ANIMACIÓN DE UN MODELO 3D DEL ROBOT DARWIN-OP UTILIZANDO KINECT

Antonio Benítez Ruiz

Universidad Politécnica de Puebla, Departamento de Posgrado

antonio.benitez@uppuebla.edu.mx

María Auxilio Medina Nieto

Universidad Politécnica de Puebla, Departamento de Posgrado

maria.medina@uppuebla.edu.mx

Miguel Ángel Medina Nieto

Universidad Politécnica de Puebla, Departamento de Posgrado

miguel.medina@uppuebla.edu.mx

Jorge de la Calleja Mora

Universidad Politécnica de Puebla, Departamento de Posgrado

Jorge.delacalleja@uppuebla.edu.mx

Resumen

Este documento describe herramientas relacionadas con la creación de video juegos, principalmente aquellas que soportan tareas como diseño y animación, con el propósito de integrar una representación gráfica en tres dimensiones del robot humanoide Darwin-OP. El objetivo se logra al animar esta representación mediante la incorporación de un esqueleto y el establecimiento de una conexión con el sensor movimientos de un Kinect. Los resultados iniciales de esta investigación de tipo exploratoria, muestran la factibilidad de animar articulaciones específicas de un personaje virtual a través de un dispositivo relativamente accesible.

Palabra(s) clave(s): Captura de movimiento, diseño 3D, kinect, motores de videojuegos.

1. Introducción

Los motores de creación de video juegos se han utilizado para simulación, animación, inmersión, realidad aumentada, interfaces humano computadora, entre otras. Éstos apoyan el desarrollo casi transparente de una diversidad de aplicaciones, por ejemplo, que se ejecutan en computadoras personales (PCs), dispositivos móviles o en consolas como Xbox, Play Station, PS Vita, Windows Phone, Windows Store Apps o Blackberry, mismas que emplean diferentes sistemas operativos. En la creación de video juegos, es frecuente la incorporación de otro tipo de hardware como giróscopos, sensores GPS, Microsoft Kinect y Oculus VR.

Los motores de video juegos son las herramientas de software que están detrás de muchos videos juegos que se encuentran en el mercado [1]. La evolución de la tecnología en las computadoras, entre otros factores, ha permitido que la industria de los videos juegos sea reconocida ampliamente por sus logros tecnológicos y financieros. Es en esta industria donde existe una demanda creciente de recursos humanos especializados que desempeñen roles como programador, diseñador de juego, ingeniero mecánico, diseñador de niveles, diseñador de personajes, diseñador de animación, escritor o diseñador de sonido [3]. De manera informal, en el vocabulario común del desarrollo de videojuegos, a los diseñadores se les conoce como artistas.

El desarrollo de un video juego requiere integrar a un conjunto de elementos como los siguientes: la historia o temática del juego (*"game design"*), la forma o estilo de interacción con los usuarios (*"game play"*); el contenido del juego (*"game content"*), que generalmente es construido por los artistas o el diseño de la interacción del contenido en tiempo de ejecución; son los programadores quienes desarrollan la aplicación que cargará el contenido y quienes integran aspectos relacionados con Inteligencia Artificial (*"game AI"*), utilizando un motor de video juegos (*"game engine"*), el cual se encarga de administrar y desplegar la información en el mundo virtual [2].

Este documento describe motores de video juegos y herramientas relacionadas con la creación de video juegos, principalmente aquellas que soportan tareas

como diseño y animación, con el propósito de integrar una representación gráfica en tres dimensiones (3D) de un robot humanoide Darwin-OP.

El documento está organizado de la manera siguiente. La sección 2 describe características básicas que se utilizan para seleccionar un motor de video juegos. La sección 3 presenta herramientas de generación de contenido. La sección 4 contiene el modelo del robot Darwin-OP. La conexión entre la interfaz de Unity con el sensor de Kinect se trata en la sección 5. Los resultados relacionados con la animación del modelo se incluyen en la sección 6. Finalmente, las conclusiones y el trabajo en proceso se presentan en la sección 7.

2. Motores para el desarrollo de videojuegos

Actualmente, existen motores para desarrollar video juegos que se distribuyen bajo licencias "*free ware"*, de código abierto o con costos que pudieran absorber organizaciones no muy grandes [4]. Esta sección describe las características y funcionalidades de algunos.

El motor de juegos Unity es uno de los más utilizados, los tipos de usuarios varían de desarrolladores novatos hasta estudios profesionales [1] y [8]. Unity pertenece a la empresa Unity Technologies, cuenta con una versión gratuita y una de paga. Los juegos se pueden desarrollarse tanto en 3D como en 2D, para plataformas como Windows, Mac y Linux. Entre las plataformas para ejecutar los juegos está la web (mediante el "plug-in Unity Web Player"), las PC, dispositivos móviles y consolas de videojuegos como Xbox, PS3 o PS4.

