RESUMEN

APLICACIÓN DE LA REALIDAD AUMENTADA AL MANTENIMIENTO DE MAQUINARIA INDUSTRIAL DE CINCO EJES: UNA INTEGRACIÓN TECNOLÓGICA

por

Cinthya Karelly García Escobedo

Asesor principal: Ramón Andrés Díaz Valladares

RESUMEN DE PROYECTO DE POSGRADO

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

Título: APLICACIÓN DE LA REALIDAD AUMENTADA AL MANTENIMIENTO DE MAQUINARIA INDUSTRIAL DE CINCO EJES: UNA INTEGRACIÓN TECNOLÓGICA

Nombre del investigador: Cinthya Karelly García Escobedo

Asesor principal: Ramón Andrés Díaz Valladares, Doctor en Nuevas Tecnologías de la Información

Fecha de terminación: Abril de 2015

Problema

La falta de implementación de tecnologías de la información ligadas al área de visión por computadora, como las librerías de realidad aumentada y en el área de simulación por computadora, como el software de motores gráficos para apoyo visual en los procesos de reparación y mantenimiento de máquinas industriales en México.

Método

Debido a la complejidad que representa la reparación y mantenimiento de máquinas industriales para los técnicos se optó por implementar la tecnología de realidad

aumentada y software de motores gráficos al proyecto. Fue necesario realizar una evaluación de las características que nos ofrecía cada componente de software seleccionado, a fin de cumplir con los objetivos propuestos en el desarrollo de la aplicación.

Conclusiones

Al finalizar el proyecto se logró la integración de las tecnologías de realidad aumentada y el software de motores gráficos, en un proyecto que dio como resultado una aplicación que permite interactuar y controlar un mundo virtual compuesto por modelos 3D, por medio de una interfaz amigable e interactiva para el técnico de máquinas industriales.

Universidad de Montemorelos

Facultad de Ingeniería y Tecnología

APLICACIÓN DE LA REALIDAD AUMENTADA AL MANTENIMIENTO DE MAQUINARIA INDUSTRIAL DE CINCO EJES: UNA INTEGRACIÓN TECNOLÓGICA

Proyecto
presentado en cumplimiento parcial
de los requisitos para el grado de
Maestría en Ciencias Computacionales

por

Cinthya Karelly García Escobedo

Abril de 2015

APLICACIÓN DE LA REALIDAD AUMENTADA
AL MANTENIMIENTO DE MAQUINARIA
INDUSTRIAL DE CINCO EJES: UNA
INTEGRACIÓN TECNOLÓGICA

Proyecto
presentado en cumplimiento parcial
de los requisitos para el grado de
Maestría en Ciencias Computacionales

por

Cinthya Karelly García Escobedo

APROBADO POR LA COMISIÓN:

Dr. Ramón Andrés Díaz Valladares
Asesor principal

Dr. Alonso Pérez Soltero
Examinador Externo

M.C.T.I. Saulo Hernández Osoria
Miembro

Dra. Raquel B. de Korniejczuk
Directora de Estudios de Posgrado

M.C. Jair A. del Valle López
Miembro

30 de Abril de 2015
Fecha de aprobación

DEDICATORIA

A mis padres porque gran parte de lo que soy se lo debo a ellos.

A familiares y amigos ya que sin su apoyo nada de esto hubiera sido posible.

TABLA DE CONTENIDO

LISTA DE FIGURAS

RECONOCIMIENTOS

Esta tesis ha sido desarrollada como parte del proyecto "**Laboratorio para el desarrollo de aplicaciones industriales ergonómicas de realidad aumentada para el aceleramiento cognitivo y fortalecimiento técnico**" que fue propuesto por la empresa Ultrakut S. A. de C. V. en la convocatoria del Programa de Estímulos a la Innovación (PEI) del año 2014. El proyecto fue apoyado con fondos del CONACYT y de la empresa.

Un agradecimiento especial al CONACYT por su apoyo al proyecto No. 209754, a la empresa y al Maestro Horacio Ríos Corzo coordinando el trabajo vinculado de la UDEM, el ITESM y el TecMilenio con la Universidad de Montemorelos.

También quiero agradecer a mis compañeros de Dirección de Investigación: Eder y Aarón quienes me introdujeron en la tecnología de realidad aumentada y compartieron conmigo su conocimiento, su experiencia y su amistad.

A los maestros que me brindaron su tiempo, en especial a mis asesores de tesis, el Dr. Andrés Díaz Valladares, M.C. Saulo Hernández y el M.C. Jair del Valle López. Gracias.

CAPÍTULO I

INTRODUCCIÓN

Antecedentes

La creciente complejidad y el avance de plantas y maquinaria industrial presentan importantes desafíos a los técnicos responsables del mantenimiento de estos equipos. La tecnología de realidad aumentada (RA) se puede utilizar para auxiliar a los técnicos de mantenimiento, proporcionando información útil a través de interacciones visuales. Por ejemplo, la superposición de las instrucciones de mantenimiento sobre el equipo real. Sin embargo, para mejorar la usabilidad de los sistemas de mantenimiento asistido con RA la información virtual proporcionada debe adaptarse al contexto pertinente. Por ejemplo, la información proporcionada debe estar de acuerdo con el estado de la actividad y el nivel de experiencia del usuario. Además, el contenido de RA no debe ser de "sólo lectura", de tal manera que los técnicos de mantenimiento se conviertan en receptores de información pasiva. Proporcionar a los técnicos los medios para interactuar con el contenido de RA es muy útil ya que podrían incluso rectificar algún contenido incorrecto que hubiera sido creado por los desarrolladores de RA (Zhu, Ong y Nee, 2013).

Justificación del problema

La realidad aumentada consiste en superponer objetos o animaciones tridimensionales creadas por computadora sobre una imagen en tiempo real que captura una

cámara web. Dicho de otra forma, es mezclar el entorno real con el entorno virtual.

La realidad aumentada no es una tecnología nueva, en 1968 Ivan Sutherland junto con su ayudante construyeron lo que se considera como el primer visor montado en la cabeza (HDM) (Sutherland, 1968). En la actualidad esta ciencia es muy versátil, puede ser aplicada prácticamente a cualquier área de la industria, como la automotriz, la medicina, pero sobre todo en el marketing con el fin de crear experiencias en los consumidores.

En este proyecto la RA se aplica al área de reparación y mantenimiento de máquinas industriales, debido a que el mantenimiento juega un papel importante en el ciclo de vida del equipo, ya que restaura y mejora el rendimiento, la fiabilidad y la seguridad de la maquinaria. Sin embargo la creciente complejidad y el avance de la tecnología presentan un reto a los técnicos de mantenimiento en la actualidad. Se han desarrollado manuales para ayudar a los técnicos a enfrentar estos desafíos sin embargo, esto no ha sido suficiente para simplificar el trabajo del especialista dado la complejidad de los sistemas que componen una máquina industrial moderna. La RA integrada en un sistema de troubleshooting representa una excelente oportunidad porque permiten simplicar los esquemas abstractos e interpretarlos de una forma sencilla e interactiva, ya que es una interfaz perfecta entre el mundo real y el virtual.

Objetivos

Los objetivos de esta investigación son los siguientes:

1. Introducción a nuevas tecnologías para su uso en investigaciones y proyectos posteriores en la Universidad de Montemorelos.

2. Desarrollo de una herramienta innovadora en el campo de la ingeniería de

manufactura avanzada, para la reparación, mantenimiento y entrenamiento en sistemas electromecánicos de alta complejidad como lo son las máquinas Walter.

3. Proporcionar apoyo visual en el entrenamiento de personal técnico por medio del uso de la tecnología que se desarrollará.

4. Prueba de la herramienta y su compatibilidad con dispositivos móviles que se encuentran en el mercado.

Limitaciones

Algunas limitantes que se tuvieron en la investigación son las siguientes:

1. Falta de una licencia del software especializado para la creación y manejo de gráficos tridimensionales debido a su elevado costo. Adicionalmente, el costo de plugins para el diseño de la interfaz, el reconocimiento de gestos y las librerías de RA que reconocen objetos tridimensionales.

2. La falta de experiencia en el desarrollo de ambientes de RA para la reparación y mantenimiento de sistemas industriales complejos influye en la duración y calidad del desarrollo del proyecto.

3. La escasez de publicaciones en el área de RA aplicada a sistemas industriales complejos.

4. El alto nivel de abstracción de los sistemas de troubleshooting de máquinas industriales complejas.

Delimitaciones

Las delimitaciones en el proyecto de investigación son:

1. La herramienta será desarrollada para las plataformas iOS y Android. Específicamente para los equipos iPhone 4, Samsung Galaxy Tab 3, iPad mini, Galaxy S3 mini y Samsung Galaxy Tab SM-T800.

2. El sistema de troubleshooting de la aplicación se desarrollará especialmente para la máquina Walter Helitronic.

3. Debido a la complejidad de la máquina Walter, la herramienta solo implementará uno de los sistemas más importantes de la maquina: el sistema de troubleshooting de los componentes mecánicos.

Definición de términos

Se presentan a continuación las definiciones de los términos que serán utilizados en este documento.

Assets: Son los objetos básicos de cualquier proyecto. Estos tipos de objetos pueden ser una imagen, un modelo 3D, un script, un prefab o un sonido.

Código fuente: Conjunto de instrucciones escritas en un lenguaje de programación concreto, estas instrucciones son traducidas a un lenguaje que se denomina código máquina, el cual podrá ejecutar cualquier computadora.

FBX: es un formato de archivo que permite exportar el modelo 3D, las texturas y animaciones en un solo componente.

Finger Gestures: Framework para el reconocimiento de gestos comunes que se realizan, ya sea con el mouse o en la pantalla táctil del dispositivo.

GameObjects (Objeto de juego): cuando usamos un Asset en una escena, este se convierte en un GameObject. Todos los GameObject tienen una componente llamada Transform, que sirve para indicar la posición, rotación y escala del objeto en el

escenario, descrito en coordenadas X, Y, Z.

Maya: Software que brinda un conjunto completo de funciones creativas para el modelado simulación, renderización, composición y animación 3D.

NGUI: Framework de gran alcance para el desarrollo de interfaces de usuario en Unity. Escrito en C# y basado en el principio de KISS. El principio propone que cualquier sistema que implementemos, un producto, un servicio, un sistema informático, las soluciones a un problema, etc.; son mucho mejores si se basan en la simplicidad y sencillez.

Prefabs (prefabricado): Es un tipo de GameObject reutilizable, es decir, que lo puedes insertar en cualquier escena y el número de veces que quieras. Cuando se agrega un Prefab a una escena, se crea una instancia del mismo. Todas las instancias del Prefab están vinculadas con el original y son copias de esté. No importa cuántos existan en el proyecto, al realizar cualquier cambio en el Prefab original los cambios se aplicarán a todas las instancias.

SDK: Kit de desarrollo de software que generalmente está formado por un conjunto de herramientas de desarrollo de software que le permite al programador crear aplicaciones para un sistema concreto.

Sqlite: Herramienta de software libre, que permite almacenar información en dispositivos, de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser un Smartphone.

Vuforia: SDK para dispositivos móviles que permite la creación de aplicaciones de Realidad Aumentada. Utiliza la tecnología de visión por computador para reconocer imágenes planas (Image Targets) y objetos 3D simples como cajas en tiempo real.

Organización de la tesis

Este documento está organizado de la siguiente manera: Se ha dividido en cinco capítulos.

El Capítulo I es una introducción general al proyecto, incluye los antecedentes, la justificación del problema, los objetivos así como las limitaciones y delimitaciones que presenta el proyecto.

El Capítulo II consiste en una presentación del estado del arte de las áreas relacionadas con el proyecto: Interacción hombre-máquina, Realidad Virtual, Interfaz Natural de Usuario y Realidad Aumentada

El Capítulo III está dedicado a las tecnologías que son la base fundamental para el desarrollo del proyecto, así como las razones por las que fueron seleccionadas.

El Capítulo IV está compuesto por la planeación del proyecto, el desarrollo del proyecto que incluye cómo fue aplicada la arquitectura MVC al proyecto, los diagramas de clases entre otros elementos y por último los logros alcanzados.

Finalmente en el Capítulo V se presentan las conclusiones, recomendaciones y trabajos futuros.

CAPÍTULO II

REVISIÓN DE LA LITERATURA

Interacción hombre-máquina

Las computadoras están presentes cada vez más en la sociedad originando la interacción natural del hombre con la computadora causando un impacto positivo cada vez mayor en su uso. Es por esto que ha habido un creciente interés en el desarrollo de nuevos enfoques y tecnologías para atenuar la barrera hombre – máquina. El objetivo final es conducir a un régimen en el que las interacciones con las computadoras serán tan naturales como una interacción entre los seres humanos (Rautaray y Agrawal, 2012).

El concepto de Interacción hombre-máquina (HCI) apareció espontáneamente con el surgimiento de la computadora. La razón es muy clara: las máquinas más sofisticadas son inútiles a menos que puedan ser utilizadas correctamente por el hombre. Este argumento básico invita a presentar los dos principales términos que deben ser considerados en el diseño de HCI: funcionalidad y usabilidad (Te'eni, Carey y Zhang, 2007).

La funcionalidad de un sistema se define por el conjunto de acciones o servicios que presta a sus usuarios. Sin embargo, el valor de la funcionalidad es visible sólo cuando se hace posible para ser utilizado de manera eficiente por el usuario (Shneiderman y Plaisant, 2004).

La usabilidad de un sistema con una cierta funcionalidad es el tipo y el grado por el cual el sistema puede ser utilizado de manera eficiente y adecuada para lograr ciertos objetivos para determinados usuarios. La eficacia real de un sistema se consigue cuando hay un equilibrio adecuado entre la funcionalidad y la facilidad de uso de un sistema (Nielsen, 1994).

Al tener estos conceptos en mente y teniendo en cuenta que los términos computadora, máquina y sistema se usan indistintamente en este contexto, HCI es un diseño que debe producir un ajuste entre el usuario, la máquina y los servicios que se requieren para lograr un determinado rendimiento tanto en la calidad como en la optimización de los servicios (Zhang y Galletta, 2006).

La computación ubicua (Ubicomp) es un importante campo de investigación en HCI. El término se utiliza a menudo de manera intercambiable por la inteligencia ambiental o la computación ubicua y se refiere a los recientes métodos de la interacción hombre-máquina, que consisten en la eliminación de un escritorio y la incrustación de la computadora en el medio ambiente para que sea invisible a los seres humanos (Karray, Alemzadeh, Saleh y Arab, 2008).

Interfaz natural de usuario

La interfaz de usuario ha sufrido una constante evolución con el transcurso de los años, todo comenzó mediante la interacción con las tarjetas perforadas y la impresora hasta los distintos estilos de interacción con el usuario (dentro de las limitaciones que impone la tecnología disponible) más próximos a la forma a la que interactuamos con los humanos (Abascal y Moriyón, 2002).

Sucesivamente surgió la interfaz basada en una línea para las instrucciones, la

basada en menús y formularios y la basada en manipulación directa representan avances cada vez más sofisticados en esta dirección, que permiten que el flujo de información en ambos sentidos entre el usuario y el computador sea mayor y se realice con menos esfuerzo (Abascal y Moriyón, 2002).

Debido a la enorme difusión de los computadores personales y la necesidad de ampliar el mercado atrayendo a los posibles usuarios que rechazaban los computadores provistos de monótonas interfaces de comandos se dio un gran impulso a la búsqueda de interfaces gráficas de usuario (GUI) cada vez más sofisticadas y que permitieran una interacción más natural (Abascal y Moriyón, 2002).

El término natural, a menudo se entiende en el sentido de la imitación del "mundo real." Las interfaces naturales de usuario utilizan interacciones naturales de los seres humanos para interactuar con objetos de la interfaz como: a) gestos y movimientos, b) multi-touch, c) expresiones faciales o lenguaje corporal, d) habla y e) eye-gazing (Wigdor y Wixon, 2011).

No hay una definición única en la literatura sobre la Interfaz de Usuario Natural (NUI). Por ejemplo, Blake (2012) define NUI como "una interfaz de usuario diseñada para reutilizar las habilidades existentes para interactuar directamente con el contenido" (p. 4). König, Rädle y Reiterer (2009) defienden que una NUI se basa en que un usuario pueda llevar a cabo movimientos naturales, movimientos o gestos y hacerles descubrir rápidamente cómo controlar la aplicación informática o manipular el contenido de la pantalla. Incluso se pueden utilizar el habla y el toque simultáneamente dando lugar a lo que se conoce como interfaces multimodales.

La utilización de las interfaces de usuario naturales (NUIS), más específicamente

el uso de gestos o movimientos en un dispositivo multi-touch, ha llegado a convertirse en una buena alternativa, debido a que el uso de dispositivos de entrada tradicionales, tales como teclados y ratones pueden transformarse en un gran obstáculo para los adultos en la interacción con los sistemas informáticos (Loureiro y Rodrigues, 2011).

El camino a las interfaces naturales aún es largo y ahora se está presenciando una naturalidad artificial. Estas interfaces son naturales, en el sentido que emplean gestos de la mano, pero también son artificiales, debido a que el diseñador del sistema impone el conjunto de gestos. La tecnología es un aspecto interesante a considerar, ya que, incluso si un gesto no es el más natural, puede llegar a ser natural debido al uso generalizado de la tecnología de adoptarlo. Sin embargo el objetivo principal de las interfaces naturales debería ser proporcionar a los usuarios de gestos a los que están más acostumbrados, teniendo en cuenta sus hábitos, antecedentes y aspectos culturales (Malizia y Bellucci, 2012).

Entre los dispositivos tecnológicos actuales considerados como Interfaz Naturales de Usuario se tienen como ejemplos: (a) Iphone, (b) Surface de Microsoft, (c) Ipad, (d) diferentes SmartPhones, (e) LeapMotion, (f) Micosoft Kinect y (g) Wiimote.

Visión por computador

La visión por computador ha recorrido un largo camino desde 1963 con la disertación por Larry Roberts del MIT, que a menudo se considera un punto fundamental en el nacimiento del campo (Turk y Hua, 2013).

