aumentar la curiosidad e interés de los visitantes de estos patrimonios cultures. La región que más ha tenido acercamiento con esta tecnología es la zona europea en donde instituciones como el museo de historia natural de Londres [6], de las américas de Madrid [7], emplearon sistemas de realidad aumentada con propósito de entretenimiento y conocimiento general de lo expuesto en ellos.

- Realidad aumentada para el museo de américa: Trabajo de grado de los estudiantes Marta Martínez, y David Hernández de la universidad de Complutense de Madrid, desarrollado para el museo de américa. El trabajo elaborado por los estudiantes consistía en un mapa mudo, que en función de la sección donde estuviera el visitante, mostraba información relevante o complementaria, así como figuras alusivas a las mismas.

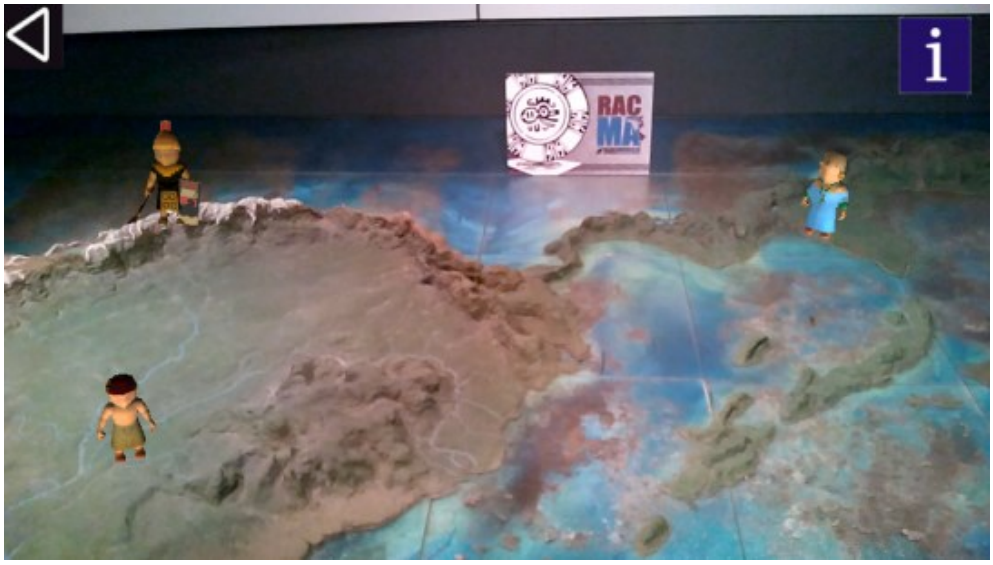


Ilustración 8: Séptima iteración del trabajo para el museo de américas, tomado de: [7]

1.2.3. Entretenimiento y Comercio

El área comercial, ventas y entretenimiento son los campos con mayor explotación en las características visuales u ópticas de productos u artículos, aquella que depende fundamentalmente del interés y gusto del público por los bienes u objetos vendidos, y en donde la RA, generaría un valor agregado a los servicios y/o productos ofrecidos. Como se mencionó anteriormente una de primeras las aplicaciones comerciales fue desarrollada en Munich, Alemania, con el propósito de promocionar el BMW Mini, así como esta se han ido

desarrollando trabajos de RA con propósitos similares para otras empresas, entre estos se encuentran:

- Diseño e implementación de aplicaciones móviles para la imagen de marca de una empresa: Trabajo de grado del estudiante Alex García Marín de la universidad de Politécnica de Catalunya, basado en un estudio sobre los productos más comercializados de la empresa Duscholux, cuyo propósito era generar curiosidad de estos productos a través aplicaciones de realidad aumentadas planteadas, evaluadas y aceptadas por dicha empresa, la cual se decantaron por: Dos juegos y dos aplicaciones.

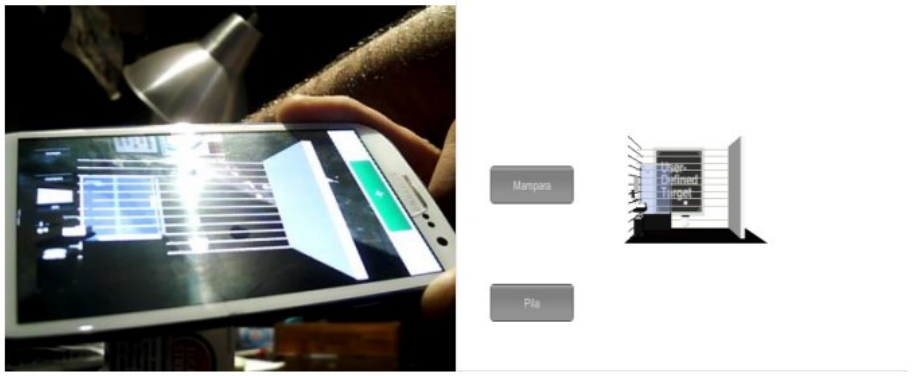


Ilustración 9: Aplicación para la empresa Duscholux, tomado de: [8]

- Probador virtual, realidad aumentada sin marcadores: Propuesta de trabajo de grado de Carlos Meca Lopez, de la universidad de Alicante, España, orientada a la venta de prendas de vestir para dama, y fundamentada en la rapidez. El trabajo propuesto por el estudiante plantea un aplicación que permitiría probarse diferentes prendas de vestir y desde diferentes ángulos, a un cliente, y ver de forma general como esta le quedaría.

Ilustración 10: Probador Virtual, tomado de: [9]

A parte de los proyectos anteriormente mencionados, existe gran variedad de proyectos para estos ámbitos u otros, que tienen un propósito específico según el área de aplicación. En educación la mayor parte de aplicaciones de RA se centran en libros interactivos, que muestras figuras alusivas a su contenido, y que pretenden suministrar información, por lo general a través de imágenes relacionados que faciliten la comprensión.

Ilustración 11: Libro interactivo, tomado de: http://museotelecomvlc.etsit.upv.es/telecochips/radiocomunicacion.html

La realidad aumentada a bordo de dos diferentes maneras la medicina, siendo la primera, aplicaciones enfocadas a la enseñanza, a través de las cuales se pueden observar de manera dinámica las diferentes partes del cuerpo humano, sistema óseo, cerebro, etc. Enriqueciendo y facilitando el aprendizaje de dichos temas.

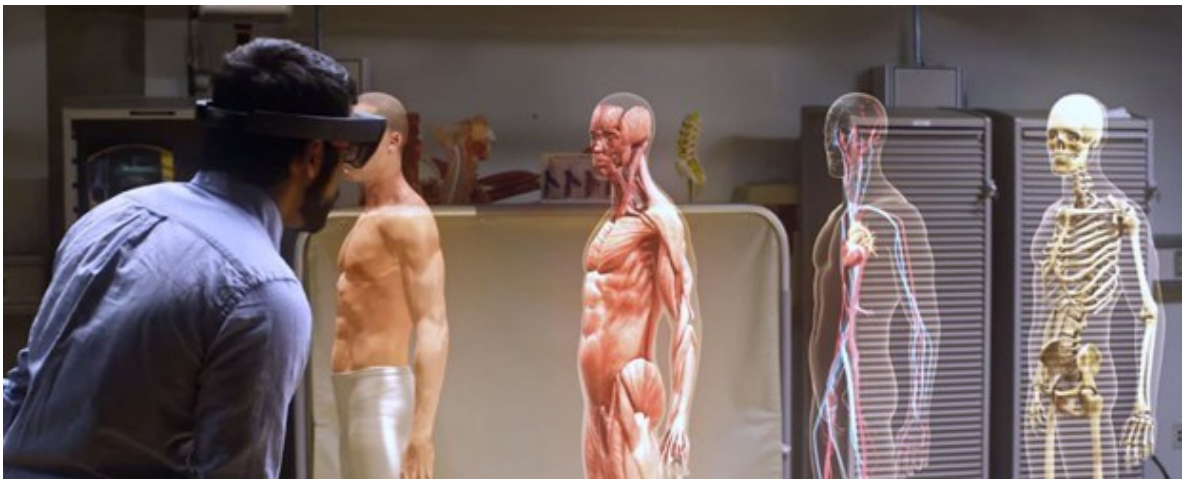


Ilustración 12: Realidad aumentada en medicina, tomado de: http://www.pangeareality.com/hololens-las-gafas-de-realidad-aumentada-que-tambien-serviran-para-la-educacion/

El segundo tipo de aplicaciones, se centran en sistemas de realidad aumentada aplicadas a entrenamiento y visualización de procedimientos quirúrgicos, modelación a través de resonancias magnéticas, tomografías, etc. Un proyecto pionero en esta área fue el desarrollado en la UNC Chapel Hill [10], en el cual se escaneaba el vientre de una mujer embarazada por medio de sensores ultrasónicos y se generada un modelo 3D del feto. [10]

Ilustración 13: Realidad Aumentada en medicina, tomado de: [10]

Otros proyectos interesantes en esta área, se basan en la implementación de sistemas de realidad aumentada empleadas para simulaciones, en la cual se puede observar la RA durante un simulacro de una biopsia de mama, el cual tiene como objetivo localizar y guiar la aguja hasta el tumor encontrado. [10]

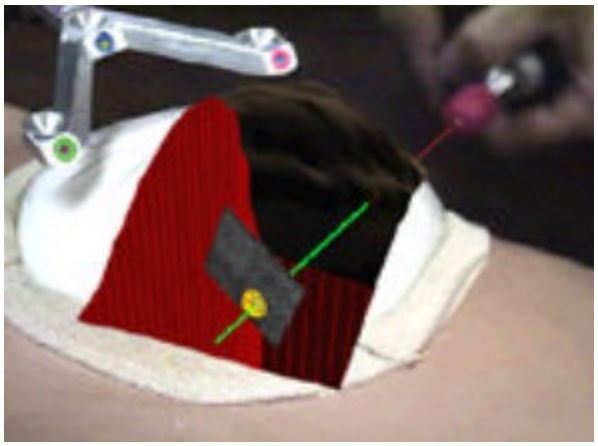


Ilustración 14: Realidad aumentada en medicina 2, tomada de: [10]

1.3. MARCO CONTEXTUAL

La realidad aumentada es una tecnología que aumenta la percepción del ser humano, empleado dispositivos electrónicos y métodos informáticos para superponer información real con la virtual (Generada por ordenador). En ésta, la información virtual sobrescribe parcial o totalmente un área del espacio en el medio visual (Monitor, pantalla, etc.).

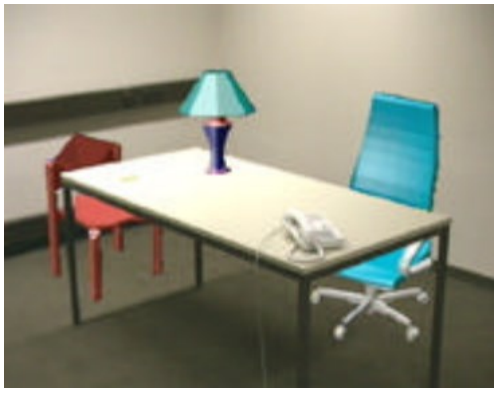


Ilustración 15: Mesa real, y lámpara y sillas virtuales, tomado de: [10]

1.3.1. Realidad aumentada

Una definición técnica aceptada de realidad aumentada es la propuesta por Azuma [10], el cual describe que, un sistema de realidad aumentada, es considerado como tal, cuando reúne las siguientes 3 características:

- Combinación del espacio real y virtual.
- Interacción en tiempo real.
- Contenido virtual en 3D. [10]

En su definición, Azuma no limita esta tecnología solo ha sistemas de visualización montados en la cabeza (HMD), mientras que el dispositivo o tecnología empleada cumpla con las 3 características esenciales anteriormente descritas. Algunas de estas alternativas son interfaces basadas en monitores, sistemas monoculares, gafas o cascos de realidad aumentada, etc.

Además describe la arquitectura básica con la que puede contar un sistema de realidad aumentada:

- Un dispositivo de visualización, que permite observar la combinación de la información real y virtual, pudiéndose ser un monitor, gafas de realidad aumentada o la pantalla de un dispositivo móvil (Smartphone, celular, table's).

- Un conjunto de sensores, que capturen la información necesaria para el funcionamiento óptimo del sistema, esta va desde cámaras, hasta sensores de posición como acelerómetros, GPS, etc.

- Un controlador, encargado de procesar la información recibida del conjunto sensor y del software encargado de combinar y calcular la posición de los objetos virtuales.

- Software especializado, en calcular la posición del objeto en el espacio real, y de sobrescribir la información recibida con la generada.

- Elemento activador o tipo de seguimiento; se definirá como tigger del sistema de realidad aumentada, éste marca el inicio del sistema de realidad aumentada y/o proporciona la posición espacial para los objetos virtuales sin se emplea un seguimiento por marcador.

Ilustración 16: Elementos básicos de un sistema de realidad aumentada, tomado de: [11]

En la Ilustración 16, se observa un ejemplo de arquitectura física o hardware necesario para el desarrollo de un sistema de realidad aumentada; en la actualidad un Smartphone o Tablet, engloba todos estos componentes, gracias al desarrollo de estos dispositivos en la última década, normalmente cuentan con una cámara en HD o VGA; acelerómetro y giroscopio, que simulan el seguidor de cabeza; la pantalla que reemplaza al monitor o display del HMD; el generador de escenas y dispositivo encargado de superponer la información real con la

virtual, en la actualidad es reemplazado con un aplicativo desarrollado por medio de librerías y software especializado en este campo, como ARToolKit o Vuforia.

1.3.2. Tipos de seguimientos

El tipo de seguimiento determina que objeto, característica o señal activara el sistema de realidad aumentada, y es uno de los campos de la misma con más proliferación en los últimos años debido al avance tecnológico de sensores, cámaras y tecnologías de visión artificial.

Las técnicas de seguimiento o "tracking" desarrolladas e implementadas en la actualidad son:

Basadas en sensores	WIFI, Bluetooth, UWB, ZibBee, RFID, Infrared, Ultrasound
Basadas en visión por computador	Basadas en marcadores Sin marcadores Marcadores naturales
Híbridas	Una combinación de diferentes métodos para mejorar la precisión. Por ejemplo, mezclar visión, localización GPS y orientación.

Tabla 1: Tipos de tracking, tomado de: [12]

El método basado en sensores y señales tales como wifi, RFID, etc., fue la técnica empleada en los primeros años de la realidad aumentada; pero estos conllevaban una serie de desventajas ligadas al tipo de sensor. Un ejemplo de esto, es que al emplear una red WIFI, este perdía precisión cuando era empleada en locaciones exteriores. El avance de las tecnologías de cómputo y técnicas de visión artificial, permitieron suplir las desventajas que este primer método conllevaba, mejorando en gran medida el método detección y seguimiento para sistemas de realidad aumentada. [12]

El método de seguimiento de visión por computador como fue denominado, aprovecha las técnicas de visión artificial, que permiten reconocer características puntuales en fotos y/o videos. En la actualidad este método se divide en:

- Identificación/localización basada en marcadores.

- Identificación/localización de características naturales de objetos o estructuras comunes (carteles, rótulos, etc.).

La identificación/locación basada en marcadores o "targets" por su nombre en inglés, fue la primera técnica en emplearse, aprovechando las características únicas de marcadores como: códigos de barras, QR, imágenes o bidi. Estos presentan alto contrasté en los elementos que los componen, permitiendo un reconocimiento fácil y ágil. En este método la cámara y sus características son esenciales, para agilizar y atenuar los efectos visuales que puede generar el entorno, así como para reconocer el marcado dentro de la secuencia de frames tomadas por ésta, que luego es procesada y sobrescrita por el software o SDK de realidad aumentada. [12]

Ilustración 17: Marcador vuforia, fuente: Autor

Entre las ventajas que ofrece éste método están, la posibilidad de calcular la distancia y ángulo desde el dispositivo de seguimiento, comúnmente la cámara hasta el marcador; además del poco poder de procesamiento requerido para la misma.

Ilustración 18: Marcador cilíndrico vuforia, tomado de: http://www.himix.lt/augmented-reality/augmented-reality-tracking-cylindrical-object/

Por otro lado la técnica de reconocimiento de características naturales de objetos o estructuras comunes, es el método con mayor tendencia de desarrollo y trabajos en sistemas de realidad aumentada, pero requiriendo de gran potencial de cálculo por parte del sistema de reconocimiento y visualización.

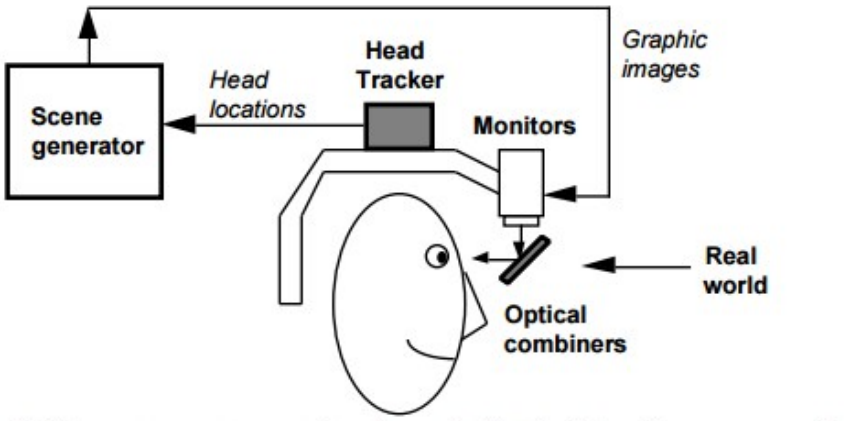
Éste método a su vez emplea dos categorías básicas, estructuras o modelos básicos tales como cubos, cilindros u otros objetos que dada sus bajas características como aristas o vértices que son fáciles de reconocer a través de visión artificial. La otra categoría, se fundamenta en

reconocer estructuras físicas que son altamente detectables, y que pueden ser modeladas de forma ágil por medio de algoritmos matemáticos que permitan reconocer las características desde diferentes distancias, algunos y niveles de iluminación. [12]

1.3.3. Tecnologías de visualización

Como se ha mencionado anteriormente, un elemento dispensable en todo sistema es medio de visualización de realidad aumentada, en la actualidad [10] define dos sistemas de visualización principales: Las pantallas óptica transparentes (Optical See-Through Display) y la pantalla de superposición de imágenes (Video Mixed Display).

- Pantalla óptica transparente: En estas pantallas, el usuario puede ver a través de sus ojos, pero teniendo delante de ellos, una pantalla transparente que muestra los elementos de la realidad virtual, superponiéndose la información real y virtual a través de este sistema de proyección óptico. Una característica de éste sistema de visualización es que tiende a reducir en gran medida la cantidad de iluminación que el usuario puede percibir.



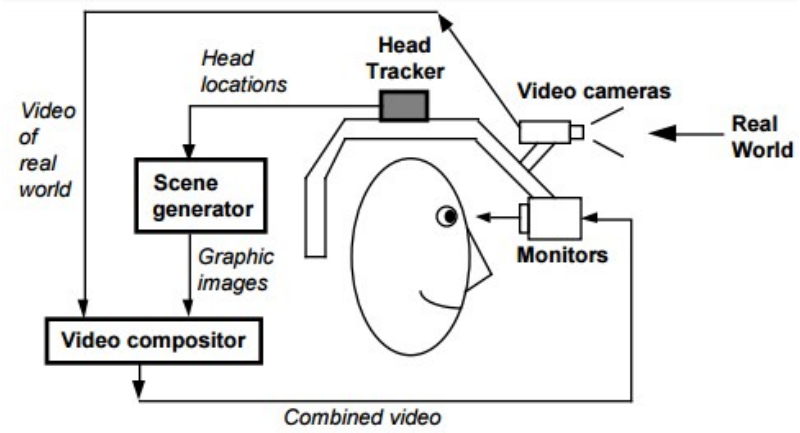
a) Diagrama conceptual pantalla óptica transparente

b) Pantallas ópticas transparentes, desarrollas por Hughes Electronics

Ilustración 19: Pantallas ópticas transparentes, tomado de: [10]

- Pantalla de superposición de imágenes: En estos dispositivos de visualización, el usuario no emplea sus ojos para percibir el mundo real, sino que lo ve a través de un

display o pantalla que muestra en tiempo real la imagen compuesta, detecta por la cámara y los elementos virtuales del sistema de realidad aumentada.



a) Diagrama conceptual pantalla de superposción de imagenes

b) Primeros prototipos de pantallas de superposción de imagenes

Ilustración 20: Pantalla de superposición de imágenes, tomado de: [10]

En su artículo Azuma [10], describe los sistemas de visualización en estas dos categorías, pero [12], a su vez describe 3 métodos de visualización que dependen del tipo de elemento o dispositivo empleado para esto.

1. Display de mano: En éste se emplean dispositivos móviles como Smartphone o Tablet, que incorporan en un ente toda la infraestructura mínima, requerida para una aplicación de realidad aumentada. En la actualidad estos disponen de un display o pantalla de tamaño considerado (2" o 3"), cámaras de alta resolución, sensores de posición, además de su alto poder de procesamiento. Éstas características hacen de los dispositivos móviles un excelente sistema para la RA, además de su portabilidad.

2. Display espacial: En esta opción, se emplean proyectores digitales para proyectar los elementos virtuales sobre los objetos físicos. La pantalla para este caso, no está sujeta

al usuario, permitiendo que la realidad aumentada sea visualizada por múltiples usuarios al tiempo.

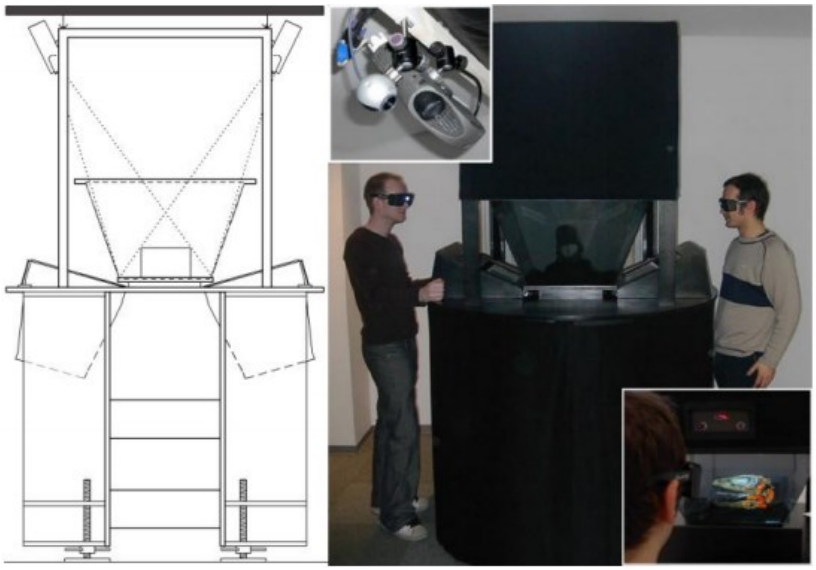


Ilustración 21: Display Espacial, tomado de: [12]

3. Dispositivo de visualización montado sobre la cabeza (HMD):

Los HMD son dispositivos de visualización que incorporan la pantalla sobre una estructura diseñada para montarse sobre la cabeza del usuario, por medio de la cual éste visualizara la realidad compuesta generada por el sistema de realidad aumentada, este a su vez se clasifica según lo descrito anteriormente, Azuma [10]. Estos dispositivos se pueden sub-dividir en monoculares y binoculares.

- Monocular: La imagen compuesta es proyecta o visualizada por un solo ojo.

- Binocular: Las imágenes compuestas son visualizadas por ambos ojos del usuario.

Comercialmente los dispositivos HMD, son desarrollados para RA, realidad virtual y realidad mixta, pudiendo ser estos cascos o gafas.

Ilustración 22: gafas de realidad aumentada, tomado de: http://www.francisortiz.com/2013/07/las-gafas-de-realidad-aumentada.html

1.3.4. Alcances y generalidades

El sistema de realidad aumentada propuesto en el presente trabajo, tiene como meta desarrollar un aplicativo de RA para ejecutarse en dispositivo móviles con plataforma ANDRIOD, enfocado a la robótica, en específico a lo referente con la cinemática de un brazo antropomórfico serial.

Para el desarrollo de este aplicativo se van a emplear los siguientes softwares:

- Modelado: Blender.
- SDK Realidad aumentada: Vuforia.
- Scripts Cinemática: Liclipse.
- Entorno de desarrollo: Unity 3D
- Entorno de programación: Monodelovep

La selección del software de modelado, SDK realidad aumentada, y entorno de desarrollo, será detallada más adelante; el software empleado para la cinemática y entorno de programación no tuvieron criterios para su selección aparte de que no requirieran licencia para su implementación.

Raplim es el brazo seleccionado, para el modelo 3D del aplicativo, éste fue diseñado en la universidad de Pamplona por los Ing.'s Alfredo Márquez y Javier Hernández, con el propósito de suplir necesidades básicas de personas con problemas motrices. A continuación se presenta las características generales de Raplim y las secuencias a realizar.

- Brazo Raplim:

Ilustración 23: Versión fisica Raplim, tomado de: Guía dibujar raplim, asignatura robótica- UP

- Dimensiones Raplim:

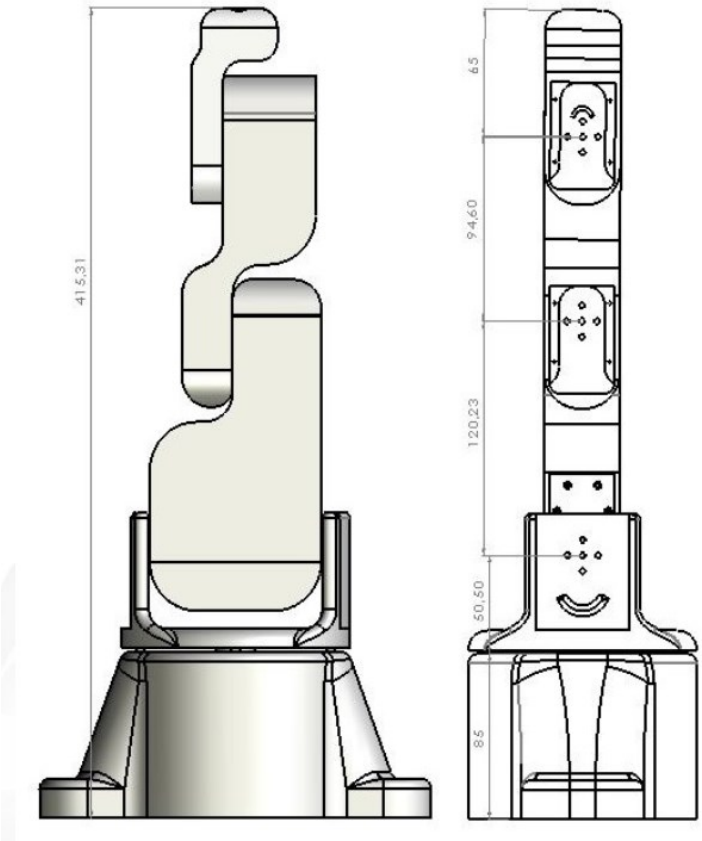


Ilustración 24: Dimensiones raplim: tomado de: Guía dibujar raplim, asignatura robótica- UP

Características generales del brazo:

- Tipo: Serial antropomórfico.
- Grados de libertad: 4.
- Complementos efector final: 3 (Cuchara, cepillo, vaso).
- Alcance máximo: 27.98 cm
- Altura: 41.5 cm.
- Articulaciones: Rotacionales.
- Número de eslabones: 4.

Éste realizara 3 secuencias básicas, cada una definida por los utensilios presentes en la plataforma: Cuchara, vaso y cepillo.

- Secuencia Cuchara: Esta primera secuencia empezara con posicionar el robot en la postura inicial (Correspondiente a posición inicial en la cinemática directa), una vez hecho esto, realizara los movimientos necesarios para que su efector final llega hasta el primer utensilio, la cuchara. Realizara un movimiento básico para incrustar la base de la misma en su efector final, para luego posicionarla encima del plato, posteriormente descenderá (Simulación de tomar liquido del plato) y ascenderá lentamente, después rotara hasta la posición contraria (Posición persona) a la base del utensilio empleado, una vez allí realizara unos pequeños movimientos correspondientes a entregar el contenido de la cuchara, para luego regresar a posición inicial la cuchara y el mismo.

- Secuencia Vaso: Las simulaciones en el presente son secuenciales, es decir, una vez terminada una, automáticamente entrara en escena la siguiente. Para esta secuencia, una vez el brazo antropomórfico regrese a su posición inicial, ejecutara los movimientos necesarios para llegar a una orientación paralela al mango de agarre del vaso, para luego generar un movimiento líneal (Incrustación vaso-efector final). Posteriormente elevara el vaso, y luego rotara 90° hasta la posición de la persona, esperara un segundo (Simulación tiempo de espera mientras la persona toma un poco del contenido del vaso), para luego realizar los movimientos requeridos para retornar el vaso a su origen y él retornara a su postura inicial.

- Secuencia Cepillo: Esta secuencia se ejecutara una vez termine la anterior, en esta secuencia, el brazo realizara los movimiento necesarios para que su efector final, este en la posición y orientación correcta, para agarrar el cepillo, primero posicionándose a pocos centímetros de la base de éste, y luego desplazándose lentamente hasta que el efector final y el complemento se unan. Una vez esto ha finalizado, el robot tomara una postura que al rotar la articulación 1 el efector y complemento no choquen contra la base de éste (complemento). Dicha articulación rotara 180 ° y posicionara el efector hasta la altura de la persona, para luego realizar pequeñas rotaciones correspondientes al movimiento de cepillado. Posteriormente retornara el cepillo hasta su posición inicial, para luego regresar a posición inicial.

Como se ha mencionado, estas simulaciones, son secuenciales, por consiguiente una vez se han realizado las 3 secuencias, se repetirán en el mismo orden.

1.3.5. Herramientas de desarrollo

En la actualidad existen en el mercado gran variedad de herramientas para el desarrollo de sistemas de realidad aumentada tanto de carácter privado (Licenciados) como de carácter público (Gratuitos).

Tecsmedia, una división de tecnología multimedia del instituto de Aragón, realizo una recopilación de los SDK actuales desarrollados para crear sistemas de realidad aumentada,

teniendo en cuenta características como el tipo de licencia, sistema de seguimiento, entre otros. En el presente se expondrán solo aquellos que cumplen los requisitos prestablecidos para el proyecto.