Algunos de los video juegos que se han desarrollado con este motor son: "Assassin's Creed Identity", "Temple Run", "Game of Thrones: Seven Kingdoms", "République", "Hearthstone: Heroes of Warcraft" [1].

La interfaz de desarrollo de Unity es relativamente fácil de utilizar. La figura 1 de la izquierda muestra una parte. Dado que la comunidad de desarrolladores es grande, los usuarios se apoyan en la documentación disponible y en los foros de ayuda.

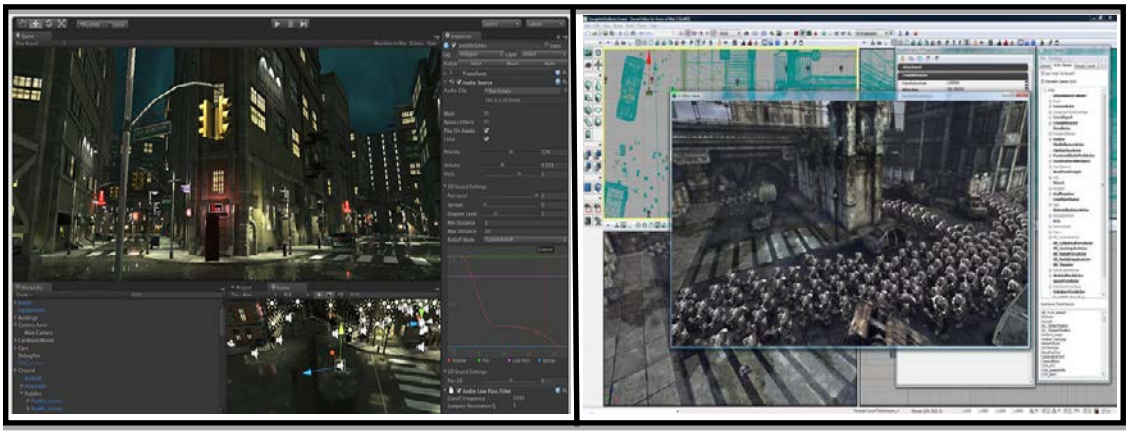

Figura 1 Ejemplos de la interfaz de desarrollo de *Unity* (izquierda) y *Unreal* (derecha).

El motor Unreal representa un conjunto de herramientas (o "*suite*") de desarrollo de video juegos [9]. Pertenece a la empresa Epic Games; es propietaria de los video juegos "Gears of War", "Haunted House: Cryptic Graves", "Tekken 7" y "Kingdom Hearts III". Los juegos desarrollados con esta herramienta se ejecutan en PC, consolas de videojuegos y dispositivos móviles.

Unreal ofrece una versión gratuita para uso académico que utiliza Windows como sistema operativo. La figura 1 de la derecha muestra parte de su interfaz. La experiencia en el uso de Unity y Unreal nos permite comentar que el desarrollo con el último es más complejo, sin embargo, sus herramientas tienen más capacidades. La documentación técnica, foros y tutoriales están disponibles en la web.

CryEngine es el motor de juegos de la empresa Crytek [10]; está orientado para desarrollar juegos que se ejecuten en consolas, la calidad gráfica puede ser superior a la obtenida con los dos motores previos. Los desarrolladores independientes pueden utilizar este motor sin costo. La figura 2 de la izquierda muestra un ejemplo de la interfaz para crear escenarios. "La saga de Crysis", "Enemy Front", "Warface", "State of Decay" y "Sonic Boom" son ejemplos de juegos desarrollados con CryEngine. La documentación es extensa y cuenta con foros oficiales para desarrolladores.

Otro motor que ha ganado popularidad es Game Maker Studio [11]. Pertenece a la empresa Yoyo Games; se ejecuta tanto en computadoras con el sistema operativo

Windows o iOS. Los juegos se ejecutan en plataformas móviles, PC, Mac, Linux, PS4 y Xbox. La figura 2 derecha muestra parte de su interfaz. Entre los juegos populares desarrollados con este motor están “Chain Braker”, “Uncanny Valley”, “Terrain Saga: KR-17” y “Arcane Soul”.