Visión por computador es un campo de rápido crecimiento enfocado al análisis, modificación, y la comprensión de alto nivel de las imágenes. Su objetivo es determinar

lo que está sucediendo delante de una cámara y utilizar ese conocimiento para controlar una computadora o sistema robótico, o para proporcionar a las personas de nuevas imágenes que son más informativas o estéticas que las imágenes originales de la cámara (Pull, Baksheev, Kornyakov y Eruhimov, 2012).

El sensor de entrada a un sistema de Vision-Based Interaction puede ser una o más cámaras de vídeo o sensores de profundidad (usando estéreo u otra tecnología de detección 3D). El ambiente podría estar bien estructurado (por ejemplo, las posiciones de iluminación y del cuerpo controladas, marcadores colocados en el participante (s)); totalmente estructurados (por ejemplo, no hay marcadores, no hay restricciones en la iluminación, los objetos del fondo, o el movimiento); o algo intermedio. Diferentes escenarios pueden limitar la entrada a determinadas partes del cuerpo (por ejemplo, la cara, las manos, parte superior del cuerpo) o movimientos (por ejemplo, las expresiones faciales, los gestos sutiles de dos manos o el movimiento de todo el cuerpo) (Turk y Hua, 2013).

La iluminación es un componente importante del sistema de visión por computador. Al igual que con los ojos humanos, los sistemas de visión se ven afectados por el nivel y la calidad de la iluminación. Los dispositivos de iluminación generan luz que ilumina los objetos de destino inspeccionados, por lo tanto, el rendimiento del sistema de iluminación puede influir en gran medida en la calidad de las imágenes y juega un papel importante en la eficiencia y la precisión del sistema (Brosnan y Sun, 2004).

Las principales áreas de aplicación de la tecnología de visión por computador incluyen: (a) video vigilancia, (b) biometría, (c) automotriz, (d) fotografía, (e) producción de películas, (f) la búsqueda Web, (g) medicina, (h) juegos de realidad aumentada y (i)

las nuevas interfaces de usuario (Pull et al., 2012).

Durante las últimas dos décadas, ha habido un creciente interés en las aplicaciones de visión por computador en la interacción humano-computadora, especialmente en sistemas que procesan imágenes de personas con el fin de determinar la identidad, la expresión, el pose del cuerpo, gesto, y la actividad. En algunos de estos casos, la información visual es una modalidad de entrada en un sistema multimodal, proporcionando señales no verbales para acompañar la entrada de voz y tal vez la interacción basada en el contacto (Turk y Hua, 2013).

Algunas representaciones de ciencia ficción de la interacción basada en visión son: (a) La realidad aumentada del cyborg de Terminator, (b) La interfaz gestual de Minority Report, (c) la interacción gestual y análisis facial de Iron Man (Turk y Hua, 2013).

Realidad virtual

En 1963 Ivan Sutherland presentó Sketchpad, un programa informático que utiliza una pantalla con vectores en xy y un lápiz óptico para el dibujo asistido por computadora. Sketchpad es posiblemente la primera interfaz gráfica de usuario interactiva para una computadora. Dos años más tarde, la llamó "pantalla inmejorable". La pantalla vendría a ser "una habitación dentro de la cual la computadora puede controlar la existencia de la materia" (Sutherland, 1965, p. 508).

Posteriormente Sutherland y su alumno, Bob Sproull, crearon lo que se considera como el primer visor montado en la cabeza o *Head Mounted Display* (HMD) para gráficos por computadora interactivos (Welch, 2009).

Su sistema generaba imágenes de acuerdo a la orientación y posición del movimiento que realizaba la cabeza. La pantalla estaba suspendida del techo y utilizaba un brazo mecánico que con la ayuda de transductores de ultrasonido realizaban el seguimiento de la cabeza (Welch, 2009).

Cuando se habla de "realidad virtual" (VR) se refiere a una simulación por computadora que crea una imagen de un mundo que aparece a nuestros sentidos de la misma forma en se percibe el mundo real, o la realidad "física". Con el fin de convencer al cerebro de que el mundo sintético es auténtico, la simulación por computadora monitorea los movimientos del participante y ajusta la pantalla sensorial para dar la sensación de estar presente en la simulación. La realidad virtual es un medio de permitir que los participantes se involucran físicamente en algún entorno simulado que es distinto de su realidad física, es un medio por el cual los seres humanos pueden compartir ideas y experiencias. La parte de la experiencia que es "el mundo" presenciado por el participante y con el que interactúan se conoce como el mundo virtual (Craig, Sherman y Will, 2009).

Hoy en día los sistemas de VR son utilizados para la simulación, el diseño industrial, terapia de fobias, la planificación y asistencia quirúrgica entre otras aplicaciones relacionadas con la salud (Welch, 2009).

Realidad aumentada

La realidad aumentada (RA) es una tecnología que permite que la información de imágenes virtuales generadas por computadora sean superpuestas directa o indirectamente en un entorno y tiempo real (Azuma, 1997; Zhou, Duh y Billinghurst, 2008).

La realidad aumentada difiere de la realidad virtual, en VR los usuarios experimentan un entorno virtual generado por computadora, mientras que en RA, el medio ambiente es real, es decir el ambiente es ampliado con la información y las imágenes del sistema. De acuerdo con Chang, Morreale y Medicherla (2010), la RA cierra la brecha entre lo real y lo virtual en una forma perfecta.

Si bien los principios de la presentación de la información y su representación han sido ampliamente investigado para entornos de escritorio, lo que lleva a los sistemas basados en Windows-Icons-Menus-Pointer (conocido por sus siglas en inglés como WIMP), los conceptos para entornos 3D aún no han sido categorizados en la misma medida. La mera capacidad de presentar información virtual en 3D ya genera un espacio de presentación que supera los conceptos de metáforas clásicas basadas en WIMP. Con el principio de la Realidad Aumentada, este espacio se amplía aún más. Un mundo virtual coexiste con el mundo físico y además está incrustado en él. Ambos mundos tienen su propia existencia en 3D en propiedades geométricas y fotométricas y tienen evolución dinámica en el tiempo (Tönnisn, Plecher y Klinker, 2013).

Representación de la información

La representación define cómo es organizada la información, mientras que la presentación muestra esos datos. Si bien hay un número infinito de formas en que finalmente la información se puede representar en RA, la investigación se ha centrado en cómo la información se puede representar en el espacio de presentación de RA (Spence, 2006).

Un sistema de realidad aumentada genera una vista compleja en el que las partes virtuales están cubiertas por elementos reales en el lugar de trabajo básico para el

usuario. Es una combinación mutua de la escena real vista por el usuario y una escena virtual generada por el núcleo lógico informático que aumenta la escena. Gracias a estas posibilidades se encuentra su utilización en muchos ámbitos industriales (Tzong-Ming y Tu, 2009).

Tönnisn et al. (2013) realizaron una investigación analítica de las aplicaciones existentes que les llevó a presentar cuatro dimensiones ortogonales que se ocupan de representación de la información para RA:

Temporalidad: Esta dimensión depende de la existencia del objeto, independientemente de si el objeto está dentro del campo de visión o no. Esta diferenciación es ya conocida por los entornos de escritorio y la interacción hombre-máquina (HCI).

Dimensionalidad: Se refiere a la cartografía y la integración de la información virtual en un entorno físico. La realidad física presenta un enorme número de propiedades de los objetos, tales como: (a) la forma, (b) el aspecto, (c) el reflejo de luz, (d) la rigidez, (e) la temperatura, (f) la radiación, (g)el sonido, etc.

Montaje: Coloca la información con respecto a otros objetos. Dos aspectos se combinan juntos aquí. En primer lugar es el montaje real, que se define por el concepto de la aplicación. En segundo lugar está el seguimiento de objetos en relación a los lugares en el mundo físico.

Tipo de referencia: es la medida en la que un objeto virtual se refiere a un objeto físico o en una ubicación. Depende de diferentes aspectos de la visibilidad de los objetos físicos. Un objeto o punto de interés puede ser directamente visible para el usuario, puede ser ocluido, o puede estar fuera del campo de visión.

Realidad aumentada en la actualidad

La comunidad científica y la industria han hecho aportaciones que permiten extender las capacidades de los dispositivos en búsqueda de propuestas innovadoras en diferentes contextos. Los objetivos varían dependiendo del interés del grupo de investigación, por ejemplo en el área automotriz, Platonov, Heibel, Meier y Grollmann (2006) desarrollaron un sistema prototipo de realidad aumentada para el mantenimiento del BMW 7 que utilizaba una cámara monocular para mostrar el contenido, aunque reportaban acumulación de error en los sistemas de rastreo.

En la universidad de Columbia, Henderson y Feiner (2011) crearon un prototipo para conducir el mantenimiento de rutina dentro de un vehículo militar, ellos usaron un casco de realidad aumentada y una computadora para procesar la información de posición mediante sensores infrarrojos que alimentaban el sistema, pero presentaba un pequeño inconveniente, el sistema necesitaba un setup exhaustivo y equipo de alto costo.

Chen en conjunto con otros autores diseñaron un ambiente que combinaba el despliegue de un automóvil a escala real y un sistema de evaluación de diseño donde usuarios múltiples podían ver simultáneamente el mismo auto para discutir los parámetros de diseño (Chen, Chen y Shen, 2008).

En área de aeronáutica Henderson y Feiner (2010) contribuyeron con lo que llamaron controles oportunísticos donde se usaban características naturales de los objetos para simular el control de la aplicación, el demostrador se realizó sobre un motor de Rolls Royce Dart 510 turboprop. Si el usuario tocaba el motor, por ejemplo en un tornillo, esto desplegaba las acciones que estaban asociadas a los procedimientos

predefinidos.

En el área de manufactura, Reinhart y Patron (2003) desarrollaron un sistema modular de realidad aumentada para guiar el ensamble manual en un sistema basado en PC. Zhang et al. (2010) integraron realidad aumentada en una máquina CNC de tres ejes, los renders apropiados para simulación de procesos de corte fueron hechos entre el cortador real y el ambiente virtual, este proyecto se centraba en la operación de la máquina y no en la reparación. Lee, Han y Yang (2011) propusieron una metodología para construir un ambiente de realidad mixta basado en un ambiente con objetos digitales con los objetos reales.

La industria médica también es uno de objetivos indirectos del desarrollo del proyecto ya que se trata de manufactura especializada, por ejemplo Nicolau et al. (2009) presentaron un sistema guía de realidad aumentada para intervenciones quirúrgicas relacionadas con el hígado humano, la precisión del sistema en ese momento fue reportada dentro de 4.5 mm. Weiss et al. (2011) propusieron un sistema que usaba rastreadores electromagnéticos para colocar agujas en la espina dorsal.

A media que la tecnología va avanzando y las capacidades de manufactura mejoran, se incrementa la necesidad de software especializado para aislar las fallas de un sistema electromecánico complejo porque básicamente todos los dominios de aislamiento no triviales son considerados NP-hard. Castellani, Grasso, O'Neill y Roulland (2009) condujeron un estudio para examinar las propiedades de diferentes ensambles de personas, recursos, tecnologías y espacios para generar un diseño de software para las diferentes situaciones de aislamientos de fallas, se cubrieron las diferentes dislocaciones entre los aspectos del estudio en los dos casos de aislamiento para el

autónomo y con ayuda de expertos, las dislocaciones encontradas fueron las físicas entre el lugar del problema y el lugar de la solución, conceptuales entre los usuarios y los expertos, y por último las dislocaciones lógicas entre los recursos de soporte y el estado del dispositivo que falló, este estudio desarrolló un sistema integrado para copiadoras que era desplegado en la pantalla guiando a los usuarios a través del proceso de aislamiento.

También se han hecho sistemas web, Liang (2008) propuso un sistema creado en línea para ayudar en la reparación de chasises automotrices, el sistema se desarrolló con animaciones virtuales e instrucciones escritas que dependían de una computadora de escritorio sin ninguna comunicación o retroalimentación del mundo real. Johnson, Rickel, Stiles y Munro (1998) crearon un sistema para desarrollar ambientes virtuales para entrenadores que usaban técnicas pedagógicas para brindar entrenamiento. Por el otro lado han sido desarrollados sistemas de aislamiento creados para expertos. Un ejemplo son los autores Ong et al. (2004) que presentaron un sistema para monitorear datos de turbinas donde el comportamiento anormal podía ser identificado para la prevención de accidentes, esta herramienta estaba completamente enfocada a personal experto por lo que solo algunos podían tener acceso.

Dugri (2004) investigó el uso de redes bayesianas para el análisis de decisiones para generar la óptima solución en aislamientos secuenciales, el modelo fue aplicado al sistema de purga del Boeing 737 pero se reportó que en algunos casos era impráctico para aplicaciones en escenarios reales debido a los costos asociados.

Como se ha mostrado en esta sección existen trabajos que han contribuido al

estado del arte de la tecnología pero aún se tiene una ventana de oportunidad de innovación en el área de manufactura avanzada de herramientas para el aislamiento de fallas usando tecnologías de realidad aumentada embebida a dispositivos móviles, el reto se encuentra en desarrollar la tecnología para que además de solucionar los problemas que tiene la industria sea a un bajo costo para crear un herramienta que pueda ser extendida fácilmente hacia otras aplicaciones.

Proceso de troubleshooting con RA

El ensamble con RA representa un tipo de inteligencia especial de herramientas de software para la comunidad de ingeniería que provee potentes elementos y equipos de hardware para la construcción de ideas que incluyen muchas piezas de montaje. La solución final de ensamble debe comprender todas las partes funcionales del montaje sin errores de ensamble. La aplicación RA de ensamble debe definir la posición y la orientación exacta para cada proceso de ensamble de los elementos. La inserción de la información digitalizada en el verdadero espacio de trabajo utilizando RA puede proporcionar a los trabajadores los medios necesarios para poner en práctica los procedimientos de ensamblaje correctos con una mayor precisión y reducir los errores (Hou, Wang, Bernold y Love, 2013; Marcincin, Barna, Janak, Marcincinova y Torok, 2012).

Durante el proceso de desarrollo el diseñador está frente a muchos problemas que afectan a la esencia misma del proceso de montaje. En la primera etapa de este proceso el modelo 3D contiene información necesaria como la orientación y la posición de las piezas, así como; el material y el valor geométrico.

En la segunda fase de la creación de la construcción el montaje de la parte 3D

solo necesita ser aplicado y proporcionado con la orientación exacta y los datos de posición ya que este modelo tiene que ser fijado al lugar particular del modelo auxiliar en el entorno real (Marcincin et al., 2012).

Aceleramiento cognitivo

Otro de los importantes campos donde la RA puede aportar beneficios es en la mejora del aprendizaje. Algunas investigaciones recientes ofrecen un panorama alentador.

Un sistema prototipo animado de RA fue configurado para tareas de montaje que normalmente se guían por referencia a la documentación y puesto a prueba mediante una serie de experimentos. Un modelo de LEGO fue utilizado para las tareas como el montaje y la prueba experimental. La prueba experimental se diseñó y llevó a cabo para validar los logros cognitivos que pueden derivarse del uso de RA para montar un modelo de LEGO.

Se realizaron dos experimentos formales con 50 participantes para comparar un sistema RA animado y el sistema manual basado en papel. El primer experimento se diseñó para medir la carga de trabajo cognitivo al utilizar el sistema de montaje, mientras que el otro experimento mide las curvas de aprendizaje de montadores novatos.

Los resultados de los experimentos revelaron que el sistema RA animado obtuvo como resultado periodos de tiempo más cortos en la finalización de la tarea, menos errores de montaje, y una menor carga de trabajo total. Los resultados también revelaron que la curva de aprendizaje de los montadores novatos se redujo y se incrementó el desempeño de tareas relevantes para la memoria de trabajo cuando se utiliza la capacitación por medio de RA (Hou et al., 2013).

CAPÍTULO III

METODOLOGÍA

Esta sección presenta una introducción de las tecnologías y conceptos fundamentales en los que se ha basado el desarrollo del proyecto. Adicionalmente se exponen las razones por las que fueron seleccionadas.

Para el desarrollo del proyecto fue necesario seguir las tareas que a continuación se presentan:

1. La elección de la librería de realidad aumentada. Esta elección se fundamentó en la aplicación de test basados en la metodología implementada por Serrano Mamolar (2012).

2. La selección del software para el modelado de objetos 3D y el motor gráfico para el control de las entradas.

3. La selección de sistemas operativos para los que se desarrollará la aplicación.

4. Se ha elegido MVC como la arquitectura de la aplicación debido a que ofrece mayores beneficios que otras arquitecturas.

5. La implementación de la base de datos Sqlite por ser una BD embebida para dispositivos móviles.

6. Se definió la utilización de plugins para realizar la conexión a la base de datos, el reconocimiento de gestos y para el diseño de la interfaz de usuario.

Elección del SDK para RA

Un factor que da lugar a la utilización más amplia de la RA es que los dispositivos móviles son cada vez más pequeños, más potentes y menos costosos.

En este proyecto la RA se aplica al área de reparación y mantenimiento de máquinas industriales. El mantenimiento juega un papel importante en el ciclo de vida de los equipos, ya que restaura y prolonga su rendimiento, así como la fiabilidad y la seguridad de los operarios. Sin embargo, en la actualidad la creciente complejidad y el avance de la tecnología añaden complicaciones al trabajo de los técnicos de mantenimiento y reparación.

Se han desarrollado manuales para ayudar a los técnicos a enfrentar estos desafíos. Sin embargo esto no ha sido suficiente dada la complejidad de los sistemas que componen las máquinas industriales. La RA permite crear una interfaz perfecta entre el mundo real y el virtual de manera que al integrarla en un sistema de troubleshooting representa una excelente oportunidad. El sistema resultante permite simplificar los esquemas abstractos e interpretarlos de una forma sencilla e interactiva.