Producto	Tipo de Licencia	Plataforma	Características	Descripción
ALVAR	Libre, Comercial	Android, iOS, Windows, Flash	Seguimiento basado en marcadores y sin marcadores	Librería software para crear aplicaciones de RA y RV. Desarrollada por el Instituto Técnico de Investigación VTT (VTT Technical Research Centre of Finland).
ARLab	Libre, Comercial	Android, iOS	GPS, sensors (IMU Sensors), búsqueda visual	ARLab ofrece un amplio portfolio de soluciones tecnológicas para RA.
ARmedia	Libre, Comercial SDK	Android, iOS, Windows, Flash	Seguimiento basado en marcadores	La plataforma ARmedia es un framework de desarrollo estructurado y modula que incluye distintos módulos software. Este framework es independiente del motor de seguimiento en tiempo real y del motor de renderización.
Arpa	Libre, Comercial SDK	Android, iOS, Windows	Seguimiento basado en marcadores y sin marcadores, GPS, Sensores (IMU sensors), tracking facial y por infrarrojos, y renderización en tiempo real	Arpa Solutions es una compañía líder en el desarrollo de productos y aplicaciones de realidad aumentada a través de su plataforma propietaria ARPA AR.
ARToolkit	Open Source, Comercial SDK	Android, iOS, Linux, OSX, Windows	Seguimiento basado en marcadores y sin marcadores	ARToolkit es una plataforma de Realidad Aumentada que está disponible para múltiples sistemas operativos: iOS, Android, Linux, Windows y Mac OS.
ArUco	Open Source	Linux, OSX, Windows	Marcadores	Librería para aplicaciones de RA basada en OpenCV
Aurasma	Libre, Comercial SDK	Android, iOS	Solución sin marcadores que se basan en las características naturales de la imagen o el objeto (bordes, esquinas o texturas), una técnica conocida como NFT (Natural Feature Tracking)	Es una solución de HP que incorpora reconocimiento automático de imágenes
Designers ARToolkit (DART)	Libre pero con el código fuente cerrado	OSX, Windows	Marcadores, ContentAPI y TrackerInterface	DART es un conjunto de herramientas de software que permite diseñar e implementar aplicaciones y experiencias de RA de forma rápida.
Koozyt	Comercial SDK	Android, iOS	Marcadores	Fundada por miembros de los laboratorios de of Sony Computer Science que desarrollaron la tecnología "PlaceEngine" en Julio de 2007. Esta tecnología conecta el mundo real y el virtual poniendo el énfasis en el comportamiento humano.

Tabla 2: SDK de desarrollo, realidad aumentada parte 1, tomado de: [12]

Producto	Tipo de Licencia	Plataforma	Características	Descripción
OpenSpace3D	Open Source	Linux, Windows	Marcadores	OpenSpace3D es una plataforma de código abierto para desarrollar proyectos de RA y RV. Su objetivo es democratizar las aplicaciones 3D en tiempo real y proporcionar herramientas para creativos.
Rox Odometry SDK	Libre, Comercial SDK	Android, iOS, Linux, OSX, Windows	Marcadores, NaturalFeature	Permite construir aplicaciones identificando con la cámara del dispositivo objetos pre-grabados y obtener de forma exacta su posición y orientación relativa en tiempo real.
SSTT	Código fuente cerrado	Android, iOS, Windows Mobile, Linux, OSX, Windows	Marcadores, NaturalFeature	SSTT Bounce es un browser de RA que usa técnicas de tracking basadas en características naturales de la imagen.
Total Immersion	Libre, Comercial SDK	Android, iOS, Windows, Flash	Marcadores, NaturalFeature, Tracking facial	Total Immersion ofrece una plataforma comercial de RA que integra gráficos 3D interactivos en tiempo real dentro del flujo de video en vivo.
UART	Open Source	iOS, OSX, Windows	Marcadores	Unity AR Toolkit (UART) es un set de plugins para el motor Unity que permite a los usuarios desarrollar y desplegar aplicaciones de RA.
Vuforia	Libre, Comercial SDK	Android, iOS	Marcadores, NaturalFeature, VisualSearch	Vuforia es una plataforma de software que permite desarrollar aplicaciones de RA para móviles y tabletas.

Tabla 3: SDK de desarrollo, realidad aumentada parte 2, tomado de: [12]

Como se puede observar el número de librerías que permiten desarrollar sistemas de realidad aumentada es extenso, y cada uno ofrece distintas características que se pueden adaptar a los múltiples proyectos que puedan derivar de esta tecnología.

Como se ha mencionado anteriormente, la plataforma de desarrollo seleccionada es VUFORIA. Ésta librería de licencia libre permite desarrollar aplicaciones de realidad aumentada con marcadores, ofreciendo no solo un tipo, sino una gama de diferentes tipos de marcadores que permiten adecuar el aplicativo a desarrollar, además cuenta con una extensa comunidad de trabajo, así como un continuo desarrollo y compatibilidad con diferentes entornos que complementan las funcionalidades que éste presenta.

1.3.6. Funcionamiento básico

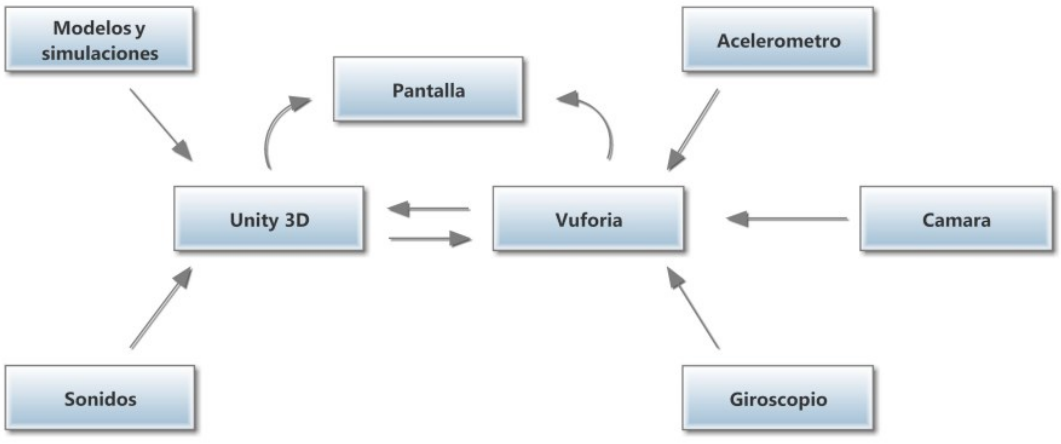


Ilustración 25. Diagrama Funcionamiento, Fuente: Autor

En la Ilustración 25, se observa la relación y funcionalidad de cada una de las partes involucradas en el desarrollo del presente proyecto. El aplicativo como se mencionó anteriormente, tendrá como propósito generar una realidad mixta (Compuesta) para el usuario, por medio de la cual éste podrá observar una rutina preestablecida de Raplim, mientras visualiza a través de una interfaz gráfica la variación articular (Cinemática inversa), y la posición del complemento o efector final(Cinemática directa).

Vuforia, es un SDK de realidad aumentada que compatible con múltiples entornos de desarrollo, dependiendo de la complejidad y necesidad del desarrollador, pudiéndose trabajar en base a algoritmos (AndroidStudio), o a través de entornos de desarrollo gráficos (Unity), con múltiples complementos y de fácil acceso.

Unity es un entorno de desarrollo enfocado a la creación de videojuegos y aplicaciones para múltiples plataformas, entre ellas, Android, Windows, Xbox, entre otras. Su extensa comunidad de desarrollo, acceso gratuito a gran parte del contenido o complementos de éste, y compatibilidad con VUFORIA, hacen que sea el entorno ideal para el desarrollo del presente proyecto. Su interfaz gráfica y multifuncionalidad, facilitan en gran medida, los objetos que se proponen.

Blender es un entorno enfocado al desarrollo de modelos y simulaciones en 2D y 3D, de acceso libre y al igual que Unity cuenta con una extensa comunidad, y complementos que agilizan el desarrollo de los proyectos a desarrollar en éste.

Ilustración 26: Unity - Vuforia, tomado de: http://www.javierlosadasanchez.com/como-desarrollar-apps-con-vuforia-y-unity/

En el presente, las simulaciones y modelado del Raplim se realizan en Blender, posteriormente se exportara a Unity el modelo y su simulación. En éste se desarrollara, el sistema de realidad aumentada, una vez sea cargado el modelo y el aplicativo VUFORIA. Unity será el encargado de posicionar en el espacio el modelo 3D, tomando como referencia la cámara de VUFORIA (Cámara de dispositivo de seguimiento), y de reproducir los sonidos alusivos a la simulación previamente cargada, las configuraciones iniciales, serán explicadas en capítulos posteriores.

1.3.7. Fundamentos de robótica

La robótica data de los años 20, cuando el escritor checo Karel Capek se refiere a está en su obra titúlala Rossum's Universal Robot. Ésta tiene como origen la palabra robota, definida como trabajo realizado de forma forzada. Pero, es Isaac Asimov el que acuña el término de robótica, en su obra "Yo robot", en la cual postula las 3 leyes fundamentales de la robótica y predice la revolución de ésta tecnología y sus impactos en la industria y la vida del hombre. [13]

En la actualidad el término robótica, no cuenta con una definición universal, por lo cual presenta una variedad de descripciones dadas por el punto de vista de diferentes autores; ésta puede ser definida según su área de aplicación u objetivo, según el término robot, etc.

Algunas de estas definiciones son:

- "La Robótica consiste en el diseño de sistemas. Actuadores de locomoción, manipuladores, sistemas de control, sensores, fuentes de energía, software de calidad--todos estos subsistemas tienen que ser diseñados para trabajar conjuntamente en la consecución de la tarea del robot"[Joseph L. Jones and Anita M. Flynn. Mobile robots: Inspirations to implementation. A K Peters Ltd, 1993].

- Ingenio mecánico controlado electrónicamente. [13]

- Máquina con comportamientos similares a una persona. [13]

1.3.7.1. Historia de la robótica – Breve resumen

Aunque la robótica es un área aceptada oficialmente hasta el siglo XX, ésta trasciende a varios siglos atrás, cuando el hombre empieza a desarrollar máquinas o autómatas para agilizar y automatizar, ciertos trabajos o tareas, que requerían gran cantidad de esfuerzo o mano de obra.

Los primeros registros de autómatas datan del año 1300 a.C, cuando Amenthotep, construye una estatua del rey de Etiopia, la cual emitía sonidos durante el alba ("Rayos del amanecer"). Así mismo grandes civilizaciones de la época como los egipcios o chinos, desarrollaron mecanismos avanzados como el reloj de agua y el ábaco respectivamente. Durante siglos estas civilizaciones fueron desarrollando tecnologías o mecanismos que proporcionan facilidad a ciertas áreas, realizaban movimientos autónomos, etc. Uno de estos fue la urraca de King-su Tse en el 500 a.C; en el año 400 Architas de Tarento considerado el precursor de la ingeniería mecánica, desarrolla una paloma que simulaba volar y/o rotar, gracias a un surtidor de agua; en el año 62 a.C Herón de Alejandría en su libro "Autómata" presenta unos esquemas de muñecos con la capacidad de moverse solos y repetitivamente. [14]

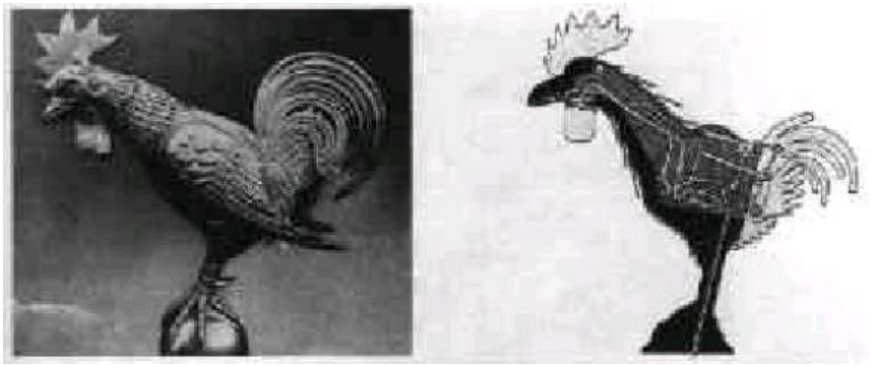


Ilustración 27: El gallo de Estrasburgo, tomado de: [15]

Durante la edad media se construyeron una gran variedad de mecanismos y autómatas, pero los más relevantes de esta época son: el gallo de Estrasburgo, el cual formaba parte del reloj de su catedral, el cual al dar la hora, éste movía el pico y las alas, este funciono desde 1352 hasta 1789; el hombre de hierro de Alberto Magno; el león mecánico de Leonardo Da Vinci presentado en honor al rey de Francia Luis XII, el cual presentaba el escudo de armas del estado cuando abría su pecho; la orquesta de Hans Bullmann del siglo XVI; el escarabajo de John Dee; entre otros. [14]

En el siglo XVIII, Joseph Jacquard crear la primera máquina textil programable mediante tarjetas perforadas; la revolución industrial fue un punto de inflexión para la robótica y máquinas automáticas, ya que este impulsa el desarrollo de equipos mecánicos tal como el torno de Babbitt. En este mismo siglo Jacques de Vauncansos construye mecanismos humanoides del tamaño de una persona; en 1805 Henri Maillardert diseña una muñeca mecánica capaz de realizar trazos, para dibujar o escribir. [16]

Ilustración 28: Muñeca dibujante de Henri Maillardert, tomado de: [15]

El desarrollo de la ingeniería en su gran cantidad de ramas como la mecánica, eléctrica, etc., así como el desarrollo de modelos de áreas fundamentales como la física, la matemática, entre otros y el avance tecnológico de las últimas décadas, desde los controladores analógicos hasta los equipos de cómputo, permitieron desarrollar máquinas autónomas o robots con gran precisión, rapidez y agilidad para el desarrollo de tareas cotidianas y detallistas como operaciones quirúrgicas y manipulación de materiales radioactivos.

Los ejemplares empleados para el desarrollo de robots u autómatas desde sus inicios, van desde gran variedad de animales hasta el propio hombre, siendo éste último, la matriz principal usada en el desarrollo de robots industriales.

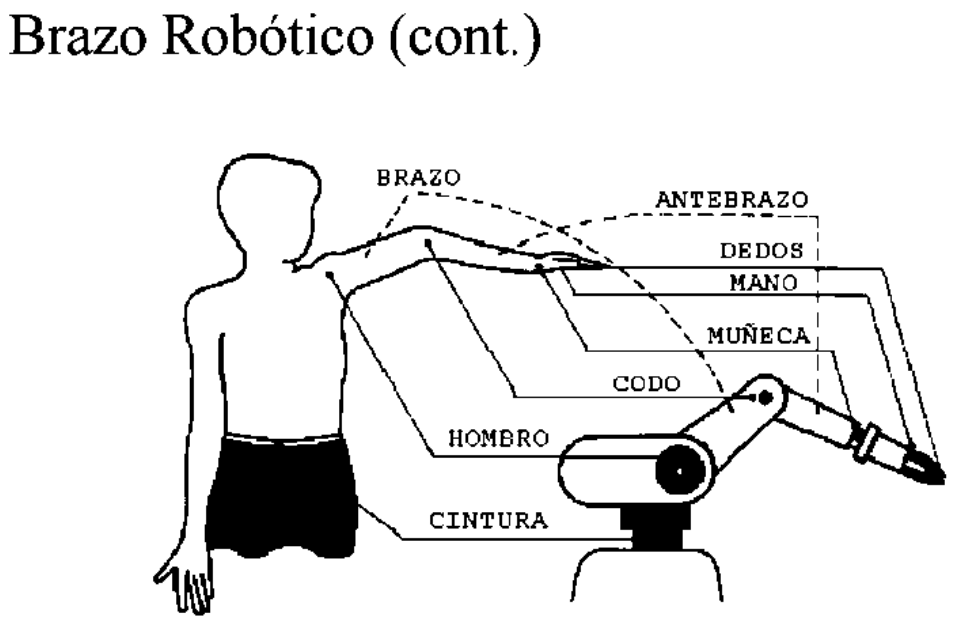


Ilustración 29: Anatomía modelo para el desarrollo de robots industriales, tomado de: [17]

Los robots industriales comúnmente conocidos como brazos robóticos fueron diseñados en similitud al brazo del hombre, la morfología de estos trata de considerar las mismas partes tales como el hombro, codo, etc; dando a estos la capacidad de manipular diferentes herramientas, u efectores finales para múltiples tareas o disciplinas.

Roselund y Pollard construyen el primer brazo robótico en 1938, para la empresa Devilviss, con el propósito de optimizar el proceso de pintado con spray, éste fue el punto fundamental para la inclusión de los robots en las líneas de producción. El avance en tecnologías como la inteligencia artificial, computadores digitales, el desarrollo de la automatización, creación de los transistores, entre otros, son hitos históricos importantes para el desarrollo de la robótica y de su incorporación en una gama extensa de áreas. [18]

Durante la segunda mitad del siglo XX, se desarrollan diversas tecnologías empleadas en diferentes campos entre ellas la robótica. En esta época se generaron diferentes estudios e investigaciones que dieron desarrollo de sensores (Ha Enst trabajo sobre sensores táctiles-1962), actuadores mecánicos (Goertz y Bergsland, investigan sobre actuadores mecánicos

activados remotamente en los años 50), hidráulicos (Desarrollo del tríodo permite la implementación de actuadores eléctricos e hidráulicos en los años 50), entre otras. [18]

Ilustración 30: Robot Da Vinci, tomado de: http://www.efesalud.com/noticias/una-decada-robot-da-vinci-espana-mejor-tecnologia-urologia/

Con el paso de los años, la robótica ha sido introducida en áreas donde la precisión, rapidez, eficacia, entre otras aptitudes son necesarias, para ejecutar labores y/o tareas críticas, tales como operaciones quirúrgicas (1985 robot PUMA es empleado para introducir una aguja en el cerebro), manipulación de materiales radioactivos (En 1958 se empiezan a desarrollar los sistemas amo-esclavo para la manipulación de éstos materiales) , líneas de producción (En 1947 DS halder afiliado a la compañía automovilística FORD emplea la primera línea de producir con brazos robóticos). [18] En la actualidad, el avance tecnológico de la robótica ha avanzado a tal punto, que se han realizado tele-operaciones donde el operador y el robot están en diferentes continentes (2001 – se realiza la primera operación transatlántica con los cirujanos y paciente en Nueva York y Francia respectivamente, empleando el robot quirúrgico Da Vinci).

1.3.7.2. Tipos de robots

Los robots no tienen una clasificación específica como se ha mencionado anteriormente, pero generalmente se pueden clasificar según su área de aplicación o estructura física. De forma general estos se clasifican en:

- Humanoides:

Ésta clase de robots están basados en la morfología humana, y tratan de imitar a grandes rasgos sus características, movimientos, expresiones, entre otras. Aunque en la actualidad no cuentan con un campo especifico de aplicación, son empleados en áreas como la medicina, milicia, etc.

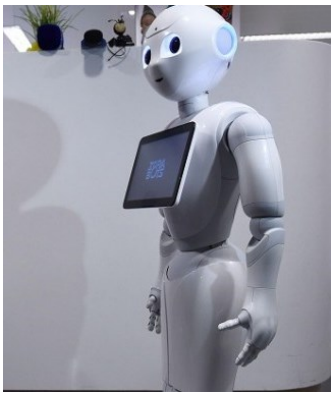


Ilustración 31: Robot Humanoide, tomado de: http://notihoy.com/los-robots-humanoides-que-trabajaran-en-hospitales-belgas/

- Moviles:

Se denominan robots móviles a aquéllos en los que su desplazamiento es debido a ruedas, orugas o patas. Son sistemas móviles que constan de todos los componentes necesarios para realizar acciones pre-programadas, para ello disponen con una serie de sensores que activan o ejecutan un movimiento o acción en función de lo sensado. Son empleados para búsqueda y rescate, transporte de mercancía (Amazon), etc. [13]

Ilustración 32: Robot móvil, tomado de: http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/movil.htm

- Robots industriales:

Los robots industriales son manipuladores reprogramables, con la capacidad de realizar múltiples trabajos o tareas con trayectorias pre-programadas o tele-operados. Son empleados en líneas de producción, o en trabajos altamente peligrosos para el ser humano. Su anatomía se basa en el brazo humano. Pueden contar con una gran variedad de actuadores finales como pinzas, fresas, taladros, entre otros, dependiendo del tipo de trabajo a realizar.

Ilustración 33: Robot industrial Kuka, tomado de: https://www.active8robots.com/robots/kuka-robots/

- Médicos:

Son aquellos empleados en las diferentes ramas de la medicina, pueden conllevar diferentes morfologías, dependiendo del área donde se implemente. Algunos ejemplos de esto son: En operaciones quirúrgicas se usan brazos robóticos tele-operados como el Da Vinci. En Japón se emplea la serie Riba, robot humanoide como asistente para tareas cotidianas de los pacientes. Veebot es un robot desarrollado para extraer sangre.

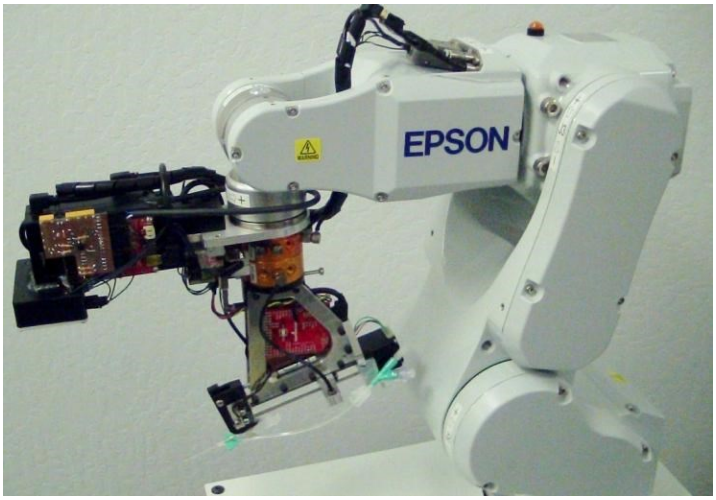


Ilustración 34: Robot Veebot, tomado de: http://www.ubergizmo.com/2013/07/veebot-draws-your-blood/

- Tele-operados:

Esta rama de los robots es desarrollado con el propósito de suplir al hombre con una herramienta para trabajos críticos y peligrosos, como se ha mencionado anteriormente, se emplean en trabajos donde se requiere alta precisión y seguridad. Los usos más comunes son la medicina, la milicia y la manipulación de materiales radioactivos. Su funcionamiento se basa en mecánicos o sistemas hápticos, que reproducen con exactitud en el robot los movimientos o acciones realizadas por el operador. En estos, no existen rutinas pre-programas, son operados manualmente. [13]

Ilustración 35: Robots tele-operados, tomado de: http://www.tiposde.org/general/460-tipos-de-robots/

1.3.7.3. Estructura de los robots industriales

Los robots industriales también llamados manipuladores robóticos, son máquinas reprogramables y multifunciones enfocados a la manipulación y desplazamiento de objetos, pudiendo ser estos operados autónomamente (Controladores lógicos) o por humanos (tele-operados). Estos comúnmente se constituyen por una serie de elementos rígidos (Eslabones) conectados mediante juntas o articulaciones, que permiten el movimiento relativo entre eslabones adyacentes (Consecutivos). [19]

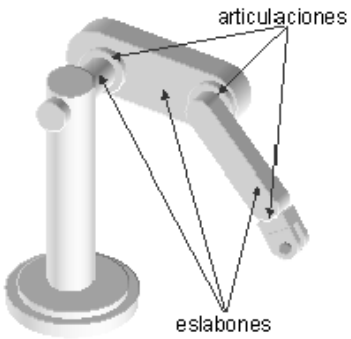


Ilustración 36: Componentes brazo robótico, tomado de: [19]

- Eslabón: “Un eslabón es un cuerpo rígido que posee como mínimo dos nodos, siendo estos los puntos de unión con otros eslabones”. [20]

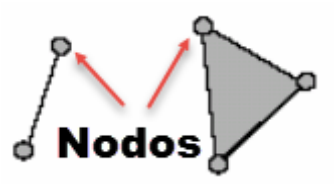


Ilustración 37: Eslabones, tomado de: [20]

Las articulaciones pueden presentar dos formas generales:

- Lineales: Se denomina lineal a la articulación que permite un movimiento traslacional o desplazamiento entre dos eslabones adyacentes, también son conocidas como articulaciones prismáticas.

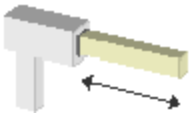


Ilustración 38: Articulación prismática, tomado de: [19]

- Rotacionales: Una articulación rotacional es aquella que permite que un eslabón gire sobre el eje solidario del eslabón anterior.

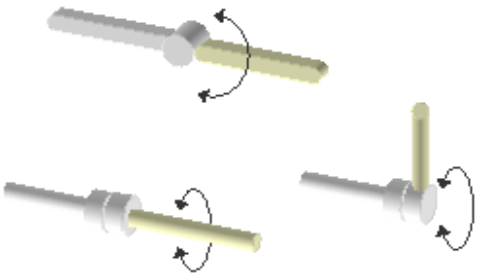


Ilustración 39: Articulaciones rotacionales, tomado de: [19]

Al conjunto de eslabones y articulaciones que permiten la ejecución de un movimiento entre los eslabones unidos se le denomina cadena cinemática. [20] Esta puede ser abierta o cerrada dependiendo dé, si el último eslabón esta enlazado con un eslabón fijo o no. En una cadena cinemática abierta el primer eslabón es aquel que es fijado a una superficie o soporte, y los eslabones posteriores son articulados a través de juntas ya sean prismáticas o rotacionales. En el extremo del último eslabón de una cadena cinemática abierta, se instalada o posiciona el elemento terminal o actuador. Aquel que determina el tipo de trabajo a realizar, este puede ser una pinza, una herramienta (fresa, broca, etc.). [19]

Los movimientos realizables por un brazo robótico, depende fundamentalmente de dos factores: Los tipos de juntas que lo componen, y el número de grados de libertad que es presenta. Se denomina grado de libertad o G.D.L al número de parámetros que deben ser definidos para establecer en cualquier instante la posición y rotación espacial de un objeto. [20]

El control sobre un robot se realizar a través de dos estudios: El estudio cinemático y el estudio dinámico, en los cuales se estudian el movimiento de éste teniendo en cuenta o no las fuerzas que lo producen, respectivamente. En el presente se empleara el estudio cinemático de Raplim, para el desarrollo de las simulaciones incluidas en el sistema de RA.

1.3.7.4. Configuraciones morfológicas comunes

- Cartesiano: Es aquel en donde el efector final es posicionado únicamente mediante articulaciones lineales. Algunas de las aplicaciones donde son empleados son: pintado, ensamblado, manipulación de máquinas herramientas y soldadura por arco. [19]

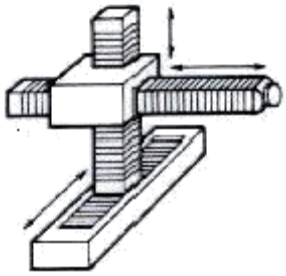


Ilustración 40: Robot cartesiano, tomado de: [19]

- Cilíndrico: Robot con base rotacional y articulaciones lineales para el resto de movimientos (Altura y radio). Empleado para operaciones de ensamblaje, soldadura por punto, fundición a presión, etc. [19]

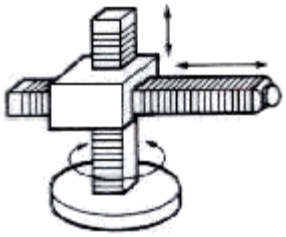


Ilustración 41: Robot cilíndrico, tomado de: [19]

- Polar: Un robot industrial polar presenta 3 grados de libertad, dos articulaciones rotacionales y una prismática. Sus ejes forman un sistema de coordenadas polares, comúnmente es usado para soldadura por punto, gas o arco; fundición a presión, manipulación de máquinas herramientas, etc. [19]

Ilustración 42: Robot polar, tomado de: [19]

- Articulado: Los robots articulados son aquellos con un mínimo de 3 articulaciones rotacionales. Presentan una morfología muy similar al brazo humano. Este puede realizar un movimiento rotacional y varios movimientos angulares. Es empleado en operaciones de ensamblaje, soldadura a gas y arco, pintado en spray, etc. [19]

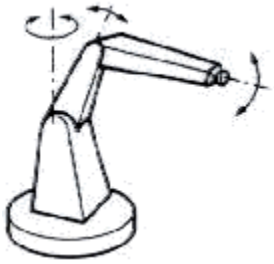


Ilustración 43: Robot articulado, tomado de: [19]

- Escara: La configuración Scara es un brazo robótico de 4 GDL caracterizado por sus altas velocidades de funcionamiento, repetitividad, y capacidad de carga. Usado en trabajos pick and place, ensamblado, clasificación, etc. [19]

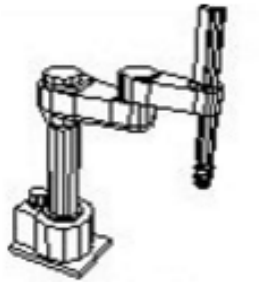


Ilustración 44: Robot Scara, tomado de: [19]

- Paralelo: Son robots de cadena cinemática cerrada, donde sus articulaciones prismáticas o rotacionales son concurrentes (paralelas). Presentan altas prestaciones de velocidad y precisión. Usados en simuladores de vuelo, clasificación, etc. [19]

Ilustración 45: Robot paralelo, tomado de: [19]

1.3.7.5. Cinemática

La cinemática es el área de la física que estudia el movimiento de un objeto, sin tener en cuenta las fuerzas o factores externos que lo producen. En el estudio o diseño de un brazo robótico, se emplean dos métodos o técnicas que permiten caracterizar el movimiento y posicionamiento espacial, en función de los parámetros conocidos del mismo. Estas son:

- Cinemática directa: La cinemática directa, es el estudio que determina la posición y orientación espacial del efector final de un brazo robótico, teniendo un sistema de referencia y conociendo los movimientos articulares de los eslabones del brazo.

  El grado de dificulta del problema cinemático directo está en función del número de grados con los que cuente el robot. Éste es resuelto a través de metodologías matemáticas que pueden ser sistemáticas o no, como lo son el método geométrico y el algoritmo de denavit hartenberg respectivamente, siendo éste último el empleado en el presente trabajo.

  El algoritmo de denavit es método sistemático, que en base a serie de rotaciones y traslaciones pre-establecidas permiten determinar la cinemática directa de cualquier robot de n eslabones.

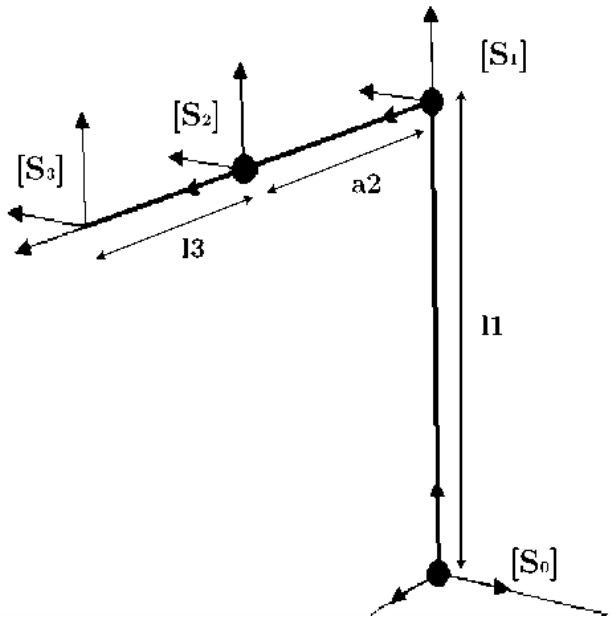


Ilustración 46: Aplicación de algoritmo dh, tomado de: [21]

Éste consiste en la siguiente serie de pasos:

1. Escoger una posición inicial para cada una de las articulaciones y/o eslabones del robot, siendo está, la más fácil de estudiar.

2. Numerar de 1 hasta n cada eslabón móvil de la cadena cinemática, y con 0 al eslabón fijo de ésta.

3. Numerar de 1 hasta n-1 las articulaciones o juntas presentes en la cadena cinemática.

4. Ubicar los ejes de rotación de cada articulación, si es rotativa será su propio eje de giro, si es prismática será el eje sobre el que se produce el desplazamiento.

5. Situar los ejes Zi sobre el eje de la articulación i+1, empezando de 0 hasta n-1.

6. Ubicar el origen del sistema de referencia (S0), sobre cualquier punto del eje Z0. Los ejes X0 y Y0, se ubicaran de modo que se forme un sistema de dextrógiro.