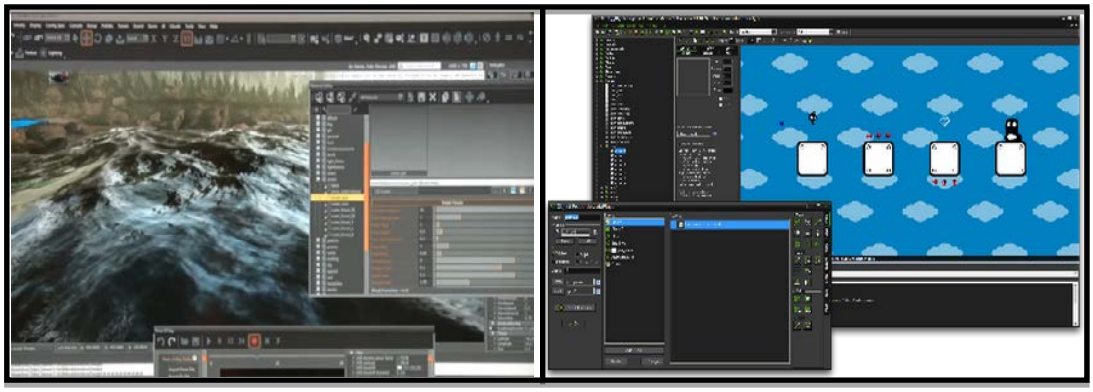

Figura 2 Interfaz de desarrollo de *CryEngine* (izquierda) y *Game Maker Studio* (derecha).

La decisión para escoger el mejor motor de juegos depende de muchos factores, aunque principalmente depende de qué se desea desarrollar y en dónde se publicará el juego. Cada motor tiene sus ventajas y limitaciones respecto a los otros. Como recomendación, se deberá tener cuidado en seleccionar aquel motor que tenga la suficiente documentación y verificar la cantidad de personas que está trabajando con él, sin descartar características internas como motor física, iluminación, manejo de shaders, manejo de sombras, animación, entre otros.

La tabla 1 muestra los nombres, la empresa y las direcciones de las páginas electrónicas principales de otros motores, se incluyen también las de los descritos.

3. Herramientas para generar contenido

Los motores de video juegos, en términos de producción, son ensambladores, eso es, casi todos los recursos (“*assets*”) se desarrollan utilizando otras herramientas y después se importan para crear el juego en sí. Por ejemplo, los modelos 3D se crean en aplicaciones como Maya [3],[12], Cinema4D [15], Blender [14], Modo, 3DS Max [13], Lightwave, Daz Studio, ZBrush [16] o Mudbox [21]; las texturas en Photoshop, Gimp [17] o Body Paint; los scripts de programación se

elaboran en Visual Studio, Mono Develop, UniSCTE o Unitron [3]; para el ambiente y diseño de un terreno, se suele usar Grome, Graphisoft ArchiCAD, SketchUp o Revit [5]. Las animaciones de los "*assets*" se definen en herramientas como Motion Builder o iPisoft Mocap Studio; incluso los sonidos también se importan. En esta sección se describen algunas de las herramientas utilizadas para implementar "assets", los cuales se consideran elementos esenciales del contenido del juego.

Tabla 1 Nombres y páginas principales de motores de desarrollo de video juegos.

Nombre del motor	**Empresa**	**Dirección electrónica de la página principal**
Cocos2DX	Cocos2d-x Developer Blog	www.cocos2d-x.org
CryEngine	Crytek GmbH.	www.cryengine.com
Game Maker Studio	Yoyo games	www.yoyogames.com/gamemaker
Gamebryo	Gamebase Co., Ltd.	www.gamebryo.com
Leadwerks	Leadwerks Software	www.leadwerks.com
Panda3D	Carnegie Mellon University	www.panda3d.org
Steam (antes Source)	Valve Corporation	www.valvesoftware.com
Stencyl	Stencyl, LLC	www.stencyl.com
Tombstone engine (antes C4 Engine)	Terathon Software	http://www.terathon.com
Torque	Garage Games	www.garagegames.com
Unity	Unity Technologies	unity3d.com
Unreal Engine 4	EPIC GAMES, INC.	www.unrealengine.com

El modelado de objetos en 3D, generalmente se realiza seleccionando uno de dos enfoques: construir o esculpir el modelo. Una de las herramientas con mayor capacidad para el primer enfoque es Maya [3], [12]. La manipulación de cámaras y de objetos se realiza con los mismos atajos que Unity. En Maya, se animan personajes que después pueden exportarse a Unity de manera sencilla. La figura 3 de la izquierda presenta un ejemplo de la interfaz en Maya.

Otra herramienta para diseño, construcción de modelos en 3D y animación es Autodesk 3DS Max; la cual emplea "*plug ins*". Con frecuencia, se le relaciona más con su uso en arquitectura que con el desarrollo de video juegos [13]. La figura 3 derecha muestra parte de su interfaz.

Figura 3 Ejemplos de la interfaz de desarrollo de *Maya* (izquierda) y *3DS Max* (derecha).