Existen varias plataformas para el desarrollo de aplicaciones con realidad aumentada como Vuforia, Metaio, ARtoolKitPlus. Sin embargo, de acuerdo a requerimientos del proyecto, tales como:

1. El desarrollo de la aplicación para dispositivos móviles.

2. La creación de una base de datos con especificaciones de fabricante.

3. El diseño de modelos 3D con número de polígonos bajo e indexación adecuada para su uso en los sistemas de realidad aumentada que se crearán.

4. La creación de la infraestructura de la biblioteca de modelos, que será una de

las principales herramientas para hacer el llamado dinámico de componentes y procesos de ensamble.

Se propuso el análisis comparativo de Vuforia SDK y Metaio SDK, dado que estas se encuentran entre las principales plataformas utilizadas para desarrollo de aplicaciones con realidad aumentada de acuerdo a Jonas (2013). La realización de este análisis ayudaría en la selección de la herramienta que mejor se adaptaba a los requerimientos de un proyecto de RA.

Vuforia y Metaio son SDK para dispositivos móviles que permiten la creación de aplicaciones de RA. Ambas utilizan la tecnología de visión por computadora para el reconocimiento de patrones en tiempo real a través de una cámara.

Entre las principales características de Vuforia SDK la literatura presenta las siguientes:

1. El desarrollo de aplicaciones requiere un poco más de tiempo, sin embargo al tener claro el concepto el desarrollo se hace más sencillo.

2. Implementa las características tracking de imageTargets, markers, imagebuttons y reproducción de video; que sirven para mostrar el modelo, animación o video de acuerdo al patrón de la imagen. Cabe señalar que Metaio SDK no implementa estas opciones.

3. Es compatible con UNITY por lo cual permite el desarrollo para distintas plataformas móviles, evitando el trabajo de realizar una versión para cada plataforma.

4. Soporta multi-tag lo que permite el seguimiento simultáneo de hasta cinco image targets.

5. No soporta 3D tracking restándole así la habilidad de relacionar el objeto del

ambiente real con el del ambiente virtual.

6. El código generado para aplicaciones para iPhone y Android no es final ya que se deben realizar ajustes en el código para su correcta ejecución.

7. Vuforia cuenta con excelente documentación.

8. Es un SDK probado y que ha mostrado ser estable.

9. La plataforma Vuforia ah permitido la interactividad del mundo real para un número creciente de marcas líderes, entre ellos Anheuser-Busch, Audi, Johnson & Johnson, Lowe de Canadá, Maybelline, Nike, Sony Pictures Entertainment, Taco Bell y Virgin Media.

Metaio también es un SDK muy popular entre los desarrolladores de aplicaciones móviles y multimedia. Entre las principales características de la plataforma se incluyen las siguientes:

1. El SDK metaio incluye un potente motor de renderizado en 3D.

2. El desarrollo de aplicaciones es dinámico, por lo que el cambio de escenas se realiza de una forma eficiente.

3. Compatible con UNITY, que al igual que Vuforia le concede las ventajas del desarrollo de aplicaciones para diversas plataformas.

4. Soporta tracking de markers, image targets, marker-less como Vuforia y adicionalmente soporta objetos 3D. Esta característica facilita el reconocimiento de objetos en ambientes reales.

5. Las aplicaciones para iPhone y Android deben ser desarrolladas por separado.

6. Reproduce animaciones MD2, lo que permite la carga de formatos antiguos.

7. No cuenta con suficiente documentación.

8. El SDK no es totalmente estable, ya que puede presentar conflictos con algunas versiones de Android.

Análisis del SDK

El estudio comparativo realizado entre el Vuforia SDK y Metaio SDK se desarrolló en base a los factores principales: (a) el reconocimiento de target en cada caso, (b) el renderizado de los objetos virtuales, (c) la robustez del tracking y (d) la compatibilidad con versiones de S.O.

El dispositivo de referencia utilizado para las pruebas ha sido una tablet Samsung Galaxy Tab SM-T800. Este dispositivo presenta las características mostradas en la Tabla 1.

Tabla 1

Características del dispositivo

Elemento	Descripción
CPU	1.9 GHz Quad-core + 1.3 GHz Quad-core Processor
RAM	3 GB
Cámara	Frontal: 2.1 MP Trasera: 8 MP
Sistema operativo	Android OS, v4.4.2 (KitKat)
Batería	7900 mAh

Reconocimiento del target

Los SDK Vuforia y Metaio ofrecen un servicio personalizado al permitir que el programador elija las imágenes que funcionarán como target para el tracking.

Vuforia dispone de una página web, en donde se crean los targets personalizados. La herramienta se encarga de evaluar la calidad de la imagen, esto permite indicar el grado de efectividad que tendrá la imagen para ser reconocida de acuerdo al número de características especiales extraídas de esta. Para mejorar la calificación de la imagen es necesario utilizar alguna herramienta para la edición de imágenes.

El proceso que nos ofrece Metaio, en lugar de ser un servicio en línea, consiste en el empleo de la herramienta Metaio Creator, que se encarga de generar un fichero XML. Este fichero deberá ser incluido en el proyecto de RA.

El propósito al utilizar Metaio Creator o el servicio web de Vuforia es obtener la evaluación de la imagen propuesta ya que esto influirá en el rendimiento de la aplicación. Estas herramientas se encargan de seleccionar la región con el mayor número de características especiales. Esto ha permitido que se pueda utilizar el mismo target tanto en Vuforia como en Metaio.

La Figura 1 muestra el target que fue utilizado para validar la funcionalidad del reconocimiento de imágenes cuando la visión de la cámara es interrumpida cuando algún objeto cubre parcialmente el target. Como se puede observar en la Figura 1, Vuforia presenta un mejor comportamiento en este experimento en comparación con el SDK Metaio que al momento de cubrir parcialmente el target con una mano pierde la referencia y como resultado no despliega el modelo 3D.

Figura 1. Evaluación del reconocimiento del target utilizando Vuforia y Metaio.

Reconocimiento según el tamaño

La Figura 2 presenta los cinco tamaños de targets impresos con el mismo código QR, que fueron empleados en la ejecución de las pruebas para cada SDK. Durante la ejecución de las pruebas el target se encontraba a una distancia de aproximadamente 50 cm de la cámara de la Tablet.

Figura 2. Diferentes tamaños del target utilizados en la prueba de reconocimiento del tamaño.

El resultado de las pruebas arrojó que el SDK Vuforia tiene un mejor comportamiento frente al tamaño del target, ya que el reconocimiento está basado en características específicas y no en la comparación pixel a pixel de la imagen.

En las figuras 3, 4, 5, se puede observar que Vuforia es capaz de reconocer la imagen cuando está reducida a un 60% del tamaño recomendado en comparación con Metaio que solo es capaz de reconocer la imagen hasta el 75% de reducción del tamaño recomendado.

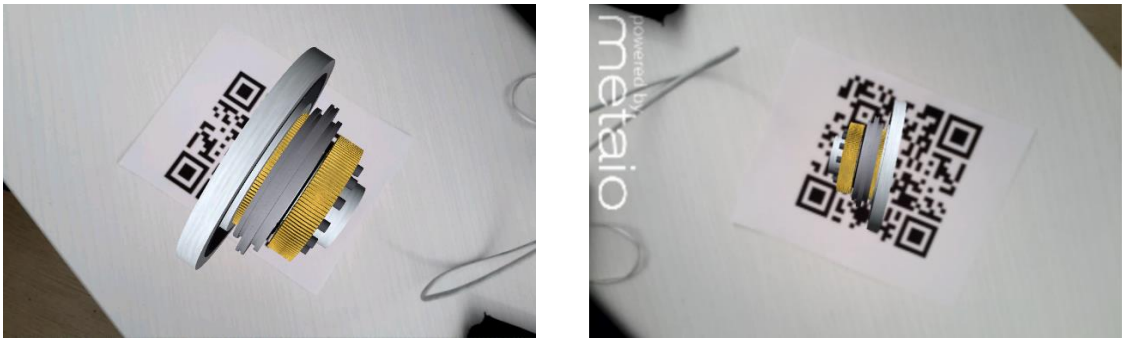

Figura 3. Reconocimiento del target por Vuforia y Metaio al 100% del tamaño.

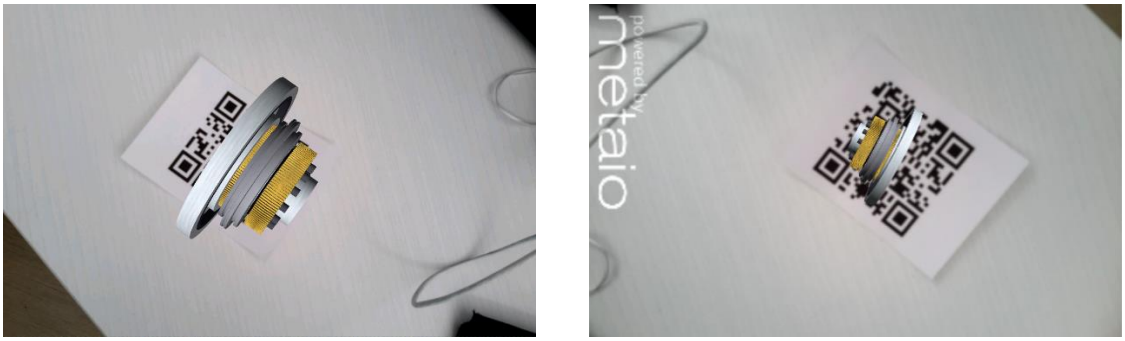

Figura 4. Reconocimiento del target por Vuforia y Metaio al 75% del tamaño.

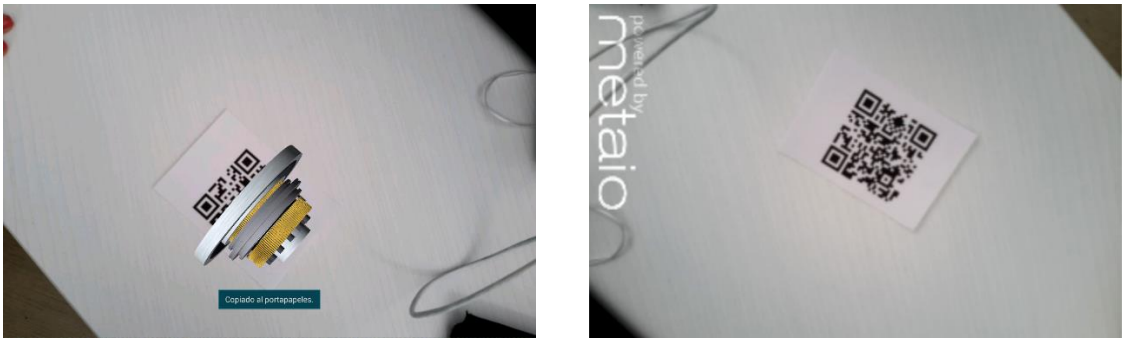

Figura 5. Reconocimiento del target por Vuforia y Metaio al 60% del tamaño.

En las figuras 6 y 7 se observa que tanto Vuforia como Metaio dejan de reconocer el target cuando este presenta un tamaño del 45% y 30% de reducción con respecto a las dimensiones del target original.

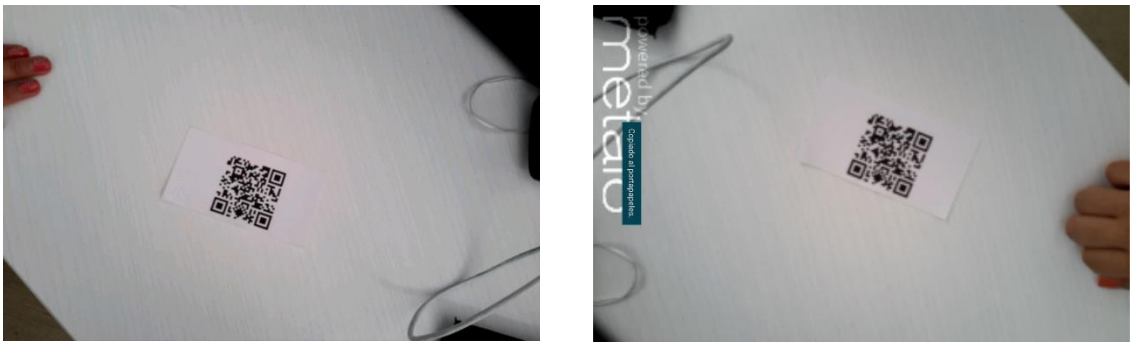

Figura 6. Reconocimiento del target por Vuforia y Metaio al 45% del tamaño.

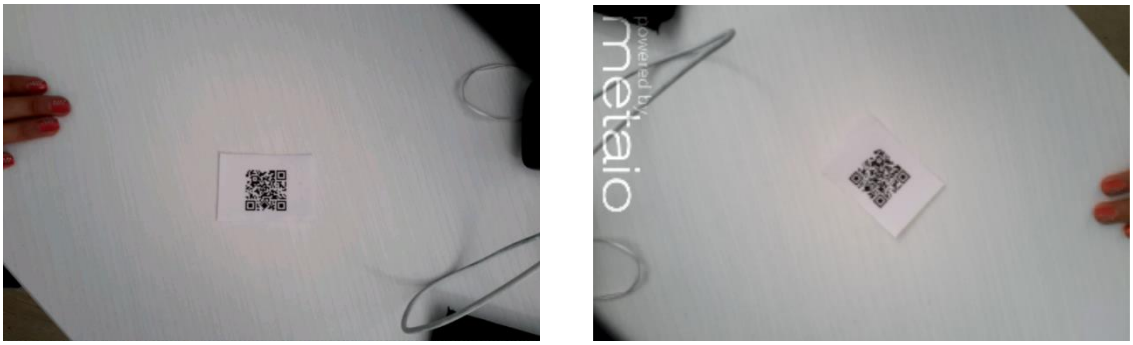

Figura 7. Reconocimiento del target por Vuforia y Metaio al 30% del tamaño.

Renderizado de los objetos virtuales

En esta etapa se estudia la tasa de frames por segundo que ofrece cada SDK frente a una situación. Es de vital importancia que los objetos 3D pueden ser importados de forma correcta desde la herramienta donde fueron diseñados. En este caso los modelos debían ser importados en formato FBX, debido a que este formato permite conservar las texturas y animaciones.

Para cada frame procesado el sistema de tracking de cada SDK necesita resolver la matriz de transformación entre la cámara y el target, para posicionar el objeto 3D. Una vez calculada esta matriz se pasa al proceso de renderizado del objeto.

Es muy común que la capacidad de renderizado de una aplicación tenga relación con la tasa de frames por segundo, ya que a medida que aumenta la complejidad de la escena a renderizar esta tasa frecuentemente baja. Una de las características básicas de una aplicación de RA es que sea en tiempo real, para ello la tasa de frames por segundo debe estar entre los 15-30 fps. El dispositivo utilizado para la realización de estas pruebas ofrece hasta 30 fps en FULL HD. Es decir, se tiene la seguridad de que el dispositivo no ofrece ninguna limitación a la hora de probar el rendimiento de renderizado de los SDK.

Renderizado de acuerdo al número de objetos.

En esta sección se realiza la prueba con el mismo modelo 3D para los dos SDK. Esta prueba permite comprobar que Vuforia y Metaio ofrecen la misma capacidad de renderizado, sin embargo la desventaja de Metaio consiste en que solo reconoce las características específicas del target a una distancia menor de lo que Vuforia lo hace.

A continuación en las Figuras 8, 9 y 10 se puede apreciar la validación de la funcionalidad de renderizado de los objetos que nos ofrece Vuforia SDK como Metaio SDK. El número de objetos a renderizar van desde cinco hasta cien.

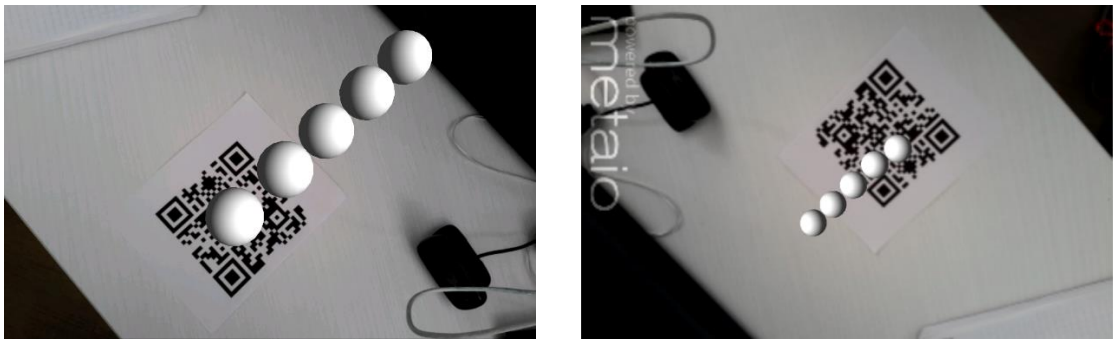

Figura 8. Evaluación del renderizado de cinco objetos con Vuforia y Metaio.

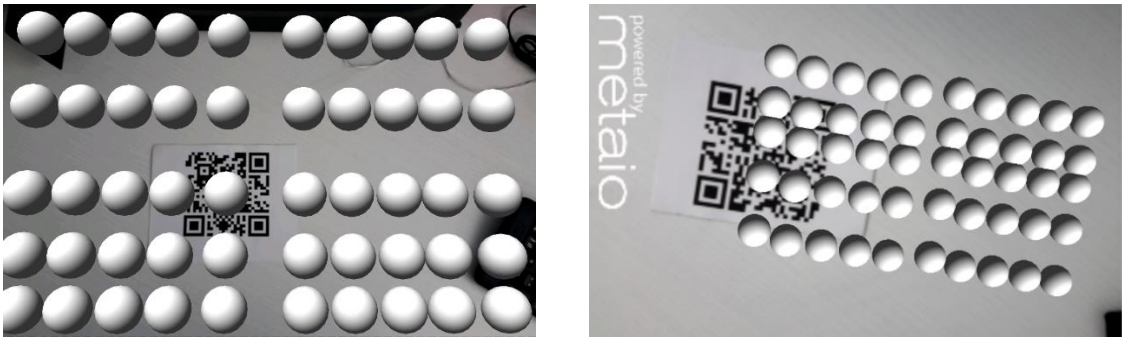

Figura 9. Evaluación del renderizado de 50 objetos con Vuforia y Metaio.

