Congr. Int. en Ing. Electrónica. Mem. ELECTRO, Vol. 40, pp. 211 - 218, Oct 2018, Chihuahua, Chih. México
http://electro.itchihuahua.edu.mx/memorias_electro/MemoriaElectro2018.zip
ISSN 1405-2172

SIMULACIÓN Y EMULACIÓN ROBÓTICA EN MEDIOS ALTERNOS

Ávila Martínez Jesús H., Baray Arana Rogelio E., García Mata, Carmen L., Márquez Gutiérrez Pedro R., Robledo Vega Isidro
Tecnológico Nacional de México/Instituto Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación
Ave. Tecnológico No. 2909, Chihuahua, Chih., México, CP 31310
e-mail: jhavila@itch.edu.mx, rbaray@itch.edu.mx, pmarquez@itch.edu.mx, clgarcía@itch.edu.mx, irobledo@itch.edu.mx

RESUMEN.

Este artículo presenta un análisis de seis plataformas de simulación 3D tipo "*open source*" para el desarrollo de simulación dinámica de robots. Estas permiten crear en un entorno virtual el robot a ser estudiado tanto en su cinemática como en su dinámica y realizar pruebas que permitan validar la planeación de movimientos y trayectorias en el espacio de trabajo del robot, esto es importante ya que en los procesos de manufactura donde se ven involucrados los robots, se requiere una considerable inversión en tiempo y costo cuando se quiere realizar una modificación de las tareas que ejecuta el robot, sin embargo, una solución es tener un ambiente virtual (aún con restricciones) donde se desarrolle la planeación, y validación de las modificaciones de las tareas que requiere ejecutar el robot dentro del proceso de manufactura, sin interrumpir el proceso real, por lo tanto, es en estas plataformas "*open source*" donde se ve la oportunidad de desarrollar e implementar un entorno de simulación en una plataforma de creación de video juegos como lo es Unity 3D donde se evalúa y se compara con otras plataformas.

Palabras clave: Simulación, Emulación, Robótica, *Unity 3D*, mundo virtual.

ABSTRACT.

This article presents an analysis of six "open source" 3D simulation platforms for the development of dynamic simulations of robots. These previous allow to create in a virtual environment the robot being studied including its kinematics, dynamics and tests that allow to validate the planning of movements and trajectories in the work space of the robot. This is important since in manufacturing processes where robots are involved, it requires a considerable investment of time and money whenever we want to make a modification of the tasks that the robot executes, however, a solution is to have a virtual environment where is possible the planning and modification of the tasks that the robot needs to execute within the manufacturing process, without interrupting the real process. Therefore, it is in these "open source" platforms where it is seen the opportunity to develop and implement a simulation environment in a videogame creation platform such as Unity 3D where it is evaluated and compared with other platforms.

Key words: Simulation, Emulation, Robotic, Unity 3D, Virtual words.

1. INTRODUCCIÓN

La simulación y emulación de sistemas robóticos en la actualidad son herramientas poderosas que permiten analizar y optimizar diseños de prototipos robóticos, modificar su operatividad dentro de los procesos productivos, así como la programación de estos, antes de su valoración en tiempo real. En los procesos industriales, cuando estos tienen que cambiar sus operaciones para nuevos productos, la planeación de las nuevas tareas hasta su validación es una de las fases que más consume tiempo, el utilizar hoy en día simuladores 3D que permitan crear una representación del proceso real con todos sus componentes en forma virtual, permite flexibilizar ese proceso de planeación y validación de un cambio de las tareas a ejecutar por un robot en el proceso productivo, sin con ello afectar el proceso actual que se está ejecutando. El desarrollo de simuladores para el área de la robótica ha crecido de forma significativa en comparación con años anteriores, por lo cual, se pueden encontrar una diversidad de paquetes de *software* tanto comerciales, como del tipo "*open source*" o "*closed source*" como lo son *Gazebo*, *V-Rep*, *Webots*, *Vortex*, *OPD* o *BULLET*, donde los dos últimos fueron de los primeros desarrollados con "*Physics Engines*" (el término en inglés "*Physics Engines*" es utilizado en los simuladores gráficos como módulos especializados para representar las propiedades físicas de los objetos, por medio de algoritmos matemáticos y en el texto se menciona como "motor de física") y los cuales son utilizados generalmente para la creación de video juegos.