Blender, a pesar de distribuirse de forma gratuita, es una herramienta potente para modelar objetos en 3D [14], implementa el enfoque de construcción. Se utiliza en sistemas Windows, Mac OS X, Linux, Solaris, FreeBSD e IRIZ. Incluye un motor de video juegos relativamente sencillo. La figura 4 de la izquierda muestra su interfaz.

Figura 4 Interfaz de desarrollo de *Blender* (izquierda) y *Cinema4D* (derecha).

Una herramienta con mayor capacidad para crear modelos 3D es Cinema 4D [15]. Entre sus características destacan la creación de "*rigs*" y animaciones avanzadas de personajes y la incorporación de efectos en 3D como pelo (ver la figura 4 de la derecha).

Aun cuando las herramientas presentadas en esta sección permiten pintar objetos y definir tipos de materiales, es común incrementar la calidad del texturizado mediante la creación de mapas UV y el uso de Photoshop o Gimp. La figura 5 muestra un ejemplo del manejo de texturas en Photoshop.

Figura 5 Mapa UV de un personaje editado en Photoshop.

Las herramientas de escultura, es decir, aquellas que implementan el segundo enfoque, no agregan objetos a un modelo, sino que a partir de un bloque 3D general, éste se esculpe. De las ventajas del segundo enfoque resalta el detalle en polígonos que se puede obtener. En el desarrollo profesional de video juegos, la literatura indica que las herramientas de escultura se emplean para modelar objetos 3D de alta calidad, es decir, aquellos que emplean varios millones de polígonos, éstos se exportan como modelos de sólo miles y gracias a mapas normales, de profundidad, de luces, entre otros, conservan su calidad.

Otra característica de las herramientas de escultura es que las técnicas de texturizado de los modelos 3D esculpidos son más intuitivas que las de los modelos hechos con herramientas de creación. Una de las herramientas de escultura más utilizadas es Zbrush, de la empresa Pixologic [16]. Es un software de escultura, pintura digital y modelado en 3D. Permite esculpir modelos detallados de un modo semejante a pintar en los mismos. Zbrush ha sido utilizado para modelar personajes de películas como Underworld o El señor de los anillos. La figura 6 de la izquierda muestra la interfaz de Zbrush. Un competidor directo de ZBrush es Autodesk Mudbox [21], (los modelos hechos en la primera versión se utilizaron en la película de King Kong. Mudbox cuenta con un ambiente 3D para

incorporar cámaras móviles, edición de mallas poligonales y subdivisión de objetos. Permite el uso de capas 3D para visualizaciones rápidas del diseño, escultura no destructiva y soporte para un número de polígonos grande (figura 6 derecha).

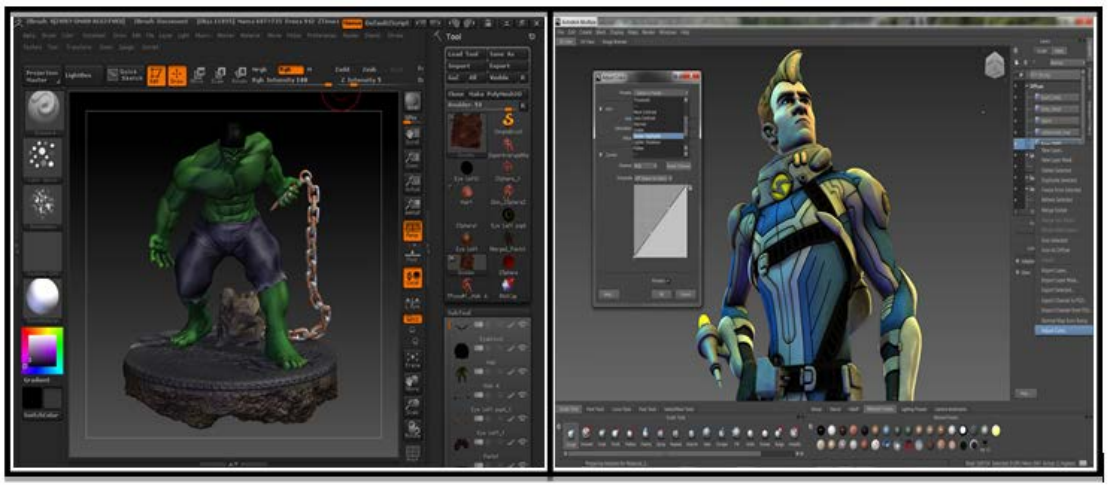

Figura 6 Interfaz de desarrollo de *Zbrush* (izquierda) y *Mudbox* (derecha).

Después de presentar algunas herramientas para desarrollar “assets”, la sección 4 explica el proceso de construcción del modelo del robot Darwin-OP.