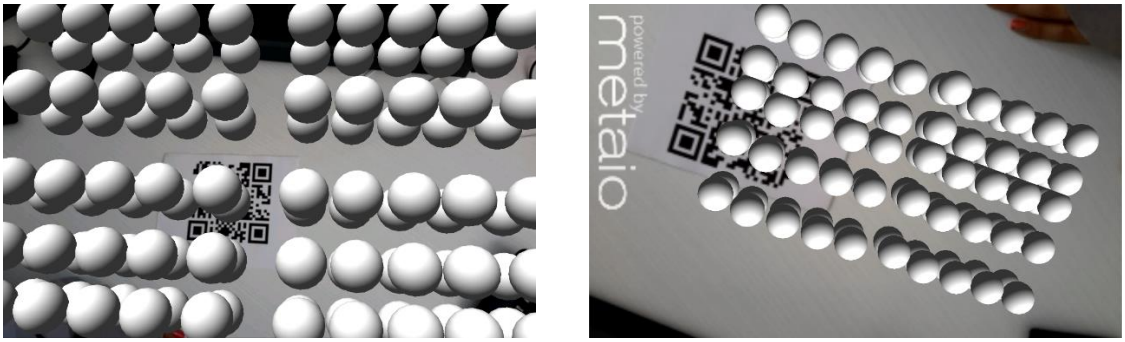

Figura 10. Evaluación del renderizado de 100 objetos con Vuforia y Metaio.

Robustez del tracking

En esta sección se realizó una serie de pruebas con un modelo 3D con el fin de comprobar el comportamiento de los targets frente al ángulo de inclinación de la cámara de cada SDK. La pieza utilizada para esta prueba es un componente del Eje C

que a su vez conforma la máquina Helitronic.

Reconocimiento del marcador según la distorsión de la perspectiva

En esta serie de experimentos la evaluación se realiza en base a la distorsión de la perspectiva del target representada mediante ángulos de visión de la cámara que van desde los 90º hasta los 30º.

En las figuras 11, 12, 13 se puede apreciar que Vuforia SDK y Metaio SDK no presentan alguna dificultad para reconocer el target cuando la cámara presenta una perspectiva de inclinación de 60°. Sin embargo en la Figura 14 y 15 demuestra que Vuforia no deja de reconocer el target en comparación con Metaio que deja de reconocer el target desde que presenta inclinación de 45°.

Figura 11. Evaluación de la perspectiva del target con inclinación a 90º con Vuforia y Metaio.

Figura 12. Evaluación de la perspectiva del target con inclinación a 75º con Vuforia y Metaio.

Figura 13. Evaluación de la perspectiva del target con inclinación a 60º con Vuforia y Metaio.

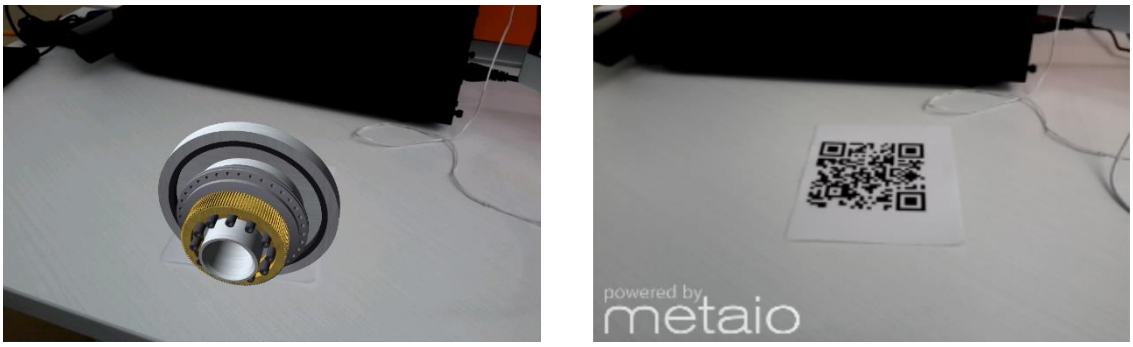

Figura 14. Evaluación de la perspectiva del target con inclinación a 45º con Vuforia y Metaio.

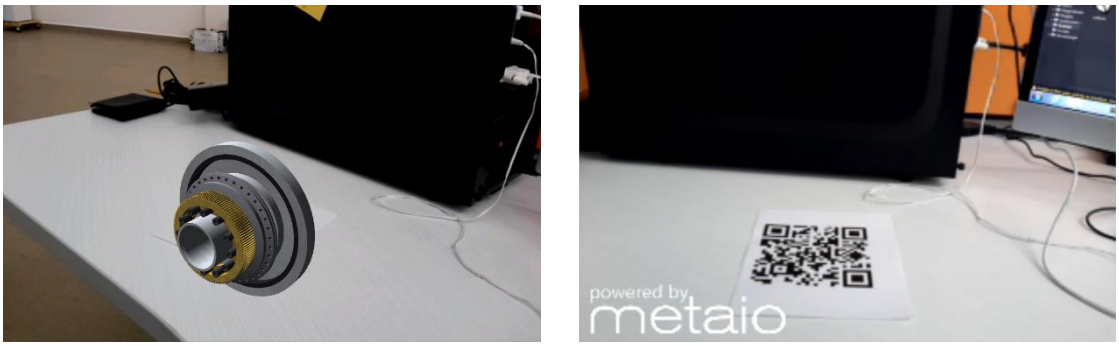

Figura 15. Evaluación de la perspectiva del target con inclinación a 30º con Vuforia y Metaio.

Modelado de objetos 3D

El modelado es el arte y la ciencia de la creación de una superficie que imita la forma de un objeto del mundo real o que se adapta a la imaginación de objetos abstractos (Blender, 2011a). Para el diseño de objetos 3D existe una gran variedad de programas que ayudan en la tarea de modelar, sin embargo el enfoque de cada programa es dirigido hacia un área específica, por lo tanto tienen diferentes características. Los programas más destacados en el área de modelado de objetos 3D son: Maya y Blender.

Maya

El software Maya es una solución integrada de modelado, animación y renderizado 3D basada en una arquitectura abierta. Este software surgió de la fusión de tres importantes compañías dedicadas al desarrollo de efectos especiales: (a) Wavefront, (b) Thomson Digital Image y (c) Alias Power Animator. En 2003 fue comprado por SGI y posteriormente, en 2006 lo adquirió Autodesk dándole el nombre de Autodesk Maya.

Este software fue elegido para el desarrollo del proyecto debido a las características que se ajustan a los requerimientos de modelado 3D. Entre estas características se presentan:

1. Contiene una amplia variedad de fotogramas, edición no lineal, y herramientas avanzadas de edición de animación de personajes. Un ejemplo son los robots de la película Transformers y las criaturas mágicas en las películas de Harry Potter, que fueron animados con esta herramienta (Maya, 2007).

2. Ofrece un completo paquete de polígonos avanzados y herramientas de asignación de texturas y modelado de superficies de subdivisión.

3. Posee herramientas para la creación de cabello, física de bala, creación de caracteres, partículas, fluidos, creación de ropa, seguimiento de la cámara, mapeado UV y coloreado por vértice.

4. Dispone de un flujo de trabajo y una interfaz unificada del usuario para renderización.

5. Incluye el lenguaje de programación Python® lo que otorga un plus al momento de personalizar algunas funciones del software. Además incluye una amplia API que permite a los desarrolladores personalizar y escalar la implementación.

6. Cuenta con versiones ejecutables para sistemas operativos Windows y OS X.

Blender

Blender es una aplicación integrada que permite la creación de una amplia gama de contenido en 2D y 3D (Blender, 2011b). El software inicialmente fue desarrollado por Ton Roosendaal co-fundador del estudio de animación holandés NeoGeo, más tarde, en 1995 fundó una nueva compañía llamada Not a Number (NaN) como un spin-

off de NeoGeo para fomentar el mercado y desarrollar Blender. Hasta el día de hoy Blender ha continuado como software libre, con el código fuente disponible bajo la licencia GNU GPL (Blender, 2011c).

Las funciones más destacables que ofrece este software son:

1. Posee una amplia gama de herramientas esenciales para la creación de contenido 3D, incluyendo modelado, mapeado uv, texturizado, rigging, skinning, animación, partículas y otras simulaciones, scripting, renderizado, composición, post-producción, y creación de juegos.

2. Es una herramienta multiplataforma, con una interfaz gráfica OpenGL que es uniforme en todas las plataformas (personalizable con scripts python), listo para usar para todas las versiones de Windows (XP, Vista, 7), Linux, OS X, FreeBSD, Sun y muchos otros sistemas operativos.

3. Dispone de una arquitectura 3D de alta calidad que permite la creación de flujo de trabajo rápido y eficiente.

4. Cuenta con el más importante apoyo de comunidad de usuarios en diversos foros especializados.

5. El ejecutable generado es pequeño lo que facilita su distribución.

Motores gráficos

La creación de videojuegos es llevada a cabo mediante software especializado en la creación de mundos virtuales. Este software es el encargado de proporcionar las herramientas para realizar las tareas de renderizado, entre las que se encuentran: (a) la reproducción de sonidos y animaciones, (b) scripting, (c) la administración de la me-

moria y (d) un escenario gráfico. El programa elegido para el desarrollo de este proyecto fue Unity.

Unity

Unity es un ecosistema completo de herramientas y servicios diseñados para el desarrollo de un negocio exitoso a través de la creación de juegos multiplataforma y contenido interactivo (Unity, 2014a).

El software, Unity Tecnhologies empezó gracias a que David Helgason, Nicholas Francis y Joachim Ante decidieran crear un motor de videojuegos que pequeñas y grandes empresas pudieran usar por igual, con el plus de tener acceso a diferentes plataformas a fin de evitar la necesidad de tener que programar el mismo juego para cada plataforma. El software fue lanzado en 2005 en el escenario de la Conferencia Mundial de Desarrolladores de Apple (WWDC) (Unity, 2010). Su núcleo está escrito en el lenguaje de programación C++ (Johansen, 2010).

Hoy Unity es el motor de juegos número uno en el mercado y en la educación. Por mucho, es la tecnología líder de juegos para el iPhone y el iPad. Unity desarrolla juegos para marcas globales como Axe, Cadbury Crunchie, Cartoon Network, Cheerios. Al mismo tiempo, decenas de miles de desarrolladores están aprovechando las innovaciones de Unity para crear innovaciones propias - nuevas interfaces de control únicos, la transmisión de deportes en tiempo real, las nuevas formas de interactuar con las redes sociales, nuevas formas de hacer publicidad de productos y juegos de realidad aumentada (Unity, 2010).

Unity Technologies ofrece el soporte de la plataforma más amplia de cualquier motor de juego en el mercado, esto se puede apreciar en la Figura 16.

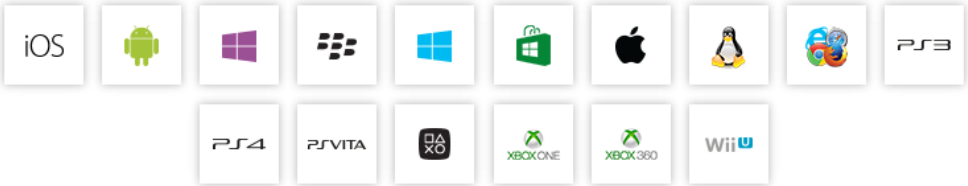

Figura 16. Lista de plataformas que soporta Unity para el desarrollo.

Unity 3D ofrece 2 versiones de software: Unity y Unity Pro. La diferencia radica en las características que ofrece cada versión: el soporte de texturas 3D, motor de física, soporte de reproducción de videos, entre otras; siendo la versión más completa Unity Pro (Unity, 2014b).

Plugins

Un plugin es un módulo de software que añade funcionalidades a un software principal. También son conocidos como complementos, o extensiones. No son parches ni actualizaciones del software.

NGUI: Next- Gen UI

NGUI fue desarrollado por Michael Lyashenko quien antes de fundar Tasharen Enterteiment era Jefe de Programación en Gameloft. Este plugin es un sistema de interfaz de usuario y un framework de notificación de eventos para las dos versiones de Unity. Además está escrito en C# y cuenta con un código limpio y simple, con un enfoque minimalista. Muchas clases de comportamiento se mantienen por debajo de las 200 líneas de código. Para un programador esto significa un tiempo mucho más fácil cuando se trata de extender la funcionalidad del kit para ajustar la existente. Para todos los demás, esto significa un mejor rendimiento, menos frustración y más diversión (NGUI, 2011).

NGUI es el futuro sistema Unity3D GUI de Unity3D. Por supuesto, como alternativa al sistema nativo (Potapenko, 2013).

Entre las principales funciones que ofrece esta plataforma se encuentran:

1. Tiene una integración completa con la pestaña de Inspector de Unity.

2. Para observar los resultados no hay necesidad de pulsar el botón Play.

3. El contenido desplegado en la vista de Scene se observa en la vista Game.

4. Su arquitectura está basada en componentes; no hay archivos DLL o recursos externos.

5. Ofrece soporte completo para iOS / Android, Blackberry, Win8, WP8 y Flash.

6. Proporciona un sistema de eventos flexible.

7. Permite construir interfaces de usuario complejas que se hacen en una llamada. Es como trabajar con Unity, solo hay que arrastrar y soltar los controles ya hechos, como botones, casillas de verificación y barras de desplazamiento, entre otros.

8. Incluye facilidades para la iluminación, mapping normal y refracción.

9. Facilita la creación de los atlas en el editor e incluso modificarlos o importarlos.

Finger Gestures

Finger Gestures es un paquete de scripting para Unity. Su función es reaccionar y detectar fácilmente los gestos de entrada comunes que son realizados con un mouse o en la pantalla táctil de un dispositivo usando uno o dos dedos (Ravaine, 2012).

Las principales prestaciones que este plugin ofrece a los desarrolladores son:

1. Ofrece soporte para gestos como tap, long-press, drag, swipe, pinch and twist, que se muestran en la Figura 17.

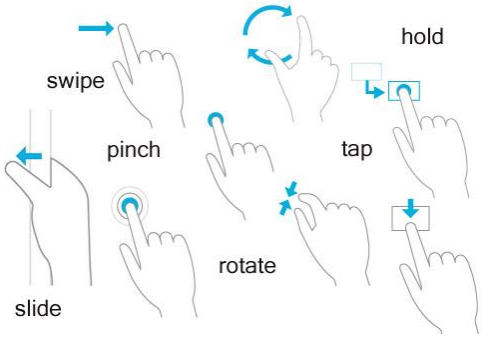

Figura 17. Principales gestos para dispositivos móviles.

2. Posibilita la detección personalizada de gestos definiendo plantillas Point CloudRecognizer. El PointCloudRecognizer compara la entrada del usuario contra un conjunto de plantillas de gestos y devuelve el valor más próximo, junto con una puntuación de adaptación y un valor de distancia. Además incluye un editor de gesto integrado. La Figura 18 muestra un ejemplo del editor de gestos.

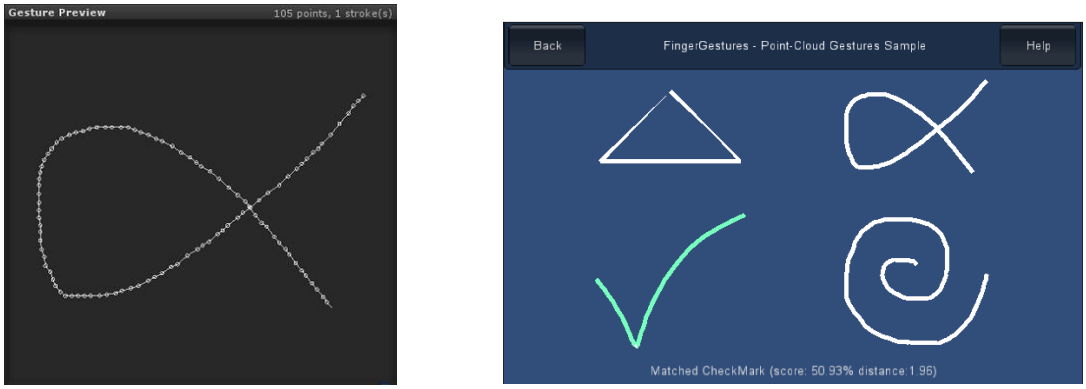

Figura 18. Plantilla PointCloudRecognizer realizada con el editor de gestos.

3. Incluye una serie de scripts para arrastrar, escalar y rotar objetos individuales además de varios controladores para cámara basados en gestos

4. Posee una arquitectura multiplataforma que ha sido probado para escritorio, iOS y Android.

5. Está escrito en C#, con un código limpio y simple. Además incluye ejemplos que presentan las funcionalidades.

Plataformas

La selección de las plataformas a las que va a ser exportado el producto constituye uno de los factores principales en su difusión ya que representa la capacidad de llegar a los usuarios. Las plataformas elegidas para el desarrollo de este proyecto fueron: iOS y Android.

iOS & Android

A continuación se presentan los hallazgos encontrados por Vision Mobile (2014a, 2014b):

1. La brecha en la cuota de mercado entre Android e iOS es aun mucho más estrecha en América del Norte que en otras regiones del mundo. Apple sigue siendo el principal fabricante de teléfonos inteligentes en el mercado de América del Norte, Samsung viene en segundo lugar con una participación de mercado de 26,7%.

2. El desarrollo para navegadores móviles es más popular en América del Norte (18% de los desarrolladores) en comparación con el resto del mundo.

3. Para la mayoría de los desarrolladores profesionales en América del Norte, que tienen como objetivo el desarrollo de aplicaciones, iOS ofrece generalmente las mejores oportunidades de monetización.

4. Se estima que en 2014 más de 2.9 millones de personas de todo el mundo estarán involucradas en el desarrollo de aplicaciones.