7. Ubicar los sistemas solidarios a cada eslabón (Si) para i de 1 a n-1, en la intersección entre la línea normal común a Zi-1 y Zi con el eje de rotación de cada articulación i. Si ambos ejes se intersectan se situaría sobre dicho punto; si son paralelos se sitúa sobre la articulación i +1.

8. Situar Xi sobre la línea normal común a Zi-1 y Zi.

9. Situar Yi de tal forma que se forme un sistema de dextrógiro con Xi y Zi.

10. Situar el sistema Sn sobre el efector final del robot, de modo que Zn y Zn-1 coincidan en dirección, y Xn sea normal a éstos. Yn, se situara de tal manera que se forme un sistema de dextrógiro.

11. Obtener Øi como el ángulo que debe girara Xi-1 entorno a Zi-1, para que éste sea paralelo a Xi.

12. Obtener di como la distancia a lo largo de Zi-1, para que Xi-1 y Xi, coincidan en un mismo punto.

13. Obtener ai como la distancia a los largo de Xi, para que los orígenes de estos sistemas Si-1 y Si, coincidan.

14. Obtener αi como el ángulo que debe rotar Zi-1 entorno a Xi, para que los sistemas Si-1 y Si coincidan totalmente.

15. Obtener las matrices de transformación homogénea i-1A1.

$$ {}^{i-1}A_i = \begin{pmatrix} \cos\theta_i & -\cos\alpha_i \cdot \sin\theta_i & \sin\alpha_i \cdot \sin\theta_i & a_i \cdot \cos\theta_i \\ \sin\theta_i & \cos\alpha_i \cdot \cos\theta_i & -\sin\alpha_i \cdot \cos\theta_i & a_i \cdot \sin\theta_i \\ 0 & \sin\alpha_i & \cos\alpha_i & d_i \\ 0 & 0 & 0 & 1 \end{pmatrix} $$

16. Obtener la matriz de transformación homogénea que relaciona el sistema base (S0), con el extremo del robot.

    T=0Ai - > 0A1 * 1A2 * …. * n-1An

- Cinemática inversa: La cinemática inversa, es aquella herramienta por medio de la cual, se determinan las rotaciones y desplazamientos articulares, necesarios para llegar a un punto conocido en el espacio. Esta al igual que la cinemática directa presenta una variedad de métodos que permiten desarrollar el problema cinemático.

  Entre estos encontramos métodos geométricos, numéricos iterativos, los cuales aunque permiten en la mayoría de casos dar solución, pueden conllevar a soluciones cerradas y rápidas, o a soluciones múltiples y lentas, respectivamente. [17]

Comúnmente se emplean los métodos geométricos para resolver el problema cinemático inverso, pero, puede requerir de modificaciones dependiendo de la dificulta del mismo, siendo necesario aplicar metodologías como el desacoplamiento cinemático, para robots con 4 o más grados de libertad.

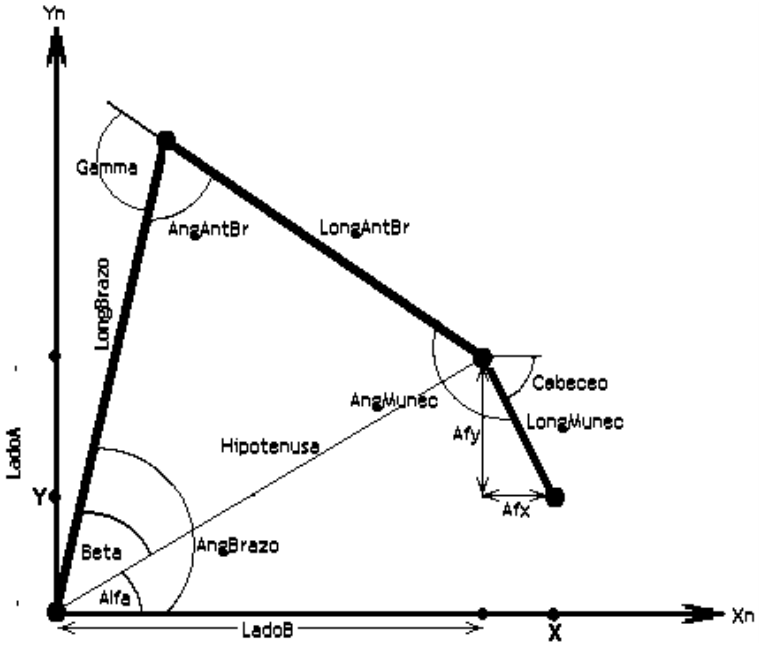


Ilustración 47: Aplicación cinemática inversa, tomado de: [22]

Éste método se fundamenta en la variación angular de los triángulos rectángulos o no que se pueden formar entre los eslabones o cadenas de un brazo robótico. Para determinar los ángulos de estos se emplean principalmente relaciones trigonométricas y leyes geométricas, tales como:

- Ley del coseno.
- Ley del seno.
- ArcoSeno
- ArcoCos
- Arcotangente

En base a estas leyes y funciones matemáticas, se establecen los valores angulares.

En esta metodología también es necesario establecer, si los movimientos se realizarán con codo arriba o codo abajo, ya que esto afectara el estudio o desarrollo de la cinemática inversa.

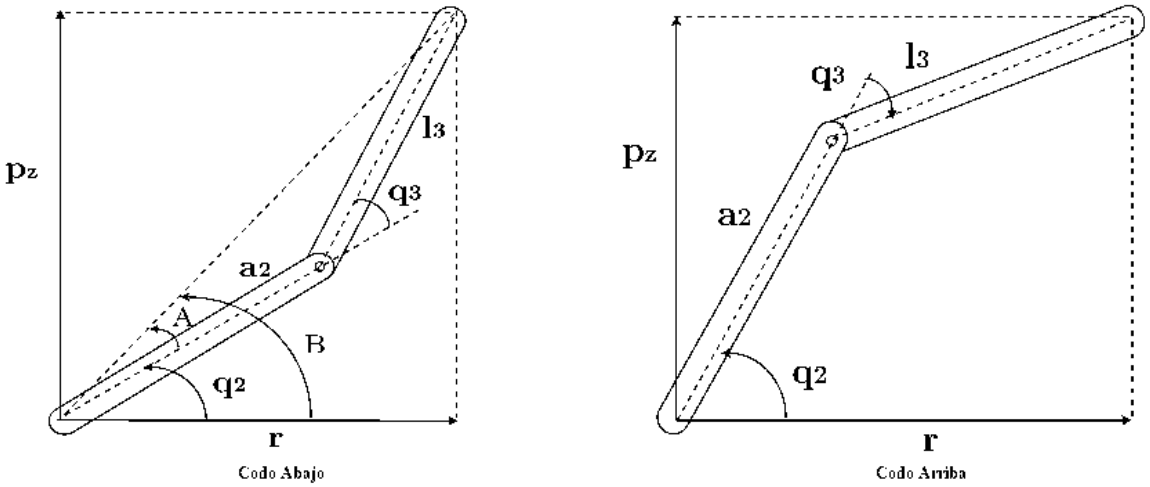


Ilustración 48: Elección de posición de codo del brazo robótico, tomado de: [21]

En el presente, la cinemática inversa se desarrollara totalmente por medio de métodos geométricos, y se programara en lenguaje python.

2. BLENDER

Blender es un software multiplaforma, enfocado a modelado y animación 3D. Desarrollado en 1998 por Ton Roosendaalm, con código abierto y distribución gratuita.

En 2003 nace Blender Foundation como medio para liberar este software bajo la licencia GNU GPL, después de que NAN (Not at number), empresa con la que inicio este proyecto cayera en bancarrota en 2002. En la actualidad Blender cuenta con soporta para Windows, Mac, Linux, Solaris, FreeBSD e IRIX, y ofrece una gran gama de herramientas y complementos para el desarrollo de contenidos bidimensionales o tridimensionales.

Ilustración 49: Logo Blender.

Blender cuenta asimismo con diferentes motores gráficos para modelar y simular texturas, materiales, iluminación, sombras, fuerzas, entre otros; que permiten realizar trabajos profesionales bajo licencia gratuita. Además cuenta con una herramienta de post-procesado para la exportación en diferentes formatos del contenido creado. Esto y muchas otras características hacen de Blender uno de los software más apetecidos y empleados a nivel mundial por profesionales, y entusiastas de la animación 3D.

2.1. CARACTERÍSTICAS

- Multiplataforma: Windows, Linux, Mac, etc., libre y gratuito.
- Desarrollado para ejecutarse en ordenadores de bajo poder de procesamiento, y requerimiento mínimo de espacio en el disco duro.
- Software integrado para desarrollar contenido 3D, desde sus inicios: Permite realizar, los modelos desde cero, animación, efectos, agregación de sonidos, programación de scripts, entre otras cosas, a través de sus múltiples herramientas de desarrollo.
- Motor 3D para el desarrollo de videojuegos, con detecciones de colisiones, físicas, partículas, etc.
- Sincronización video-audio.

- Animación y modelado a través de código o scripts desarrollados en lenguaje Phyton, a través de la consola integrada.

2.2. MOTORES GRÁFICOS

Los motores gráficos son los espacios de trabajo de un software, diseñado para crear y desarrollar videos juegos. Estos han de ofrecer funcionalidades básicas al desarrollador. Proporcionando comúnmente un motor de renderizado ("Render") para 2D y 3D, un motor para la detección de colisiones, fuerzas, físicas, etc.; y una respuesta a éstas: animaciones, sonidos, entre otras cosas. [23]

Blender cuenta con un motor gráfico, enfocado al desarrollo interactivo de animaciones o videojuegos. La estructura de su motor, se fundamenta en lo que ellos denominan ladrillos de lógica, cuyo propósito es proporcionar un interfaz amigable que permita el desarrollo de aplicaciones interactivas, sin necesidad de tener conocimientos profundos en lenguajes de programación. Esta estructura cuenta con 3 tipos: Lógica, sensores; controladores y actuadores. [24]

Actualmente Blender cuenta con 2 motores gráficos de pre-renderizado (Renderizado durante el modelado) y uno de tiempo real (Post-procesado). Los 2 motores de pre-renderizado son enfocados al realismo: Blender Render o interno, motor por defecto al iniciar la aplicación, y cycles, su motor más reciente basado en fuentes de iluminación.

2.2.1. Interno

Motor por defecto de Blender, empleado comúnmente en la creación de escenas básicas, en las cuales se requiere alta velocidad al renderizar. Emplear este motor requiere de gran conocimiento por parte del usuario para lograr efectos fotorealistas en su trabajo.

Ilustración 50: Motores de pre-renderizado, fuente: Autor

2.2.2. Cycles

"Motor introducido en la versión 2.61, basado en el trazado de rayos de luz, y más en concreto en la técnica conocida como BRDF". Empleado para obtener trabajos más profesionales y con efectos realistas, en base a materiales y texturas que el usuario puede crear en Blender. Este motor tiende a ser más lento durante el renderizado pero requiere menor esfuerzo por parte del desarrollador para obtener efectos fotorealistas.

2.3. INSTALACIÓN

La instalación del software Blender para la plataforma Windows, no requiere de conocimientos previos en códigos DOS (CMD), para realizar dicho proceso, ya que en la actualidad cuenta con un ejecutable o instalador que facilite en gran medida este procedimiento.

La instalación cuenta de los siguientes pasos:

1. Descarga el software de la página oficial de Blender:

   http://www.blender.org/download/get-blender/

2. Ejecutar el instalador de Blender como administrador en Windows.

3. Seguir el proceso de instalación de instalación del software. Especificando en él:

- El directorio de instalación.
- Los componentes a instalar.
- El directorio de información de la aplicación.

Para plataforma Windows, Blender cuenta con versión portable, la cual se puede ejecutar sin realizar el proceso de instalación en el ordenador. Esta versión igualmente puede ser descargada a través de la página oficial de Blender.

2.4. ENTORNO DE TRABAJO

El entorno de trabajo de Blender por defecto es default, el cual cuenta con todas las herramientas para el diseño de modelos 3D y propiedades de esto. Pero este puede cambiar dependiendo del trabajo a realizar. En la Ilustración 51 podemos 4 de las posibles alternativas para el entorno de trabajo: Default empleada en el diseño de elementos o estructuras; Animation enfocado en la producción de simulaciones o animaciones; Scripting orientado al diseño y animaciones a través de código (programación); y por último UV editing, especializado en la alteración de mapas UV o texturas por cada cara presente en los sólidos o modelos.

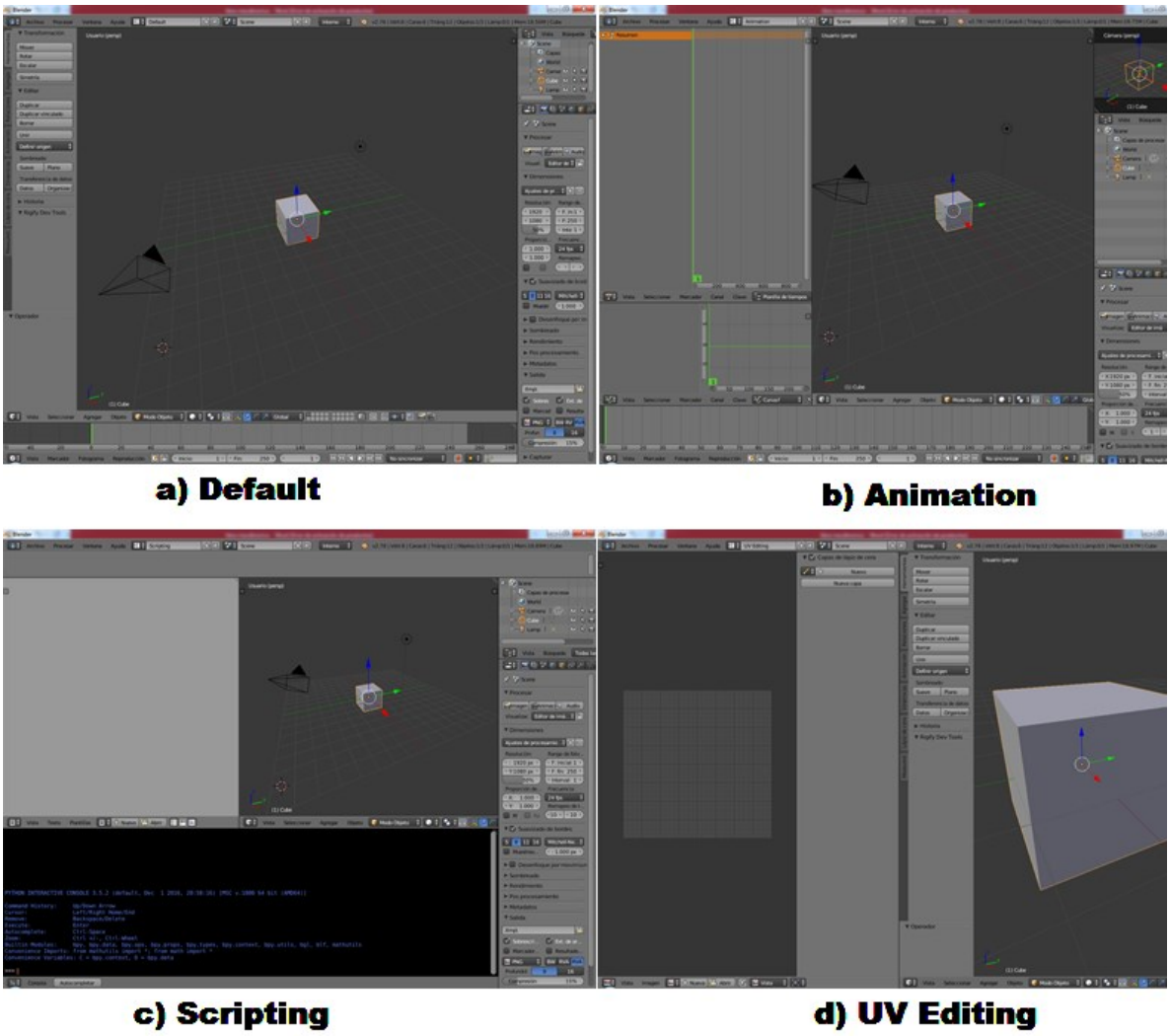


Ilustración 51: Entornos de trabajo Blender, Fuente: Autor

Para seleccionar el tipo de entorno de trabajo, al iniciar el programa, en la parte superior derecha, se encuentra un menú desplegable que permite elegir entre las posibles opciones.

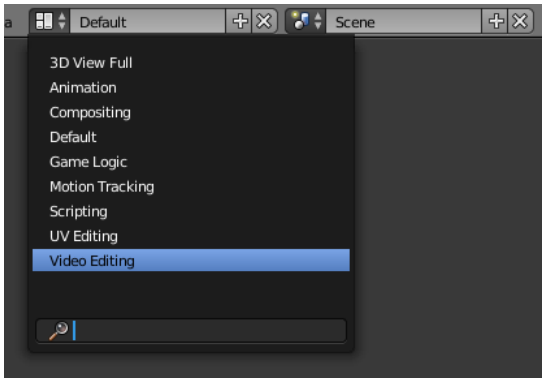


Ilustración 52: Selección de entorno de trabajo, Fuente: Autor

En el presente trabajo, se empleó la interfaz Default y Animation, así como variaciones de ésta, para diseño y tratamiento de los eslabones de Raplim y la simulación correspondiente.

2.4.1. Entorno de trabajo: Default

1. Vista 3D o espacio de trabajo: Esta zona dentro del entorno, hace referencia a la sección donde se diseñaran, crearan o posicionaran los objetos y/o estructuras a modelar o animar.

2. Barra de herramientas: Esta sección se encuentra todas las herramientas de trabajo, para diseñar, insertar, desplazar, etc., los elementos básicos del espacio de trabajo de Blender.

3. Barra de propiedades: Esta barra indica las propiedades del objeto seleccionado dentro del espacio de trabajo de Blender. En esta se contratara información sobre su posición respecto al origen o a su objeto padre, sus rotaciones, escala, dimensiones, etc.

4. Barras de menú: En esta barra, se encuentran todas las funciones de guardado del proyecto, exportaciones, tipo de entorno, motor gráfico, entre otras. En general son las encargadas de modificar la apariencia y las preferencias dentro de Blender.

5. Barra de tiempos o animación: Esta barra es empleada al realizar animaciones básicas. Su funcionamiento se basa en la secuencia de frames, en donde cada frame contendrá o no, una posición, escala u otra característica definida diferente respecto a las encontradas en otros frames.

6. Lista: Esta sección del entorno, muestra los objetos dentro del espacio de trabajo, la relación (padre-hijo) entre ellos. Por medio de ésta se pueden ocultar objetos, renombrarlos, etc.

7. Propiedades de área u objeto: Esta sección permite modificar propiedades referentes al tipo de objeto seleccionado (Cubo, cámara, textura, etc.), además a través de esta se aplicaran modificadores (Reflexión, biselar, etc.) sobre los objetos, que corresponden a métodos específicos para facilitar el trabajo sobre ellos. Así mismo por medio de esta podemos configurar la calidad, fps formato, etc., para la exportación y post-procesado.

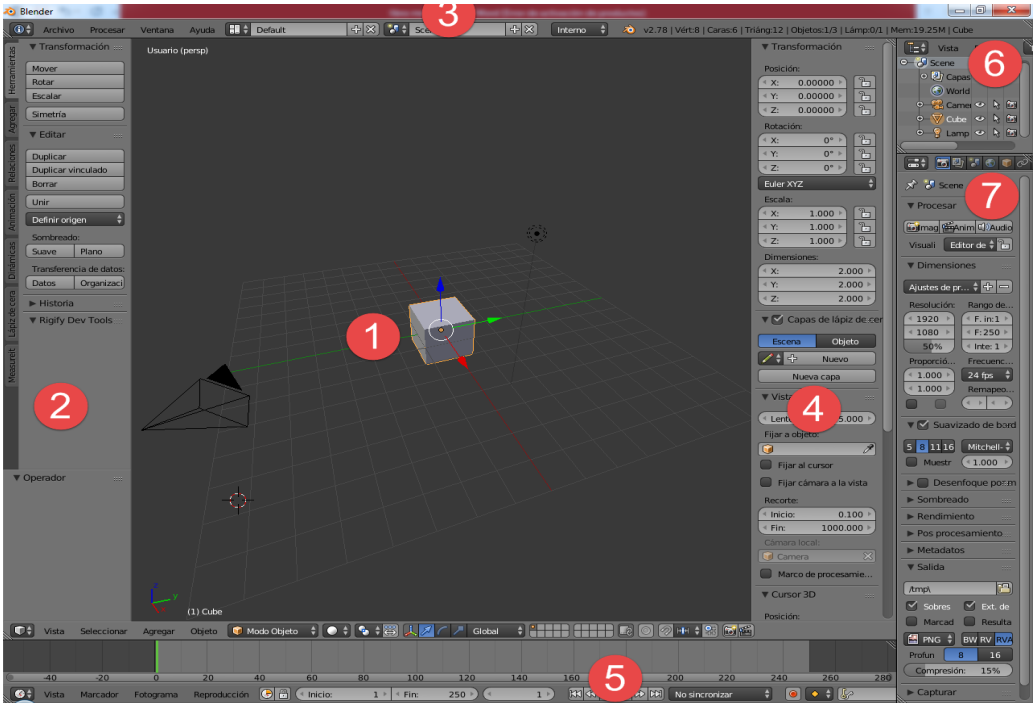


Ilustración 53: Entorno de trabajo Blender, Fuente: Autor

3. UNITY – VUFORIA

Unity es un entorno de desarrollo de videojuegos multiplataforma (Windows, Linux, Android, Xbox, etc.), así como en plataforma de escritorio y principales navegadores de internet. Este entorno integrado permite la creación de videojuegos desde cero, permitiendo manipular y crear en su totalidad los elementos que lo van a integrar. Éste a su vez integra un IDE para programar en lenguaje JavaScript, C# o Boo, para facilitar la animación, dinámica e interacción entre los elementos empleados en él. [23]

El sistema de funcionamiento de Unity se basa en escenas, cada una corresponde a un nivel dentro del videojuego. El contenido de estas va desde escenarios, pasando por interfaces hasta videos ilustres a la introducción y e interfaz inicial.

La unidad básica de Unity es el asset, correspondiente a cualquier elemento empleado en el proyecto. Estos pueden ser desarrollados en Unity o por herramientas externas, para luego ser exportados al área de trabajo. Unity maneja gran diversidad de formatos para archivos como modelos 3D (Maya, Blender, etc.), archivos de sonido, imágenes y videos. [25]

Aparte de estos tipos de assets Unity emplea algunos de carácter especial que solo pueden ser creados en su entorno a través de los múltiples herramientas con las que. [25]

En el presente Vuforia es el asset fundamental del proyecto, ya que provee las características y elementos necesarios para desarrollar sistemas de realidad aumentada en Unity, reemplazando a assets específicos del entorno.

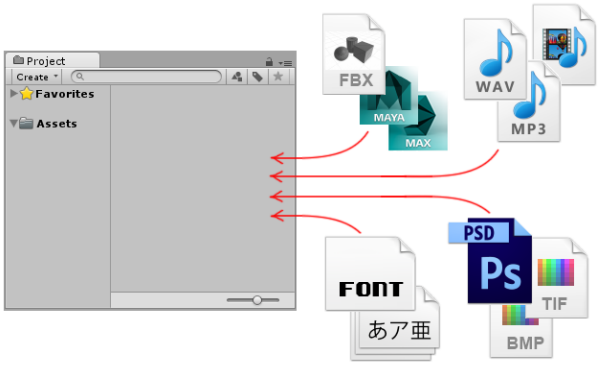


Ilustración 54: Assets Unity, tomado de: [25]

3.1. INSTALACIÓN UNITY

El instalador de Unity, desde la versión 5.0 cuenta con un asistente que permite descargar cada uno de los paquetes necesarios para el funcionamiento de la aplicación. Para dicho instalable se debe acceder a: http://unity3d.com/download. Cuando es necesario descargar una versión alterna a la última disponible, ingresar a: https://unity3d.com/es/get-unity/download/archive

En el presente la versión de Unity seleccionada es: Unity 5.3.0f4 (32-bit), versión compatible con Vuforia 5.5.9.

Una vez ingresado a esta página web, seleccionar la versión gratuita de esta plataforma de desarrollo, descargado el asistente, ejecutarlo, y seleccionar los componentes necesarios para el proyecto.

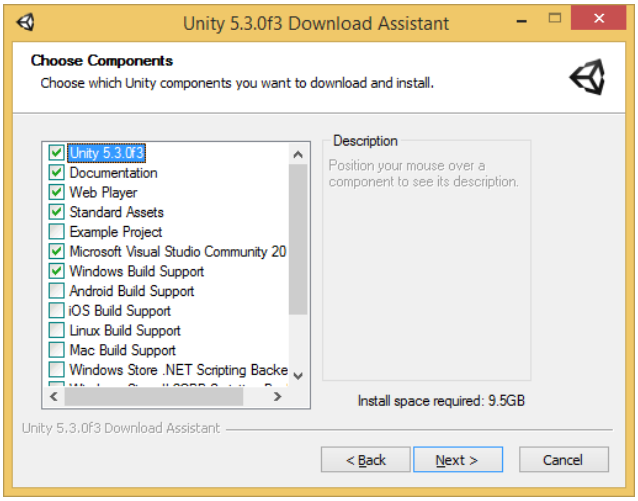


Ilustración 55: Asistente de instalación, tomado de: https://docs.unity3d.com/es/current/Manual/InstallingUnity.html

Los componentes a fundamentales a seleccionar son:

- Unity, instalador principal de la aplicación.

- Web player, corresponde al complemento de Unity para ejecutar el proyecto dentro del propio entorno de desarrollo.

- Standard Assets: Paquete básico de assets para el funcionamiento del aplicativo.

El resto de componentes disponibles en la lista, corresponden a los archivos necesarios para exportar el proyecto a diferentes plataformas, para el presente se seleccionaron:

- Android Build Support.

- Windows Build Support.

El proceso de instalación una vez, seleccionados los elementos requeridos, corresponde al mismo procedimiento para instalar una aplicación en el sistema operativo Windows.

3.2. CONFIGURACIONES UNITY

La exportación correcta para aplicaciones Android dentro de unity, requiere de la instalación o asignación del directorio de las siguientes aplicaciones:

- SDK Tools: Paquete de desarrollo necesario para compilar o ejecutar aplicaciones desarrolladas en Android en cual versión de éste. Puede ser descargado desde la página oficial de Android Studio: https://developer.android.com/studio/index.html

- JDK: Paquete de herramientas necesarias para desarrollar aplicaciones en lenguaje java, del cual deriva la plataforma Android. Éste se descarga de la página oficial de oracle: http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Una vez terminado el proceso de instalación de estos paquetes de desarrollo, se debe asignar su directorio de instalación dentro de Unity. Para configurar estos directorios, se debe dirigir a: Edit -> Preferencias -> Herramientas externas.

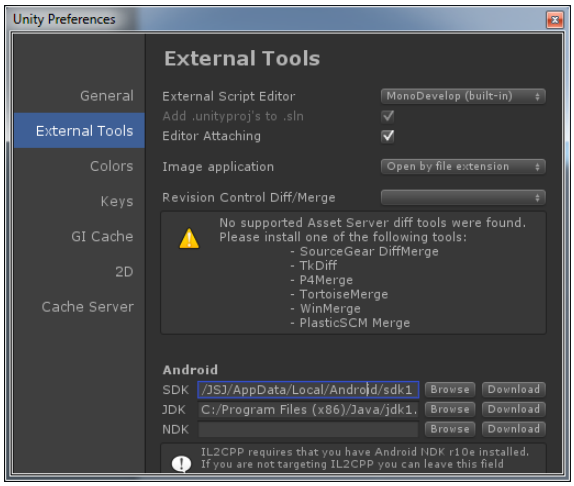


Ilustración 56: Preferencias Unity, Fuente: Autor

3.3. VUFORIA

Vuforia es una de las múltiples herramientas o SDK's de la actualidad, empleadas para diseñar aplicaciones o sistemas de realidad aumentada. Ésta utiliza la pantalla del dispositivo donde se ejecuta la aplicación como lente o pantalla de visualización, por medio del cual se observa la realidad mixta genera por el SDK. Los marcadores son el tipo de seguimiento empleado por Vuforia para la activación de la RA.

3.3.1. Marcadores Vuforia

Vuforia ofrece una gama de marcadores como medio de activación o tipo de seguimiento para el sistema de RA. Estos pueden asumir diferentes formas o tipos desarrollados a partir de archivos previamente procesados por el generador de targets de Vuforia.

- FrameMarker: Este tipo marcador emplea una de las 512 posibles opciones que ofrece Vuforia, estas son caracterizadas por un código binario codificado a lo largo de sus bordes.

- ImageTarget: Éste es una variación del FrameMarker, que permite al usuario o desarrollador empelar cualquier tipo de imagen como marcador para el sistema de realidad aumentada. Éste marcador debe ser generado por medio del aplicativo puesto por Vuforia en su página oficial. Entre mayor detalle presente ésta, mejor actuara el algoritmo de reconocimiento.

- VuMarks: Este tipo de marcado trabaja similar a un código QR. Este puede asumir diferentes formas y colores, así como adaptarse a gran variedad de entornos. La mayor ventaja que presenta de los VuMarks, es que el algoritmo de reconocimiento los encontrara fácilmente.

- CylinderTarget, CubeTarget: Estos targets permiten emplear paquetes de productos como marcadores. Empleando la forma primitiva (Cilindro o cubo) como medio de identificación, e interactuando sobre el mismo objeto de reconocimiento.

- ObjectTarget: Éste método de identificación, permite emplear figuras o modelos complejos como celulares, computadores o juguetes como marcadores para la activación del sistema AR. Es necesario obtener la data o información del modelo empleando la herramienta: Vuforia Object Scanner Tool, para poder generar el marcado por medio del generador de marcadores Vuforia. [26]

3.3.2. Creación de marcadores Vuforia

Vuforia dispone en su página oficial de un aplicativo para generar targets compatibles con su sistema, empleando imágenes, cubos, cilindros u objetos completos. Para el presente se expondrá como generar un marcador tipo imagen a través de su aplicativo. Para esto es necesario crear e iniciar una cuenta en Vuforia.

Con la cuenta creada e iniciada, acceder a la pestaña Develop, posteriormente a Target Manager. Una vez localizados en esta sección añadir una base de datos (Add Database), asignar un nombre y escoger tipo: Device, luego crear.

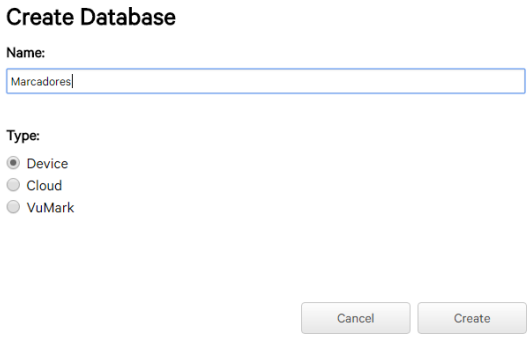


Ilustración 57. Creando base de datos para marcadores, Fuente: Autor

Una vez creada la base de datos, acceder a ésta, y dar en añadir marcador (Add target).