4. Modelado del robot Darwin-OP en Maya

El modelo 3D del robot Darwin-OP requirió de la integración de las piezas disponibles en formato STL obtenidas desde el sitio web [20]; estas piezas se importaron en Maya y se integraron mediante manipulación simple y operaciones de escala, traslación y rotación. La Figura 7 muestra el conjunto de piezas que forman el modelo del robot. Tanto el robot original (el físico), como el modelo, tiene 20 articulaciones asociadas a las piernas, brazos y cuello. Los pasos realizados en el modelado del robot son:

- Paso 1: *creación de materiales*. - Se crearon materiales nuevos para las piezas del robot, con el propósito de colorear el modelo y que éste sea lo más fiel posible al robot original (ver la figura 7).

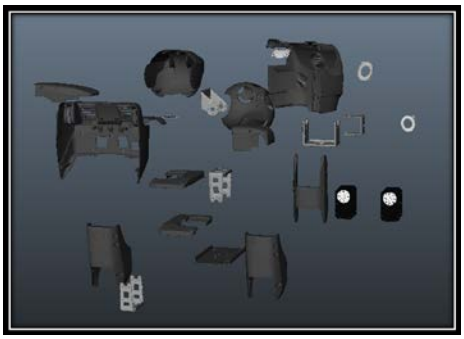

Figura 7 Piezas del robot Darwin OP importadas en *Maya*.

- Paso 2*: aplicación de escala*. - La manipulación de las piezas requirió de la especificación de una escala común para armar el modelo.
- Paso 3*: definición de jerarquía entre piezas.* - Una vez armado el modelo, se definió la jerarquía entre las diferentes piezas, la cual consiste en asociar conjuntos de partes del robot como piernas y brazos para unirlas entre sí, definiendo cadenas cinemáticas [7]. Esta asociación permite que el modelo se pueda animar o dotar de movimiento. La figura 8 muestra las piezas utilizadas para una pierna, el color verde se emplea para resaltar cómo se realizó la asociación.

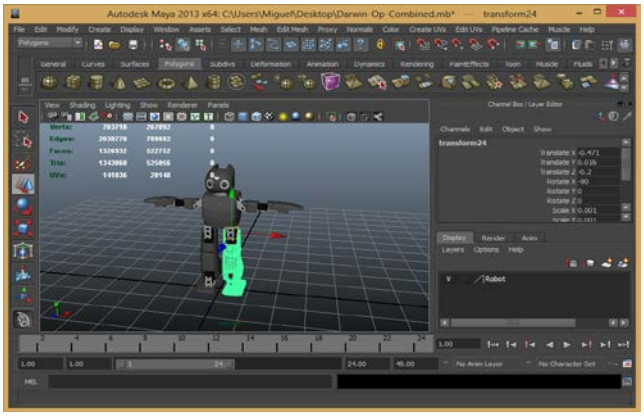

Figura 8 Asociación de piezas para la pierna del robot Darwin OP en *Maya*.

- Paso 4*: Definición de articulaciones.* Una de las tareas más importantes en el proceso de animación, consiste en definir la posición de las articulaciones que se asocian al modelo. En este caso, las articulaciones se representan en el modelo por motores y corresponden a elementos del robot como rodillas, codos o muñecas.

- Paso 5*: Asignación de rangos de movilidad para las articulaciones. -* Además de definir la posición de las articulaciones, se requiere establecer rangos de movilidad (o restricciones) de la parte del cuerpo que se desea animar, con el propósito de evitar daños en el robot original al intentar ejecutar un movimiento considerado como no permitido. La Figura 9 muestra una articulación y su rango de movilidad [7]. Los rangos se ajustan a cada uno de los ejes cartesianos en un espacio tridimensional (*x, y, z*).

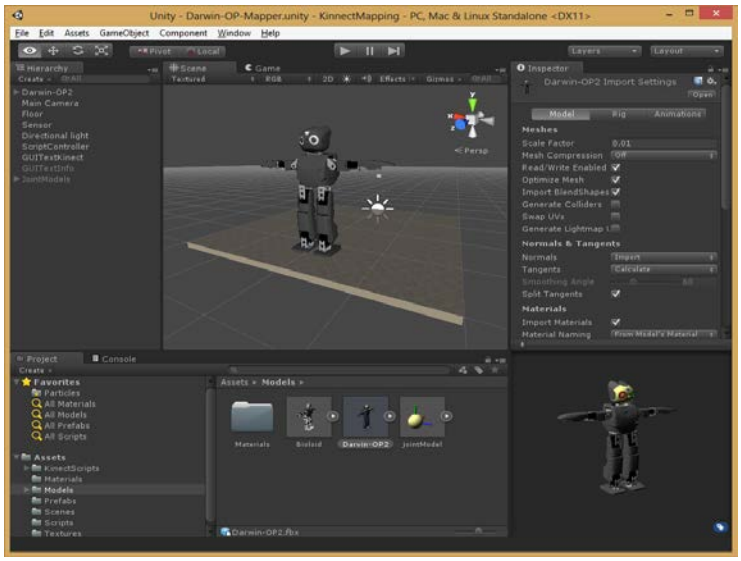

Figura 9 Modelo en 3D del robot Darwin OP importado en *Unity*.