5. Android y iOS representaron más del 94% de las ventas de teléfonos inteligentes, en el último trimestre de 2013.

6. De los 2,3 millones de desarrolladores que se han dirigido a las plataformas móviles en el 2013, el 71% desarrollan para Android, el 55% para iOS y 37% para uso de HTML5 para crear aplicaciones web o sitios web móviles.

7. El 59% de los desarrolladores de iOS en primer lugar construirían para iOS y luego para otras plataformas.

8. Los desarrolladores de aplicaciones en América del Norte o Europa Occidental prefieren construir para iOS en primer lugar, mientras que en el resto del mundo los desarrolladores de aplicaciones prefieren construir primero para Android.

Arquitectura de la aplicación: MVC

Controlar la complejidad es la esencia de la programación (Kernighan y Plauger, 1976). Una de las formas de reducir la complejidad es escribir software modular, separando las funcionalidades en mecanismos y reglas. Esto dará como resultado que la complejidad global será mucho menor que si se implementara un solo proceso monolítico para alcanzar la misma funcionalidad. Al emplear esta forma de trabajo se reducirá la vulnerabilidad a los errores y disminuirán los costos del ciclo de vida del desarrollo. Otra buena práctica en el desarrollo es implementar las reglas en el Front End y los mecanismos en el Back End (Raymond, 2003).

El patrón MVC (Model- View-Controler) es una arquitectura de diseño de software que divide los componentes de una aplicación en tres niveles, descritos a continuación:

Modelo: Es el dominio de la aplicación, contiene los datos con los que trabajará la aplicación, es la “lógica del negocio”. Además es independiente de la capa de Vista y Controlador.

Vista: Esta capa es la presentación de la capa Modelo. Está conformada por las interfaces con las que el usuario interactúa, recibiendo y enviando información

Controlador: es la encargada de reaccionar a la petición del cliente, ejecutando la acción adecuada y creando el modelo pertinente.

En la Figura 19 puede apreciarse de manera simplificada la forma en la que fue implementado el modelo de arquitectura MVC en la aplicación TroubleshootingAR.

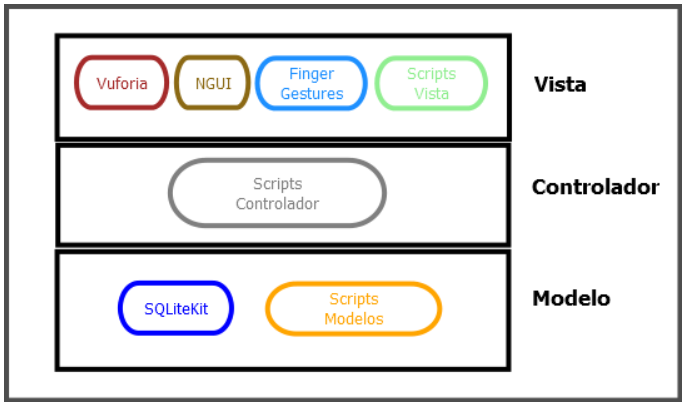

Figura 19. Arquitectura general de la aplicación.

Beneficios de utilizar MVC

Algunos beneficios directos que se obtienen al implementar el patrón MVC son (Prakash, Kumar y Mishra, 2013; Sridaran, Padmavathi, Iyakutti y Mani, 2010):

1. Facilita la reutilización de las clases, al disminuir la interdepencia del código.

2. La interacción y la forma como se ve el software puede ser cambiado sin afectar la lógica de negocio.

3. El patrón MVC proporciona escalabilidad y un mantenimiento más sencillo de las aplicaciones, ya que separa la lógica de negocio (el modelo) y la presentación (la vista).

4. Es fácil de delegar el trabajo de desarrollo dentro de los desarrolladores o los miembros del equipo y distribuir el esfuerzo total. El patrón MVC se asegura de que los cambios en una aplicación no afectarán a la otra aplicación.

5. No afecta el flujo de trabajo de la aplicación. Como ejemplo el diseñador web y el desarrollador web pueden trabajar de forma independiente.

6. La migración de programas heredados ha llegado a convertirse en una tarea menos compleja desde que la capa modelo y la capa controlador están totalmente separados en MVC.

Base de datos

El uso de las bases de datos ya se ha extendido de los servidores hacia los dispositivos móviles, convirtiéndose en una herramienta indispensable en el desarrollo de las aplicaciones, debido a que esta permite almacenar un conjunto de datos pertenecientes a un mismo contexto. De acuerdo a los requerimientos de la aplicación en este proyecto fue seleccionada la Base de Datos SQLite.

SQLite

SQLite es una herramienta de software libre, que permite almacenar información en dispositivos empotrados de una forma sencilla, eficaz, potente, rápida y en equipos con pocas capacidades de hardware, como puede ser una PDA o un teléfono celular. SQLite implementa el estándar SQL92 y también agrega extensiones que facilitan su

uso en cualquier ambiente de desarrollo. Esto permite que SQLite soporte desde las consultas más básicas hasta las más complejas del lenguaje SQL. Más importante aún es que se puede usar tanto en dispositivos móviles como en sistemas de escritorio, sin necesidad de realizar procesos complejos de importación y exportación de datos, ya que existe compatibilidad al 100% entre las diversas plataformas disponibles, haciendo que la portabilidad entre dispositivos y plataformas sea transparente (Rómmel, 2007). Las plataformas principales dónde SQLite se encuentra funcionando son: a) Windows 95, 98, ME, 2000, XP y Vista, b) Windows CE & Pocket PC, c) Mac OSX, d) Linux y e) Symbian.

CAPÍTULO IV

RESULTADOS

En este capítulo se describen los procesos seguidos para crear la aplicación TROUBLESHOOTING AR para la máquina Helitronic. Se abordan los desafíos y se describe la forma en la que se solucionaron y también se analizan los logros obtenidos en función de los requerimientos del proyecto.

Planificación del proyecto

Walter es considerado líder mundial en el rectificado de herramientas, comprende una amplia gama de productos que van desde rectificadoras de herramientas de uso general y rectificadoras para micro-herramientas, pasando por máquinas para implantes ortopédicos complejos y mucho más (Walter, 2014a).

En el mercado de las herramientas de precisión para metal y madera la máquina Helitronic Basic es capaz de diseñar cada una de las piezas eficazmente. Por este motivo, Helitronic Basic es la primera opción para realizar trabajos de reafilado (Walter, 2014b).

Su campo de aplicación comprende las herramientas simétricas rotativas para el mecanizado de metal y madera, herramientas especiales y piezas con geometrías complejas, como: fresa frontal HM, broca escalonada helicoidal, fresa de desbaste para madera, broca para herrajes, broca para agujeros de clavijas, fresa de corte dual.

Las características principales que ofrece la máquina se presentan a continuación: (a) baja vibración, fundición gris sólido, (b) ejes lineales X, Y, Z con accionamiento lineal tipo bola, (c) ejes A y C con accionamientos de rosca giratoria, (d) husillo accionado por correa con dos extremos y (e) FANUC, el estándar mundial para equipos de control.

Como se puede observar el funcionamiento de las máquinas industriales es muy complejo y por lo tanto extenso. Debido a la relevancia de algunos procesos para las máquinas Helitronic Basic, se consideraron cuatro escenarios principales para su estudio: (a) aislamiento de fallas del Power Supply Module (PSM), (b) montado/desmontado de los ejes lineales, (c) desmontando del eje A, (d) desmontando del eje C y (e) reconstrucción del eje C.

Desarrollo del proyecto

Uno de los objetivos del proyecto consistió en desarrollar una aplicación para la mejora de los tiempos en reparación o mantenimiento de las máquinas Walter. Se espera obtener a mediano plazo un aceleramiento cognitivo en los técnicos de la planta industrial y en segunda instancia la disminución en los costos de reparación de maquinaria industrial, específicamente la máquina Walter Helitronic.

Arquitectura de software

La elección del modelo de arquitectura constituye un factor importante ya que de ello depende la facilidad para agregar nuevos módulos a la aplicación así como el mantenimiento eficiente. El resultado final de la implementación del modelo de arquitectura MVC en la aplicación se presenta a continuación:

Capa Modelo

La capa modelo del proyecto es el nivel básico, está estructurada por los componentes que se muestran en la Figura 20.

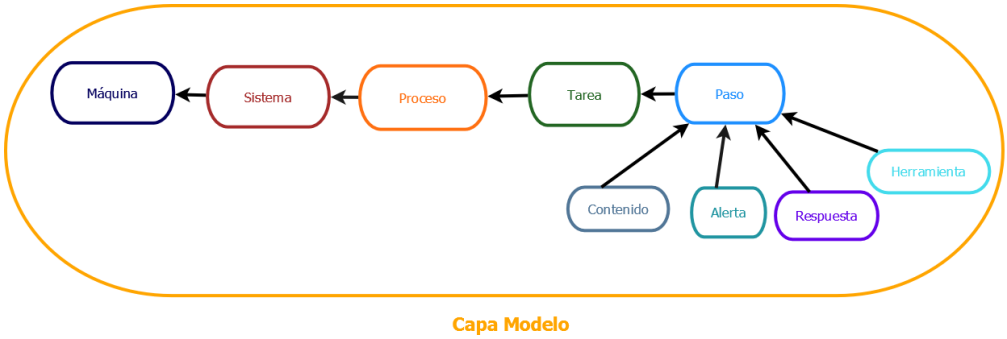

Figura 20. Componentes de la aplicación presentes en la Capa Modelo.

A continuación se da una breve explicación de la información que proporciona cada componente.

Máquina: Este componente carga los procesos pertinentes a la maquina elegida.

Sistema: Este componente maneja la información relativa al tipo de sistema que se pretende reparar, por ejemplo Sistema Eléctrico, Sistema Mecánico, Sistema Hidráulico entre otros.

Proceso: Esta clase lista los procesos que se pueden realizar en cada sistema, por ejemplo inspección del eje A, chequeo doble del circuito de seguridad, resolución de problemas del controlador del motor.

Tarea: En este componente se manejan las tareas que están asociados con los procesos, por ejemplo módulo de la fuente de alimentación, modulo del servoamplificador.

Paso: En esta clase se manejan los pasos requeridos para completar una tarea,

además cuenta con subcomponentes para una mayor flexibilidad por parte de la aplicación en el manejo de los datos.

Alerta: Debido a la delicadeza de algunas piezas y el riesgo potencial presentado por otras es necesario que cada paso sea capaz de contener alguna alerta para el usuario.

Contenido: La aplicación está preparada para el manejo de información como modelos 3D, imágenes y vídeos. Estos son los elementos necesarios para dar las indicaciones que sirven para completar el paso.

Herramientas: Especifica las herramientas necesarias para ejecutar el paso.

Respuesta: Especifica los posibles estados que pueda tener la máquina durante el paso así como el siguiente relacionado con ese estado.

Cada uno de estos componentes a su vez está compuesto por dos clases, una clase que funciona a manera de contenedor de la información y otra que se encarga de la gestión de los datos. Estas relaciones pueden ser apreciadas en la Figura 21.

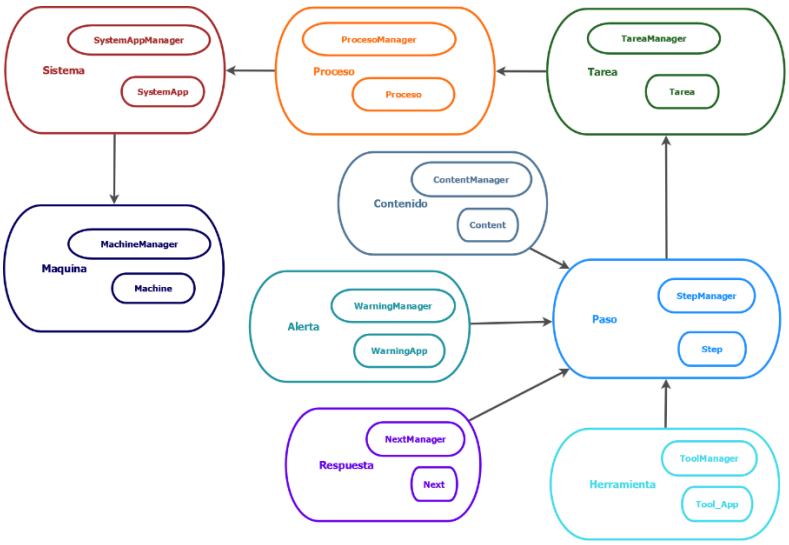

Figura 21. Clases principales de la aplicación.

Capa Vista

Las tecnologías mostradas en la Figura 22 son utilizadas en la Capa Vista como se muestra en el siguiente diagrama:

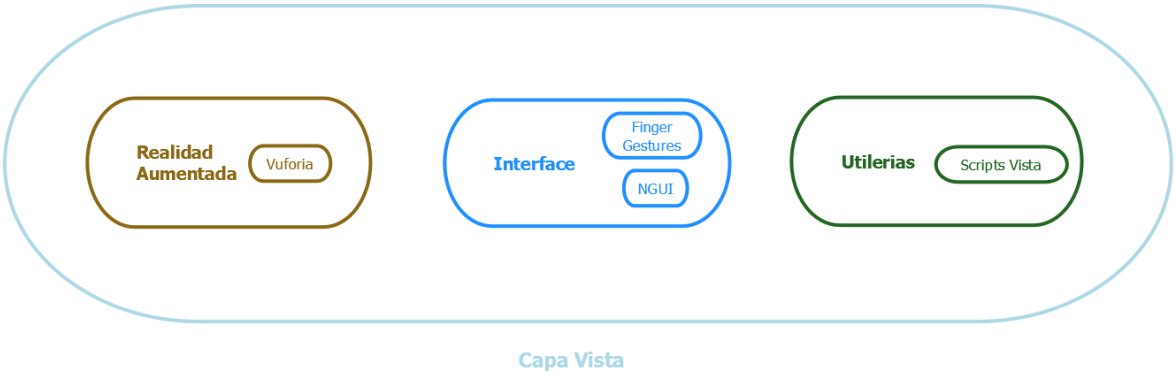

Figura 22. Tecnologías utilizadas en la Capa Vista.

De acuerdo al análisis de los requerimientos el prototipo final se compone de seis escenas principales: (a) 01-InitDB, (b) 02-Portada, (c) 03-Menú, (d) 04-StepView, (e) 00-Calibración y (f) 05-Catálogo.

El diagrama de la Figura 23 muestra en resumen cada escenario y su estructura interna:

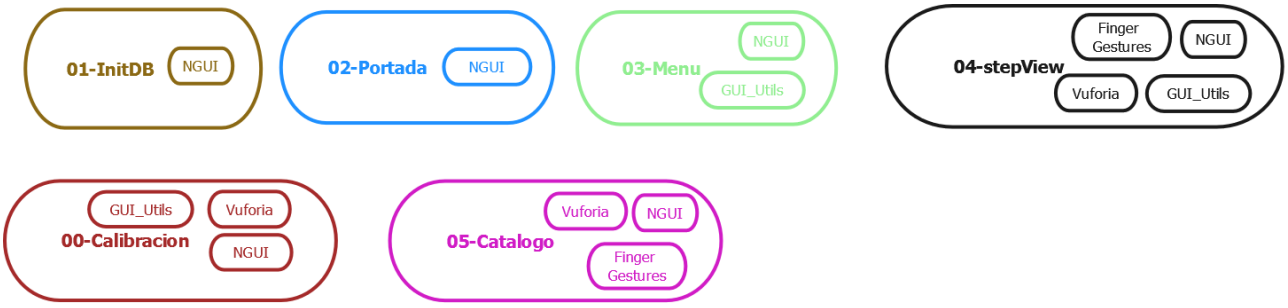

Figura 23. Estructura interna de cada escenario.

En las siguientes secciones se presenta una explicación de cada escenario.

Escenario 01-InitDB

La aplicación Troubleshooting RA inicia con una pantalla de espera tal como se muestra en la Figura 24. Esta pantalla tiene una duración de dos segundos y su función es configurar el dispositivo realizando algunos procesos internos, tal como: copiar la base de datos, para asegurar el correcto funcionamiento de la aplicación para su uso posterior. Esto se logra mediante la implementación de co-rutinas. La función *MonoBehaviour.StartCoroutine()* devuelve una *Coroutine*. Las instancias de esta clase son usadas solamente como referencia de esas co-rutinas y no tienen propiedades ni funciones. Una co-rutina es una función que puede suspender su ejecución (yield) hasta que la orden *YieldInstruction* concluye.

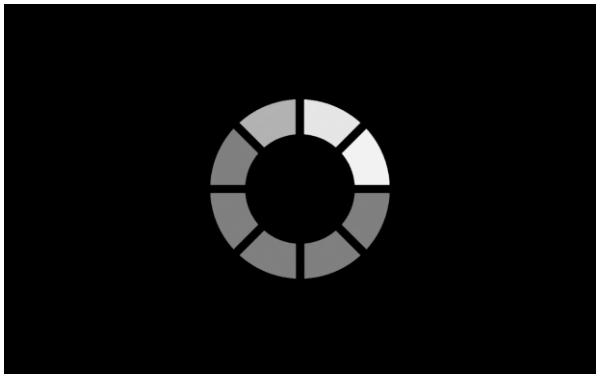

Figura 24. Pantalla de espera mientras se copia la base de datos.

La interacción en este nivel se da a través del algoritmo *CopyDataBase()* el cual se encarga de copiar la base de datos que se encuentra en la carpeta StreamingAssets del proyecto de Unity a una carpeta del móvil, ya que durante la copia de la base de

datos se tiene el problema de que al buscarla no se encuentra debido a que se transfiere como un archivo de solo lectura y no se logra establecer la conexión con la base de datos, haciéndose necesario crear una carpeta en el dispositivo móvil para copiar el archivo de la base de datos a ese directorio. Este contratiempo se resuelve implementando el método *Application.PersistentDataPath*(). El valor que se requiere es una ruta de un directorio donde se espera que los datos se mantegan almacenados durante la ejecución de la aplicación. Al publicar en iOS y Android el *PersistentDataPath* apuntará a un directorio público en el dispositivo. Los archivos en esta ubicación no se borrarán con cada actualización de la aplicación. Sin embargo, se debe considerar que esta acción no es infalible contra las acciones de los usuarios.