Este artículo forma parte de los resultados del trabajo de implementar una Interface Hombre Máquina (*Human Machine Interface*, HMI) para un brazo robótico PUMA TQ MA2000 incorporando de forma integral, el control, simulación, emulación y visualización de este dentro de la misma Interface, incorporando modelos 3D. Es un tema vigente y en el cual hay aportaciones muy interesantes como el presentado por Serena Ivaldi *et al* [1] donde se deduce que los paquetes de *software* que pueden satisfacer la necesidad del proyecto son: *Gazebo*, *V-Rep*, *Webots*, *Vortex*, *OPD* o *BULLET* que son herramientas utilizadas dentro de los robots móviles. En este artículo se busca que un paquete de *software* "*open source*", que cuente con un "*Physics Engines*" y además sea compatible con algún *middleware* robótico [2] como por ejemplo *ROS*, *YARP OROCOS*, *MIRO*, *Urbi*, *Orca*, *OpenRDK*, etc. Esto con el fin de llegar a integrarlos en trabajos futuros. Considerando estos

puntos, los paquetes de *software* que cumplen con esos requerimientos son *Gazebo* y *V-Rep.*

En el 2014, Alen y Silva [3] presentan una aplicación para un robot móvil KUKA desarrollando la simulación el ambiente del paquete de *software* V-REP. Los autores utilizan V-REP como simulador del robot móvil Kuka en forma virtual y para crear un escenario también virtual en donde el robot debe de realizar tareas de navegación, orientación, etc. en un ambiente industrial. Los autores destacan principalmente entre otras características que V-REP permite manejar programas elaborados en otros lenguajes como LUA, así como también de la simulación de sensores, para darle percepción al robot móvil, además de involucrar el análisis mecánico y cinemático del robot.

Una investigación presentada en el 2015 por Christoph Bartneck *et al*[4] demuestran lo eficiente que resulta implementar *Unity 3D* en el área de la robótica, ellos desarrollan "*El Motor de Robot*" (TRE por sus siglas en inglés) el cual permite conectar los movimientos en el mundo virtual a los motores y sensores del mundo real. Su enfoque se basa en la evolución que ha tenido la investigación multidisciplinaria en el campo de la "*Interacción Hombre-Robot*" (HRI por sus siglas en inglés) y en el desarrollo de juegos de computadora ya que estos dos campos de estudio se enfrentan al mismo problema: lograr resultados satisfactorios en HRI y videojuegos sin recurrir a programar directamente en lenguajes de bajo nivel, de aquí que la solución fue el desarrollo de los motores de juegos (*Game Engines*) los cuales incluyen editores gráficos para crear animaciones y la interacción con otros usuarios.

Por otro lado un trabajo reciente como el de Yoshiaki Mizuchi *et al* [5], presentan una nueva arquitectura de software para simular la interacción de un robot humanoide (HRI) y un medio ambiente virtual (VR), su característica principal es que es una arquitectura basada en la nube para resolver el problema de la captura y almacenamiento de una cantidad masiva de datos con los cuales se recrea la experiencia de interacción HRI-VR, proponiendo una nueva arquitectura de sistema para "*SIGVerse*" utilizando *Unity* y *ROS* [6][7].

En este artículo se presenta una etapa que forma parte de un proyecto para desarrollar un brazo articulado, y el objetivo de esta etapa es mostrar los criterios que se toman en consideración para la selección de la plataforma en el desarrollo una interface HMI, sustentada en base a la investigación y pruebas que se realizan en las diversas plataformas planteadas como medios alternos en la literatura revisada, con el fin de desarrollar interfaces que permitan simular y emular brazos articulados en diversos escenarios industriales.

2. ANÁLISIS DE HERRAMIENTAS Y AMBIENTES VIRTUALES

Dentro del área de la simulación de sistemas robóticos existen diversos paquetes de *software* que satisfacen diferentes necesidades, dependiendo del campo de investigación del usuario, los principales paquetes de *software* son: Gazebo, V-Rep, Webots, Vortex o ARGoS. [1]

La intención del proyecto es realizar una HIM para un brazo robótico PUMA TQ MA2000 con los siguientes requerimientos:

- Control del brazo robótico.
- Comunicación inalámbrica entre ordenador y robot.
- Monitoreo de posición.
- Simulación 3D del robot.
- Seguimiento 3D del robot.
- Compatibilidad con algún *middleware* robótico.
- Plataforma de desarrollo de licencia *"open source"*

Dentro de los paquetes de *software* utilizados en la simulación de sistemas robóticos, se menciona que alguno de ellos fueron desarrollados con ayuda de algún "motor de física", que son comúnmente utilizados en la industria de video juegos para la creación de estos. Una vez realizado el análisis de las propuestas publicadas, y teniendo en consideración las propuestas planteadas en los artículos anteriormente mencionados y en busca de las mejores opciones para el desarrollo del proyecto se propone incluir a *Unity 3D* durante el proceso de evaluación.