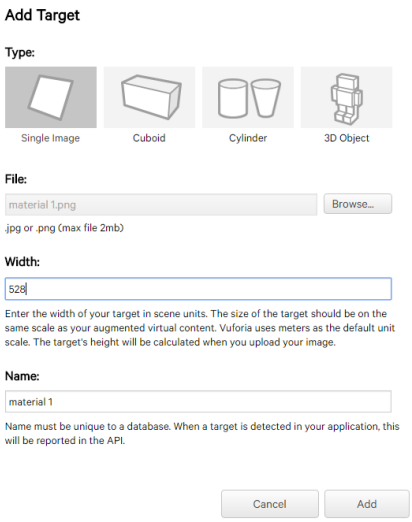


Ilustración 58: Creando y añadiendo marcador a la base de datos, Fuente: Autor

Una vez realizado esto el aplicativo pedirá características y datos correspondientes al tipo de marcado a crear. Este tipo requiere el archivo o imagen a emplear, su ancho y nombre a asignar. Luego dar en añadir.

Una vez creado el marcador, descargar la base de datos, para cargarla en Unity.

3.3.3. Creando licencia para el aplicativo

Uno de los principales requisitos que exige el prefab ARcamera para funcionar, es la asignación de una key o llave valida generada en la página oficial de Vuforia.

Al igual que para la creación de marcadores es necesario tener una cuenta en el portal de Vuforia, e iniciada sesión ir a la pestaña Develop, luego a License manager. Allí se añadirá una nueva licencia (Add License key). El aplicativo preguntara que tipo de proyecto se está desarrollando: No comercial (Development), comercial (Consumer) y empresarial (Enterprise). Seleccionar No comercial (Development).

Una vez seleccionado el tipo de proyecto asignar un nombre a la licencia (Comúnmente el nombre de la aplicación), y continuar con el proceso. Una vez termine esto, se presentara la información relevante sobre la licencia a generar.

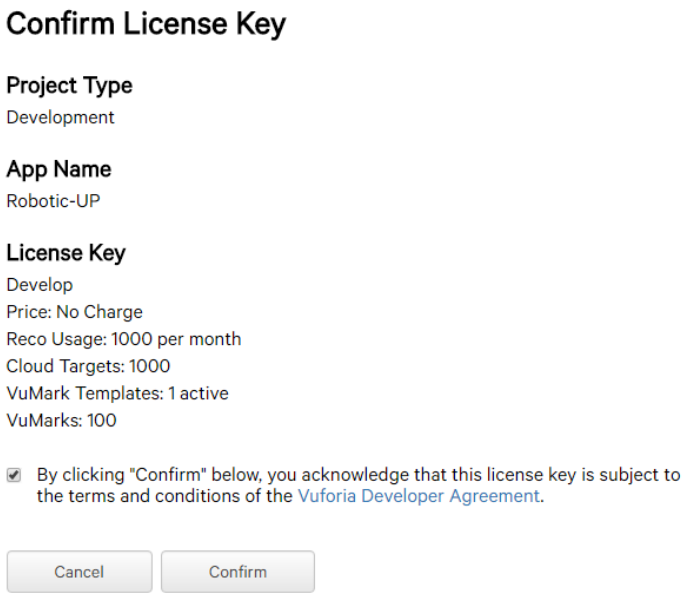


Ilustración 59: Creando licencia, Fuente: Autor

El proceso termina con la confirmación de las características de la licencia generada. Posteriormente regresar License Manager, seleccionar la licencia creada y copiar el contenido, dentro de esta.

3.3.4. Instalación Vuforia-Unity

Vuforia es un aplicativo que puede ser empleado en múltiples plataformas de desarrollo, Unity, es una de las posibles opciones para trabajar con este SDK. Para emplearlo, este debe cargarse como un Asset de Unity en el proyecto que se esté desarrollado.

El aplicativo de Vuforia compatible con Unity puede ser descargo en el siguiente enlace:

https://developer.vuforia.com/downloads/sdk

En el presente se emplea la versión 5.5.9 de Vuforia, la cual es compatible con la versión empleada de Unity.

Una vez descargo el aplicativo, ir a Unity -> Assets -> Import Package -> Custom Package

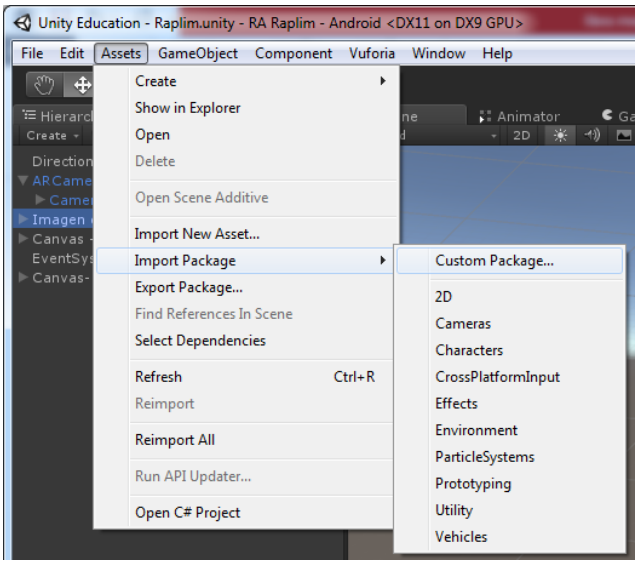


Ilustración 60: Importar Vuforia a Unity, Fuente: Autor

Y seleccionar el directorio donde este almacenado este paquete, posteriormente dar en importar en el cuadro que emerge una vez son cargados todos los elementos de Vuforia al proyecto.

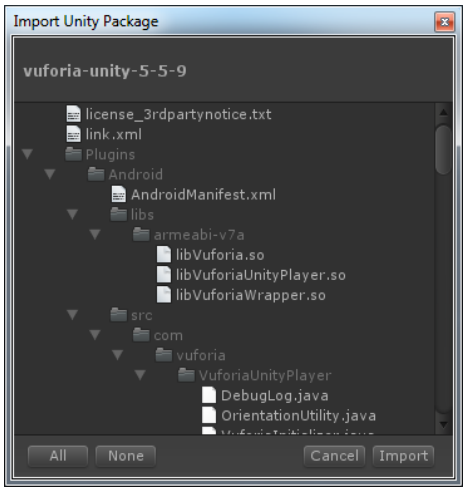


Ilustración 61: Importación de Vuforia, Fuente: Autor

Una vez realizado esto, Unity cargara todos los prefabs (Plantilla para instanciar objetos dentro de Unity), entre ellos los dos a emplear en el presente proyecto: ARcamera e ImageTarget.

4. DESARROLLO

El desarrollo del proyecto de Ra aumentada fue un proceso iterativo que se fundamento en 3 características esenciales:

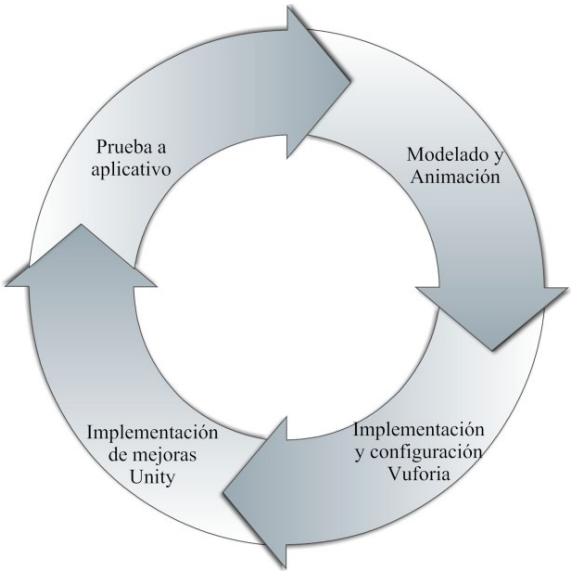


Ilustración 62: Diagrama de desarrollo del proyecto. Fuente: Autor

4.1. CINEMÁTICA

4.1.1. Cinemática directa

Para el problema cinemático directo empleado el algoritmo de denavit, se emplea la siguiente posición inicial.

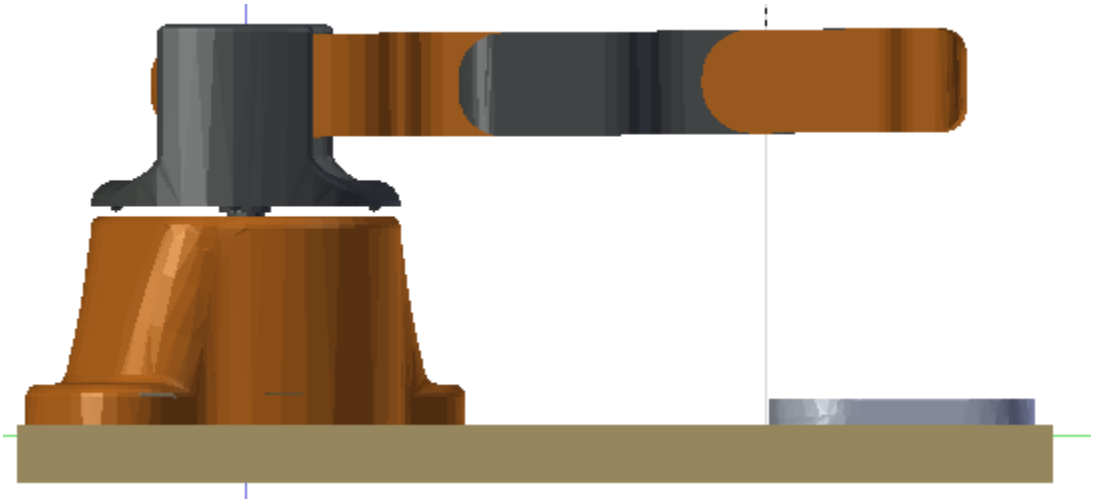


Ilustración 63: Posición inicial Raplim, cinemática directa, fuente: Autor

El estructura cinemática de Raplim es:

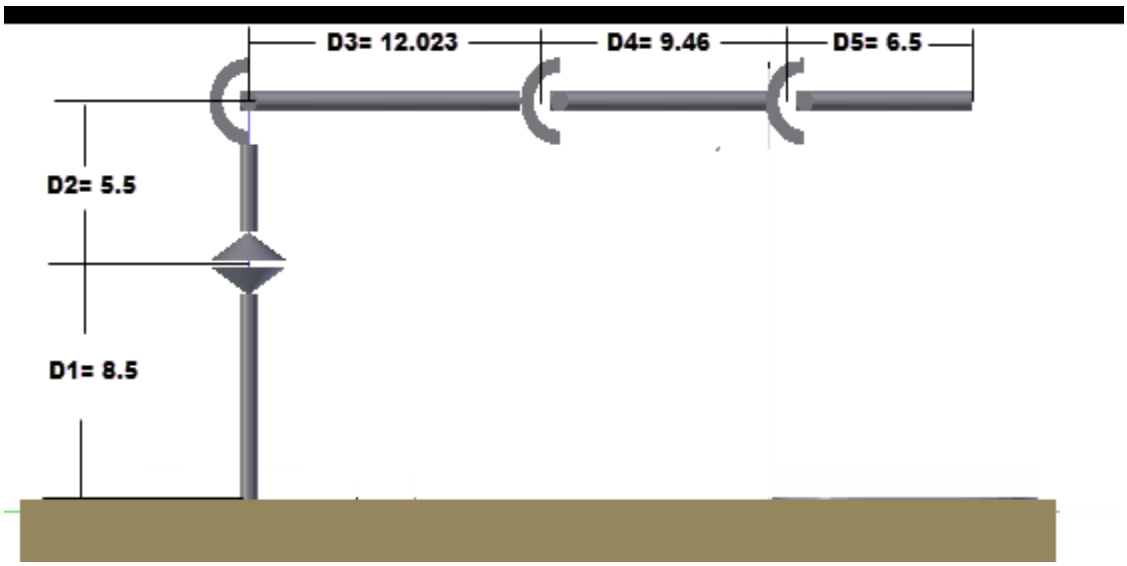


Ilustración 64: Estructura cinemática Raplim, fuente: Autor

Enumerando articulaciones y eslabones:

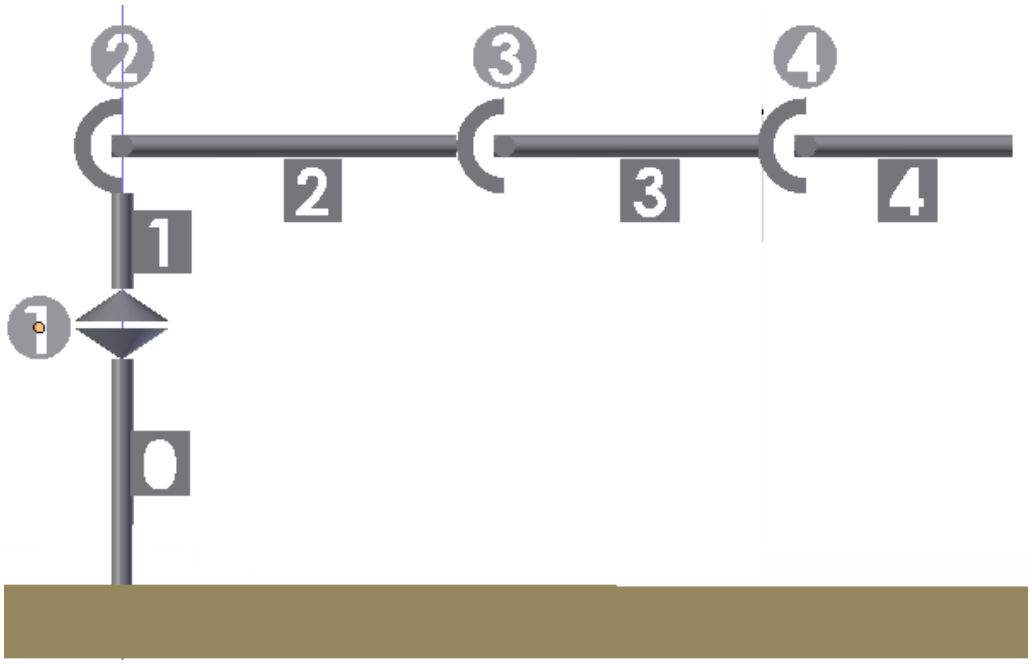


Ilustración 65: Eslabones y articulaciones Raplim, fuente: Autor

Ubicación de ejes de rotación:

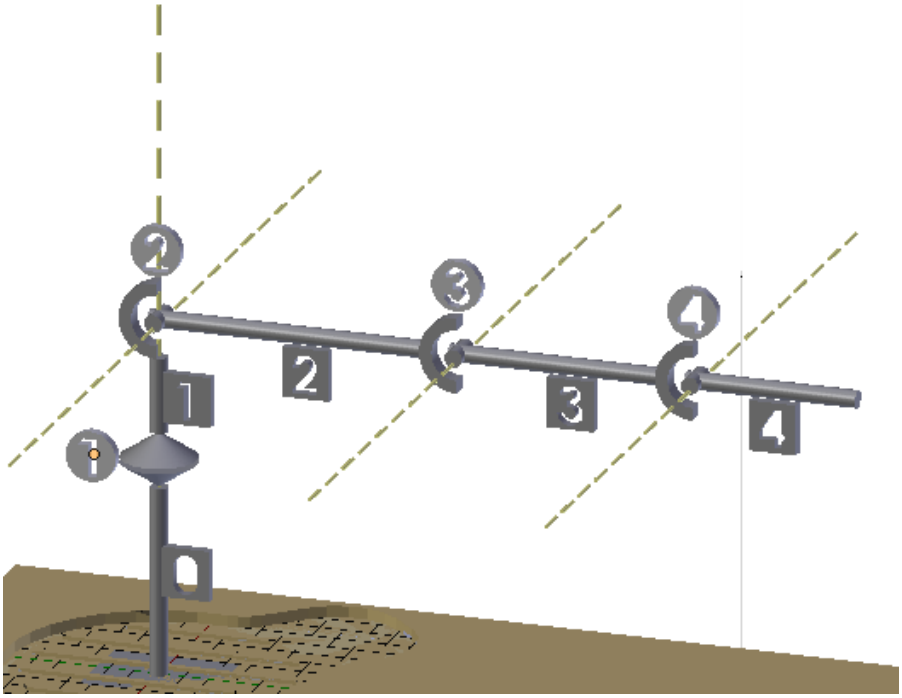


Ilustración 66: Ejes de rotación Raplim, fuente: Autor

Ubicación de los ejes Zi:

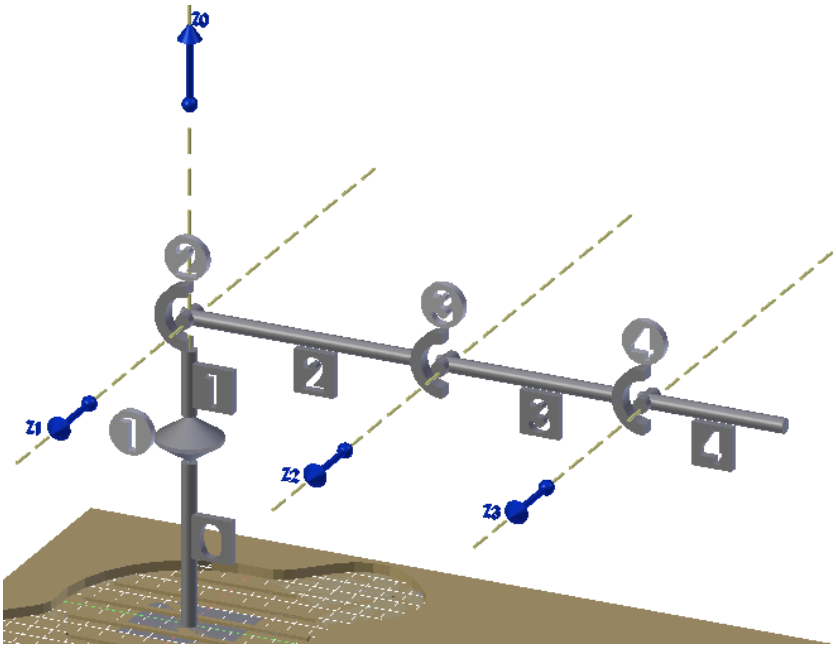


Ilustración 67: Ejes Zi Raplim, fuente: Autor

Ubicación del sistema S0 (Base):

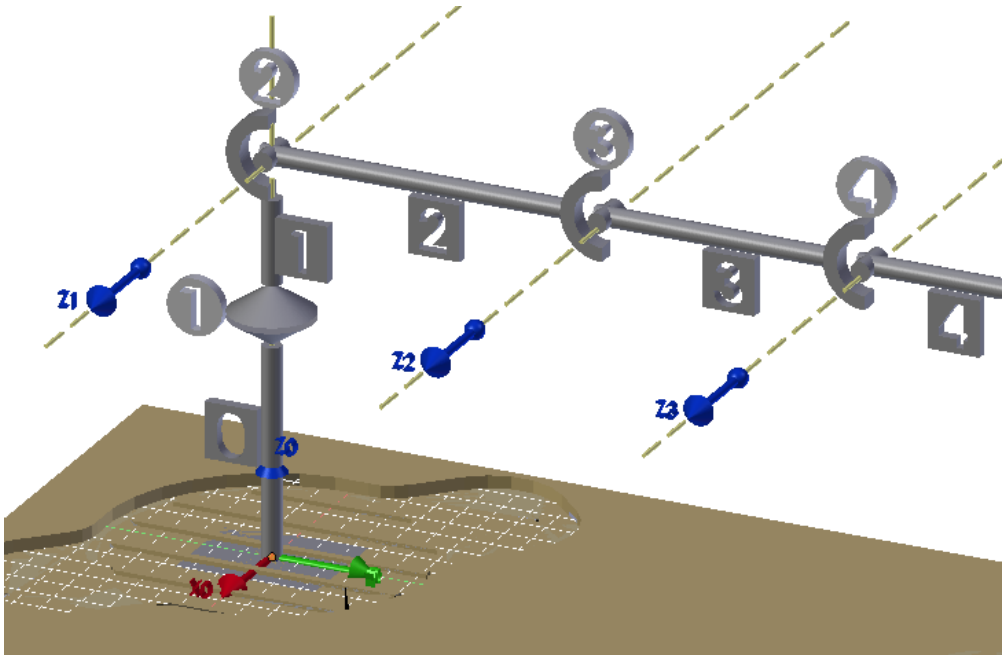


Ilustración 68: Sistema S0 Raplim, fuente: Autor

Posicionamiento de los ejes Z1 a Zn-1:

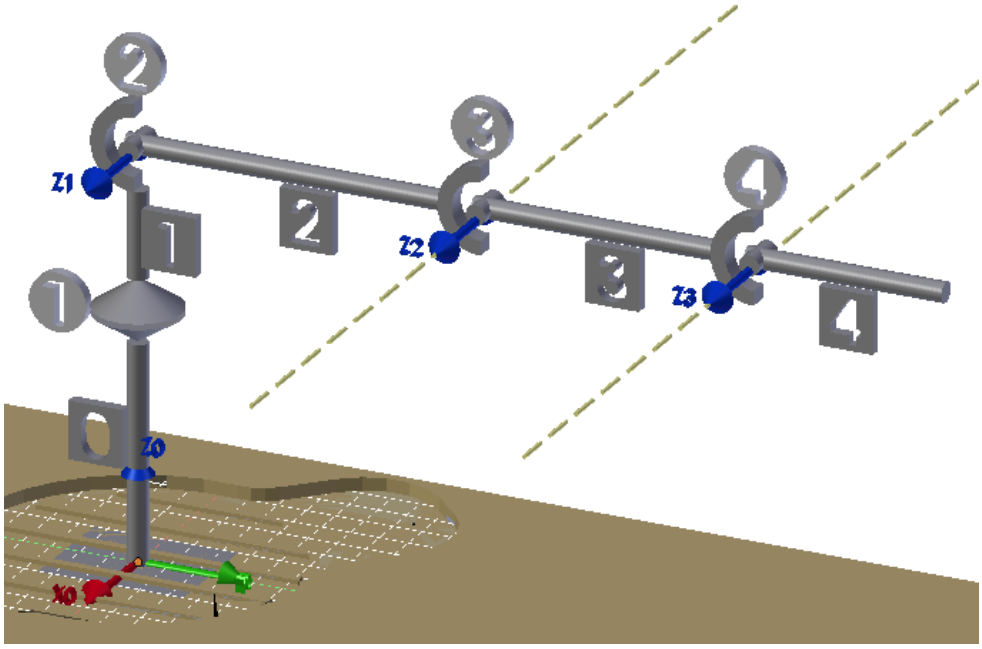


Ilustración 69: Ubicación ejes Z1 a Zn-1, fuente: Autor

Ubicación de X1 a Xn-1:

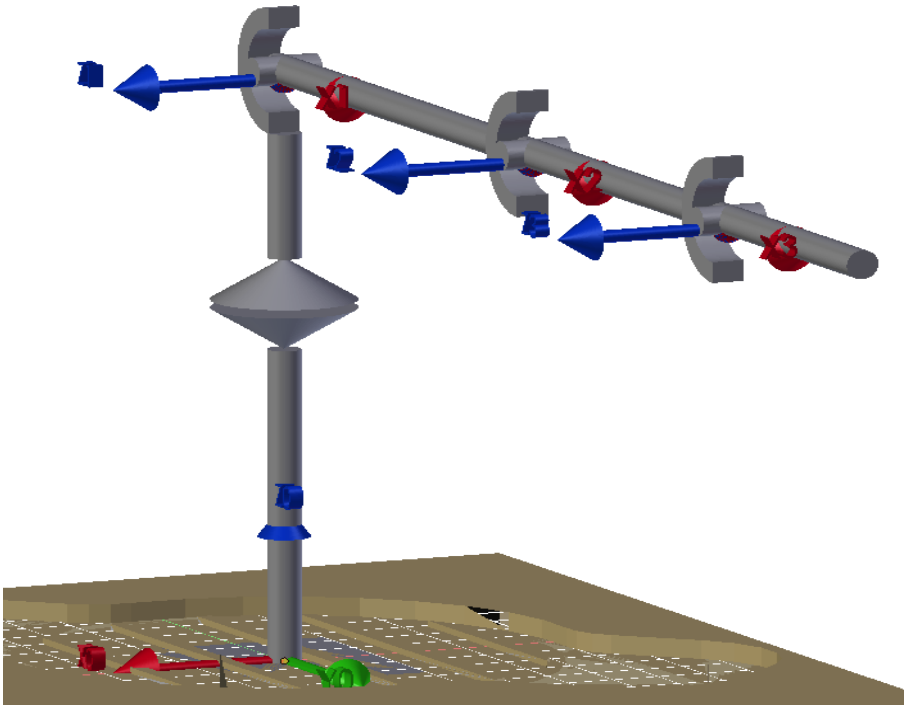


Ilustración 70: Posición X1 a Xn-1 Raplim, fuente: Autor

Ubicación de Y1 a Yn-1 para formas sistemas dextrógiros:

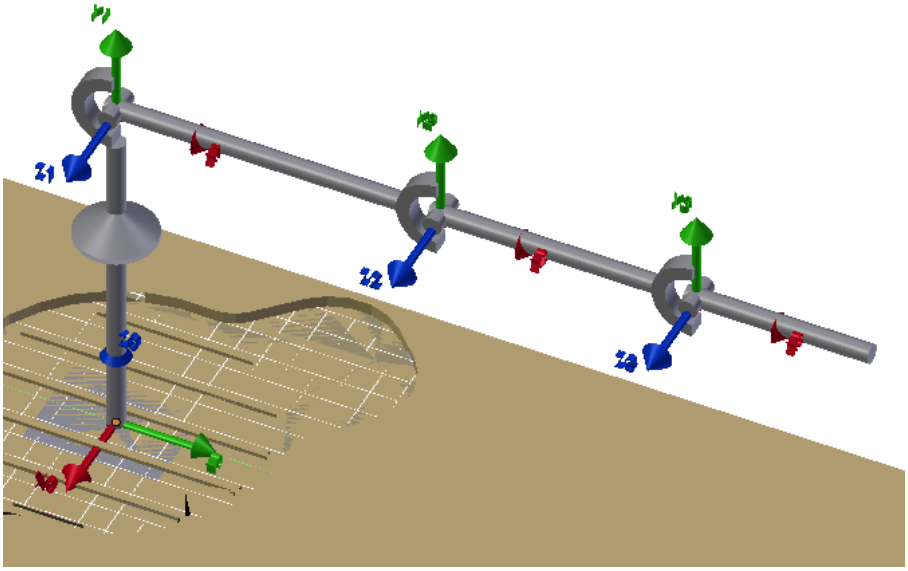


Ilustración 71: Sistemas S1 a Sn-1 Raplim, fuente: Autor

Ubicación sistema S4 o Sn:

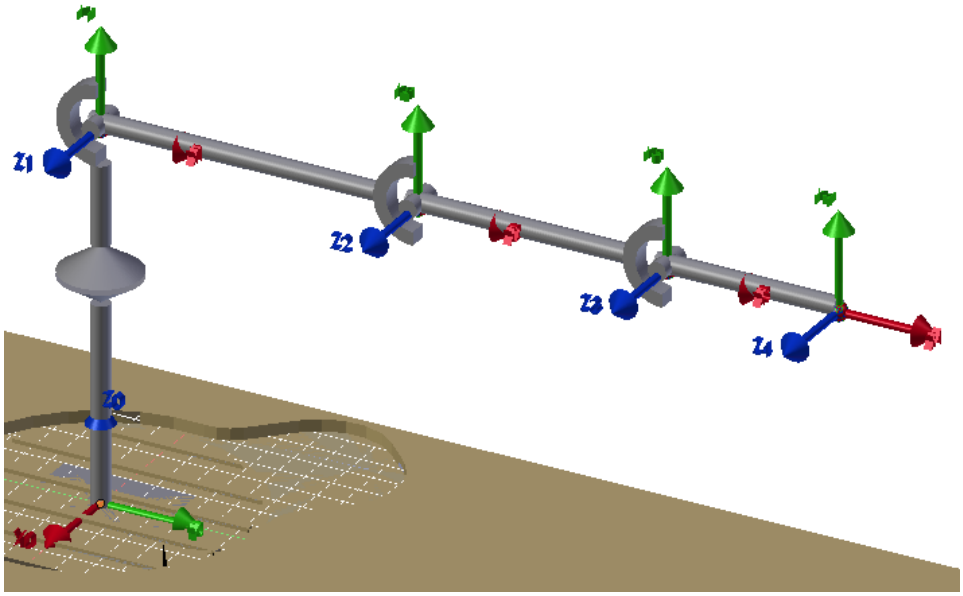


Ilustración 72: Sistema S4 Raplim, fuente: Autor

Parámetros dh Raplim:

Parámetros DH Raplim				
Relación Eslabones	Zi-1		Xi	
	Øi	di	ai	αi
0 - > 1	Q1+90	D1+D2	0	90
1 - > 2	Q2	0	D3	0
2 - > 3	Q3	0	D4	0
3 - > 4	Q4	0	D5	0

Tabla 4: Parámetros DH Raplim

El desarrollo de las matrices, se realizara en tiempo real a través de un algoritmo realizado en Unity en lenguaje de programación C#.

Cada reglón de los parámetros expuestos en la anterior tabla, establecen las rotaciones y translaciones necesarias, para que los sistemas solidarios a cada eslabón se conviertan en el sistema adyacente, es decir, para S0 se convierta en S1, y sucesivamente.

4.1.2. Cinemática inversa

- Determinando q1:

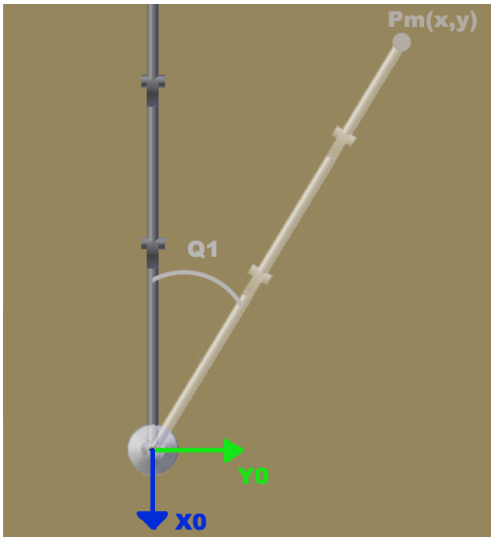


Ilustración 73: Vista superior Raplim, fuente: Autor

Q1 se establece, por medio de las componentes X y Y del punto conocido Pm. Con estos valores se puede establecer el ángulo que roto la articulación 1, para llegar a dicho punto, de forma que:

- Punto conocido:

$$Pm = [X, Y, Z]$$

- Q1: Se calcula como :

$$q1 = a\tan 2(Y, -X)$$

- Al rotar un ángulo Q1, el robot se desplaza de sus planos iniciales, generando un nuevo plano de trabajo. Además sobre este plano se puede establecer el alcance máximo(r) que tendrá el robot, este se define como:

$$r = \sqrt{X^2 + Y^2}$$

  Este nueva variable, determinara en todo momento el alcance que tendrá el robot, y dará un referente para el cálculo de las siguientes articulaciones.

- Determinando Q3:

Para determinar Q3 y Q2, se tomaran en cuenta una serie de triángulos que se pueden establecer entre la segunda y la cuarta articulación. Así mismo se supondrá que la distancia D5 correspondiente al último eslabón no afectara su cálculo, restando su magnitud al alcance máximo anteriormente determinado.