Para un mejor mantenimiento de la aplicación se opta por crear una clase llamada Constantes, en la cual se encuentran el valor de las variables que contienen la ruta de los directorios, los nombres de las escenas y los valores del menú.

Considerando la magnitud del proyecto y el dinamismo de las aplicaciones, se observa que era necesario la implementación de una base de datos, de manera que se opta por SQLite es una base de datos embebida para dispositivos móviles.

La base de datos está organizada en nueve tablas: contents, machines, nexts, process, steps, systems, tasks, tools y warnings. A continuación se da una breve descripción:

La tabla contents almacena los modelos, la carpeta en la que se encuentran y con qué tarea están relacionados.

La tabla de machines almacena la información referente a las máquinas para la cual la aplicación está preparada para trabajar.

La tabla de nexts es la que controla el flujo de la aplicación, establece el orden de los pasos a realizar para la ejecución de esa tarea.

La tabla de process indica los procesos de mayor importancia que pertenecen a un sistema.

La tabla steps almacena la serie de pasos que deben ser realizados en base a una tarea para la solución del problema.

La tabla systems es la principal de la aplicación, almacena los sistemas de acuerdo al tipo de máquina industrial.

La tabla tasks consiste en las actividades que se realizan con regularidad para la verificación de un problema con la máquina.

La función de la tabla tools es indicar al técnico la herramienta que debe ser utilizada para desarmar el componente que está siendo revisado.

Por último la tabla warnings almacena los avisos que deberán ser presentados al técnico durante el uso de la aplicación en caso de que la tarea afecte a una pieza que sea delicada. Para una mejor comprensión de la estructura de la base de datos se presenta un diagrama entidad-relación de la base de datos en la Figura 25.

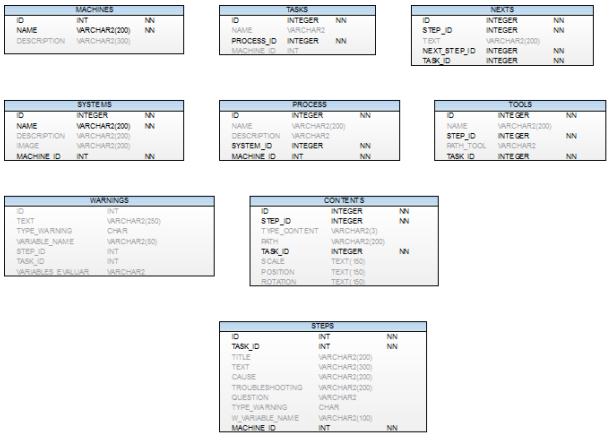

Figura 25. Diagrama entidad-relación de la BD de la aplicación.

Para acceder a la base de datos fue necesario implementar el plugin SQLiteKit y modificar la clase de la conexión a la base de datos de acuerdo a los parámetros de la aplicación. La clase quedó como se muestra en la Figura 26.

```
5 public class Conexion  {
6
7     protected SQLiteDB db = null;
8
9     public void Open(){
10        try{
11            string dbPath= Application.streamingAssetsPath+Constantes.DB_PATH;
12            db=new SQLiteDB();
13            db.Open(dbPath);
14        }catch(Exception e){
15            Console.WriteLine(e.ToString());
16        }
17
18    }
19    public void Close(){
20        db.Close ();
21    }
22 }
```

Figura 26. Clase encargada de hacer la conexión a la base de datos.

El siguiente nivel que carga la aplicación es el escenario de la portada.

Escenario 02-Portada

La segunda pantalla que se muestra en la aplicación es la portada. Esta incluye las entidades que están involucradas en el desarrollo del proyecto, la máquina industrial Helitronic y el sistema de ejes que lo conforma. En la Figura 27 se muestra la portada.

La pantalla de portada está formada por la cámara de NGUI y un Game Object vacío que se le agrega la textura de la portada. Se agregó el componente UIPanel que contiene los anchors para calcular el tamaño de la pantalla del dispositivo móvil y que la portada se ajuste.

Figura 27. Portada de la aplicación móvil.

En este nivel la interacción se da a través del script de *Portada.cs*. En esta clase se crea un botón que abarca toda la pantalla y al momento de tocar la pantalla se ejecuta una serie de eventos, uno de ellos es la llamada al método *OnCreatedButton-Click()*, que tiene como función cargar la pantalla del menú de la aplicación. En la Figura 28 se muestra el script *Portada.cs*.

```
4 public class Portada : MonoBehaviour {
5     public UIButton clickable;
6     void Start () {
7         EventDelegate.Add(clickable.onClick,OnCreatedButtonClick);
8     }
9
10    void Update () {
11        if (Input.GetKeyDown (KeyCode.Escape)) {
12            Application.Quit();
13        }
14    }
15
16    void OnCreatedButtonClick() {
17        Application.LoadLevel (Constantes.MENU_SCENE);
18    }
19 }
20
```

Figura 28. Script que crea el botón en la portada para pasar al nivel del menú

Escenario 03-Menú

El menú consta de un solo escenario, en el cual todos los niveles y subniveles

que lo integran se van cargando de forma dinámica de acuerdo a los elementos capturados en la base de datos. Este está dividido en los siguientes niveles: máquinas, sistemas de la máquina seleccionada, procesos, tareas y por último los pasos que constituyen la realización de la tarea elegida.

La Figura 29 hay que mencionar que es el menú principal de la aplicación, muestra las máquinas que pueden ser reparadas mediante el uso de la aplicación. La Figura 30 es el nivel dos del menú, despliega los sistemas que integran la máquina elegida.

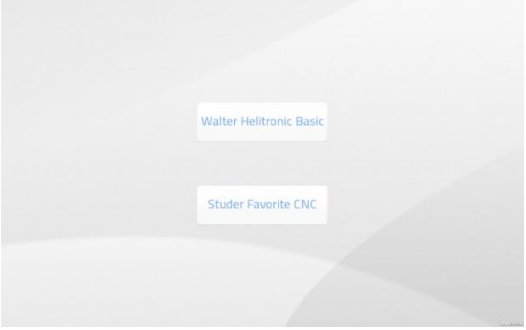

Figura 29. Listado de máquinas Industriales.

Figura 30. Sistemas de la máquina.

La Figura 31 muestra los procesos de acuerdo al sistema elegido para su verificación mediante el uso de la aplicación. Dentro de cada proceso existen tareas relacionadas con el mismo. Esto puede apreciarse en la Figura 32.

Figura 31. Procesos de la máquina.

Figura 32. Tareas de un proceso.

El escenario de menú está integrado por la cámara de NGUI y un GameObject vacío al cual se le agregó la textura asignada para el menú mediante el plugin NGUI para agregar el componente UITexture. Para el desplazamiento entre los niveles del menú se utilizó la animación tween, sin embargo fue necesario quitar el anchor del

UIPanel y generarlo nuevamente en tiempo de ejecución porque al momento de realizar la animación la pantalla parecía que temblaba.

La interacción en este subnivel se da mediante dos scripts: *UI_Manager.cs* y *Menu_Manager*.cs. El script de *UI_Manager.cs* es el encargado de manejar la escena, escuchando los eventos mediante el método *EscuchaEventos()* que es el encargado de asignar a las variables los valores. El método *BackEnd_Utils.activaHijosPorIndex()* es el encargado de regresar los elementos que están activos de acuerdo al id para que se desplieguen en la pantalla. Mientras que el script *Menu_Manager.cs* es el encargado de construir los elementos para los botones que deben ser desplegados.

Además fue necesario implementar una clase estática llamada *GeneralData.cs* con el fin de compartir información entre las escenas.

Escenario 04-StepView

Una vez que ha sido seleccionada la tarea se presenta la pantalla más relevante de la aplicación, se encarga de desplegar el paso actual, mostrar las herramientas necesarias para su ejecución y los posibles estados de la máquina de los cuales depende el siguiente paso a ejecutar. La Figura 33 muestra el escenario por medio del cual es posible llevar a cabo con éxito una tarea previamente seleccionada.

El escenario de StepView está compuesto de cinco luces a fin de asegurar la correcta iluminación de cada modelo 3D, la cámara del SDK Vuforia, el prefab del plugin Finger Gestures que permitirá el reconocimiento de gestos para rotar o escalar el modelo 3D y el prefab ImageTarget de Vuforia.

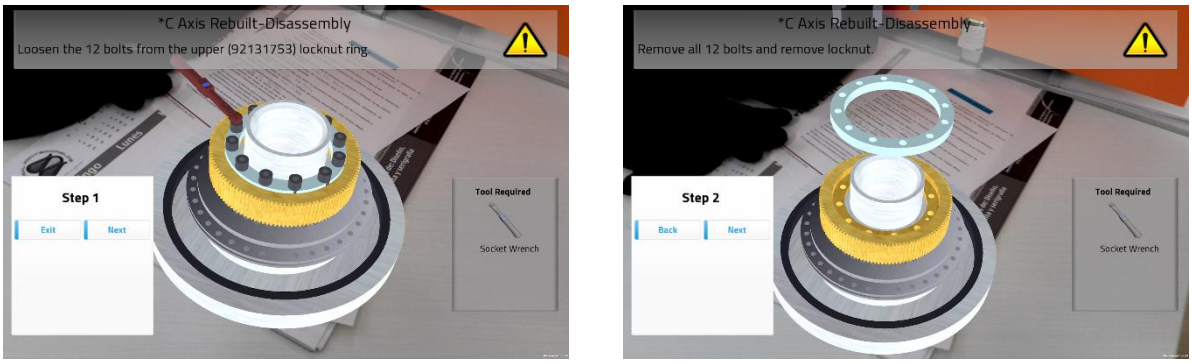

Figura 33. Captura de pantalla paso uno con alerta y paso dos.

Para el desplazamiento entre estos niveles se desarrollaron dos scripts: *SceneAR_Manager.cs* y *AR_Manager.cs*. El script *SceneAR_Manager.cs* es el que se encarga de escuchar los eventos mediante el método *EscuchaARScene(object o, EventosArgsAR e)* desde donde se llama al método *CargaPaso()* que se encuentra en la clase AR_Manager.cs. Este es el encargado de traer el siguiente paso o regresar al menú, eliminar el fbx, y desplegar la tabla de herramientas e instrucciones del paso. El resultado final puede verse en el Apéndice A.

Escenario 00-Calibración

El escenario de calibración está compuesto por cuatro botones: (a) Back, (b) Next, (c) Apply y (d) Graba DB. Además incluye seis Text Boxes, que permiten introducir valores de acuerdo a la rotación, posición y escala que se desea que adquiera el FBX como se muestra en la Figura 34.

Figura 34. Pantalla de calibración de modelos 3D.

El script desarrollado para la interacción en este nivel se llama *CalibracionManager.cs*. Este contiene los siguientes métodos:

1. *resetVectores()*, su tarea es cambiar los String guardados en los campos scale, position y rotation de la tabla contents a Vector3. En caso de que no tuvieran algún valor en la base de datos se le asigna 0.5 a la escala y cero a la posición y rotación.

2. *clickApply()*, que a su vez llama a las funciones *descargaValores()*, *aplicaValores ()* y *cargaValores ()*. Siguiendo el orden en que se encuentran listados, sus funciones son respectivamente, subir los valores a memoria, asignar el valor a las variables, y cargar los valores en la interfaz de la aplicación.

3. *Onclick Write()*, es el encargado de almacenar los valores en la base de datos utilizando el método *Write_Files.guardaCalibracion()* su tarea es crear un archivo de tipo CSV, en el que de acuerdo al id de la tarea se escribe la línea en el mismo renglón que coincide con el id de la tarea. Fue necesario crear un archivo debido a que la base de datos se copia cada vez que se inicia la aplicación.

4. *clickChangeProcess()*, para modificar los valores de calibración del modelo 3D

busca el proceso en la carpeta Resources/FBX/Process+taskId y en la interfaz carga los valores que están almacenados en la tabla contents.

Escenario 05-Catálogo

La pantalla que se muestra en la Figura 35 tiene la función de mostrar las piezas de la máquina. Internamente está conformada por una luz direccional, el plugin Finger Gestures para la manipulación del modelo 3D, el Image Target y los elementos del UIRoot del plugin NGUI. La interacción en este escenario es controlada por el script PartController.cs, el método principal es *loadContent()* y su función es cargar dinámicamente el modelo FBX. El resultado final puede apreciarse en el Apéndice A.

Figura 35. Pantalla de catálogo de partes.

Capa Controlador

En la Capa Controlador se encuentran los 51 scripts que se encargan de ejecutar los procesos que son responsabilidad de la capa Modelo, para tener como resultado las interacciones que lleva a cabo el usuario en la capa Vista.

Como puede apreciarse en la Figura 36 se cuenta con un controlador para cada

escenario, cada controlador está compuesto por uno o más scripts encargados de ejecutar las llamadas a la capa Modelo o en su defecto ejecutar scripts que realizan tareas específicas.

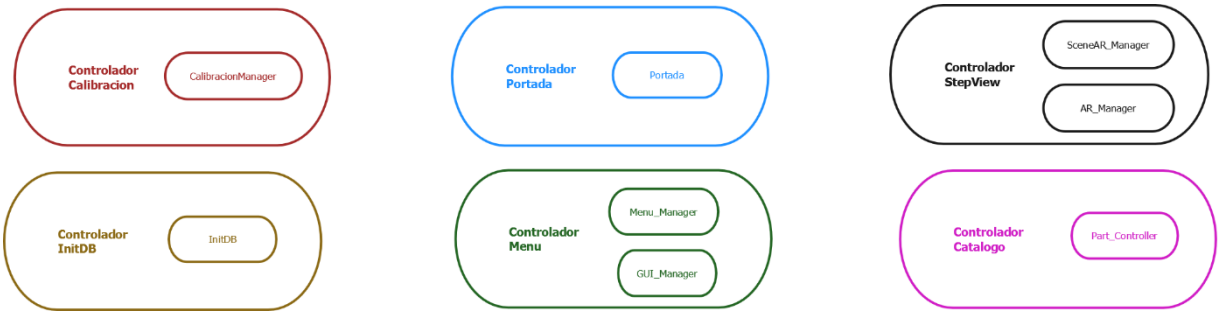

Figura 36. Relación script-controlador.

CAPÍTULO V

CONCLUSIONES Y TRABAJOS FUTUROS

En este documento se ha presentado un proyecto interdisciplinario donde se vieron envueltos desarrolladores, diseñadores y el gerente de una planta industrial. El resultado fue una aplicación móvil y multiplataforma que une tecnologías recientes como librerías de realidad aumentada y motores gráficos. Esto le da un enfoque novedoso al conocimiento básico que los técnicos de la planta emplean diariamente para el desempeño eficiente de sus labores.

Conclusiones

Los resultados que se desprenden de este proyecto se consideran positivos ya que la aplicación cubre una serie de procesos que son básicos en la reparación y mantenimiento de la máquina Walter Helitronic, aportando la ventaja del uso de las TIC para la compresión de procesos complejos. La sección de conclusiones se divide en: (a) conclusiones de la aplicación y (b) conclusiones del problema.

Conclusiones de la aplicación

La aplicación puede ser mejorada en diversos aspectos. Sin embargo, cumple con los objetivos planteados: una aplicación multiplataforma para la reparación y mantenimiento de la máquina industrial Walter Helitronic en el que las acciones como el desplazamiento y la selección de objetos son controlados por medio de gestos.

Conclusiones del problema

Las tecnologías implementadas en el proyecto como: (a) librerías de realidad aumentada (b) diseño de modelos 3D y (c) software de motores gráficos, proporcionan un enfoque novedoso a procesos industriales que podrían resultar difíciles de comprender sin una representación visual.

Al decidir realizar una aplicación para la reparación y mantenimiento de algunos procesos de máquinas industriales se percibe cómo la tecnología podría llegar a ser de gran ayuda en entornos industriales propiciando un aceleramiento cognitivo en los técnicos de las plantas de producción industrial y una mejora en los tiempos y costos.

En cuanto a la tecnología empleada en la mayoría de los casos el software elegido cumplió con las expectativas esperadas al ofrecer características potentes para el cumplimiento de los objetivos planteados.

Trabajos a futuro

El desarrollo de este proyecto ha abierto un panorama más amplio a los involucrados en el desarrollo de aplicaciones enfocados al entorno industrial y promete ser un detonante que logre interesar a más profesionales de las TIC a emprender proyectos de investigación que contribuyan con generación de más conocimiento para el crecimiento en estas áreas.

Mediante la experiencia obtenida en el desarrollo de este proyecto se ofrecen las siguientes recomendaciones o trabajos futuros.

Mejoras a la aplicación móvil

En esta sección se describen los cambios que pueden aplicarse a la aplicación:

1. Desarrollo de un algoritmo que realice una comparación de los campos de la base de datos existente contra los campos modificados de la nueva base de datos con el fin de evitar el reemplazo de todas las tablas.

2. Probar y analizar la librería de Metaio SDK para el desarrollo de la aplicación.

3. Creación de modelos tridimensionales con la misma escala para la mejor visualización de estos en la aplicación.

Otros usos a las tecnologías

1. Estudio que determine si la aplicación influye positivamente en el conocimiento de los técnicos propiciando una mejora en los tiempos de reparación y una disminución en los costos.

2. Desarrollo de prototipos para el aprendizaje basado en juegos relacionados con la educación en diversas materias elementales, tales como matemáticas o español. Esto debido a que en México, el nivel de ansiedad hacia las matemáticas es alta, de acuerdo a un estudio realizado por la OECD (2012) el 75% de los alumnos mexicanos declara estar de acuerdo o muy de acuerdo con la afirmación “frecuentemente me preocupa que tendré dificultades en clases de matemáticas.

3. Desarrollo de un prototipo utilizando AssetBundles para la actualización de los modelos de la aplicación en caso de que se requiera agregar otra máquina industrial para su mantenimiento y reparación.