Como primer prueba de evaluación y como se muestra en la tabla 1 se hace una comparativa entre los paquetes de *software* que se mencionaron anteriormente, para así lograr seleccionar los más aptos a las necesidades del proyecto. Primeramente uno de los requisitos de selección y que nos obliga a descartar automáticamente a *ARGoS*, *Webot* y *Vortex Dynamics* es que estos no son paquetes de *software "open source"* o no tienen soporte con algún *middleware* robótico, característica esencial para el proyecto. En segundo se observa que paquetes como *V-rep* y *Gazebo* son compatibles con el *middleware* robótico de ROS, esta característica permite aprovechar los recursos proporcionados por ROS para la simulación y manipulación de robots, por lo cual se seleccionan ambos paquetes de *software* para ser evaluados. Así mismo está selección es muy utilizada en el medio, sin descartar a *Unity*, que a pesar de no soportar directamente algún *middleware* robótico, existen trabajos de investigación donde esta plataforma logra vincularse con ROS y con YARP.

Durante la segunda etapa de la evaluación de los paquetes de *software* seleccionados se realizan pruebas de funcionalidad de cada una de las herramientas que proporcionan, para familiarizarse con sus comandos e ir conociendo las ventanas y

su interacción con esas herramientas de desarrollo, así como evaluar los módulos especiales que cada uno de los paquetes ofrece, y valorar el nivel de complejidad de su operación, y la eficiencia de la implementación de proyectos en el ambiente de desarrollo que cada paquete de software proporciona al usuario.

Tabla 1: Evaluación de *software* etapa 1

Plataforma	Gráficos	Robotic middleware	Simulación multiusuario	Comunidad de desarrolladores	S.O	Tipo de licencia
Gazebo	Muy buenos	ROS	n/a	Amplia	GNU / Linux	Open source
ARGoS	Buenos	n/a	n/a	Baja	GNU / Linux, MAC OS	Open source
Webots	Excelentes	ROS	Soportada	Media	GNU / Linux, MAC OS, Windows	comercial
V-Rep	Muy buenos	ROS	Soportada	Media	GNU / Linux, MAC OS, Windows	Open source
Vortex Dynamics	Excelentes	ROS	Soportada	Baja	GNU / Linux, Windows	comercial
Unity Engine	Excelentes	n/a	Soportada	Extensa	GNU / Linux, MAC OS, Windows	Open source

Esas pruebas funcionales se realizan bajo la aplicación de ciertas características a valorar en cada uno de ellos como son: motor de física, soporte en la nube, comunicación TCP/IP, etc., las cuales se listan en la tabla 2 y como se puede observar en la se da a conocer un conjunto de características específicas de cada uno de los paquetes de software evaluados, donde nos podemos percatar de las ventajas que tienen unos sobre otros, por ejemplo *Gazebo* y *V-Rep* resultan ser compatible con "motores de física" como ODE, *Bullet*, entre otros, mientras que *Unity* solo cuenta con *PhysX*. Por otro lado tenemos a *V-Rep* con sus módulos especializados para implementar funciones de detección de colisiones y realizar cálculos de cinemática directa e inversa. En el caso de Gazebo y Unity, esas funciones se pueden realizar, sin embargo, no cuentan con un módulo especializado que realice esos procesos.

Tabla 2: Evaluación de *software* etapa 2

CARACTERISTICAS	Software		
	Gazebo	V-Rep	Unity
Motor de física.	✓✓	✓✓	✓
Soporte en la nube.	✓	✓	✓
Comunicación TCP/IP.	✓	✓	✓
Compatible con API Remotas.	✓	✓	n/a
Exportación de modelos CAD.	✓	✓	✓
Detección de colisiones.	✓	✓✓	✓
Cinemática Inversa y directa.	✓	✓✓	✓
Adquisición de datos externo.	✓	✓	✓
Middleware (Robótica).	✓✓✓	✓✓	✓
API externa de control.	✓	✓	n/a
Compatible con Matlab.	✓	✓	✓
HMI (interna en producto final).	x	x	✓
Instalación Multiplataforma.	x	✓	✓
Exportación Multiplataforma.	x	x	✓
Comunidad de desarrolladores.	✓✓	✓	✓✓✓

Otra consideración que se debe de tomar en cuenta, es el hecho de que *Unity* a diferencia de Gazebo y *V-Rep* ofrece la ventaja de implementar una HMI que facilita el control de forma interna sin la necesidad de tener que desarrollar una API externa en otro paquete de *software* como por ejemplo en *Microsoft VisualStudio*. Otra ventaja que nos brinda es al momento de la creación de proyectos, ya que este nos permite exportarlo directamente como una aplicación para *Windows*, MAC OS o *Linux* inclusive como un archivo *Web*, todo esto sin la necesidad de realizar modificaciones agresivas dentro del código. También a pesar de no ser un paquete de *software* de desarrollo en el área de la robótica cuenta con una extensa comunidad que lo emplea con ese propósito, ya que la finalidad de *Unity* es proporcionar a los usuarios una herramienta de desarrollo interactivo de 2D y 3D de la manera más accesible.