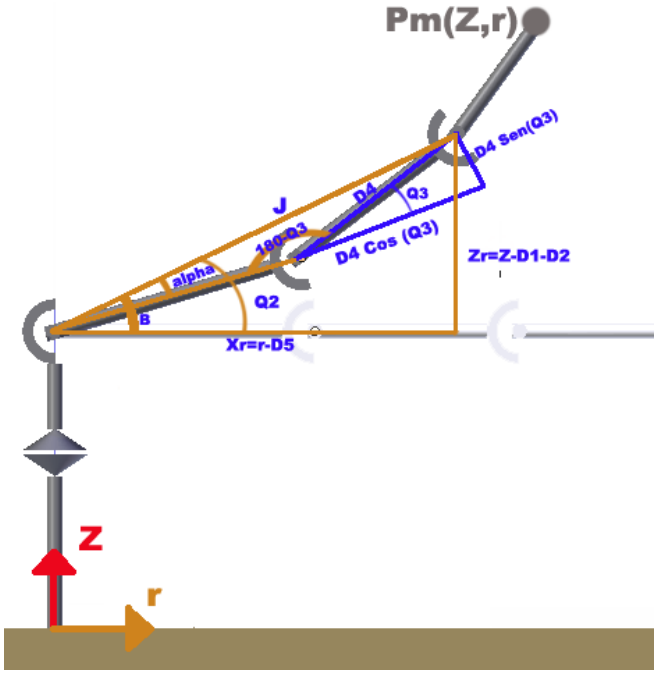


Ilustración 74: Vista desde el plano r parte 1, fuente: Autor

Para el cálculo de Q3, primeramente, se obtienen dos variables auxiliares que determinar la altura y el alcance real entre los eslabones 3 y 4. Para ello, se le resta al alcance máximo del robot D5, y al punto conocido las alturas correspondientes a los dos primeros eslabones.

$$Xr = r - D5$$

$$Zr = Z - (D1 + D2)$$

Con el valor de estas dos variables, se obtendrá dos de los 3 valores del triángulo rectángulo que se forma entre la articulación 2 y 4 y el plano r, además con ello se puede determinar el ángulo B, auxiliar necesario para el cálculo posterior de Q2.

$$\beta = a\tan 2(Zr, Xr)$$

El paso a seguir es determinar la hipotenusa, del triángulo rectángulo, la cual proporciona la magnitud necesaria para aplicar la ley del coseno al triangulo oblicuo (no rectángulo) que se forma entre las articulaciones 2, 3 y 4.

$$J = \sqrt{Xr^2 + Zr^2}$$

Posterior al cálculo de la hipotenusa J, se aplica el teorema del coseno, para dicho lado, el cual tiene como ángulo opuesta a 180-Q3:

$$J^2 = D_3^2 + D_4^2 - 2D_3D_4Cos(180 - Q3)$$
$$J^2 = D_3^2 + D_4^2 + 2D_3D_4Cos(Q3)$$

De la anterior ecuación se puede establecer Q3, al despejar el coseno, y aplicarle su inversa (coseno a la menos uno), pero realizar esto, no se tendría control sobre el tipo de codo a emplear por el brazo, por ello se decanta por el cálculo de Q3, obteniendo las componentes seno y coseno:

$$Cos(Q3) = \frac{J^2 - D_3^3 - D_4^2}{2D_3D_4}$$
$$Sen(Q3) = \pm\sqrt{1 - Cos^2(Q3)}$$
$$Q3 = a\tan 2(Sen(Q3), Cos(Q3))$$

- Determinando Q2:

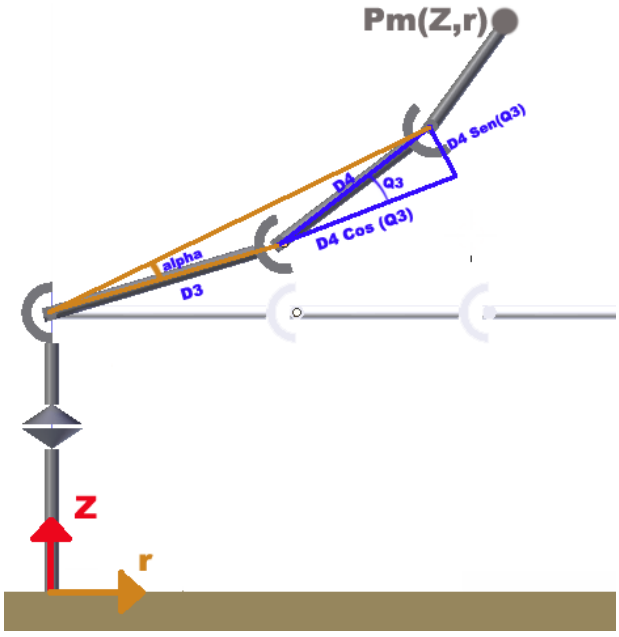


Ilustración 75: Vista desde plano r parte 2, fuente: Autor

Para determinar Q2, se establece un triángulo rectángulo entre la articulación 2, 4 y paralelo a la orientación del eslabón 3. Para ello es necesario determinar la proyección del eslabón 4 sobre el tercer eslabón, y sobre un plano perpendicular a este (Elevación del eslabón 4):

$$D3x = D_3 + D_4 Cos(Q3)$$
$$D3y = D_4 Sen(Q3)$$

Estos dos parámetros son necesarios para determinar Q2, ya que establecen los lados del triángulo rectángulo, necesarios para determinar el ángulo alpha (α), ángulo auxiliar para la determinación de Q2:

$$\alpha = a\tan 2(D3y, D3x)$$

Con alpha y beta (β), determinados, Q2 se define como:

$$Q2 = \beta - \alpha$$

- Determinando Q4:

Q4 se determina de manera que este siempre sea paralelo al plano XY o XR, para ello se debe determinar el ángulo Q4, con respecto al eslabón 3. Para ello se deben determinar las proyecciones del quinto eslabón sobre el eje X o r y sobre el eje Z.

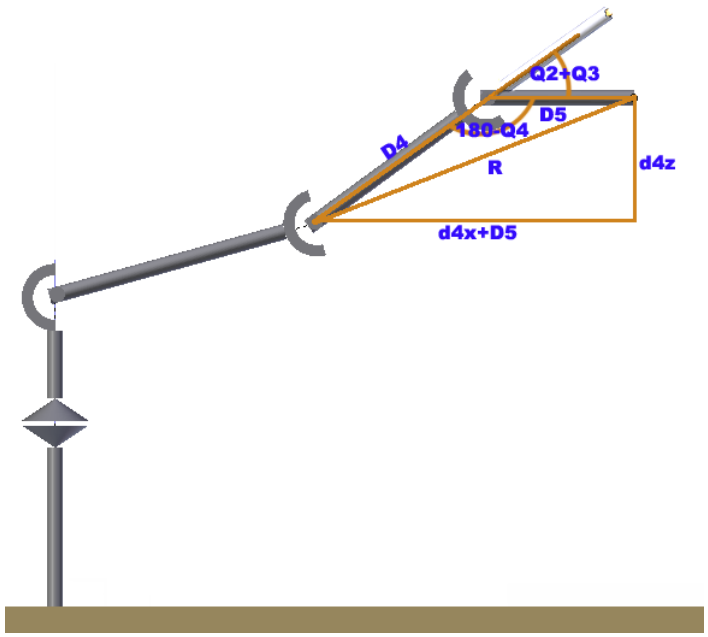


Ilustración 76: Vista desde plano r parte 3, fuente: Autor

Proyecciones de quinto eslabón, el cual esta rotado respecto al eje inicial Q2 y Q3, por lo tanto se definen estas como:

$$d4x = D_4 Cos(Q2 + Q3)$$
$$d4z = D_4 Sen(Q2 + Q3)$$

Con estas proyecciones se puede caracterizar el triángulo rectángulo que se forma entre la articulación 3, 4, permitiendo calcular la hipotenusa de dicho triangulo:

$$R = \sqrt{\left(d4z^2\right) + \left(D5 + d4x\right)^2}$$

Así mismo entre la articulación 3, 4 y el punto conocido o extremo final del último eslabón, se puede establecer un triángulo oblicuo, que tiene como lado y ángulo opuesto a R y 180-Q4, respectivamente. Aplicando la ley del coseno:

$$R^2 = D_4^2 + D_5^2 - 2D_4 D_5 Cos(180 - Q4)$$
$$R^2 = D_4^2 + D_5^2 + 2D_4 D_5 Cos(Q4)$$

De la ecuación de la ley del coseno, se despeja y calcula el coseno de Q4, y posteriormente el seno y Q4, como:

$$Cos(Q4) = \frac{R^2 - D_4^2 - D_5^2}{2D_4 D_5}$$
$$Sen(Q4) = \pm\sqrt{1 - Cos^2(Q4)}$$
$$Q4 = a\tan 2(Sen(Q4), Cos(Q4))$$

El +/- del seno de q3 y q4, se configuran como variables que determinan el tipo de codo y muñeca arriba o abajo dentro del algoritmo programado en python.

4.2. MODELADO Y ANIMACIÓN

Como se ha mencionado anteriormente, Blender es una herramienta dedicada al desarrollo de modelos 3D y producción de animaciones tridimensionales, empleado comúnmente en áreas de diseño y multimedia, pero poco empleada en la ingeniería. Siendo un reto para todo aquel que no haya tenido contacto con dicha herramienta.

Para la creación de los modelos y animaciones, fue necesario referenciarse e instruirse en los conceptos básicos de modelado y secuencias, que permitieran desarrollar óptimamente el diseño de Raplim.

El concepto de modelado en 3D, y animación no es un área desconocida en la ingeniería mecatrónica, dada la necesidad de desarrollar diseños mecánicos, o prototipos empleando software's como Solidworks o SolidEdge. En los cuales la unidad básica de diseño son los croquis, los cuales una vez trazados, se le aplican operaciones como extrudir saliente, revoluciones, extrudir de corte, etc., según lo requerido.

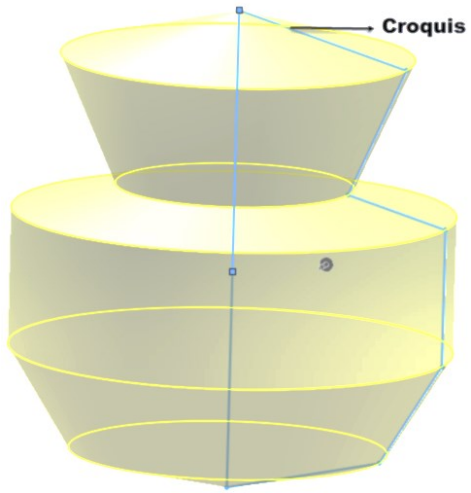


Ilustración 77: Operación de revolución, Fuente: Autor

Pero, en Blender el modelado se basa en trabajar sobre curvas u objetos primitivos, que deben ser alterados o trabajados para desarrollar arquitecturas o modelos más complejos. Para tener un primer contacto fructífero con Blender se requiere conocer su interfaz, y la función básica de cada una de las paletas o barras que esta presenta; se requiere también aprender a añadir curvas o mallas primitas sobre el espacio de trabajo, como modificarlas o unirlas unas con otras según lo requerido.

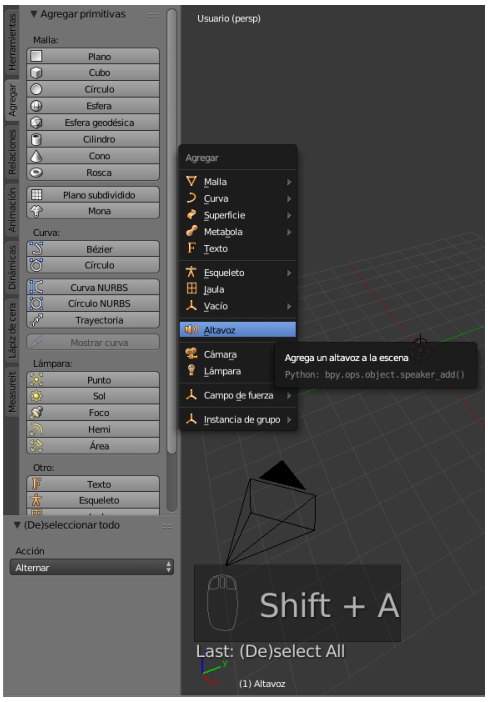


Ilustración 78: Agregar Primitivas, Fuente: Autor

Para trabajar en Blender es necesario conocer las combinaciones de teclas [27], o el menú donde se encuentran las herramientas a emplear.

Combinaciones tales como Shift + A, Shift + S u otras son comúnmente empleadas para el diseño, posicionamiento, rotaciones, enlaces, etc., dentro del entorno de Blender. Para ello se consultaron una variedad de fuentes de información, tales como foros [28] , blogs [29], comunidad Blender [30], video tutoriales, manual Blender [31], entre otros medios que facilitaran la compresión y desarrollo de modelos en esta herramienta.

Una vez obtenidos los conceptos básicos y las herramientas necesarias, se diseñaron diferentes modelos ajenos al proyecto, con el fin de aplicar y asentar lo aprendido durante la investigación y consultas. A continuación se mostrara dos de estos modelos:

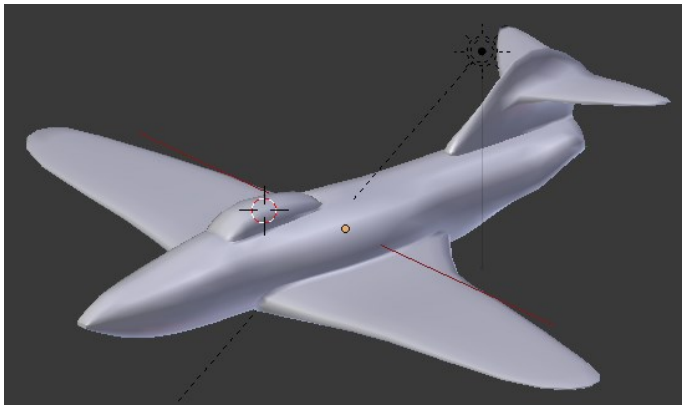


Ilustración 79: Modelo de avión Blender, Fuente: Autor

Ilustración 80. Modelo escudo universidad de Pamplona, Fuente: Autor

Una segunda fase necesaria para el presente proyecto, era conocer el motor de texturas y materiales, que emplea Blender a los modelos. Como se ha mencionado anteriormente Blender cuenta con una serie de motores gráficos, que según su elección, conllevara resultados rápidos y profesionales. Se investigaron dos métodos de coloreado en Blender:

1. Materiales: Blender cuenta con una herramienta básica para asignar y crear diferentes tipos de materiales, aplicando las extensas gamas de efectos que ésta tiene por defecto. Pudiéndose recrear materiales como acero, vidrio, plástico, entre otros con una calidad apreciable, ligada a la capacidad de procesamiento del ordenador. Éste resultado puede ser logrado de dos formas diferentes, dependiendo del motor gráfico empleado (Interno o Cycles).

   La asignación de un material es un proceso sencillo, que requiere de los siguientes pasos.

   1. Seleccionar el objeto o modelo al que se le aplicara el material.
   2. Ir a la pestaña material en la barra de propiedades.
   3. Añadir un nuevo material (Material base del objeto).
   4. Asignar las características del material (Color, traslucidez, etc.).

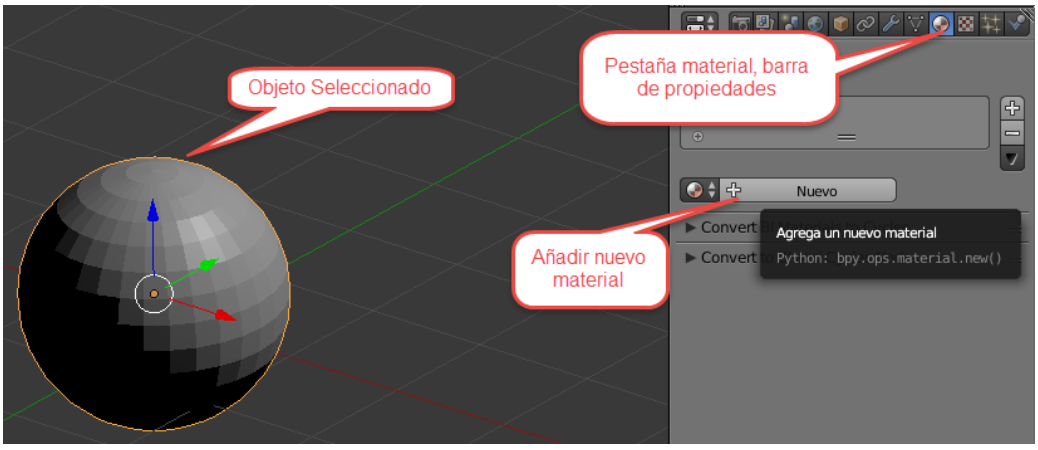


Ilustración 81: Creando materiales parte 1, Fuente: Autor

5. Si el objeto presenta dos tipos de materiales diferentes, ingresar al modo edición del objeto (Con objeto seleccionado presionar Tab).

6. Seleccionar las caras, o secciones con diferente material (La selección de varias caras se realiza con click derecho + Shift o con la tecla c).

7. Con el área de trabajo seleccionada, repetir pasos 2 al 4.

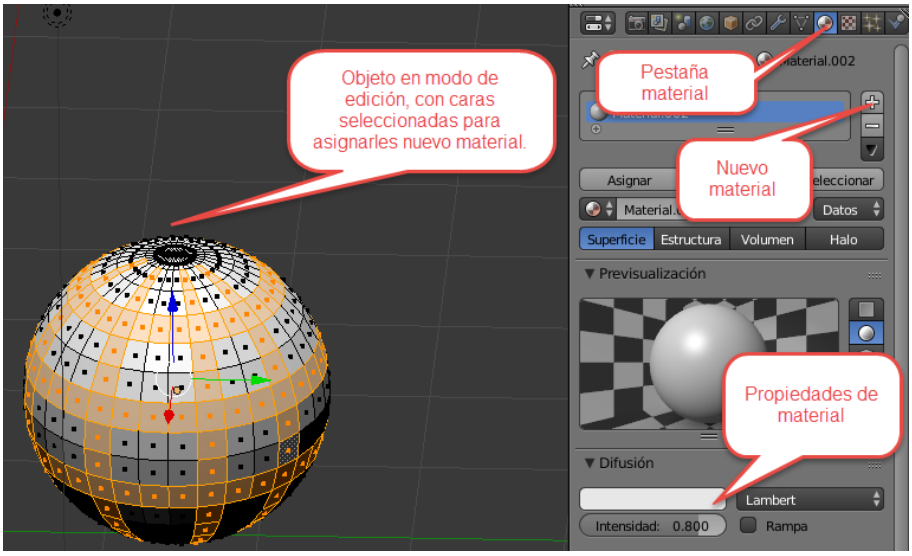


Ilustración 82: Creando materiales parte 2, Fuente: Autor

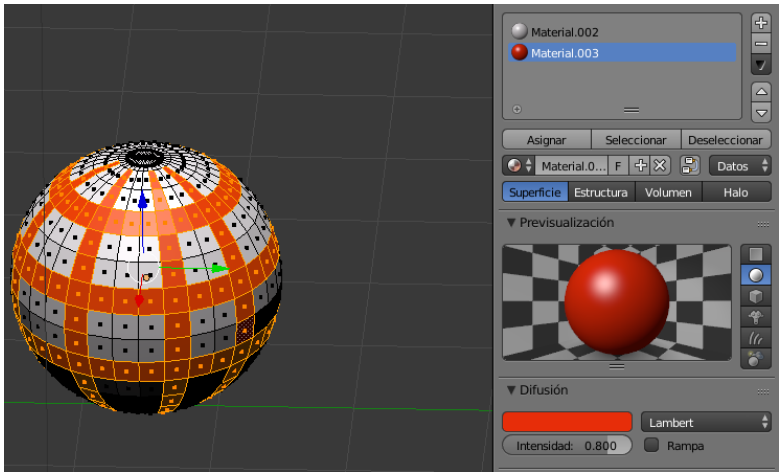


Ilustración 83: Creando material parte 3, Fuente: Autor

Lo expuesto anteriormente es el procedimiento más básico para la asignación de materiales a un modelo en Blender, pero éste no se limita a estas funcionalidades, también cuenta con otra series de opciones y herramientas, que permiten mejorar y dar foto realismo a los modelos, así como crear efectos cada vez más complejos como hologramas, metales preciosos, etc. Entre estos se encuentran los nodos de composición.

La correcta selección del motor gráfico es muy influyente ya que, al ser pre-renderizado el modelo, se obtendrás un mayor realismo si se emplea Cycles, en comparación con el pre-renderizado del motor gráfico interno.

A continuación se exponen los resultados obtenidos en los pre-renderizados dependiendo del motor escogido.

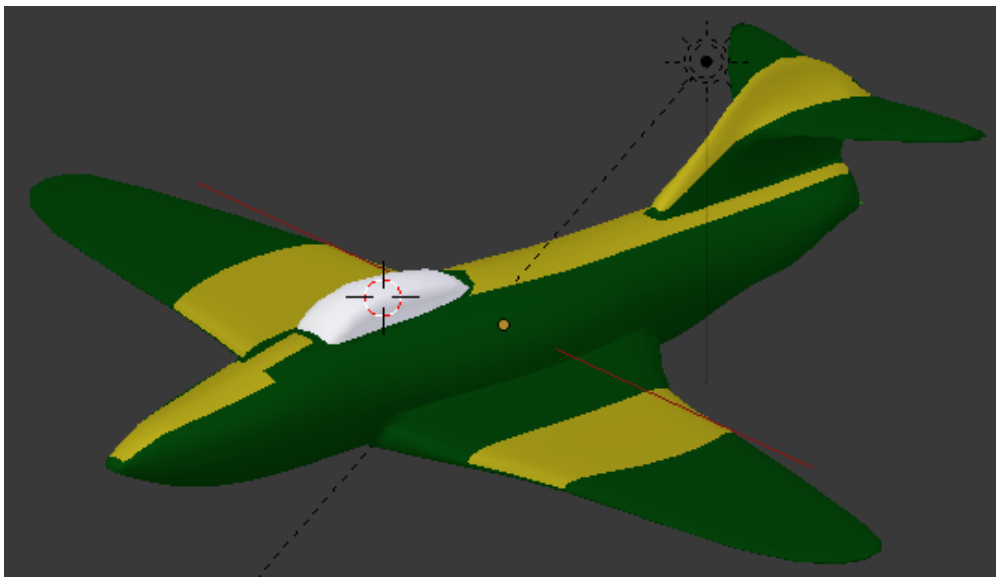


Ilustración 84: Pre-renderizado motor gráfico interno, Fuente: Autor

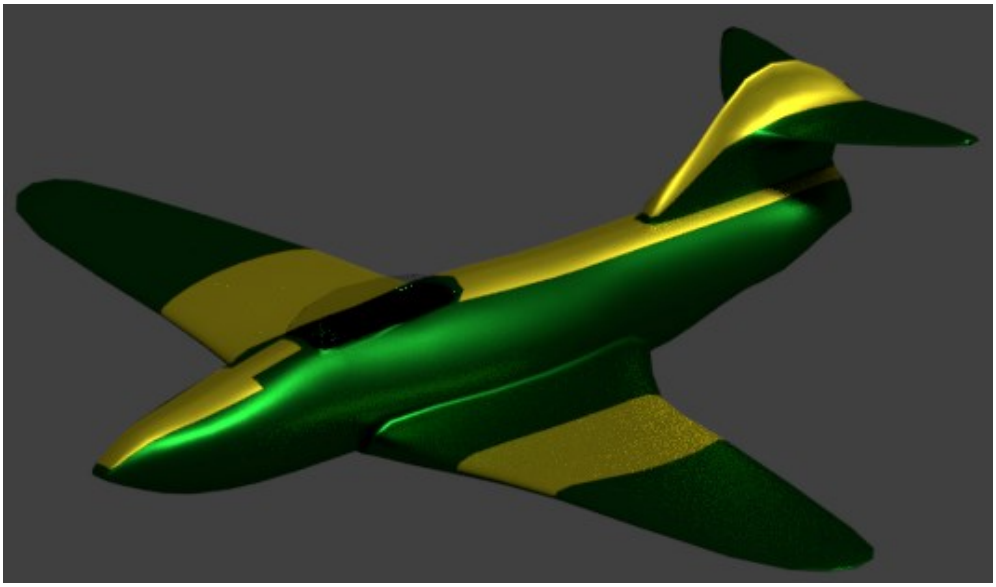


Ilustración 85: Pre-renderizado motor gráfico cycles, Fuente: Autor

Como se puede observar la diferencia es enorme, y es debido a como procesan la luz y las sombras ambos motores gráficos. En el interno, las sombras y la luz son efectos planos, es decir, su intensidad no varía con las distancias y ángulos del punto de luz. Por otro lado el

motor cycles, genera efectos suaves y realistas, teniendo en cuenta la posición y orientación del punto de luz respecto al objeto.

Efectos más complejos y realistas pueden ser desarrollados a través de una herramienta denominada nodos de composición, la cual permite aplicar ajustes a los elementos de la escena, recalcar materiales, aumentar la cantidad de detalles, etc. Esto se logra a través de la combinación de más dos o más efectos básicos sobre un mismo objeto, pudiéndose aplicar traslucidez, emisión u otro efecto simultáneamente.

Esto se realiza empleado el aplicativo editor de nodos integrado en blender.

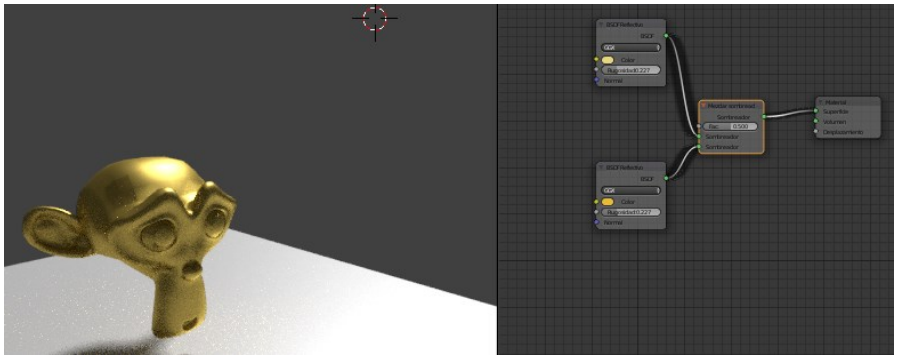


Ilustración 86: Mejora de materiales con nodos de composición, Fuente: Autor

La tercera fase necesaria para el modelado de Raplim en Blender. Era contextualizar los métodos para que un objeto fuera solidario a otro, es decir, que dos elementos o más elementos solidarios se orienten en el espacio como un conjunto. Siendo uno la referencia de orientación del otro.

En Blender esto se puede realizar a través de emparentar objetos, relacionándolos como padre e hijo, siendo el primero superior al segundo, y en donde la orientación espacial del padre se determina respecto al sistema coordenado de Blender, y el hijo respecto al sistema coordenado del padre.

Ilustración 87: Emparentar por Huesos, Fuente: Autor

Blender cuenta con una gran variedad de tipos para emparentar los objetos presentes en el proyecto: Entres estas encontramos deformación de armadura cuyo propósito es emparentar dos objetos, en el cual el las variaciones o deformaciones dentro del objeto padre, generan deformaciones en el objeto hijo; huesos, es un tipo de emparentar empleado para la animación de personajes, en el cual los huesos simulan en movimiento del cuerpo y estos a su vez deforman o varían el modelo según la posición del hueso; emparentar por objeto, es la forma más básica de emparentar objetos en Blender, en la cual el objeto hijo, asimila los movimientos y rotaciones del objeto padre, este tipo es el empleado en el presente proyecto. [28] [32]

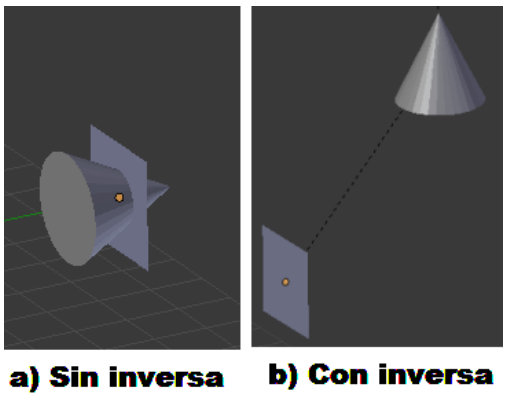


Ilustración 88: Emparentar por objeto, Fuente: Autor

Este tipo para emparentar objetos puede realizarse de dos formas, difiriendo en los resultados obtenidos, y en matemática que esta emplea.

- Sin inversa (Shift – Ctrl – P): Por este método, si el objeto padre ha sufrido cambios en su rotación antes de emparentar, estos serán reflejados en el objeto hijo (a Ilustración 88), además la posición del objeto hijo es reiniciada o setteada al origen del objeto padre.

- Con inversa (Ctrl – P): Al emparentar por este método, las rotaciones del objeto padre antes de emparentar no afecta al objeto hijo, además este se posicionara respecto al origen del objeto padre, a la misma distancia respecto al origen de Blender, que tuvo antes de ser emparentado.

La última fase necesaria corresponde a la producción de la simulación, para esto Blender cuenta con un sistema de captura de keyframes, en donde cada keyframe almacena la información de cada objeto en el espacio, siendo estos secuenciales, permiten generar un efecto de movimiento, orientando los elementos según la posición requerida.

En la producción de animaciones en Blender, no es necesario determinar los movimientos de cada objeto keyframe por keyframe, basta con indicar la posición inicial y final de los elementos en el keyframe inicial y final respectivamente, los movimientos intermedios entre

estos son generados por Blender, empleando una serie de herramientas y métodos de cálculo que determinan la posición intermedia entre ambos puntos (posición final e inicial).

Para la captura de keyframes, Blender recurre a una metodología en el cual el usuario debe indicar que información almacenar en cada keyframe, siendo los datos almacenados más comunes, la posición, rotación, escala, dimensión, visibilidad, entre otros; los cuales permiten obtener diferentes efectos durante la animación.

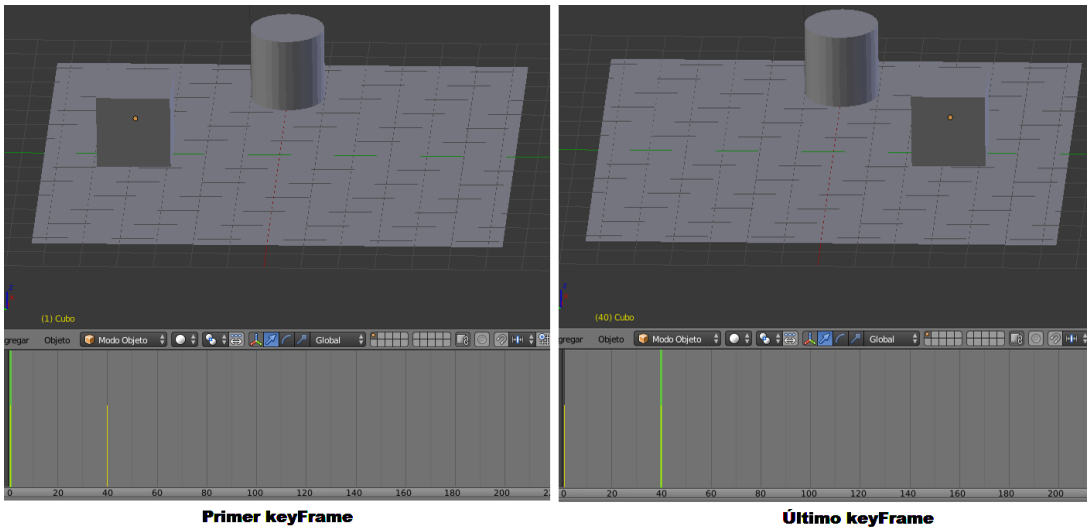


Ilustración 89: Animación Blender, Fuente: Autor

4.2.1. Modelado Raplim

El modelado de Raplim para el sistema de realidad aumentada, se realizó a través de modelos STL, desarrollados por los Ing's Alfredo Márquez y Javier Hernández. Con dichos modelos como base, se adecuaron de tal manera que no presentaran una carga excesiva para el sistema de RA.