APÉNDICE A

CÓDIGO FUENTE EN C#

GUI_Manager.cs

```
using UnityEngine;
using System.Collections;
/**
 * La clase GUI_Manager realiza las operaciones a nivel Escena para la escena 03-menu
 *
 * @author Eder Martinez
 */
public class GUI_Manager : MonoBehaviour {

	//
	private GameObject maquinas;
	private GameObject sistemas;
	private GameObject subsistemas;
	private GameObject procesos;
	void Awake(){
	 Debug.Log ("Awake GUI_Manager");

	 foreach (Transform child in transform) {
		if(child.gameObject.name.Equals(Constantes.MAQUINAS_MENU)){
			maquinas =child.gameObject;
		}else if(child.gameObject.name.Equals(Constantes.SISTE-
MAS_MENU)){
			sistemas =child.gameObject;
		}else if(child.gameObject.name.Equals(Constantes.SUBSISTE-
MAS_MENU)){
			subsistemas =child.gameObject;
		}else if(child.gameObject.name.Equals(Constantes.PRO-
CESOS_MENU)){
			procesos =child.gameObject;
		}
	 }


	 BackEnd_Utils.activaHijosPorIndex (transform.gameObject,0);
	}

	void OnEnable(){
	 ScriptEventos.menuHandler += new ScriptEventos.EventMenuHandler (Escu-
chaEventos);
	}
	void OnDisable(){
	 ScriptEventos.menuHandler -= new ScriptEventos.EventMenuHandler (Escu-
chaEventos);
	}

	void EscuchaEventos(object o,EventosArgsMenu e){
	 Debug.Log ("-----EscuchaEventos");
	 if (e.tipo == Constantes.MAQUINAS_MENU) {
		Debug.Log("Entro a maquinas-----------");
```

```
                GeneralData.maquina_id = e.identificador;
                //Debug.Log (sistemas);
                StartCoroutine (espera (maquinas,2, 1));
        } else if (e.tipo == Constantes.SISTEMAS_MENU) {
                Debug.Log("------------"+e.nombre);
                if(e.identificador==4|e.identificador==5){
                        Debug.Log("Carga Partes");
                        Application.LoadLevel (Constantes.PARTS_CATA-
LOG_SCENE);
                }else{
                        GUI_Utils.quitaAnchors(sistemas);
                        GeneralData.sistema_id = e.identificador;
                        //Debug.Log (sistemas);
                        StartCoroutine (espera (sistemas,2, 2));
                }
        } else if (e.tipo.Equals (Constantes.SUBSISTEMAS_MENU)) {
                GUI_Utils.quitaAnchors(subsistemas);
                GeneralData.proceso_id = e.identificador;
                StartCoroutine (espera (subsistemas, 2, 3));
        } else if (e.tipo.Equals (Constantes.PROCESOS_MENU)) {
                Debug.Log(Constantes.PROCESOS_MENU);

                GeneralData.tarea_id = e.identificador;
                GeneralData.paso_id=Constantes.ID_STEP_INICIAL;
                Application.LoadLevel (Constantes.STEP_VIEW_SCENE);
        } else {
                Debug.Log("GO HOME!!");
                Debug.Log("****** "+e.tipo+" ++++ "+e.identificador);
                if(e.identificador==1){
                        Debug.Log("Sistemas");

                        GeneralData.maquina_id=0;
                        GeneralData.tarea_id=0;
                        GeneralData.proceso_id=0;
                        GeneralData.sistema_id=0;
                        BackEnd_Utils.activaHijosPorIndex (transform.gameObject,0);

                }else if(e.identificador==2){
                        Debug.Log("Procesos");

                        GeneralData.tarea_id=0;
                        GeneralData.proceso_id=0;
                        GeneralData.sistema_id=0;
                        BackEnd_Utils.activaHijosPorIndex (transform.gameObject,1);

                }
                else if(e.identificador==3){
                        Debug.Log("Tareas");

                        GeneralData.tarea_id=0;
                        GeneralData.proceso_id=0;
                        BackEnd_Utils.activaHijosPorIndex (transform.gameObject,2);
```

```
                }
        }
        }

        IEnumerator espera(GameObject go,int segundos,int index){
         //Objeto a Mover
         Debug.Log ("espera");
         TweenPosition.Begin(go,segundos*1f,new Vector3(-2000,0,0));
         yield return new WaitForSeconds((segundos*1f)/2);
         BackEnd_Utils.activaHijosPorIndex(transform.gameObject, index);
        }

        // Use this for initialization
        void Start () {

        }

        // Update is called once per frame
        void Update () {
         if (Input.GetKeyDown (KeyCode.Escape)) {
                if(!sistemas.activeSelf){
                        Debug.Log("Debo Salir");
                        Application.Quit();
                }else{
                        GeneralData.maquina_id=0;
                        GeneralData.tarea_id=0;
                        GeneralData.proceso_id=0;
                        GeneralData.sistema_id=0;
                        BackEnd_Utils.activaHijosPorIndex (transform.gameObject,0);
                }
         }
        }
}
```

Menu_Manager

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
/**
 * La clase Menu_Manager realiza las operaciones a nivel pantalla para la escena 03-menu
 *
 * @author Eder Martinez
 */
public class Menu_Manager : MonoBehaviour {
        //Prefab del boton a usar en la GUI
        /**
         * GameObject que permite hacer la interaccion script-escena y que contiene el
```

```
valor del tipo de boton a desplegar en la pantalla
 */
public GameObject prefab_boton;
/**
 * GameObject que permite hacer la interaccion script-escena y que contiene el
valor del tipo de boton de Home
 */
public GameObject home_button;
/**
 * GameObject que permite hacer la interaccion script-escena y que contiene el
valor del Titulo
 */
public GameObject titulo;

void Awake(){
 Debug.Log ("Awake "+name);
}
//se ejecuta cuando se activa
void OnEnable(){
 //Debug.Log ("OnEnable "+name);
 this.transform.position = Vector3.zero;
 Debug.Log (this.transform.position);
 //hacer el fade
 generateGUI ();
 Debug.Log (this.transform.position);
}

void OnDisable(){
 Debug.Log ("OnDisable "+name);
 this.transform.position = Vector3.zero;
 Destroy (gameObject.GetComponent<UITweener>());
 BackEnd_Utils.limpiaHijos (gameObject);
}

// Use this for initialization
void Start () {

}

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

/**
 * Metodo que ejecuta la generacion de los elementos mostrados en pantalla,
esto basado en el nombre del objeto al que esta asociado este script
 */
public void generateGUI(){
 if (this.name == Constantes.MAQUINAS_MENU) {
        Debug.Log ("Crea GUI Maquinas");
        generateMenuMaquinas ();
 } else if (this.name == Constantes.SISTEMAS_MENU) {
```

```
                Debug.Log ("Crea GUI Sistemas");
                generateMenuSistemas ();
            } else if (this.name.Equals (Constantes.SUBSISTEMAS_MENU)) {
                Debug.Log ("Crea GUI SubSistemas");
                generateMenuProcesos ();
            } else if (this.name.Equals (Constantes.PROCESOS_MENU)) {
                Debug.Log ("Crea GUI Procesos");
                generateMenuTareas ();
            }
            }
            /**
             * Metodo que genera el menu para el elemento Maquina
             */
            public void generateMenuMaquinas(){
             Debug.Log ("Crea Menu Maquinas");
             MaquinaManager mm = new MaquinaManager ();
             List<Maquina> lista = mm.getMaquinasTodas ();
             GUI_Utils.agregaTabla(transform.gameObject);
             UITable tabla = GetComponentInChildren<UITable> ();
             tabla.columns = 1;
             tabla.padding = new Vector2 (Constantes.PADDING_TABLE_X,Con-
stantes.PADDING_TABLE_Y);
             Debug.Log ("--------"+lista.Count);
             foreach(Maquina m in lista){
                Debug.Log(m.id+" - "+m.nombre);
                GUI_Utils.makeButtonMenu(m.id,m.nombre,Constantes.MA-
QUINAS_MENU,prefab_boton,tabla.gameObject);
             }
             GUI_Utils.posicionaTabla (tabla,prefab_boton);
             tabla.Reposition ();
             Debug.Log (this.transform.position);
            }

            /**
             * Metodo que genera el menu para el elemento Sistema
             */
            public void generateMenuSistemas(){
             Debug.Log ("Crea Menu Sistemas");
             //Carga Maquinas
             MaquinaManager mm = new MaquinaManager ();
             Debug.Log (GeneralData.maquina_id);
             Maquina maquina=mm.getMaquinaById(GeneralData.maquina_id);
             //
             SistemaManager sam = new SistemaManager ();
             Debug.Log ("Maquina ----------"+GeneralData.maquina_id);
             //List<Sistema> lista = sam.getSistemaTodos ();
             List<Sistema> lista = sam.getSistemasByMaquinaId (GeneralData.ma-
quina_id);
             GUI_Utils.agregaTabla(transform.gameObject);
             UITable tabla = GetComponentInChildren<UITable> ();
             tabla.columns = 2;
```

```
tabla.padding = new Vector2 (Constantes.PADDING_TABLE_X,Con-
stantes.PADDING_TABLE_Y);
Debug.Log ("--------"+lista.Count);
foreach (Sistema sa in lista) {
        //Debug.Log("Sistema "+sa.name);
        GUI_Utils.makeButtonMenu(sa.id,sa.nombre,Constantes.SISTE-
MAS_MENU,prefab_boton,tabla.gameObject);
}
GUI_Utils.posicionaTabla (tabla,prefab_boton);
tabla.Reposition ();
Debug.Log (this.transform.position);
GUI_Utils.createLabel (gameObject,titulo,maquina.nombre);
setButtonHome (1);
}
/**
 * Metodo que genera el menu para el elemento Proceso
 */
public void generateMenuProcesos(){
Debug.Log ("Crea Menu SubSistemas");
//
SistemaManager sam=new SistemaManager();
Sistema sa=sam.GetSistema(GeneralData.sistema_id);
//
ProcesoManager ssm = new ProcesoManager ();
List<Proceso> lista = ssm.getProcesoBySistemaId (GeneralData.sistema_id);
GUI_Utils.agregaTabla(transform.gameObject);
UITable tabla = GetComponentInChildren<UITable> ();
tabla.columns = 1;
tabla.padding = new Vector2 (Constantes.PADDING_TABLE_X,Con-
stantes.PADDING_TABLE_Y);
foreach(Proceso ss in lista){
        GUI_Utils.makeButtonMenu(ss.id,ss.nombre,Constantes.SUBSISTE-
MAS_MENU,prefab_boton,tabla.gameObject);
}
GUI_Utils.posicionaTabla (tabla,prefab_boton);
tabla.Reposition ();
GUI_Utils.createLabel (gameObject,titulo,sa.nombre);
setButtonHome (2);
}
/**
 * Metodo que genera el menu para el elemento Tarea
 */
public void generateMenuTareas(){
Debug.Log ("Crea Menu Procesos");
//
ProcesoManager ssm = new ProcesoManager ();
Proceso ss = ssm.GetProceso (GeneralData.proceso_id);
//
TareaManager pm = new TareaManager ();
List<Tarea> lista = pm.getTareasByProcesoId (GeneralData.proceso_id);
GUI_Utils.agregaTabla(transform.gameObject);
UITable tabla = GetComponentInChildren<UITable> ();
```

```
                        tabla.columns = 1;
                        tabla.padding = new Vector2 (Constantes.PADDING_TABLE_X,Con-
stantes.PADDING_TABLE_Y/4);
                        foreach(Tarea p in lista){
                                GUI_Utils.makeButtonMenu(p.id,p.name,Constantes.PRO-
CESOS_MENU,prefab_boton,tabla.gameObject);
                        }
                        GUI_Utils.posicionaTabla (tabla,prefab_boton);
                        tabla.Reposition ();
                        GUI_Utils.createLabel (gameObject,titulo,ss.nombre);
                        setButtonHome (3);
                       }

                       /**
                        * Metodo encargado de agregar el boton Home en las pantallas que asi lo re-
quieren
                        */
                       public void setButtonHome(int index){
                        //Debug.Log ("SET HOME BUTTON");
                        GUI_Utils.addBoton (transform.gameObject,home_button);
                        foreach(Boton_Componente bc in GetComponentsInChildren<Boton_Compo-
nente>()){
                                if(bc.name=="Home(Clone)"){
                                        bc.setValores(index,Constantes.HOME_MENU,Constan-
tes.HOME_MENU);
                                        UISprite _sprite=bc.GetComponent<UISprite>();
                                        UIRect _rect=_sprite.GetComponent<UIRect>();
                                        Debug.Log(_rect);
                                        int h=_sprite.height;
                                        int w=_sprite.width;
                                        int l=30;
                                        int b=20;
                                        _rect.SetAnchor(transform);
                                        _rect.leftAnchor.Set(0,l);
                                        _rect.bottomAnchor.Set(0,b);
                                        _rect.rightAnchor.Set(0,(l+w));
                                        _rect.topAnchor.Set(0,(b+h));

                                        _rect.ResetAnchors();
                                        _rect.UpdateAnchors();
                                }
                        }
                       }
}
```

SceneAR_Manager.cs

```
using UnityEngine;
using System.Collections;
/**
```

```
 * La clase SceneAR_Manager maneja a nivel global la escena de augmentacion
*/
public class SceneAR_Manager : MonoBehaviour {

                                void Awake(){
                                 #if UNITY_EDITOR
                                 GeneralData.tarea_id=3;
                                 #endif
                                 if(GeneralData.paso_id==0){
                                            GeneralData.paso_id=Constantes.ID_STEP_INICIAL;
                                 }
                                }

                                void OnEnable(){
                                 ScriptEventos.arHandler += new ScriptEventos.EventARHandler (Escu-
chaARScene);
                                }
                                void OnDisable(){
                                 ScriptEventos.arHandler -= new ScriptEventos.EventARHandler (Escu-
chaARScene);
                                }

                                // Use this for initialization
                                void Start () {
                                }

                                // Update is called once per frame
                                void Update () {
                                 if (Input.GetKeyDown (KeyCode.Escape)) {
                                            Debug.Log("Debe Regresar al paso anterior");
                                            Debug.Log("-------------");

                                            foreach(int num in GeneralData.historialPasos){
                                                       Debug.Log(num);
                                            }
                                }
}
public void EscuchaARScene(object o,EventosArgsAR e){
                                 Debug.Log ("EscuchaARScene");
                                 Debug.Log ("Datos "+e.paso_id+"-"+e.tarea_id);
                                 GeneralData.paso_id = e.paso_id;
                                 GeneralData.tarea_id = e.tarea_id;
                                 Debug.Log (name);
                                 AR_Manager manager = GetComponentInChildren<AR_Manager> ();
                                 manager.CargaPaso ();
                                 //GeneralData.changePaso = true;
                                }
}
```

AR_Manager.cs

```
public void instanciateModel(){
                    if (fbx == null) {
                        Debug.Log(paso_actual.contenido.Path);
                        fbx=Instantiate(Resources.Load(paso_actual.contenido.Path)) as
GameObject;
                    fbx.transform.parent=target.transform;
                    calibraModelo();

                    }
}
                    void calibraModelo(){
                     //asignar escalas
                     target.transform.localScale=BackEnd_Utils.convertStringToV3(paso_ac-
tual.contenido.escala);
                     fbx.transform.localScale=BackEnd_Utils.convertStringToV3(paso_actual.con-
tenido.escala);

                     //asignar posicion
                     fbx.transform.localPosition=BackEnd_Utils.convertStringToV3(paso_ac-
tual.contenido.posicion);

                     //asignar rotation
                     fbx.transform.localEulerAngles = BackEnd_Utils.convertStringToV3(paso_ac-
tual.contenido.rotacion);
                    }


}
```

GeneralData.cs

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;
/**
 * La clase GeneralData es una clase estatica lo cual permite usarla para compartir informacion entre
escenas
 *
 * @author Eder Martinez
 */
public static class GeneralData  {
                    /**
                     * Numero entero que mantiene el identificador de una Maquina para su uso
entre escenas
                     */
                    public static int maquina_id;
                    /**
```

```
                                 * Numero entero que mantiene el identificador de un Sistema para su uso en-
tre escenas
                                 */
                                public static int sistema_id;
                                /**
                                 * Numero entero que mantiene el identificador de un Proceso para su uso en-
tre escenas
                                 */
                                public static int proceso_id;
                                /**
                                 * Numero entero que mantiene el identificador de una Tarea para su uso entre
escenas
                                 */
                                public static int  tarea_id;
                                /**
                                 * Numero entero que mantiene el identificador de un Paso para su uso entre
escenas
                                 */
                                public static int  paso_id;
                                /**
                                 * Lista que mantiene el historial de los pasos seguidos
                                 */
                                public static List<int> historialPasos=new List<int>();
                                /**
                                 * Variable que contiene un valor para saber si es necesario cambiar de paso y
mantener el actual
                                 */
                                public static bool cambiarPaso = true;
                                //Diccionario para las entradas
                                /**
                                 * Variable que guarda un par de valores (nombre de la variable y valor) captu-
radas por una Advertencia de tipo dialogo
                                 */
                                public static Dictionary<string,string> variables=new Dictionary<string, string>();
                                /**
                                 * Variable que contiene el nombre de las variables a validar antes de realizar
las operaciones pertinentes
                                 */
                                public static string variables_validar="";
}
```