- Paso 6*: Exportación del modelo.-* Una vez definidas las articulaciones y sus rangos de movilidad, el modelo resultante se exporta en formato FBX y se importa en Unity, con la intensión de establecer una conexión entre el modelo y el dispositivo Kinect. Se requiere crear una escena para incorporar al modelo (ver la figura 9). La conexión entre la interfaz de Unity con el sensor de Kinect se trata en la segunda parte de la sección 5.

5. Integración de MS Kinect en Unity

Desde su concepción, Kinect ha revolucionado el campo de las interfaces naturales, en inglés "*Natural User Interface*" (NUI), así como el diseño de video juegos cuya interacción no requiere de utilizar las manos. Dado que la interacción

con Kinect se realiza con gestos y voz, sus aplicaciones se han extendido en áreas como cuidado de la salud, robótica, procesamiento de imágenes, seguridad o educación [6].

Kinect es un dispositivo de captura de movimiento, lo cual implica que utiliza métodos de procesamiento de imágenes y su digitalización para detectar la posición de las articulaciones de las personas con el propósito de interactuar con sistemas de software [19]. A pesar de que el lanzamiento de Kinect para Xbox fue en noviembre de 2010, no fue sino hasta junio de 2011 cuando Microsoft liberó la versión beta del SDK[1] [18]. Esto permitió que los desarrolladores pudieran tener acceso al sensor a través de librerías y APIs[2] para implementar aplicaciones de software utilizando al sensor como una interfaz multimodal.

La organización OpenNI fue de las primeras en crear una conexión entre los motores de video juegos y Kinect. Posteriormente, a partir de la versión 1.7 del SDK y la versión 4.5.2 de Unity, se pueden encontrar diferentes "*plug ins*". La conexión que se reporta en este documento utiliza dichas versiones. Los pasos realizados para establecer la conexión entre el modelo del robot y Kinect son:

- Paso 1: *Animación del modelo*. - En Unity, se busca en su Asset Store y se seleccionó el *"plug in"* "Kinect with MS-SDK" de RF Solutions. Este elemento carga automáticamente las librerías disponibles del SDK de Kinect para que pueda controlarse desde Unity; contiene scripts de ejemplos y se distribuye de forma gratuita.
- Paso 2: *Mapeo de articulaciones*. - El sensor de Kinect identifica la posición de hasta 20 articulaciones de dos jugadores. Dado que las articulaciones del robot no son las mismas que las articulaciones de una persona y que los rangos de movilidad son también distintos, se hizo un mapeo entre las articulaciones del Kinect y las del robot. La Figura 10 muestra en color naranja las articulaciones reconocidas por el sensor, el lado derecho contiene las asociadas al robot. En la figura 11 se observa copm se asocian las articulaciones del cuerpo humano a las articulaciones del robot.

[1] SDK son las siglas de "Software Development Kit", herramienta de desarrollo de software.
[2] API son las siglas en inglés de "Application Programming Interface", interfaz de programación de aplicaciones

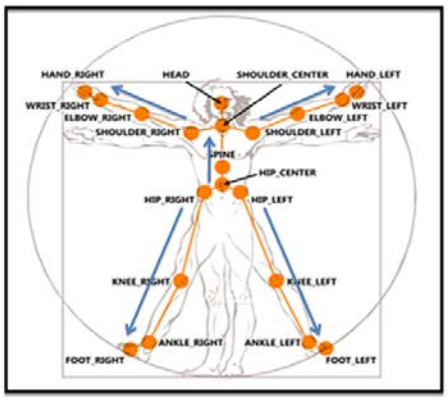

Figura 10 Articulaciones identificadas por el sensor del Kinect.