La calidad de los modelos empleados por el software Unity depende de la cantidad de caras, aristas y/o bordes con los que estos cuentan, siendo a mayor cantidad mayor calidad del modelo, pero sobreponiendo un mayor consumo de recursos para el procesamiento de los mismos. Por ello sobre los modelos previamente proporcionados se realizaron dos actividades fundamentales:

- Reducir el número de aristas, bordes y caras de cada uno de los modelos STL, con el fin de reducir dicha carga.

- La rotación y desplazamiento dentro de Blender depende del punto pivote o centro de masa del modelo, para ello, esta cuenta con una herramienta que permite desplazar este punto, a cualquiera dentro del espacio de trabajo.

El procedimiento a seguir para esto fue el siguiente:

1. Importar el modelo a Blender.

2. Escalar el modelo, a dimensiones reales. Blender no cuenta con un sistema de unidades dentro de su entorno, generando un sobre escalamiento de cualquier modelo desarrollo en otro software CAD. Por ello aunque en otro este sea exportado en cm o m, este tomara la magnitud o valor de ellas. Por ende un modelo de 120 cm en un software como SolidWorks, en Blender puede ser tratada 120 cm o metros. El factor de escala empleado es 0.1.

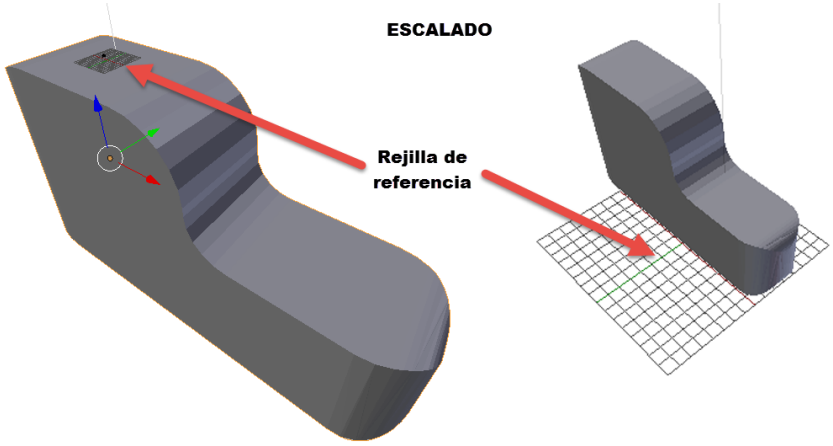


Ilustración 90: Escalado modelo en Blender, fuente: Autor

3. El punto pivote original de algunas de las piezas del modelo Raplim original difieren con los pivotes asignados durante el desarrollo de la cinemática directa en el presente trabajo. Además dado que Blender emplea otro sistema de posicionamiento espacial es necesario, que el punto pivote de toda pieza este asignado sobre el eje de rotación de la misma. Esto es necesario sobre todo en el último eslabón donde comúnmente se asigna el pivote en el extremo del efector final.

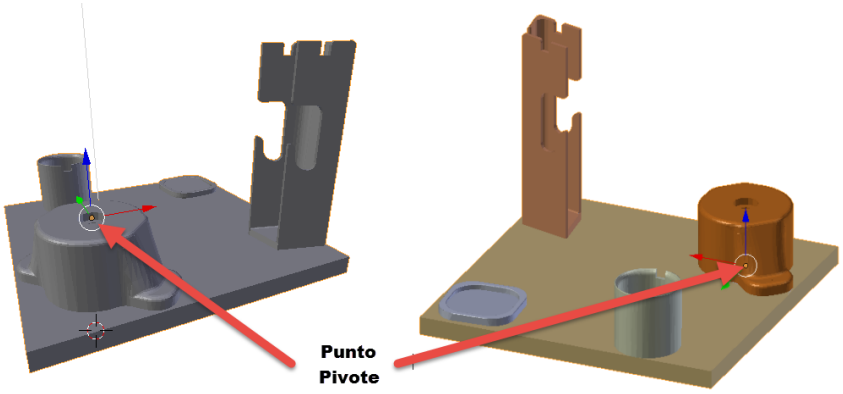


Ilustración 91: Cambio punto pivote modelos, fuente: Autor

4. Asignación de materiales, sobre cada objeto presente de los modelos. Como se puede observar en la Ilustración 91: Cambio punto pivote modelos.

5. Ensamblaje del modelo, relación padre e hijo entre cada elemento de Raplim. Para ello se emparento desde el efector final hasta la base, por medio de emparentar sin inversa (Shift – Ctrl – P).

6. Desarrollo de otros utensilios; el modelo original cuenta con un solo efector final: la cuchara, por ello para el presente se desarrollan de otros dos: vaso y cepillo con el fin de aumentar el número de rutinas de la simulación.

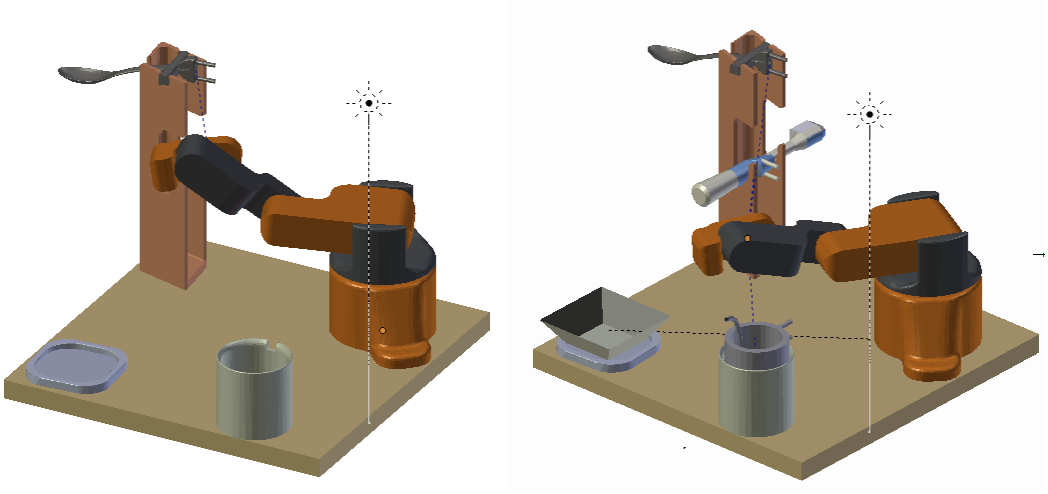


Ilustración 92: Desarrollo de nuevos efectores finales – Blender, fuente: Autor

4.2.2. Animación Raplim

El desarrollo de la simulación en Blender se basa en los puntos de interés dentro del espacio de trabajo, es decir, se definen aquellas posiciones espaciales a los cuales Raplim debe llegar con el fin de tomar uno de sus efectores finales: Para ello se definen los siguientes puntos:

Posiciones definidas efectores finales			
	X	Y	Z
Cuchara	24	-1	26
Cepillo	24.46	-0.5	12.5
Vaso	1.7	23.34	9.5

Tabla 5: Posiciones actuadores Raplim

Una vez Raplim tome uno de sus efectores finales es necesario realizar ajustes en sus dimensiones, para el cálculo de los valores angulares del mismo, que estos varían o aumentan el alcance inicial que posee Raplim.

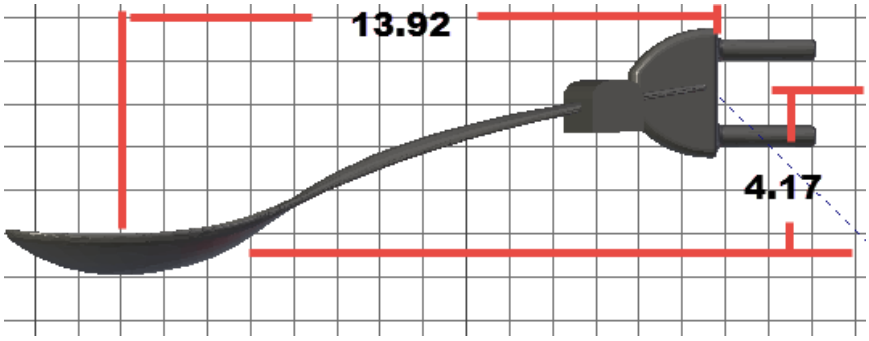


Tabla 6: Dimensiones cuchara, fuente: Autor

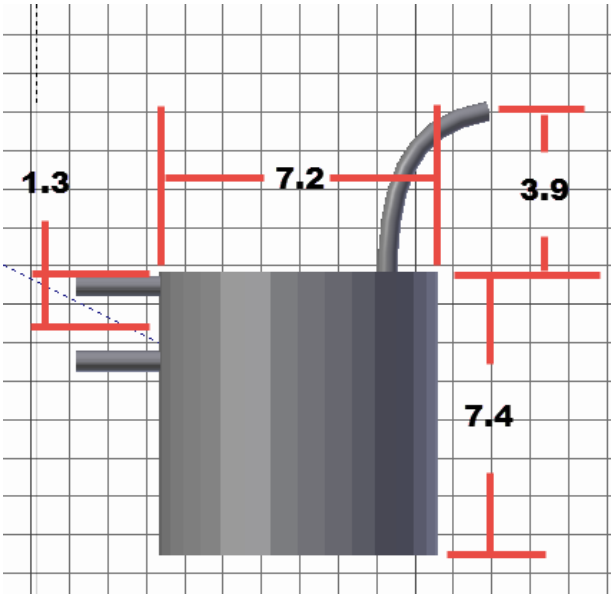


Tabla 7: Dimensiones vaso, fuente: Autor

Para el tercer efector: cepillo, se desprecian las variaciones que provocan sus dimensiones, dado que por su orientación, la variación generada por él, no supera el los dos centímetros.

Teniendo en cuenta estas consideraciones, se definen los siguientes puntos intermedios

Posiciones intermedias definidas			
	X	Y	Z
Plato	25	25	6.7
Punto intermedio	25	25	19.4
Persona rutina 1	-21.3	0	19.4
Devuelve la cuchara a posición inicial			
Robot a posición inicial: Valores angulares a 0			

Punto intermedio para empezar rutina de vaso	0	15.8	25.5
Punto intermedio una vez toma el efector final vaso	2	22.9	15.46
Persona rutina 2	-17	0	18.6
Devuelve vaso a posición inicial			
Posición intermedia para empezar rutina de cepillo	0	15.8	25.5
Persona rutina 3	-19.4	0	17.82
Devuelve cepillo a posición inicial			

Tabla 8: Puntos intermedios predefinidos

Los puntos anteriormente expuestos son los de mayor relevancia en la simulación, los puntos intermedios entre estos son calculados internamente por Blender.

Los puntos de interés definidos se pasan como parámetros al algoritmo de la cinemática inversa desarrollado en lenguaje pyhton, con el propósito de obtener los valores angulares para cada articulación para posicionar el efector final en éstos.

Durante las simulaciones, el robot se acopla o toma cada uno de sus efectores finales, los cuales siguen la posición y orientación del último eslabón de Raplim, para ello Blender cuenta con una serie de propiedades que permiten interactuar objetos dentro de su entorno de trabajo. Algunas de estas son: las restricciones de movimiento, los modificadores, etc. Con ellos se pueden recrear físicas como campos de gravedad, colisiones, bisagras, entre otras.

En el presente, se empleó la restricción subordinar en el entorno de Blender, la cual obliga a un objeto a seguir los movimientos y orientaciones que tome el objeto padre, teniendo en cuenta, su posición original antes de la subordinación.

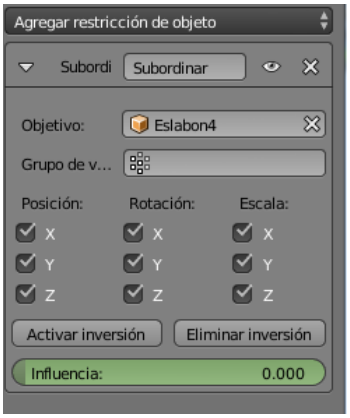


Ilustración 93: Restricción subordinar parámetros, fuente: Autor

Todos los efectores finales presentan esta característica, y se emplea con el propósito de que el objeto siga el movimiento de Raplim una vez el eslabón final de éste se acerque a su poción inicial. Un aspecto fundamental de subordinar es determinar el keyframe desde el cual el objeto debe seguir a su objeto padre, para ello se emplea la influencia de ésta restricción, la cual toma valores entre 0 y 1.

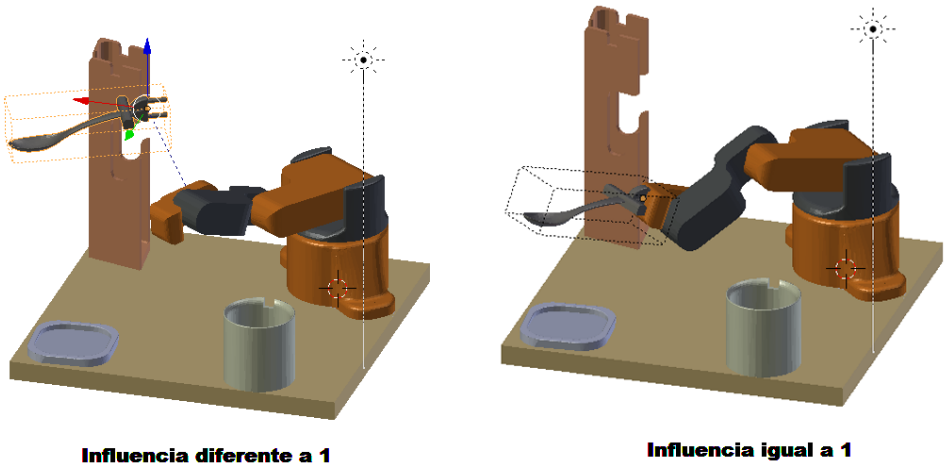


Ilustración 94: Restricción subordinar, fuente: Autor

En la anterior ilustración se puede observar el resultado que genera el valor asignado de la influencia en la acción subordinar, determinando que el valor a emplear para lograr el resultado querido es 0 o 1. También es necesario asignar el método de variación de éste parámetro dentro de un lapso de tiempo, o entre keys de la simulación, para esta tarea Blender maneja un editor de curvas que permite modificar el tipo de interpolación que se va a efectuar entre los dos keys que tengan influencia diferente.

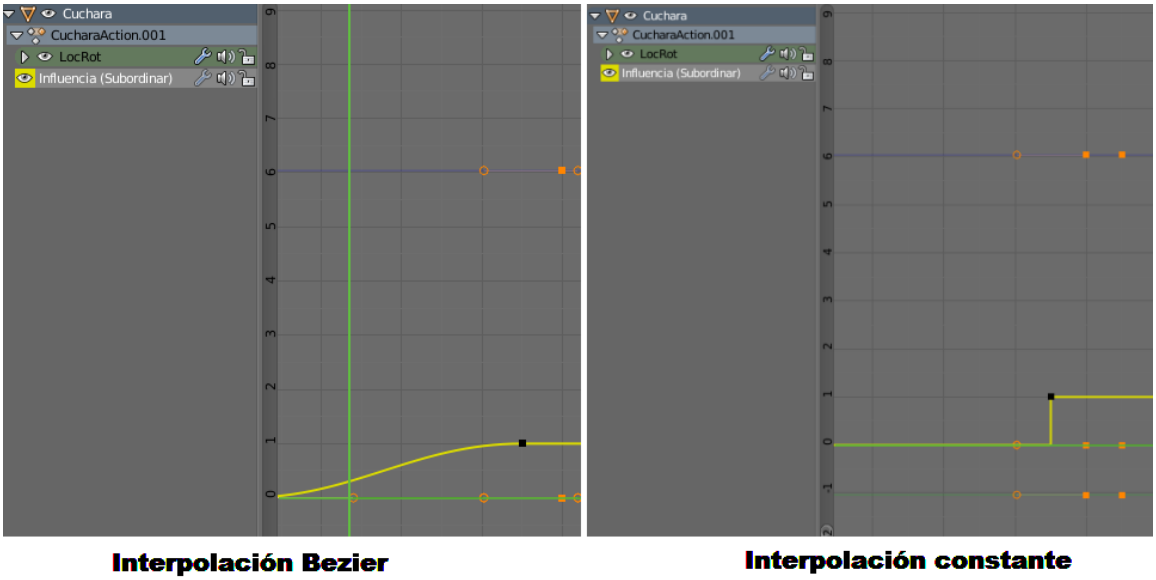


Ilustración 95: Interpolación curvas, fuente: Autor

Blender maneja 3 tipos de interpolación: Constante, lineal y bezier; el tipo de interpolación por defecto en Blender es bezier, en esta cualquier tipo de keyframe o parámetro empleado para la simulación varia su valor según una curva del tipo bezier, de menor a mayor o viceversa entre los dos keyframes. Éste tipo de funcionamiento era inadecuado para la simulación del sistema de RA, por ello se configuro como interpolación constante la influencia de la restricción subordinar, de modo que, el valor de influencia variaría instantáneamente solo en el keyframe, donde el eslabón final de Raplim estuviera en la posición indicada para tomar cada uno de los efectores finales.

4.2.3. Algoritmo cinemática inversa Python – Liclipse

```
'''
Created on 4/04/2017

@author: JSJ
'''
# coding=utf-8
# Definiendo librerías

import math



# Parámetros robot
    # Longitud Eslabones

D1=8.5
D2=5.5
D3=12.023
D4=9.46
D5=6.5


# Factores de conversión

Ang2rad=math.pi/180 # Conversión grados a radianes
rad2ang=180/math.pi # Conversión radianes a grados
# Parámetros cinemática


#Pc=[24.13,-1.10,26.1]
#Pc=[24.22,-1.10,26.1] # Punto del efector final
Pc=[-19.52,0.917,18.36]
#Pc=[24.2,-1.01,26]


while (dc == 0) :
    # Determinando q1

    q1=math.atan2(-Pc[0],Pc[1])
    r=math.sqrt((Pc[0])**2 + (Pc[1])**2)
```

```
    # Determinando q3

    Xr=r-D5 #-8.57   #-13.92    # 8.57 , 13.92 Es la magnitud del vaso y la
cuchara respectivamente, se descuenta cuando la cuchara es el efector final
    Zr=Pc[2]-(D1+D2)#-5.2 # 5.2 Es la altura efectiva del vaso
    Beta=math.atan2(Zr,Xr)
    J=math.sqrt((Xr**2)+(Zr**2))
    cosq3=((J**2)-(D3**2)-(D4**2))/(2*D3*D4)
    senq3=math.sqrt(1-((cosq3)**2))
    q3=math.atan2((codo*senq3),cosq3)
    #q3=math.acos(cosq3)

    # Determinando q2

    ax=D3+(D4*math.cos(q3))
    ay=D4*math.sin(q3)
    alpha=math.atan2(ay,ax)

    q2=Beta-alpha

# Determinando q4

    d4x=D4*math.cos(q2+q3)
    d4z=D4*math.sin(q2+q3)
    R=math.sqrt((d4z**2)+((d4x+D5)**2))
    cosq4=((R**2)-(D4**2)-(D5**2))/(2*D4*D5)
    senq4=math.sqrt(1-(cosq4**2))
    q4=math.atan2((muneca*senq4),cosq4)

# Convirtiendo a grados los ángulos

    q1= q1* rad2ang
    q2 = q2*rad2ang
    q3= q3* rad2ang
    q4= q4* rad2ang

    if ( q2 < 0):
        muneca=1
        codo=-1
    else :
        muneca=-1
        codo=1
        dc=1
# Imprimiendo resultados

print q1
print q2
print q3
print q4
```

4.2.4. Configuraciones de exportación y modelado Blender

Para el presente trabajo, se tomaron las siguientes preferencias y configuraciones en Blender, para el modelado y simulación, con objeto de obtener el mejor comportamiento al realizar la exportación e implementación del modelo a Unity.

- Versión blender: 2.78c.

- Motor de pre-renderizado: Se selecciona el motor de pre-renderizado como interno, debido a su compatibilidad con el motor de Unity, es decir, las texturas o materiales aplicados por medio de este motor de Blender, serán tomados en cuenta por Unity.

- Fps: 24 frames por segundo.

- Resolución: 720* 486

- Formato de salida: H.264

- Emparentar sin inversa, se emplea este método por ser compatible con el usado por Unity para éste mismo fin.

- Formato de exportación. FBX Versión 6.1 ASCII

4.3. CREACIÓN DE PROYECTO EN UNITY

Una vez terminada la modelación y simulación a emplear, se debe crear el proyecto en Unity, y cargar los elementos o prefeabs a emplear, entre ellos:

- Modelo Cad en formato fbx exportado desde Blender.

- Base de datos generada en la página oficial de Vuforia.

- Sonidos e imágenes alusivos al proyecto.

Para importar los modelos cad, simulaciones, sonidos e imágenes basta con copiar y pegar estos archivos dentro del directorio Assets encontrado en la carpeta del proyecto. Por otro lado todo prefab creado por medio de una herramienta de Unity, como lo es el generador de marcadores, genera un instalador para su posterior exportación al entorno de trabajo.

Con los elementos exportados a Unity, se carga el prefab ArCamera e ImageTarget de Vuforia al entorno de trabajo, arrastrándolos desde la pestaña Project hasta Hierarchy. Siendo el primero, el reemplazo de la cámara por defecto en Unity, y el segundo, el correspondiente tipo de seguimiento, marcador.

4.3.1. Primer ensayo

El primer prototipo desarrollado en el presente proyecto, fue planteado de la siguiente forma:

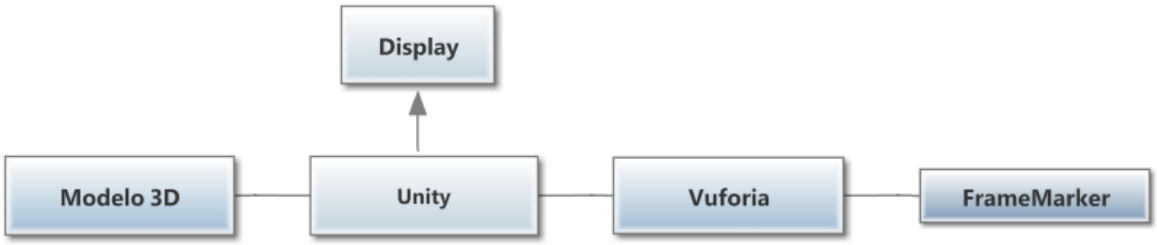


Ilustración 96: Diagrama esquemático primer prototipo, fuente: Autor

Este primer ensayo se realiza con los siguientes propósitos: Establecer las configuraciones más estables basadas en la previa investigación sobre los métodos de la ARCamera; rendimiento y estabilidad de las configuraciones y del FrameMarker con modelos simples (Figuras primitivas); igualmente establecer el rendimiento y estabilidad de las configuraciones y FrameMarker para modelos complejos; por ultimo determinar la relación marcador/escala.

El primer modelo empleado es un cubo, generado Unity (gameObject – 3D Object – Cube), con escala a 0.8, referenciada al marcador tipo frameMarker. Siendo este el padre del cubo.

Para lograr esto en Unity, se arrastra el modelo 3D hacia el marcador en la pestaña Hierarchy, esto se realiza con el propósito que objeto aparezca, solo cuando sea detectado el marcador.

Cuando se emplea un marcador tipo FrameMarker, solo se le debe asignar el ID, de uno de uno de los posibles marcadores ofrecidos por Vuforia, como se puede observar en la siguiente en la Ilustración 97.

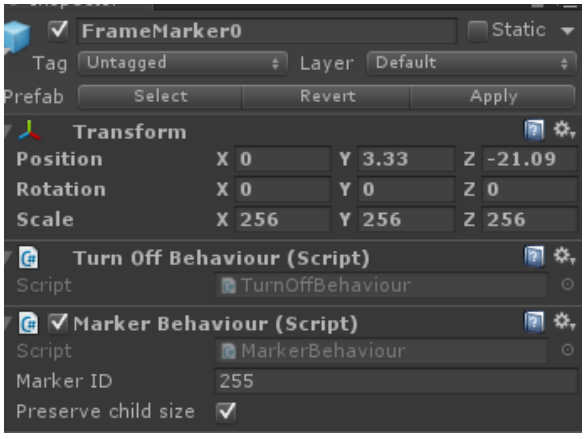


Ilustración 97: Propiedades (Inspector), FrameMarker, Fuente: Autor

El prefab Arcamera de Vuforia presenta dos componentes principales: El primero es la ARCamera, a la cual se le aplican una serie de métodos y propiedades del tipo Vuforia y derivados de clase monobehavior (Clase de Unity). El segundo elemento es la Camera, encarga de todas las propiedades de obturación: Como profundidad, apertura de obturación entre otros.

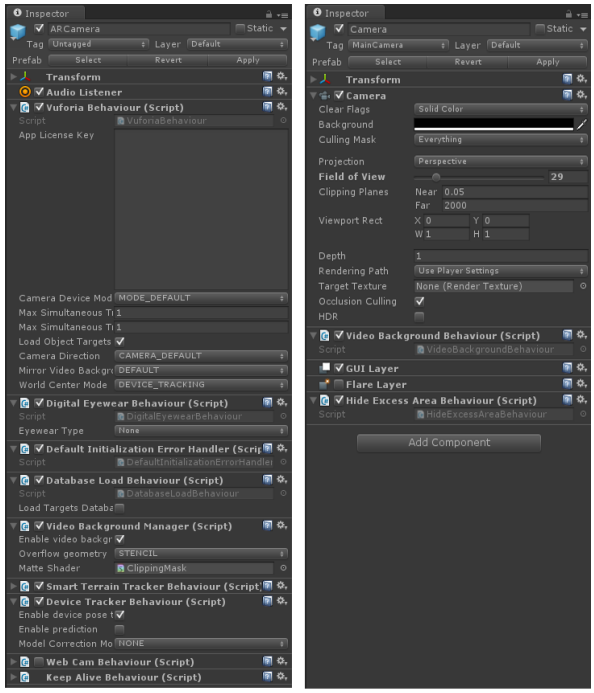


Ilustración 98: Propiedades Camera y ARcamera, fuente: Autor

Los métodos de ARCamera configurados para este primer acercamiento son:

- App License key: Este método de la clase Vuforia Behaviour es el campo donde se debe asignara el key de la licencia previamente creada, sin la asignación del mismo, se presentara error, al intentar compilar o ejecutar la aplicación.

- Camera device Mode: Método que determina la calidad empleada en el renderizado, del modelo 3D, y la velocidad de fotogramas de la aplicación, esta función prioriza una de estas opciones dependiendo de las capacidad de procesar del equipo empleado [33], además Mode-Default provee resoluciones de 640*480 según [34].

- Camera direction: Método por medio del cual selecciona la cámara del dispositivo a emplear: Back (Cámara trasera), Front (Camara frontal). En modo default este método prioriza la cámara trasera sobre la frontal, excepto cuando el dispositivo presenta una sola cámara, la frontal. [35]

- Mirror Video Background: Esta propiedad el comportamiento de la imagen mostrada, según la orientación del dispositivo y la dirección de cámara seleccionada. Según el

artículo [35], la opción default prioriza la configuración ON y OFF, si es usada la cámara delantera o trasera respectivamente. Además se recomienda dejar el parámetro por defecto. Así mismo en [36], expresa que existen cuatro posibles casos posibles en función de las configuraciones de dirección de la cámara y fondo espejo:

1. Cámara delantera con espejo OFF: Mostrara imagen invertida en modo vertical (Dispositivo móvil), y texto sin inversión.

2. Cámara delantera con espejo ON: Se espera un funcionamiento correcto de la orientación de todos los elementos mostrados tanto de la imagen y texto sin importar la orientación del dispositivo.

3. Cámara trasera con espejo ON: Imagen invertida en orientación vertical, además de texto reflejado.

4. Cámara trasera con espejo OFF: Comportamiento esperado sin importar la rotación u orientación del dispositivo.

- World Center Mode: Define que instancia u objeto dentro (Camara, marcadores), será empleado como la referencia (0,0,0) del sistema de realidad aumentada, está presente 4 posibles opciones: [33] [37]

1. Specific_target: Opción que sujeta la referencia del sistema a un objeto especificado durante el desarrollo del sistema.

2. First Target: Opción que configura la referencia 0 del sistema al primer marcador encontrada por el dispositivo.

3. Camera: En esta opción, la cámara del dispositivo, será la referencia cero del sistema de realidad aumentada.

4. Device tracking: Esta opción es usada cuando el dispositivo o infraestructura empleada es una pantalla óptica transparente (Visor), o una pantalla de superposición de imágenes (VR Googles). [38]

La configuración empleada para este la primera prueba es:

1. Camera:

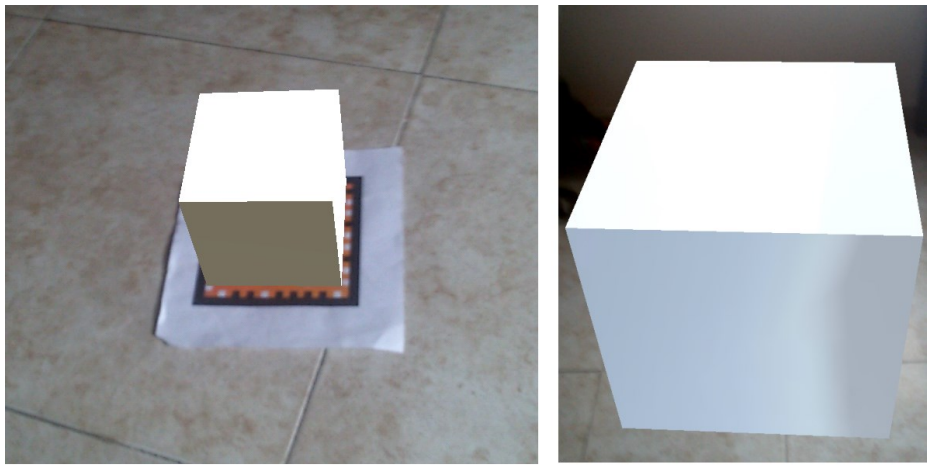
- ID Marker: 0

2. ARCamera:

- Camera device Mode: Default_Mode.
- Camera direction: Default_Mode.

- Mirror Video Background: Default_Mode.