Gesture_Manager.cs

```
using UnityEngine;
using System.Collections;
/**
 * Esta clase permite manejar los gestos usando las clases proporcionadas por el plugin FingerGes-
ture
 *
*/
```

```
public class Gesture_Manager : MonoBehaviour {
                bool rotating = false;
                bool pinching = false;
                /**
                 * Variable que contiene el valor al que va cambiar la escala cuando se detecte
el evento Pinch
                 */
                public float pinchScaleFactor = 0.02f;

                /**
                 *Este Enum contiene los tipos de interacciones que puede recibir este Compo-
nente
                 */
                public enum InputMode
                {
                 PinchOnly,
                 TwistOnly,
                 PinchAndTwist
                }

                bool Rotating
                {
                 get { return rotating; }
                 set
                 {
                        if( rotating != value )
                        {
                                rotating = value;
                        }
                 }
                }
                void OnTwist(TwistGesture gesture) {
                 Debug.Log ("OnTwist");
                 if( gesture.Phase == ContinuousGesturePhase.Started )
                 {
                        Debug.Log ("Twist gesture started");
                        Rotating = true;
                 }
                 else if( gesture.Phase == ContinuousGesturePhase.Updated )
                 {
                        if( Rotating )
                        {
                                Debug.Log("Rotation updated by " + gesture.DeltaRotation + "
degrees");

                                // apply a rotation around the Z axis by rotationAngleDelta de-
grees on our target object
                                transform.Rotate( gesture.DeltaRotation, 0, 0 );
                        }
                 }
                 else
                 {
```

```
                    if( Rotating )
                    {
                        Debug.Log ("Rotation gesture ended. Total rotation: " + ges-
ture.TotalRotation);
                        Rotating = false;
                    }
                }
            }

            bool Pinching
            {
                get { return pinching; }
                set
                {
                    if( pinching != value )
                    {
                        pinching = value;
                    }
                }
            }

            void OnPinch(PinchGesture gesture) {
                /* your code here */
                if( gesture.Phase == ContinuousGesturePhase.Started )
                {
                    Pinching = true;
                }
                else if( gesture.Phase == ContinuousGesturePhase.Updated )
                {
                    if( Pinching )
                    {
                        // change the scale of the target based on the pinch delta value
                        transform.localScale += gesture.Delta.Centimeters() *
pinchScaleFactor * Vector3.one;
                    }
                }
                else
                {
                    if( Pinching )
                    {
                        Pinching = false;
                    }
                }
            }

            void Awake(){
                transform.localScale = new Vector3 (0.5f,0.5f,0.5f);
            }
            // Use this for initialization
            void Start () {
            }
```

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

}
```

Part_Controller.cs

```
public class Part_Controller : MonoBehaviour {
public GameObject imageTarget;
public GameObject botonBack;
public GameObject botonNext;
public GameObject descripcion;

private ParteManager pm;
private List<Parte> partesPorMaquina;
private Parte actual;
private int index=0;
private GameObject fbx=null;

void OnEnable(){
 Debug.Log ("OnEnable");
 UIButton tmpB = botonBack.GetComponent<UIButton> ();
 UIButton tmpN = botonNext.GetComponent<UIButton> ();
 EventDelegate.Add (tmpB.onClick,parteAnterior);
 EventDelegate.Add (tmpN.onClick,parteSiguiente);


 pm = new ParteManager ();
#if UNITY_EDITOR_WIN
 GeneralData.maquina_id=1;
#endif
 partesPorMaquina = pm.getPartesByMaquinaId (GeneralData.maquina_id);
 Debug.Log (partesPorMaquina.Count);
 loadContent ();
 Debug.Log (actual.Path_folder);
}
void OnDisable(){
 Debug.Log ("OnDisable");
 UIButton tmpB = botonBack.GetComponent<UIButton> ();
 UIButton tmpN = botonBack.GetComponent<UIButton> ();
 EventDelegate.Remove (tmpB.onClick,parteAnterior);
 EventDelegate.Remove (tmpN.onClick,parteSiguiente);
}
// Use this for initialization
void Start () {

}

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

```
                    }

                    void parteSiguiente(){
                     GameObject.DestroyImmediate(fbx);

                     index = index + 1;
                     if(index>partesPorMaquina.Count-1){
                              index=0;
                     }
                     loadContent ();
                    }

                    void parteAnterior(){
                     GameObject.DestroyImmediate(fbx);

                     index = index - 1;
                     if(index<0){
                              index=partesPorMaquina.Count-1;
                     }
                     loadContent ();
                    }

                    void loadContent(){
                     if (fbx == null) {
                              Debug.Log (index);
                              actual = partesPorMaquina [index];
                              BackEnd_Utils.setTexto (descripcion,actual.nombre+": "+actual.des-
cripcion);

                              fbx = Instantiate (Resources.Load (actual.Path_folder + ac-
tual.path_content)) as GameObject;
                              fbx.transform.parent = imageTarget.transform;

                              //asignar escalas
                              imageTarget.transform.localScale=BackEnd_Utils.convert-
StringToV3(actual.escala);
                              fbx.transform.localScale=BackEnd_Utils.convertStringToV3(ac-
tual.escala);
                              //asignar posicion
                              imageTarget.transform.localPosition=BackEnd_Utils.convert-
StringToV3(actual.posicion);
                              //asignar rotation
                              imageTarget.transform.Rotate(BackEnd_Utils.convertStringToV3(ac-
tual.rotacion));
                     }
                    }
}
```

LISTA DE REFERENCIAS

Abascal, J. y Moriyón, R. (2002). Tendencias en interacción persona-computador. *Revista Iberoamericana de Inteligencia Artificial*, *16*, 9-24. doi:10.4114/ia.v6i16.750.

Azuma, R. T. (1997). A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, *6*(4), 355-385.

Blake, J. (2012). The natural user interface revolution. En J. Blake (Ed.), *Natural User Interfaces in .NET* (pp. 4-35). Londres: Manning Publications.

Blender. (2011a). *Modelado en blender*. Recuperado de http://wiki.blender.org/index.php/Doc:2.6/Manual/Modeling

Blender. (2011b) *¿Qué es blender?* Recuperado de http://wiki.blender.org/index.php/Doc:2.6/Manual/Introduction

Blender. (2011c). *Historia de Blender*. Recuperado de http://wiki.blender.org/index.php/Doc:2.6/Manual/Introduction/History

Brosnan, T. y Sun D. W. (2004). Improving quality inspection of food products by computer vision—a review. Journal of Food Engineering, 61, 3-16. doi:10.1016/S0260-8774(03)00183-3

Castellani, S., Grasso, A., O'Neill, J. y Roulland, F. (2009). Designing technology as an embedded resource. *Computer Supported Cooperative Work, 18*, 199-227. doi:10.1007/s10606-008-9088-1

Chang, G., Morreale, P. y Medicherla, P. (2010). Applications of augmented reality systems in education. En Proceedings of Society for Information Technology & Teacher Education International Conference (pp. 1380-1385). Chesapeake, VA: AACE.

Chen, K. M., Chen, L. L. y Shen, S. T. (2008). Development and comparison of a full-scale car display and communication system by applying augmented reality. *Displays, 29*(1), 33-40. doi:10.1016/j.displa.2007.07.002

Craig, A. B., Sherman, W. R., y Will, J. D. (2009). *Developing virtual reality applications*. San Francisco: Elsevier.

Dugri, P. (2004). *Optimal troubleshooting plan for a complex electromechanical system*. (Tesis de Maestría, Oregon State University). Recuperado de https://ir.library.oregonstate.edu/xmlui/bitstream/handle/1957/30053/DugriParthsarathy2004.pdf?sequence=1

Henderson, S. y Feiner, S. (2010). Opportunistic tangible user interfaces for augmented reality. En: *IEEE transaction on visualization and computer graphics, 16*(1), 4-16. doi:10.1109/TVCG.2009.91

Henderson, S. y Feiner, S. (2011). Exploring the benefits of augmented reality documentation for maintenance and repair. En *IEEE Transactions on Visualization and Computer Graphics*, *17*(10), 1355-1368. doi:10.1109/TVCG.2010.245

Hou, L., Wang, X., Bernold, L. y Love, P. D. (2013). Using animated augmented reality to cognitively guide assembly. *Journal of Computing In Civil Engineering, 27*(5), 439-451. doi:10.1061/(ASCE)CP.1943-5487.0000184

Johansen, Emil. (2010). *Is Unity engine written in Mono/C#? or C++*. Recuperado de http://answers.unity3d.com/questions/ 9675/is-unity-engine-written-in-monoc-or-c.html

Johnson, W. L., Rickel, J., Stiles, R. y Munro, A. (1998). Integrating pedagogical agents into virtual environments. *Presence: Teleoperators and Virtual Environments, 7(6),* 523-546.

Jonas. (2013). *Qualcomm's Vuforia (QCAR) vs. Metaio SDK vs. D'Fusion Mobile vs. Layar SDK.* Recuperado de http://stackoverflow.com/ questions/ 9227962/ qualcomms-vuforia-qcar-vs-metaio-sdk-vs-dfusion-mobile-vs-layar-sdk

Karray, F., Alemzadeh, M., Saleh, J. A. y Arab, M. N. (2008). Human-computer interaction: Overview on state of the art. *International Journal on Smart Sensing and Intelligent Systems, 1*(1), 137-159.

Kernighan, B. W. y Plauger, P. J. (1976). Reading, MA: *Software Tools*. Addison-Wesley.

König, W.A., Rädle, R. y Reiterer, H. (2009). Squidy: A zoomable design environment for natural user interfaces. En *Proceedings of the 27th International Conference Extended Abstracts on Human Factors in Computing Systems* (pp. 4561-4566). New York: ACM. doi:10.1145/1520340.1520700

Lee, J., Han, S. y Yang, J. (2011). Construction of a computer-simulated mixed reality environment for virtual factory layout planning, *Computers in Industry*, *62*, 86-98*.* doi: 10.1016/j.compind.2010.07.001

Liang, J. S. (2008). The troubleshooting task implementation in automotive chassis using virtual interactive technique and knowledge-based approach, *Journal of Network and Computer Applications, 31*, 712-734. doi:10.1016/ j.jnca.2007 .11.001

Loureiro, B. y Rodrigues, R. (2011). Multi-touch as a natural user interface for elders: A Survey. En: *Conference 6th Iberian on information systems and technologies* (pp 1-6). Chaves, PT: IEEE

Malizia, A. y Bellucci, A. (2012). The artificiality of natural user interfaces. *Communications of the ACM, 55*(3), 36-38. doi:10.1145/2093548.2093563

Marcincin, J. N., Barna, J., Janak, M., Marcincinova, L. N. y Torok, J. (2012).Visualization of intelligent assembling process by augmented reality tools application. En: *4th IEEE International Symposium on Logistics and Industrial Informatics*, 33-36. Smolenice, SK: IEEE. doi:10.1109/LINDI.2012.6319505

Maya. (2007). *Cine Autodesk Maya*. Recuperado de http://latinoamerica.autodesk .com/adsk/servlet/item?siteID=7411870&id=11277888

NGUI. (2011). *NGUI: Next-Gen UI kit*. Recuperado de_http://www.tasharen.com/ ?page_id=140

Nicolau, S. A., Pennec, X., Soler, L., Buy, X., Gangi, A., Ayache, N. y Marescaux, J. (2009). An augmented reality system for liver thermal ablation: Design and evaluation on clinical cases, *Medical Image Analysis, 13*(3), 494-506. doi:10.1016/j.media.2009.02.003

Nielsen, J. (1994). *Usability engineering*. San Francisco: Morgan Kaufman.

Ong, M., Ren, X., Allan, G., Kadirkamanathan, V., Thompson, H. y Fleming, P. (2004). Decision Support System on The Grid. En *The 2004 International Conference on Knowledge-Based Intelligent Information & Engineering Systems* (pp. 699-710). Wellington, NZ: Springer.

Platonov, J., Heibel, H., Meier, P. y Grollmann, B. (2006). A mobile markerless AR system for maintenance and repair*. En: IEEE/ACM International Symposium on Mixed and Augmented Reality* (pp. 105-108). Santa Barbara, CA: IEEE/ACM doi:10.1109/ISMAR.2006.297800

Potapenko, D. (2013). *Will NGUI be next "oficial" 4.x GUI?* Recuperado de http://forum .unity3d.com/threads/will-ngui-be-next-official-4-x-gui.173384/

Prakash, S., Kumar, A. y Mishra, R. B. (2013). MVC architecture driven design and agile implementation of a web-based software system. *International Journal of Software Engineering & Applications, 4*(6), 13-28. doi:10.5121/ijsea.2013.4602

Pull, K., Baksheev, A., Kornyakov, K. y Eruhimov, V. (2012). Real-time computer vision with Open CV. *Communications of the ACM, 55*(6), 61-69. doi:10.1145/2184319 .2184337

Rautaray, S. S. y Agrawal, A. (2012). Vision based hand gesture recognition for human computer interaction: A survey. *Artificial Intelligence Review, 43* (1-54). doi:10 .1007/ s10462-012-9356-9

Ravaine, W. (2012). *Finger Gestures*. Recuperado de http://fingergestures.fatalfrog .com/

Raymond, E. S. (2003). *The Art of Unix Programming*. Reading, MA: Addison-Wesley

Reinhart, G. y Patron, C. (2003). Integrating augmented reality in the assembly domain-fundamental, benefits and applications, *CIRP Annals Manuafacturing Technology,* 5-8. doi:10.1016/S0007-8506(07)60517-4

Rómmel, F. (2007). Sqlite: La base de datos embebida. *Software Guru, 5*(17), 12-13.

Serrano Mamolar, A. (2012). *Herramientas de desarrollo libres para aplicaciones de Realidad Aumentada con Android. Análisis comparativo entre ellas*. (Tesis de Maestria, Universidad Politecnica de Valencia). Recuperado de https://riunet .upv.es/bitstream/handle/10251/18028/Memoria%20TFM%20Ana%20Serrano. pdf?sequence=1

Shneiderman, B. y Plaisant, C. (2004). *Designing the user interface: Strategies for effective human-computer interaction*. Boston: Pearson/Addison-Wesley.

Spence, R. (2006). *Information visualization: design for interaction*. Upper Saddle River, NJ: Prentice Hall.

Sridaran, R., Padmavathi, G., Iyakutti, K. y Mani, M.N.S. (2010). *SPIM architecture for MVC based web applications*. Recuperado de http://arxiv.org/abs/ 1006.2702

Sutherland, I. E. (1965). The ultimate display .En *Proceedings of International Federation for Information Processing Congress 1965*, 2, 506 –508. New York: Macmillan.

Sutherland, I. E. (1968). A head mounted three-dimensional display. En *Proceedings of the* Fall Joint Computer Conference (pp. 757-764). New York: ACM. doi:10. 1145/1476589.1476686

Te'eni, D., Carey, J. y Zhang, P. (2007). *Human computer interaction: Developing effective organizational information systems*. Hoboken, NJ: John Wiley & Sons.

Tönnisn, M., Plecher, D. A. y Klinker, G. (2013). Representing information–classifying the augmented reality presentation space. *Computers and graphics*, *37*(8), 997-1011. doi:10.1016/j.cag.2013.09.002

Turk, M. y Hua, G. (2013). *Vision-Based Interaction*. San Rafael, CA: Morgan & Claypool. doi:10.2200/ S00536ED1V01Y201309COV005.

Tzong-Ming, C. y Tu, T. (2009). A fast parametric deformation mechanism for virtual reality applications. *Computers & Industrial Engineering, 57*(2), 520-538. doi:10.1016/j.cie.2008.08.008

Unity. (2010). *Unity Tecnhologies celebrates 5 years of unity*. Recuperado de https://unity3d.com/http%3A//unity3d.com/company/public-relations/ news/unity-5year-press

Unity. (2014a) *¿Qué es Unity?* Recuperado de https://unity3d.com/es/pages/ what-is-unity

Unity. (2014b). *Comparaciones de Licencias*. Recuperado de https://unity3d.com/ es/unity/licenses

Vision Mobile. (2014a). *Developer Economics Q3 2014: State of the Developer Nation*. Recuperado de http://www.visionmobile.com/product/developer-economics-q3-2014/

Vision Mobile. (2014b). *North American App Developer Trends 2014*. Recuperado de http:// www.visionmobile.com/product/north-american-app-developer-trends-2014/.

Walter. (2014a). *Rectificadoras de herramientas y máquinas de medición WALTER - el líder mundial en tecnología de rectificado de herramientas*. Recuperado de http://www. grinding.com/es/rectificadoras/rectificado-de-herramientas-y-medicion/walter.html

Walter. (2014b). *Helitronic Basic Rectificadora universal CNC para trabajos de reafilado.* Recuperado de http://www.interempresas.net/FeriaVirtual/ Catalogos_y_documentos/ 162109/helitronic_basic_es_100dpi.pdf

Weiss, C.R., Marker, D.R., Fischer, G.S., Fichtinger, G., Machado, A.J. y Carrino, J. A. (2011). Augmented reality visualization using image-overlay for mr-guided interventions: system description, feasibility, and initial evaluation in a spine phantom. *American Journal of Froentgenology, 96*(3), w305-w307. doi:10.2214/ AJR.10.5038.

Welch, G. F. (2009). History: The use of the kalman filter for human motion tracking in virtual reality. *Presence: Teleoperators & Virtual Environments, 18*(1), 72-91.

Wigdor, D. y Wixon, D. (2011). *Brave NUI World: Design natural user interfaces for touch and gesture*. Burlington: Elsevier.

Zhang, J., Sheng, Y., Hao, W., Wang, P. P., Tian, P., Miao, K. y Pickering, C.K. (2010). A context-aware framework supporting complex ubiquitous scenarios with Augmented Reality enabled. En: *5th International Conference on Pervasive Computing and Applications*, 69-74. doi:10.1109/ICPCA.2010.5704077

Zhang, P. y Galletta, D. (2006). *Human-Computer Interaction and Management Information Systems: Foundations*. Armonk, NY: M.E. Sharpe.

Zhou, F., Duh, H. B. L. y Billinghurst, M. (2008). Trends in augmented reality tracking, interaction and display: A review of ten years of ISMAR. En *Proceedings of the 7th IEEE/ACM International Symposium on Mixed and Augmented Reality* (pp. 193-202). Washington, DC: IEEE. doi:10.1109/ISMAR.2008.4637362

Zhu, J., Ong, S. K. y Nee, A. Y. C. (2013). An authorable context-aware augmented reality system to assist the maintenance technicians. *International Journal of Advanced Manufacturing Technology*, *66*(9-12), 1699-1714. doi:10.1007/ s00170-012-4451-2