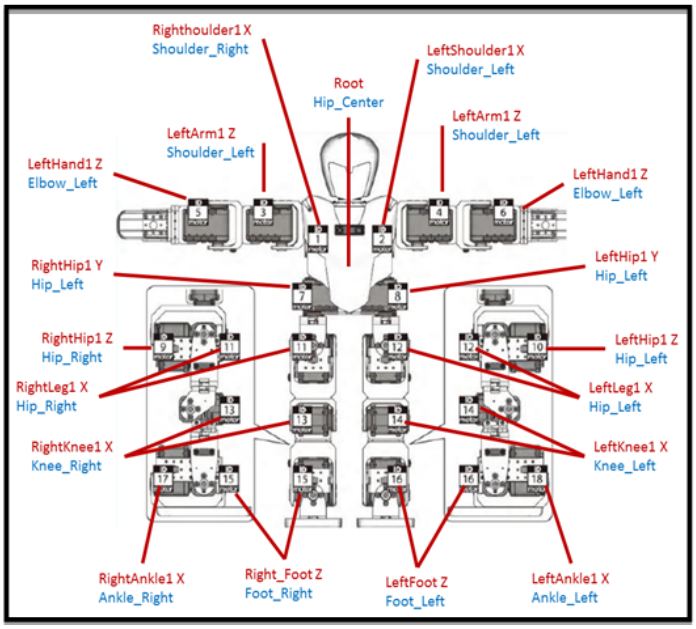

Figura 11 Articulaciones identificadas modelo 3D del robot Darwin OP importado en Unity.

- Paso 3: Implementación de scripts. Después del mapeo, se modifican los scripts de ejemplo para mover las articulaciones del modelo 3D del robot con base en la captura de movimientos del usuario. Los resultados preliminares se incluyen en la sección 6.

6. Resultados

Los resultados del modelado del robot Darwin OP se organizan en dos grupos: los relacionados con la funcionalidad y los relacionados con el diseño y

conectividad. Con respecto a la funcionalidad, el modelo se asoció a través de un mapeo con las 20 articulaciones del cuerpo humano que detecta el sensor de Kinect, (ver paso 2 de la sección 5). A través de un conjunto de pruebas y observaciones directas, se verificó la reacción correcta del modelo a los movimientos del usuario. La figura 12 de la izquierda muestra la posición de inicio del modelo en Unity, a partir de la cual se puede iniciar la animación. Del lado derecho se muestra cómo se ve el modelo cuando el usuario coloca los brazos hacia abajo.

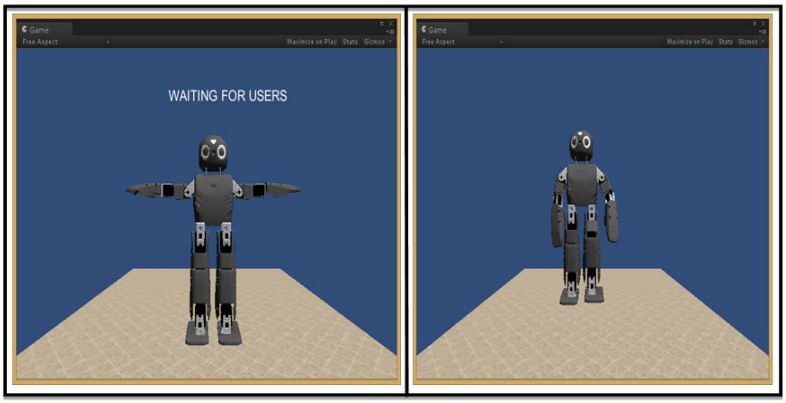

Figura 12 Posición inicial de modelo (izq.) y captura de un movimiento del usuario (der.).

De manera experimental, cuando un usuario movía el cuerpo considerando las articulaciones del modelo, se pudieron observar las limitaciones de la animación.
En relación con el diseño, se tiene lo siguiente:

- Un modelo en 3D del robot Darwin OP que se integró en maya, al cual se le asociaron cadenas cinemáticas para su animación.
- La identificación de rangos de movilidad para las articulaciones asociadas al robot. En relación a la conectividad, se logró el establecimiento de una conexión del motor de videojuegos de Unity con el dispositivo de Kinect para animar el modelo del robot.

7. Conclusiones

Este documento describió un conjunto de herramientas utilizadas para el desarrollo de video juegos, con el propósito de utilizar algunas que permitan dotar

de animación a un modelo en 3D del robot Darwin-Op utilizando Kinect. Para ello se integraron y asociaron un conjunto de piezas que forman las cadenas cinemáticas. El modelo se integró en un escenario en Unity, en el cual se pueden analizar los valores de posición de las articulaciones y tareas como rotación y traslación.

Actualmente, el modelo 3D emplea un gran número de polígonos. Como trabajo a futuro, se propone reducir el número de polígonos para incrementar la fluidez y velocidad de ejecución del modelo, de manera que los movimientos se perciban más naturales. Se planea también la implementación en el modelo de rutinas de movimiento determinadas.

8. Bibliografía y Referencias

[1] Goldstone W, Unity 3.x Game Development Essentials. 2011. Packt Publishing.

[2] D. H. Eberly, 3D Game Engine Design, a practical approach to real-time computer graphics. 2006. Morgan Kaufmann Publishers.