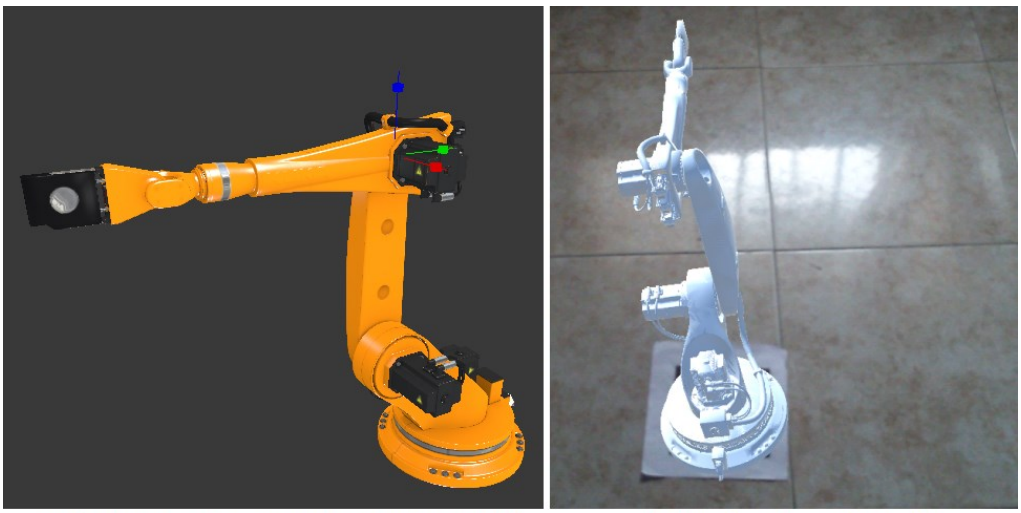
- World Center Mode: Camera.



a) Escala modelo: 0.8 a) Escala modelo: 4

Ilustración 99: Resultados primera prueba, Fuente: Autor

En este primer ensayo se realiza una segunda prueba, empleando las mismas configuraciones y tipo de marcador, pero tomando como modelo, un robot industrial descargado de: [39], y licenciado bajo dominio público. Esta se realiza para confirmar el rendimiento de la aplicación con modelos con gran cantidad de detalles; y la compatibilidad del motor gráfico de mapas UV de Blender con Unity.



a) Modelo original Blender con mapa UV, descargado de www.blendswap.com b) Resulta del sistema RA

Ilustración 100: Resultados segunda prueba, Fuente: Autor

4.3.2. Segundo ensayo

El objetivo del segundo ensayo era determinar el comportamiento de las simulaciones realizas en Blender en el entorno Unity; determinar la compatibilidad entre los motores gráficos de materiales de Unity y Blender; considerar los ImageTarget como tipo de seguimiento.

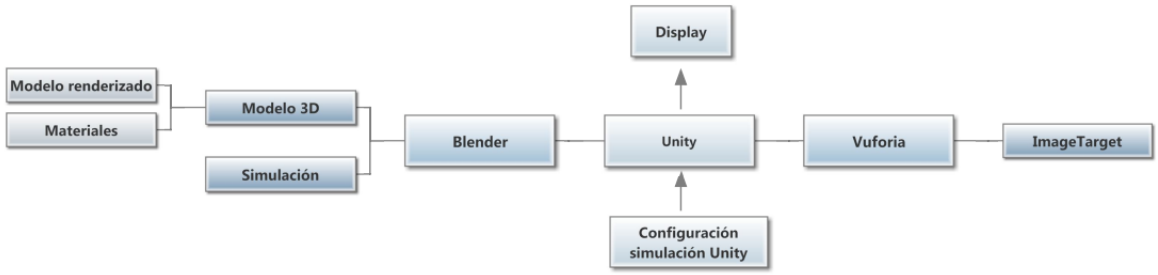


Ilustración 101: Diagrama conceptual segundo prototipo, Fuente: Autor

Para este ensayo se emplearon dos modelos 3D, y simulaciones:

- Mazo con punto de pivote en el mango: A cada elemento, mango, mazo, soporte, y cilindro se le asignó un material, en el motor gráfico interno de Blender. Así mismo, se realizó una animación básica. En esta el mazo gira ciertos grados alrededor de su pivote hasta golpear un cilindro de vidrio.

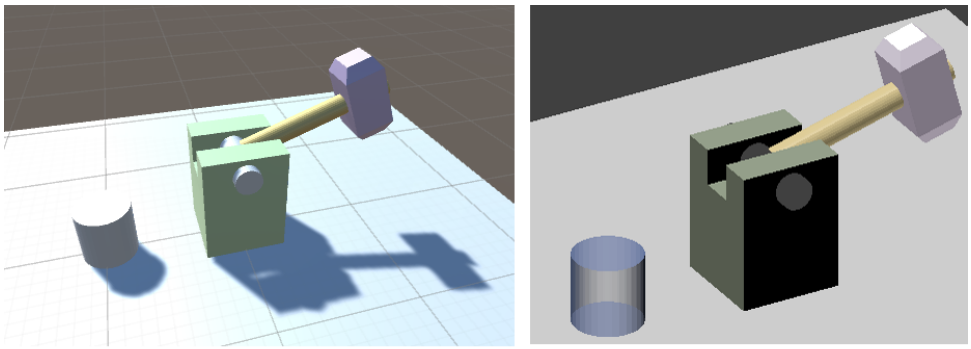


Ilustración 102: Motor gáfico Unity Vs Blender, fuente: Autor

Unity, reconoce los materiales asignados en Blender, cuando el modelo y animación es realizado con el moro gráfico Blender render (Interno); el motor de Unity, reconoce estos materiales y los procesa de una forma diferente, obteniendo un resultado más realista para el mismo diseño. Además Unity cuenta con herramientas que permiten realizar ajustes sobre estos materiales, que permiten obtener un efecto más fotorealista.

- Raplim: El modelo y simulación Raplim empleado, es el descripto en los apartados 4.2.1 y 4.2.2, del presente trabajo.

Unity, considera el modelo y la simulación como dos assets diferentes que deben enlazarse dentro de su entorno. Las formas posibles de enlazar estos dos elementos son: Sobreponer el asset simulación sobre el asset modelo, generando Unity un enlace automático entre los dos elementos; el segundo método es a través de programación, empleado la variedad de clases con las que cuenta Unity.

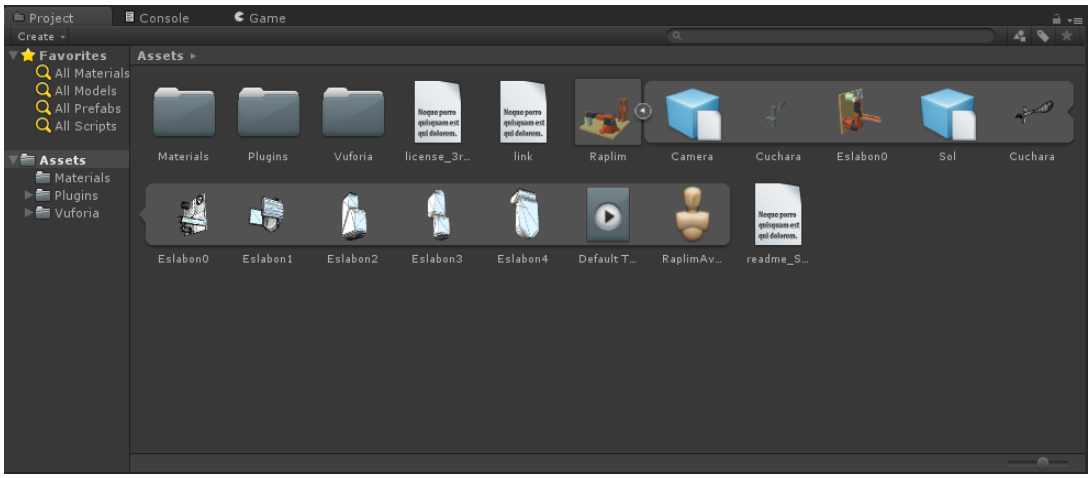


Ilustración 103: Biblioteca de assets proyecto - Pestaña project, fuente: Autor

El primer método se realiza arrastrando el asset Raplim hacia pestaña Hierarchy, esto cargara el modelo en el entorno de trabajo. Una vez realizado esto, se arrastra el asset simulación comúnmente llamado Default_Take encima de Raplim dentro del Hierarchy. Se genera un nuevo elemento de la clase Animator dentro del directorio de assets del proyecto. El cual indicara que ambos elementos han sido enlazados correctamente.

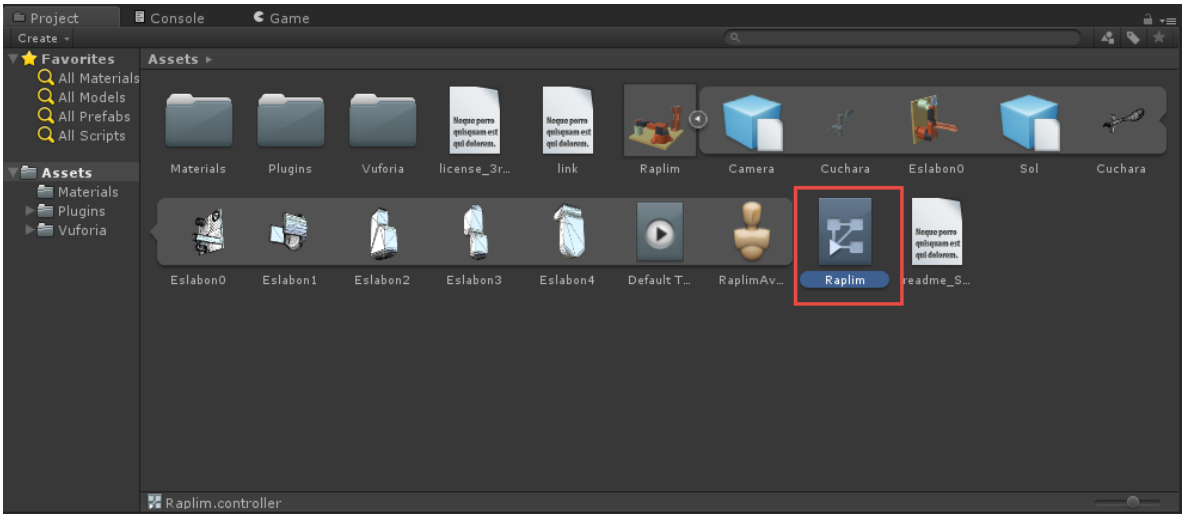


Ilustración 104: Enlace Asset Modelo-Simulación, fuente: Autor

El segundo método, se basa en algoritmos desarrollados en C# dentro de Blender, y en la aplicación de los métodos y propiedades de las clases Animation y Animator.

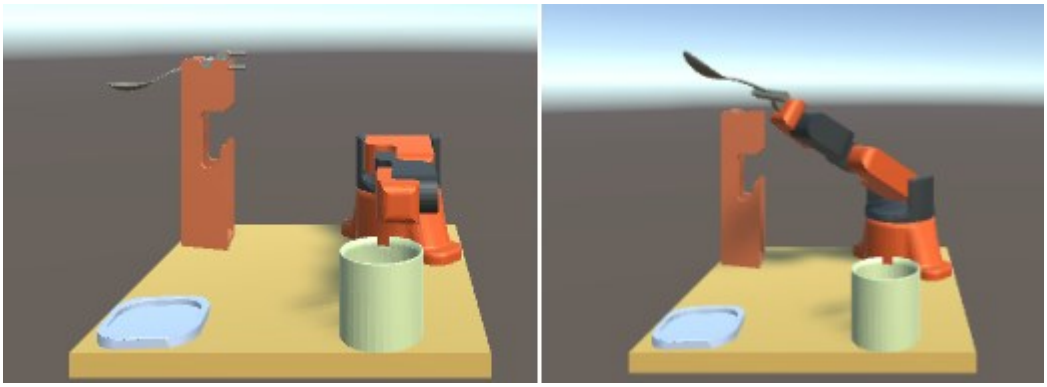


Ilustración 105: Animación Raplim en Unity, fuente: Autor

Con la animación de Raplim funcionando dentro del entorno de Unity, se procede a la configuración de Vuforia, tal como se ha expuesto anteriormente, con el propósito de visualizar las dimensiones, velocidad, calidad del mismo, durante la ejecución de la aplicación en la plataforma Android.

La primera prueba, se realiza con Raplim sin materiales, escala 4 veces el tamaño del target, imagen target: target marker y sin animación, con esta se pretender verificar la estabilidad, el tamaño y la máxima distancia a la que debería estar el observador del target.

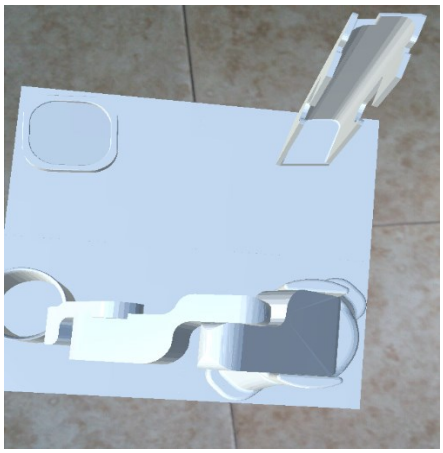


Ilustración 106: Raplim - Primera prueba RA, fuente: Autor

En esta prueba se empleó, como imagen marcador uno de los modelos propuestos por Unity, obteniendo como resultado, la incompatibilidad entre este método de rastreo e imágenes simétricas, o con características similares. Reduciendo en gran medida la distancia del operador al target, poca estabilidad, se presenta alteraciones intempestivas en el modelo, y orientación incorrecta.

En la segunda fase se emplea una imagen target con mayor cantidad de detalles, Raplim texturizado, simulación, y las mismas configuraciones de Vuforia. Con esta se pretende igualmente determinar la distancia, calidad, y estabilidad del sistema de realidad aumentada.

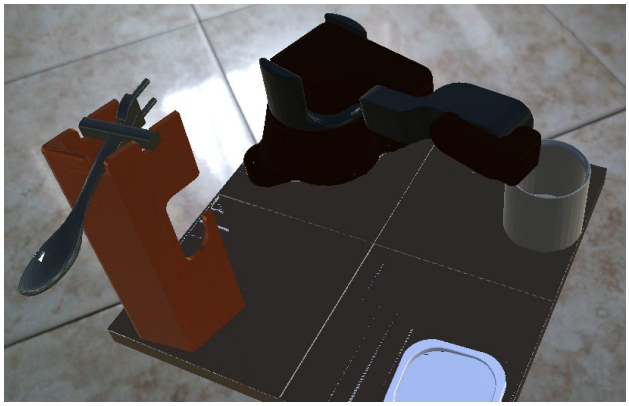


Ilustración 107: Raplim - Segunda prueba RA, fuente: Autor

En esta segunda prueba se verifico, que imágenes con gran cantidad de detalles permiten al algoritmo de rastreo detectar más eficazmente la imagen usada, mayor estabilidad del modelo, pero se determinan las siguientes restricciones: La distancia en primera detección del image target debe ser pequeña, una vez es detectado el sistema reconocerá la posición de ésta a mayores distancias; la orientación del objeto es afectado por reflejos sobre la image target, dobladuras sobre el papel impreso, u otras elevaciones del mismo.

En ésta también se emplea el motor de materiales de Unity, aplicándole más realidad a los ya implementados en Blender, obteniéndose un modelo más foto realista para la aplicación de RA.

4.3.3. Tercer ensayo

El tercer ensayo se realiza como puesta a punto del sistema de RA, en esta se introduce el audio de las simulaciones y las interfaces gráficas para la visualización de los valores angulares de las articulaciones y la posición del efector final.

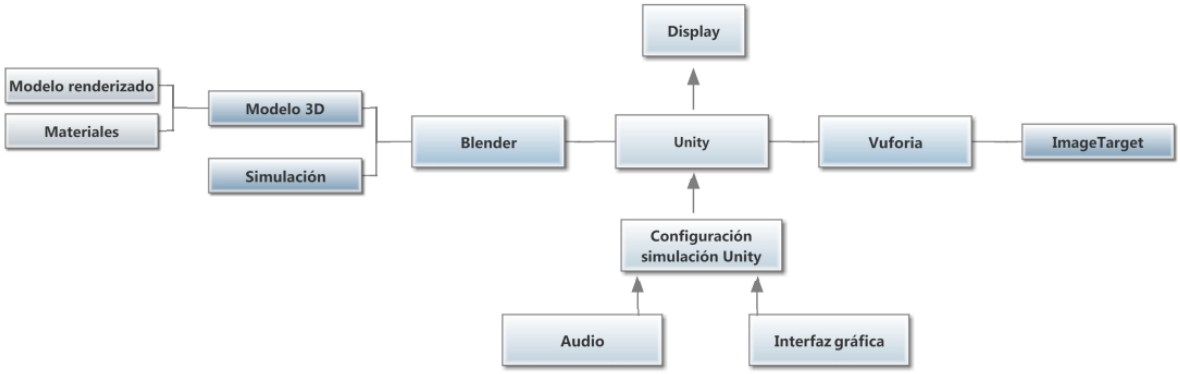


Ilustración 108: Diagrama conceptual tercer prototipo, fuente: Autor

La primera fase de este ensayo, es la incorporación de la interfaz gráfica, por medio de la cual se le presenta al observador el valor angular de cada articulación en tiempo real.

En aplicaciones desarrolladas en Unity se puede presentar información al observador de 2 maneras: La primera es empleado el GameObject 3D Text o a través de UI's desarrollados en el propio Unity.

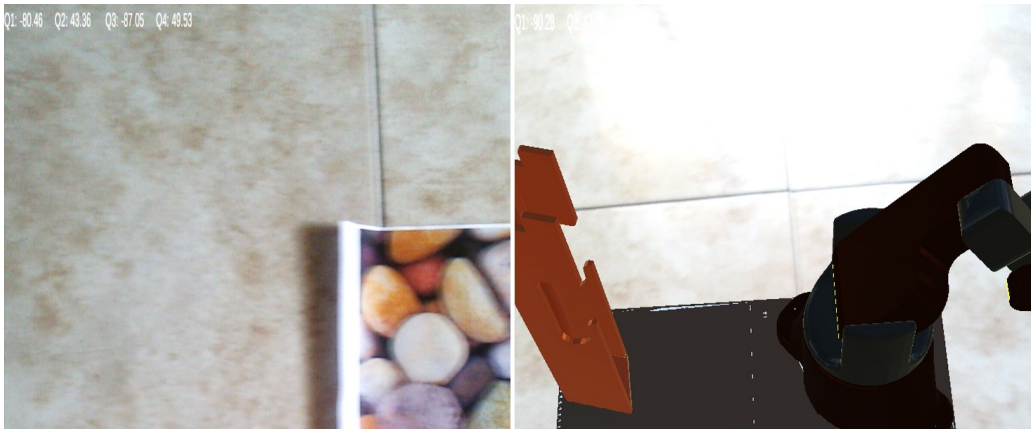


Ilustración 109: Primera Interfaz, fuente: Autor

La primera interfaz se realizó a través de un Skin integrado por 4 3D Text, los cuales corresponden a la variación angular de cada uno de las articulaciones de Raplim. Esta a su vez presenta dos sub-fases de desarrollo, en la primera, el skin es visualizado en todo momento desde el inicio de la aplicación, sin necesidad de ser encontrado el image Target.

La segunda sub-fase, es la implementación de un algoritmo que restringe la visualización de los valores angulares, siendo mostrados cuando el target es detectado y Raplim es visualizado a través de la aplicación, para ello se emplea un método el método de la clase Vuforia (OnTrackableStateChanged).

Ilustración 110: Segunda interfaz, fuente: Autor

Éste método solo puede ser implementado a través de códigos que generen y alteren la información de los 3DText. La alteración de las propiedades y campos de éstos, solo es posible a través de métodos preestablecidos, poco sofisticados y atractivos para el desarrollador.

Durante el desarrollo de estas interfaces, se detectaron dos errores: El primero, la visualización de la información en todo momento, estuviera o no presente el modelo en la pantalla de visualización, esto dio como resultado el desarrollo de la segunda interfaz; el segundo error encontrado, la reproducción automática de la simulación, esta iniciaba antes de ser encontrado el target, y el usuario no observaba los inicios de ésta.

Para los problemas encontrados durante el desarrollo de estas interfaces, se plantean las siguientes soluciones: primero, desarrollar la interfaz a través de la herramienta UI suministrada por Unity; desarrollar un algoritmo para controlar la reproducción de la simulación en función de la detección del marcador.

Se desarrollan dos interfaces, que permitan visualizar de formas más concisa la información que se quiere mostrar, para ello se desarrolla pastillas sobre las cuales se expondrán los valores angulares y la poción del efector final.

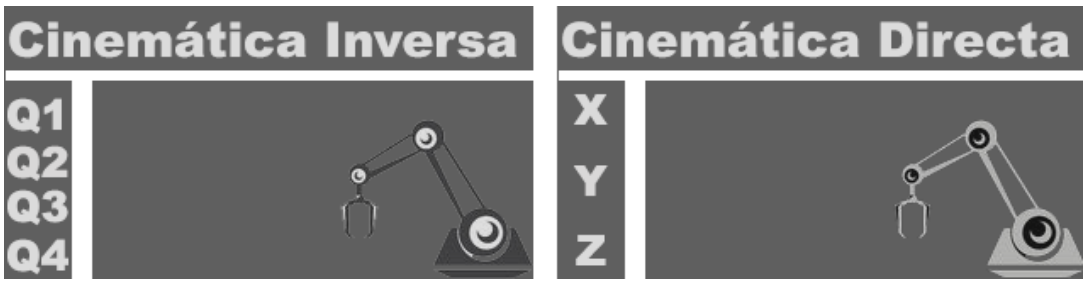


Ilustración 111: Interfaz final aplicativo, fuente: Autor

Estas interfaces se desarrollan del tal manera que permiten visualizar sobre y a través de ella, es decir, la posición que ocupan sobre la pantalla no bloquea totalmente la toma capturada por la cámara.

El funcionamiento de estas estos UI se basan en una plantilla vacía denominada CANVAS, sobre la cual se pueden integrar diferentes tipos de elementos según el tipo de menú o interfaz que se quiera desarrollar, entre las opciones que proporciona Unity está el texto, botones, imágenes, paneles, etc.

En el presente las dos interfaces gráficas se componen de 3 elementos: El canvas el cual está configurado de la manera que se adapte a las dimensiones de las pantallas, desfasado 100 unidades respecto a la posición de la cámara, las dimensiones de los elementos que lo componen serán constantes y presentaran la mayor calidad posible; una imagen vacía, esta referencia el aspecto de la interfaz desarrollado en editores de imágenes, caracterizados por su transparencia y adecuación al sistema, cada imagen vacía se desarrolló para mostrar sobre él los resultados del problema cinemático directo e inverso; el componente texto, es aquel componente variable de las interfaces que permitirán mostrar la información requerida.

Se selecciona este método sobre el anterior, dado que presenta una mayor gama de ajustes permitiendo una mejor ubicación y tratamiento sobre los elementos textos e imágenes a mostrar, generando una interfaz con aspecto más profesional y amigable.

Para el funcionamiento de estas interfaces se desarrollaron métodos de la clase inversa información, por medio de los cuales se les asigna en tiempo real los valores calculados de la cinemática directa (Posición efector final), y los valores adquiridos de la simulación a través de clases y métodos proporcionados por Unity.

Ilustración 112: Aplicativo con interfaz final, fuente: Autor

Con el desarrollo de la interfaz final se plantean, la integración de sonidos alusivos a los movimientos presentados durante la simulación y el desarrollo de una clase que permitiera controlar la ejecución de la simulación, para ello se desarrollan sonidos con duración respectiva a los movimientos de Raplim, en el software Audacity, basados en audios de robots industriales en funcionamiento; y métodos que permitieran pausar, continuar y e iniciar la simulación, respectivamente.

Para ello se tomó y categorizo los movimientos en función de su cantidad de keyframes, con el propósito de generar el sonido con la duración y tono adecuado para cada categoría. Con esta información y el número de keyframes por según (24 fps) se determinó la duración de cada audioclip.

Para el desarrollo del componente sonido para el aplicativo, se examinaron diferentes maneras, con el propósito de encontrar la que proporcionara el mejor resultado. La primera prueba realizada, consistió en montar el audio sobre la animación en la plataforma Blender, posteriormente exportada a Unity. Blender dispone de un editor de secuencia de video que permite integrar audio a las simulaciones desarrolladas en él. Luego de cargar y acomodar el audio sobre la simulación, posteriormente se exporto a Unity, con el propósito de ver los resultados, el desenlace de esta prueba comprobó que el Audio es un componente inexportable en Blender para el formato FBX, y tampoco es aceptado por el software Unity.

Para la siguiente prueba, se empelaron métodos y objetos de la clase Audio proporcionada por Unity, los cuales funcionan en base a un algoritmo que carga los audioclips previamente importados a la carpeta Assets del proyecto, al entorno de trabajo, estos son reproducidos cuando la animación llegan a ciertos valores de keyframes, que de antemano han sido especificados. El método de la clase desarrollada para la reproducción de los sonidos, tiene como parámetro de entrada, el número de keyframe, el cual genera la ejecución del código dentro del condicional que presente dicho valor. Para enviar este parámetro de entrada a la función primeramente, se elaboró un método que determinara en tiempo real el keyframe actual, para luego ser enviado a dicha función. Unity en la actualidad no cuenta con una clase que permita determinar el número de keyframe en reproducción, siendo necesario emplear métodos de tiempo como delta time, y otros para obtener una aproximación del frame actual, conllevando a resultados inapropiados, dado que se genera saltos entre keyframe, obviando una gran cantidad de ellos.

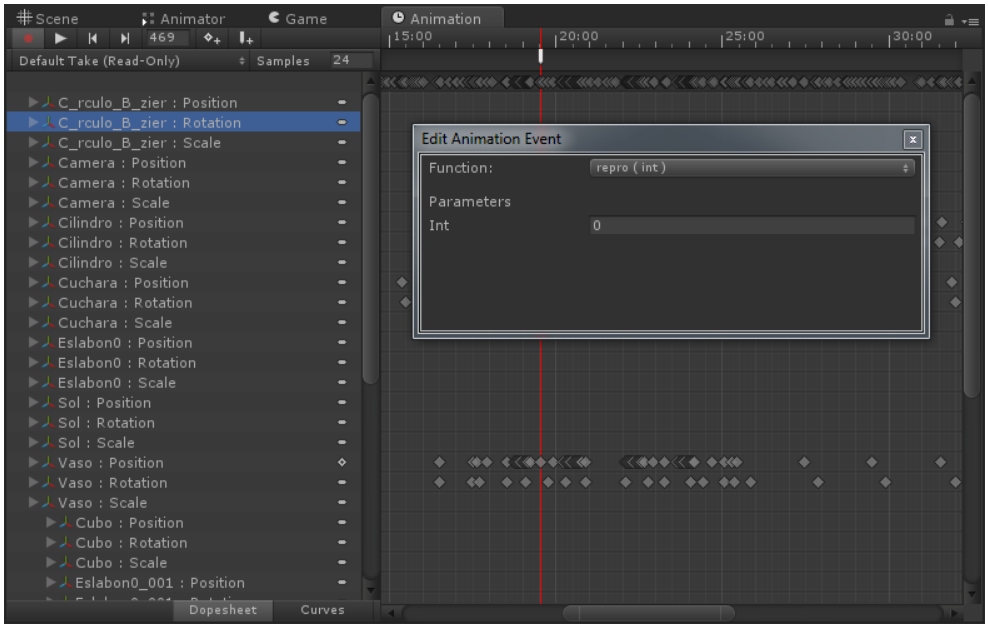


Ilustración 113: Componente Animation, fuente: Autor

Para mitigar este problema, Unity dispone de la herramienta Animatión, la cual permite generar eventos sobre el editor de simulaciones de Blender. Estos, básicamente funcionan como tiggers o disparadores para métodos y funciones desarrolladas en el proyecto. Estos eventos envían el valor especificado cuando la simulación pasa por dicho keyframe, generando un efecto más coherente entre la parte visual y auditiva del proyecto.

La clase generada para el control sobre la animación, es una clase derivada de la clase Animation de Unity. En esta se emplean métodos que permiten variar la velocidad de reproducción de la animación, de tal manera que genere el efecto de pausa y reproducción.

4.3.4. Clases C#

4.3.4.1. Clase inversa información

Ésta clase, es el algoritmo principal de funcionamiento del aplicativo de RA aumentada, a través de éste se envía la información hacia las interfaces gráficas, se emplean los métodos de las demás clases desarrolladas, etc.

```
using UnityEngine;
using Vuforia;
using UnityEngine.UI;


public class Inversa_Información : MonoBehaviour, ITrackableEventHandler
{
    public Activaranimacion activar; //  Construcutor clase activaranimacion
    public Animation ani2;


    // Variables auxiliares texto GUI
    public Text Q1;  // Declarando variables para los Text del GUI
    public Text Q2;
    public Text Q3;
    public Text Q4;
    public Text x;
    public Text y;
    public Text z;
    // Variables auxiliares animación
    public int n=0;
    //public int i;
    //public float k;
    // variables auxiliares para posición angular
    public float Eslabon1 = 0; // Declarando variables para almacenar los valores angu
lares de cada articulación
    public float Eslabon2 = 0;
    public float Eslabon3 = 0;
    public float Eslabon4 = 0;

    // Variables auxiliares posición efecto final
    public double[] xyz = new double[] {0,0,0};
```

```
   private bool mShowGUIButton = false; // Variable de informa si la target es detectad
o o no

void Start()
   {
      mTrackableBehaviour = GetComponent<TrackableBehaviour>();
      if (mTrackableBehaviour)
      {
         mTrackableBehaviour.RegisterTrackableEventHandler(this);
      }
      visualizar (); // Iniciando parametros iniciales de los GUI Text
      visualizar_2 ();
      //i = 0;
   }

   public void OnTrackableStateChanged(            // Metodo vuforia para saber si el tar
get es detectado o no
   TrackableBehaviour.Status previousStatus,
   TrackableBehaviour.Status newStatus)
   {
      if (newStatus == TrackableBehaviour.Status.DETECTED ||
      newStatus == TrackableBehaviour.Status.TRACKED)
      {
         mShowGUIButton = true; // target detectado
         n = n + 1;
      }
      else
      {
         mShowGUIButton = false; // target no detectado
      }
   }

   void Update ()
   {

      if (mShowGUIButton) // Condicional para determinar el estado de bandera ( Dete
ctar target)
      {
         //contador ();
         activar.animacion (n);
         obtenerq ();
```

```
            visualizar ();
            visualizar_2 ();
            xyz=dh.obtenerorientacion ();
        }
        if(!mShowGUIButton)
        {
            activar.animacion2 ();
        }
    }
    void visualizar () // Función o método para enviar información a los GUI Text
{
            Q1.text = "" + Eslabon1.ToString ()+"°";
            Q2.text = "" + Eslabon2.ToString ()+"°";
            Q3.text = "" + Eslabon3.ToString ()+"°";
            Q4.text = "" + Eslabon4.ToString ()+"°";
}

    void visualizar_2 ()
{
        x.text = "" + xyz [0].ToString ("N1");
        y.text = "" + xyz [1].ToString ("N1");
        z.text = "" + xyz [2].ToString ("N1");
}
    void obtenerq () // función o método para obtener los ángulos de las articulaciones en ti
empo real
{
            Eslabon1 = GameObject.FindGameObjectWithTag ("Eslabon1").transform.localE
ulerAngles.z;
            if (Eslabon1 > 180) {
                Eslabon1 = Eslabon1 - 360;
            }
            Eslabon1 = Mathf.Round (Eslabon1 * 1f) / 1f;
            Eslabon2 = GameObject.FindGameObjectWithTag ("Eslabon2").transform.localE
ulerAngles.x;
            if (Eslabon2 > 180) {
                Eslabon2 = Eslabon2 - 360;
            }

            Eslabon2 = Mathf.Round (Eslabon2 * 1f) / 1f;
            Eslabon3 = GameObject.FindGameObjectWithTag ("Eslabon3").transform.localE
ulerAngles.x;
```

```
        if (Eslabon3 > 180) {
            Eslabon3 = Eslabon3 - 360;
        }
        Eslabon3 = Mathf.Round (Eslabon3 * 1f) / 1f;
        Eslabon4 = GameObject.FindGameObjectWithTag ("Eslabon4").transform.localE
ulerAngles.x;
        if (Eslabon4 > 180) {
            Eslabon4 = Eslabon4 - 360;
        }
        Eslabon4 = Mathf.Round (Eslabon4 * 1f) / 1f;
}

}
```

4.3.4.2. Clase dh

Esta clase es la encarga de realizar el cálculo de la cinemática directa, y de todas las operaciones necesarias para la mismas.

```
using UnityEngine;
using System;


public static class dh  {
    // Definiendo funcion de la clase
    ///<summary>
    /// Clase que representa la cinemática directa de un robot antropomofico
    ///</summary>

////////////////////////////////////////////////////////////////////////////// Definiendo metodos

 //// Metodo 1

/// <summary>
/// Multiplicacion entre matrices: Se cálcula la multiplicacion de matrices cuatro por cu
atro
/// </summary>
/// <param name="m1">M1.</param>
```

```
/// <param name="m2">M2.</param>
   public static double[,] multiplicacion (double[,] m1 , double[,] m2)
   {
   double[,] matrizRespuestas = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0
}};// Creando la matriz resultado
       for (int i=0;i<=3;i++)
       {
          for (int j=0;j<=3;j++)
          {
             for (int k=0;k<=3;k++)
             {
                matrizRespuestas [i, j] = matrizRespuestas [i, j] + (m1 [i, k] * m2 [k, j]);
             }
          }
       }
       return matrizRespuestas;
   }

/// <summary>
/// Denavit: cálcula las matrices de transformacion homogenea para diferentes sistem
as, ingresando los parametros de denavit
/// </summary>
/// <param name="p1">P1.</param>

       public static double[,] denavit (double[] p1)
   {

   double[,] mdh = new double[,]  {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};// Creand
o la matriz resultado m. denavit

       mdh [0, 0] =Math.Cos (p1 [0]);
       mdh [0, 1] =-1*Math.Sin(p1[3])*Math.Cos(p1[0]);
       mdh [0, 2]= Math.Sin(p1[3])*Math.Sin(p1[0]);

       mdh [0, 3]= p1[2]*Math.Cos (p1 [0]);
       mdh [1, 0] = Math.Sin (p1 [0]);
       mdh [1, 1] = Math.Cos (p1 [3]) * Math.Cos (p1 [0]);
       mdh [1, 2]= -1*Math.Sin (p1 [3])* Math.Cos (p1 [0]);
```

```
        mdh [1, 3]= p1[2]*Math.Sin (p1 [0]);
        mdh [2, 0] = 0;
        mdh [2, 1]= Math.Sin (p1 [3]);
        mdh [2, 2]= Math.Cos (p1 [3]);
        mdh [2, 3] = p1 [1];
        mdh [3, 0] = 0;
        mdh [3, 1] = 0;
        mdh [3, 2] = 0;
        mdh [3, 3] = 1;
        return mdh;
    }

//// Metodo 3
/// <summary>
/// Obteniede las coordenada (X,Y,Z) del efector final, tomada de la matriz de tansfor
macion denavit A04
/// </summary>


    public static double[] obtenerorientacion ()
    {
    double pp1 = GameObject.FindGameObjectWithTag ("Eslabon1").transform.localE
ulerAngles.z;
    double pp2 = GameObject.FindGameObjectWithTag ("Eslabon2").transform.localE
ulerAngles.x;
    double pp3 = GameObject.FindGameObjectWithTag ("Eslabon3").transform.localE
ulerAngles.x;
    double pp4 = GameObject.FindGameObjectWithTag ("Eslabon4").transform.localE
ulerAngles.x;
    double[] p1 = new double[]  {Deg2rad(pp1)+ Deg2rad(90),13.49,0,Deg2rad(90)};
    double[] p2 = new double[]  {Deg2rad(pp2), 0, 12.02, 0 };
    double[] p3 = new double[]  {Deg2rad(pp3),0,9.45,0};
    double[] p4 = new double[]  {Deg2rad(pp4),0,6,4,0};

    double[,] A00 = new double[,] {{1,0,0,0},{0,1,0,0},{0,0,1,0},{0,0,0,1}};
    double[,] A01 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
```

```
    double[,] A02 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
    double[,] A03 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
    double[,] A04 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
    double[,] A12 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
    double[,] A23 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};
    double[,] A34 = new double[,] {{0,0,0,0},{0,0,0,0},{0,0,0,0},{0,0,0,0}};

        A01 = denavit (p1);
        A12 = denavit (p2);
        A23 = denavit (p3);
        A34 = denavit (p4);

        A01 = multiplicacion (A00, A01);
        A02 = multiplicacion (A01, A12);
        A03 = multiplicacion (A02, A23);
        A04 = multiplicacion (A03, A34);

        double x= (A04 [0, 3])*1f/1f;
        double y=(A04 [1, 3])*1f/1f;
        double z=(A04 [2, 3])*1f/1f;

        double [] xyz = new double[3];
        xyz [0] = x;
        xyz [1] = y;
        xyz [2] = z;
        return xyz;
    }
    public static double Deg2rad (double x)
    {
        double y;
        y = (x * Math.PI) / 180;
        return y;
    }
}
```

4.3.4.3.Clase Activar animación

Clase encargada de la reproducción y control de las animaciones.

```
using UnityEngine;


public class Activaranimacion : MonoBehaviour
{

   public Animation ani;

   public void animacion (int n) // Play y Continue animacion
   {

      if (n <= 1) {
         ani ["Default Take"].speed = 1;
         GetComponent<Animation> ().Play ();
         print ("Play");
      }
      if (n > 1) {
         ani ["Default Take"].speed = 1;
         print ("Continua");
         }
   }
   public void animacion2 (){ // Stop animacion
      ani ["Default Take"].speed = 0;
      print ("Stop");
}


}
```

4.3.4.4.Clase Sonido2

Clase encargada de reproducir el audioclip, dependiendo del número de keyframe actual en la simulación.

```
public class Sonido2 : MonoBehaviour {

    public AudioClip s5f;
    public AudioClip s10f;
    public AudioClip s20f;
    public AudioClip s30f;
    public AudioClip s40f;
    public AudioClip s50f;
    AudioSource fuenteSonido;

    // Use this for initialization
    void Start () {
        fuenteSonido = GetComponent<AudioSource> ();
    }

    void repro (int f)
    {
            if (f == 5)
        {
            fuenteSonido.clip = s5f;
            fuenteSonido.Play ();
        }
            if (f == 10)
        {
            fuenteSonido.clip = s10f;
            fuenteSonido.Play ();
        }
            if (f == 20)
        {
            fuenteSonido.clip = s20f;
            fuenteSonido.Play ();
        }
```

```
        if (f == 30)
        {
            fuenteSonido.clip = s30f;
            fuenteSonido.Play ();
        }
        if (f == 40)
        {
            fuenteSonido.clip = s40f;
            fuenteSonido.Play ();
        }
        if (f == 50)
        {
            fuenteSonido.clip = s50f;
            fuenteSonido.Play ();
        }

    }

}
```

RESULTADOS

El desarrollo de RA Raplim conllevo una serie de fases que permitieron determinar las características, propiedades, y configuraciones tanto del modelo 3D, las simulaciones y los softwares implicados en el desarrollo del aplicativo de realidad aumentada, que consiguieran los mejores resultados posibles en el aspecto, velocidad de ejecución, y funcionalidad.

El desarrollo del modelo 3D, fue una progresión desde la parametrización y adecuación de cada eslabón y su posición durante el ensamble de Raplim. Así mismo se asignaron texturas y materiales a cada componente dentro del modelo, que proporcionaran un aspecto con la suficiente realidad para obtener una realidad mixta coherente entre los objetos virtuales y reales del aplicativo.

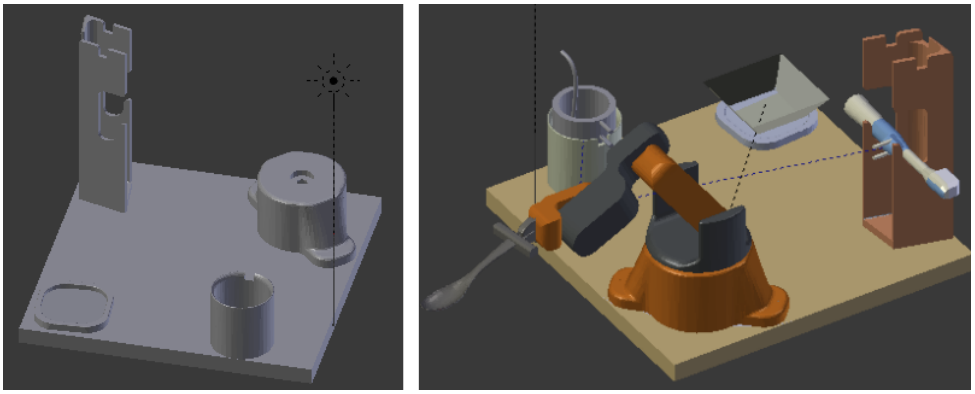


Ilustración 114: Evolución del modelo 3D, fuente: Autor

La producción de las rutinas de Raplim, implico el desarrollo de diferentes efectores finales, y un algoritmo para determinar los valores angulares necesarios para que el efector final se posicionara sobre los puntos de interés, tales como la ubicación de la cuchara, el cepillo y el vaso. Así como la configuración de las herramientas que Blender proporciona para el desarrollo de animaciones dentro de su entorno de trabajo.

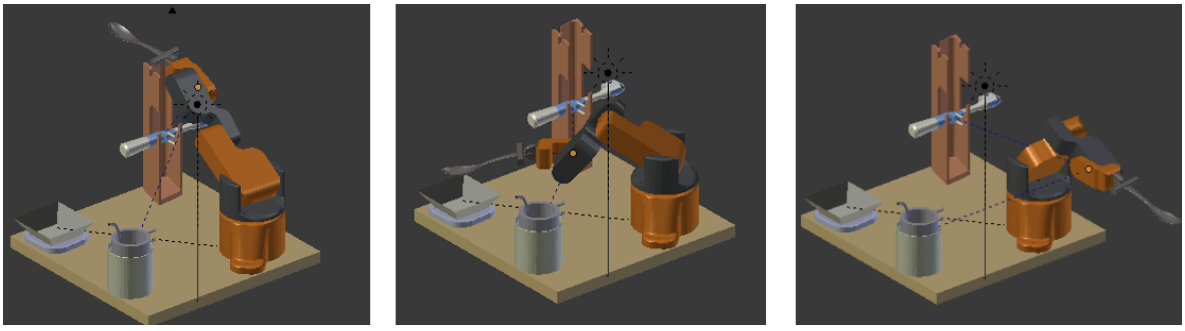


Ilustración 115: Secuencia cuchara entorno Blender, fuente: Autor

El desarrollo del modelo CAD y las simulaciones del mismo, fueron las primeras fases para el desarrollo del aplicativo; posteriormente se establecieron las configuraciones de Unity y Vuforia, que lograran obtener una velocidad de ejecución, detección, y aspecto semi-profesional, aceptable para la publicación de RA Raplim, y que eliminaran la mayoría de defectos y problemas encontrados durante el desarrollo, tales como la distancia entre el marcador y el usuario; y la vibración del modelo en la realidad mixta.

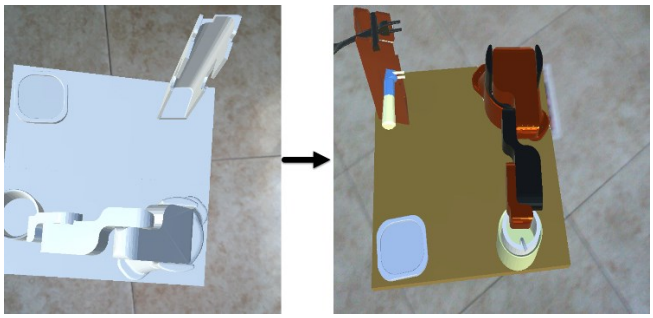


Ilustración 116: Evolución del aplicativo de RA, fuente: Autor

Para la versión final del aplicativo, se trabajaron aspectos tan básicos como el icono del aplicativo, y tan fundamentales como las interfaces de visualización. Así mismo una vez cumplidos los objetivos principales planteados, se planea la integración de audios alusivos a los movimientos de Raplim, que eliminaran el entorno plano que llevaba hasta el momento el aplicativo.

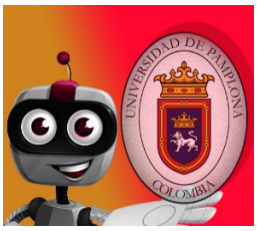


Ilustración 117: Icono RA Raplim

En el desarrollo de las interfaces de visualización, se comprobaron cada una de los métodos proporcionados por Unity para la visualización de información, y se selecciona aquel que concediera el mejor aspecto y calidad visual al usuario.

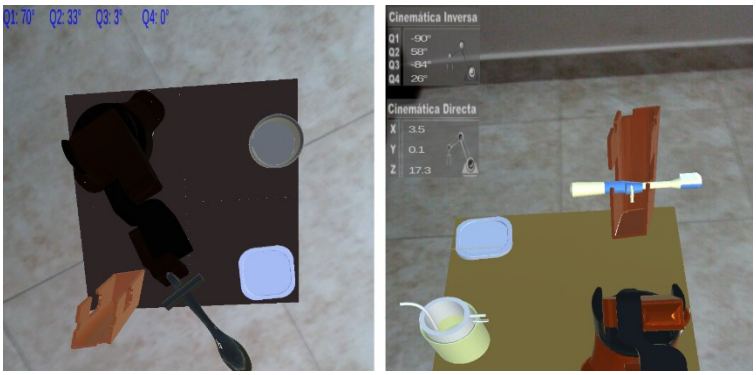


Ilustración 118: Evolución interfaces de visualización, fuente: Autor

El resultado final y las secuencias que observara el usuario, se presenta en la siguiente ilustración:

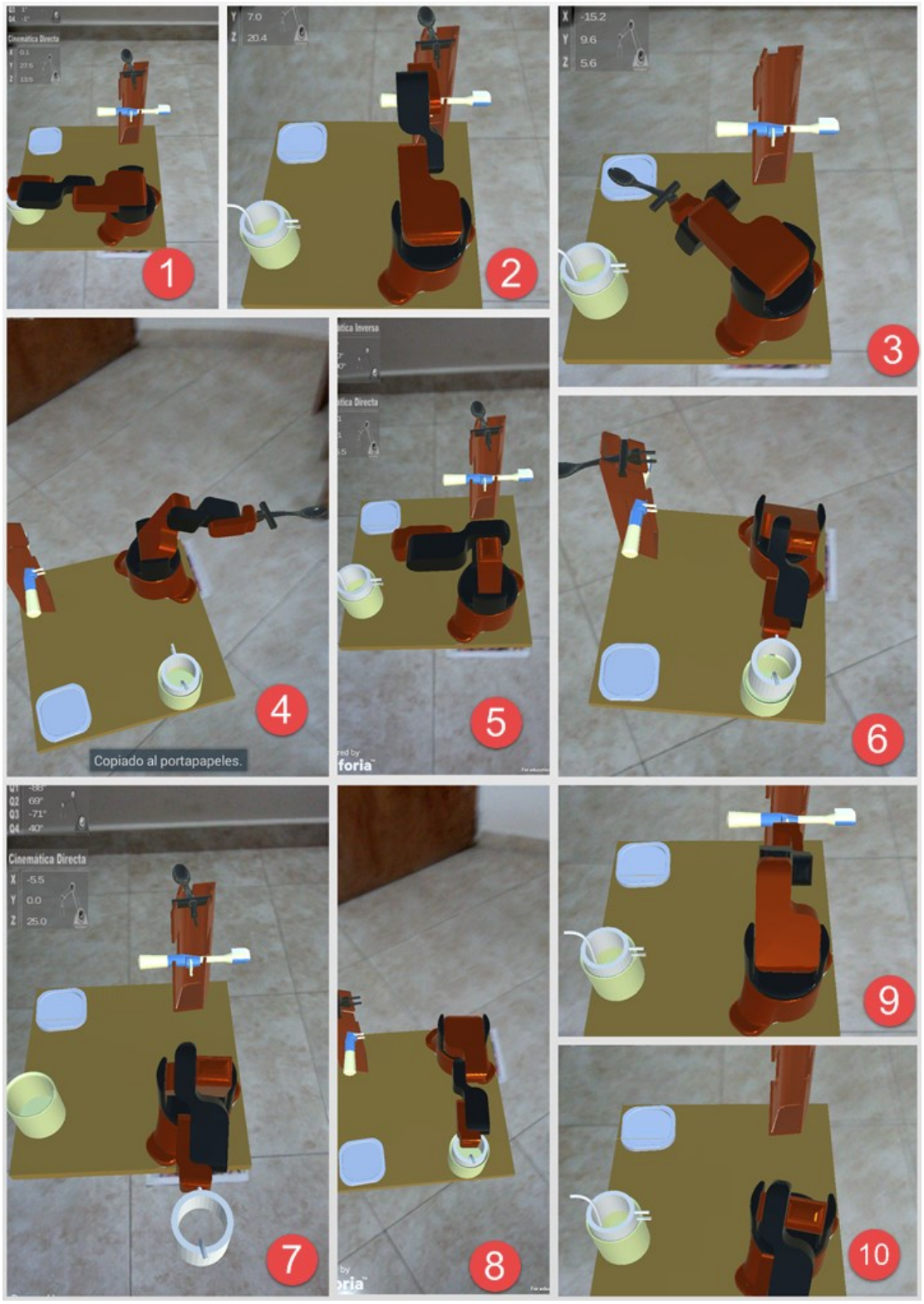


Ilustración 119: Secuencias en aplicativo RA Raplim

CONCLUSIONES

Como se ha mencionado anteriormente en el presente trabajo, la realidad aumentada es una tecnología en apogeo, que recrea una realiza mixta superponiendo información virtual y real, que será presentada al usuario a través de diferentes medios de visualización. La cual tiene un gran potencial para ser la tecnología con mayor campo de aplicación en áreas como educación, industria y demás, ya que sin tener en cuenta el soporte físico o hardware, está limitada únicamente a la creatividad e imaginación del ser humano, dado que a través de esta se pueden realizar infinitos entornos y contenidos para cualquier área de aplicación.

La implementación de la realidad aumentada en la educación desde sus inicios hasta la formación profesional desarrollaría en los estudiantes un mayor interés por el conocimiento, los alcances y posibilidades de éste; incentivaría la creatividad y entendimiento en temas que requieren de modelos para su explicación.

El desarrollo de una aplicación de RA aumenta, no conlleva grandes costos iniciales para el ámbito educativo, ya que en la actualidad se cuenta con una gran variedad de softwares que permiten desarrollar cada uno de los elementos (SDK de realidad aumentada, entornos de trabajo, modelado 3D, etc.) necesarios para implementar un sistema RA, con resultados casi profesionales y atractivos para los consumidores o beneficiarios.

Vuforia es un SDK completo, que gracias a su distribución gratuita, comunidad de desarrollo y su constante mejoría, lo conviertes en excelente paquete de desarrollo de esta tecnología, a su vez permite generar sistemas de realidad aumentada de gran calidad, y para una variedad de infraestructuras, permitiendo la implementación tanto en dispositivos móviles (Tablet, Smartphone) como en infraestructuras más avanzadas como gafas de realidad aumentada. Cuenta con la desventaja que sus marcadores no deben ser obstruidos, y estar sobre una superficie plana y en hojas no corrugadas, para no obtener resultados indeseados.

Blender es un entorno de desarrollo de modelado y animación tridimensional que cuenta con herramientas y aplicativos sofisticados para realizar trabajos con aspecto profesional y en lapsos de tiempo reducidos. Pero que deben emplearse adecuadamente para obtener los resultados queridos, ya que este, emplea algoritmos o métodos que aun con su grado de desarrollo, se pueden generar errores, debido a su mala implementación, esto se pudo observar en la interpolación automática durante el desarrollo de las simulaciones. En las cuales por la falta de asignación de puntos de interés, se presentaban colisiones entre elementos.

Unity es un entorno de trabajo interactivo, y amigable con el usuario, que permite realizar infinidad de trabajos gracias a su compatibilidad con diferentes formatos, extensiones y plataformas de desarrollo, entre ellas Android, convirtiéndolo en una gran herramienta para aquellos entusiastas y desarrolladores de video juegos y sistemas de realidad aumentada.

Ra Raplim se puede considerar una aplicativa de realidad aumentada, ya que cuenta con los elementos mínimos que según Azuma debe contener un sistema de realidad aumentada para

considerarse como tal. Cuenta con una realidad mixta, genera a través de Unity y Vuforia, encargados de superponer y proyectar al usuario la sinergia entre el contenido virtual y real.

Es una aplicación interactiva, ya que el usuario observa como su entorno y el modelo participa activamente en la realidad mixta, ya que cualquier alteración, intervención externa sobre el marcador, u acercamiento de objeto o persona al modelo, será visto en tiempo real por el usuario de la aplicación.

Cuenta con un contenido virtual en 3D: Raplim, el cual fue modelado y animado empleado el entorno de trabajo Blender, el cual posteriormente es exportado y procesado por Unity y Vuforia para la creación de la realidad mixta.

Bibliografía

[1] Harvard Business Review, «The Mainstreaming of Augmented Reality: A Brief History,» 2016. [En línea]. Available: https://hbr.org/2016/10/the-mainstreaming-of-augmented-reality-a-brief-history. [Último acceso: 01 Marzo 2017].

[2] AUGMENT, «Infographic: The History of Augmented Reality,» 2016. [En línea]. Available: http://www.augment.com/blog/infographic-lengthy-history-augmented-reality/. [Último acceso: 01 Marzo 2017].

[3] Informit - Pearson, «Introduction to Augmented Reality,» 2016. [En línea]. Available: http://www.informit.com/articles/article.aspx?p=2516729&seqNum=2. [Último acceso: 02 Marzo 2017].

[4] Universidad de la Laguna, «Entorno de aprendizaje ubicuo con realidad aumentada y tabletas para estimular la comprensión del espacio tridimensional,» 2013. [En línea]. Available: http://revistas.um.es/red/article/view/234041/179811. [Último acceso: 20 Marzo 2017].

[5] Vienna University of Technology , «Construct3D - An Augmented Reality System for Mathematics and Geometry Education,» [En línea]. Available: https://www.ims.tuwien.ac.at/projects/construct3d. [Último acceso: 20 Marzo 2017].

[6] C. D. C. Arzola, «Realidad aumentada utilizando un iPad,» [En línea]. Available: https://www.cs.cinvestav.mx/TesisGraduados/2013/TesisCesarCorona.pdf. [Último acceso: 25 Marzo 2017].

[7] M. y. H. H. D. Caro Martínez, «Realidad aumentada para el Museo de América,» [En línea]. Available: http://eprints.ucm.es/32915/1/Realidad aumentada para el Museo de América.pdf. [Último acceso: 28 Marzo 2017].

[8] A. G. Marín, «Diseño e implementación de aplicaciones móviles para la imagen de marca de una empresa,» [En línea]. Available: https://upcommons.upc.edu/bitstream/handle/2099.1/20782/memoria.pdf?sequence=1. [Último acceso: 08 Abril 2017].

[9] C. M. Lopez, «Probador virtual sin marcadores,» [En línea]. Available: https://rua.ua.es/dspace/bitstream/10045/40271/1/Espejo_Virtual_MECA_LOPEZ_CARLOS.pdf. [Último acceso: 09 Abril 2017].

[10] R. T. Azuma, «A Survey of Augmented Reality,» [En línea]. Available: http://www.cs.unc.edu/~azuma/ARpresence.pdf. [Último acceso: 10 Abril 2017].

[11] Fundación telefónica, «Realidad Aumentada: una nueva lente para ver el mundo,» [En línea]. Available: https://books.google.es/books?hl=es&lr=&id=OXHmCgAAQBAJ&oi=fnd&pg=PA10&dq=realidad+aumentada&ots=3qu5U-elv9&sig=DgUbH7rYOh8tFswt_soumhjJPwI#v=onepage&q&f=false. [Último acceso: 15 Abril 2017].

[12] Tecsmedia, «Análisis: Realidad Aumentada aplicada a entornos industriales,» 2015. [En línea]. Available: http://www.aragon.es/estaticos/GobiernoAragon/Departamentos/InvestigacionInnovacionUniversidad/Areas/Sociedad_Informacion/Documentos/Estado del arte de Realidad Aumentada.pdf. [Último acceso: 15 Abril 2017].

[13] V. R. González, «Introducción a la Robótica,» 2002. [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/. [Último acceso: 15 Abril 2017].

[14] M. R. F. S. B. J. P. R. J. E. F. E. F. S. V. M. H. Sánchez Martín FM, «Historia de la robótica: de Arquitas de Tarento al robot Da Vinci (Parte I),» [En línea]. Available: http://scielo.isciii.es/pdf/aue/v31n2/original1.pdf. [Último acceso: 16 Abril 2017].

[15] V. R. González, «Origen y desarrollo de la Robótica,» [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0204/cyr_01/robotica/historia.htm. [Último acceso: 19 Abril 2017].

[16] V. E. Miranda, «Modelado y simulación del robot PASIBOT : estudio de la rigidez y mejora en la prevención del vuelco lateral,» 2009. [En línea]. Available: http://e-archivo.uc3m.es/bitstream/handle/10016/7792/PFC Victor_Espantoso_Miranda.pdf?sequence=1. [Último acceso: 16 Abril 2017].

[17] M. K. R. Benavides, «Cinemática inversa,» [En línea]. Available: http://www.kramirez.net/Robotica/Material/Presentaciones/CinematicaInversaRobot.pdf. [Último acceso: Abril 25 2017].

[18] J. S. P. M. R. F. S.-B. J. M. F. V. P. R. J. V. M. H. Sánchez-Martín F.M., «Historia de la robótica: de Arquitas de Tarento al Robot da Vinci (Parte II),» [En línea]. Available: http://scielo.isciii.es/scielo.php?script=sci_arttext&pid=S0210-48062007000300002. [Último acceso: 26 Abril 2017].

[19] A. L. C. J. A. C. E. Víctor R. González Fdez, «Robots industriales,» [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0708/archivos/_15/Tema_5.4.htm. [Último acceso: 20 Abril 2017].

[20] Universidad de Huelva, «Conceptos y definiciones en cinemática,» [En línea]. Available: http://www.uhu.es/rafael.sanchez/ingenieriamaquinas/carpetaapuntes.htm/Apuntes Tema 1.pdf. [Último acceso: 20 Abril 2017].

[21] Grupo AUROVA, «Cinemática Directa. Concepto teórico,» [En línea]. Available: http://www.aurova.ua.es/robolab/EJS2/RRR_Intro_2.html. [Último acceso: 20 Abril 2017].

[22] Proyectos róboticos, «Cinemática Inversa II,» [En línea]. Available: https://sites.google.com/site/proyectosroboticos/cinematica-inversa-ii. [Último acceso: 25 Abril 2017].

[23] TecsMedia, «Análisis: Motores gráficos y su aplicación en la industria,» [En línea]. Available: http://www.aragon.es/estaticos/GobiernoAragon/Departamentos/InvestigacionInnovacionUniversidad/Areas/Sociedad_Informacion/Documentos/Estado del arte GameEngines y su impacto en la industria.pdf.

[24] Blender, «Descripción del Motor de Juegos de Blender,» [En línea]. Available: https://wiki.blender.org/index.php/Doc:ES/2.4/Manual/Game_Engine. [Último acceso: 16 Abril 2017].

[25] Unity3D, «Flujo de trabajo de los Assets (Asset Workflow),» [En línea]. Available: https://docs.unity3d.com/es/current/Manual/AssetWorkflow.html. [Último acceso: 20 Abril 2017].

[26] envatotuts+, «introducing baugmented reality with vuforia,» [En línea]. Available: https://code.tutsplus.com/tutorials/introducing-augmented-reality-with-vuforia--cms-27160. [Último acceso: 20 Abril 2017].

[27] Blender, «Window HotKeys,» [En línea]. Available: https://wiki.blender.org/index.php/Doc:2.4/Reference/Hotkeys/All. [Último acceso: 10 Marzo 2017].

[28] Blender Stack Exchanges, «How do I "parent" objects?,» [En línea]. Available: https://blender.stackexchange.com/questions/26108/how-do-i-parent-objects. [Último acceso: 20 Marzo 2017].

[29] KatsBits TM, «Shortcuts (Hotkey) & Charts For Blender,» [En línea]. Available: https://www.katsbits.com/tutorials/blender/useful-keyboard-shortcuts.php. [Último acceso: 11 Marzo 2017].

[30] Blender, «Blender HotKeys In-depth Reference,» [En línea]. Available: http://download.blender.org/documentation/BlenderHotkeyReference.pdf. [Último acceso: 10 Marzo 2017].

[31] Blender, «Doc:ES/2.6/Manual/,» [En línea]. Available: https://wiki.blender.org/index.php/Doc:ES/2.6/Manual/. [Último acceso: 02 Marzo 2017].

[32] Blender 3D, «Blender 3D: Noob to Pro/Practicing Good Parenting,» [En línea]. Available: https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Parenting. [Último acceso: 22 Marzo 2017].

[33] Foro Vuforia , «How To Use the Vuforia Object Recognition Unity Sample,» [En línea]. Available: https://library.vuforia.com/articles/Solution/How-To-Use-the-Vuforia-Object-Recognition-Sample. [Último acceso: 25 Abril 2017].

[34] DavidBeard - Foro Vuforia, «Camera Mode details,» [En línea]. Available: https://developer.vuforia.com/forum/qcar-api/camera-mode-details. [Último acceso: 25 Abril 2017].

[35] AlessandroB - Foro Vuforia, «Unity - How to select Camera and mirroring,» [En línea]. Available: https://developer.vuforia.com/forum/faq/unity-how-select-camera-and-mirroring. [Último acceso: 25 Abril 2017].

[36] NalinS - Foro Vuforia, «Flipping Video Input,» [En línea]. Available: https://developer.vuforia.com/forum/unity-extension-technical-discussion/flipping-video-input. [Último acceso: 25 Abril 2017].

[37] AlessandroB , Foro Vuforia, «What is World Center Mode ?,» [En línea]. Available: https://developer.vuforia.com/forum/unity-extension-technical-discussion/what-world-center-mode. [Último acceso: 26 Abril 2017].

[38] Foro Vuforia, «Configuring Rotational Tracking in Unity,» [En línea]. Available: https://library.vuforia.com/articles/Solution/Configuring-Rotational-Tracking-in-Unity. [Último acceso: 26 Abril 2017].

[39] underCreative Commons Cero, «Robot_ Industrial,» [En línea]. Available: http://www.blendswap.com/blends/view/67820.

[40] M. C. Martínez y D. H. Hernández, «Realidad aumentada para el museo de América,» [En línea]. Available: http://eprints.ucm.es/32915/1/Realidad aumentada para el Museo de América.pdf. [Último acceso: 02 Abril 2017].

[41] Ecured, «Blender,» [En línea]. Available: https://www.ecured.cu/Blender. [Último acceso: 20 Abril 2017].

[42] Dan Casas, «Cinemática inversa,» [En línea]. Available: http://dancasas.github.io/teaching/AC-2016/docs/2.2-Cinematica-inversa-v2016.pdf. [Último acceso: 18 Abril 2017].

[43] Universidad de la república Urugay , «Fundamentos en Robótica,» [En línea]. Available: https://eva.fing.edu.uy/pluginfile.php/109572/mod_resource/content/2/1.1 - Intro robótica e IA.pdf. [Último acceso: 18 Abril 2017].

[44] A. L. C. J. A. C. E. Víctor R. González Fdez, «Control y Robótica,» [En línea]. Available: http://platea.pntic.mec.es/vgonzale/cyr_0708/index.htm. [Último acceso: 19 Abril 2017].