[3] A. Watkins, Creating games with Unity and Maya. Focal Press. 2011.

[4] Una visión de la industria de videojuegos en México. http://www.motor dejuegos.net/uploads/1/5/4/7/15475184/mdj_reporte2013.pdf. Junio del 2016.

[5] S. Boeykens, Unity for architectural visualization. 2013. Packt Publishing.

[6] J. Abhijit, Kinect for Windows SDK Programming Guide. 2012. Packt Publishing.

[7] A. Benitez, I. Huitzil, A. Casiano, J. De la Calleja, M. A. Medina, “PUMA 560: Robot prototype with graphic simulation environment”. Advances in Mechanical Engineering. Vol 2. No. 1. ISSN: 2160-0619. 2012. Pp. 17-24.

[8] Unreal Development Kit. https://www.unrealengine.com/. Junio del 2016.

[9] CryEngine 3. http://www.crytek.com/cryengine/cryengine3/overview. Junio del 2016.

[10] Game Maker Studio. https://www.yoyogames.com/studio. Junio del 2016.

[11] Autodesk Maya. © 2016 Autodesk Inc. http://www.autodesk.mx/products /3ds-max/overview Junio del 2016.

[12] Autodesk 3DS Max. © 2013 - 2016 YoYo Games Ltd. https://www.yoyogames.com/studio. Junio del 2016.

[13] Blender. http://www.blender.org/. Junio del 2016.

[14] Cinema 4D. © MAXON Computer. http://www.maxon.net/es/products /cinema-4d-studio.html. Junio del 2016.

[15] ZBrush. © 2016 Pixologic, Inc. http://pixologic.com/zbrush/ features/ZBrush4R6 /. Junio del 2016.

[16] Gimp. © 2001-2016. The GIMP Teamhttp://www.gimp.org/. Junio del 2016.

[17] Microsoft Kinect SDK 1.7. © 2016 Microsoft. http://www.microsoft.com/en-us/download/details.aspx?id=36996. Junio del 2016.

[18] Kinect asset para Unity. https://www.assetstore.unity3d.com/en/#!/ content/7747. Junio del 2016.

[19] Modelo 3D del robot Darwin-OP. © 2016 MakerBot ® Industries, LLC. http://www.thingiverse.com/thing:9793. Junio del 2016.

[20] Autodesk Mudbox. © 2016 Autodesk Inc http://www.autodesk.com/ products/mudbox/overview. Junio del 2016.

9. Autores

El Dr. Antonio Benitez Ruiz realizó el Doctorado en Ciencias de la Computación y la Maestría en Ciencias Computacionales en la Universidad de las Américas Puebla (UDLAP). Es egresado de la Licenciatura en Computación de la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla (BUAP). Su trabajo de investigación está relacionado con las áreas de robótica reactiva, algoritmos de planificación de movimientos, graficación por computadora e interacción humano - robot. Cuenta con reconocimiento a Perfil Deseable del PRODEP desde 2007.

La Dra. María Auxilio Medina Nieto realizó el Doctorado en Ciencias de la Computación y la Maestría en Ingeniería en Sistemas Computacionales en la Universidad de las Américas Puebla (UDLAP). Es egresada de la Licenciatura en

Computación de la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla (BUAP). La Dra. Medina ha participado en proyectos relacionados con agentes, bibliotecas digitales, bases de datos y redes sociales; cuenta con el reconocimiento a perfil deseable de PRODEP desde 2005. Es miembro nivel candidato del Sistema Nacional de Investigadores (SNI). Actualmente, su investigación está dirigida a temas como web semántica, representación del conocimiento a través de ontologías, tecnologías de la información y comunicación (TICs), e interacción humano – computadora.

Dr. Jorge de la Calleja Mora realizó el Doctorado en Ciencias Computacionales y la Maestría en Ciencias Computacionales en INAOE. Es egresado de la Licenciatura en Computación de la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla (BUAP). Cuenta con el reconocimiento a perfil deseable de PRODEP desde 2007 y es miembro nivel candidato del Sistema Nacional de Investigadores (SNI). Su trabajo de investigación está relacionado con las áreas de Aprendizaje Automático y Visión por Computadora. Cuenta con reconocimiento a Perfil Deseable del PRODEP desde 2007.

El Lic. Miguel Ángel Medina Nieto. Es egresado de la Licenciatura en Computación de la Facultad de Ciencias de la Computación (FCC) de la Benemérita Universidad Autónoma de Puebla (BUAP). Actualmente es estudiante de la Maestría en Ingeniería en Sistemas y Cómputo Inteligente en la Universidad Politécnica de Puebla. Este programa de Maestría se encuentra adscrito al PNPC de CONCACYT. Sus áreas de interés son Aprendizaje Automático y Visión por Computadora.