Dentro de las actividades realizadas en este período que cubre la operación y control del paquete de *software* se obtienen los resultados que se muestran en la tabla 3 donde es posible observar un período de tiempo de 30 horas de desarrollo e implementación en ciertas actividades específicas en cada una de las plataformas (no se consideró monitorear el tiempo de capacitación para el manejo de estos paquetes de *software* por lo complejo y diferentes que son cada uno ellos).

Como se puede notar en los resultados de la evaluación, se realizan una serie de pruebas de funciones que todos los paquetes de *software* tienen en común, con algunas excepciones como son la "Comunicación serial a la UI" o "Escritura del API remota", las cuales no son necesarios dentro de *Unity* ya que este cuenta con herramientas que suplen estas funciones, esto permitió concentrarse en otras pruebas funcionales y obtener avances significativos en comparación a *Gazebo* o *V-Rep*.

Los objetivos de realizar esta evaluación es el conocer y comprobar la eficiencia, potencia y practicidad de estas herramientas. En el estudio y análisis de esas herramientas se encuentra que cada una de ellas tienen características a favor y otras en contra, pero en particular para las metas que se quieren alcanzar en la presente investigación se definen como las mejores opciones a *Gazebo* y a *V-rep*, ya que estos son paquetes de *software* especializados en ambiente de robótica, pero estos demandan una cantidad de tiempo mayor para desarrollar habilidades que permitan el uso de estos a un nivel que se pueda implementar la solución al problema planteado. Los tiempos anteriormente mostrados son los que se invirtieron en dichas pruebas funcionales, no se incluye el tiempo invertido en la investigación y lectura de manuales de las herramientas para obtener conocimiento acerca del correcto manejo de estas.

Por otro lado tenemos a *Unity*, un motor gráfico de video juegos que no cuenta con módulos especializados en el campo

de la robótica, sin embargo cuenta con componentes muy similares a *Gazebo* y *V-Rep*, así como otras herramientas que lo vuelven competitivo. Teniendo esto en cuenta los resultados al evaluarlo son mejores con respectos a los demás, logrando un mejor rendimiento y en menor tiempo. Esto no significa que *Unity* sea mejor pero sí que muestra mejores resultados a corto plazo, por ende lo vuelve la opción más viable para realizar el proyecto

Tabla 3: Evaluación de *software* etapa 3

Actividades	Software					
	Gazebo		V-Rep		Unity	
	Estado	Tiempo	Estado	Tiempo	Estado	Tiempo
Instalación.	✓	3	✓	1	✓	2
Compresión de la GUI.	✓	3	✓	3	✓	3
Exportación de modelos SVG.	✓	1	✓	2	✓	1
Exportación de modelos 3D CAD.	✓	2	✓	3	✓	1
Construcción de escenas.	✓	5	✓	5	✓	2
Escritura del API remota.	X	5	X	2	N/A	0
Comunicación serial a la UI.	X	3	X	2	N/A	0
Importación de modelos de brazos robóticos.	X	3	X	2	✓	2
Simulaciones.	✓	5	✓	5	✓	5
Simulaciones de brazos robóticos.	X	0	✓	5	✓	5
Comunicación con tarjeta adquisitoria de datos.	X	0	X	0	✓	4
Tiempo total (horas)	6	30	7	30	9	25

3. SIMULACIÓN EN *UNITY*

El modelado 3D es el proceso de representación matemática de cualquier objeto tridimensional a través de alguna herramienta que permita desarrollar la acción. El producto de esto se le conoce como modelo 3D.

El modelado 3D juega un papel de suma importancia debido al diseño de la interface de usuario con que permita la simulación 3D tanto del brazo robótico como su entorno, esto con la finalidad de facilitar al usuario el control y la programación de este.

Durante este parte del desarrollo se definen las herramientas que se utilizan para desarrollar el proyecto, tales como la selección del motor gráfico, lenguaje de programación y el paquete de software. Después de realizar la evaluación se opta por usar las siguientes herramientas:

- *Unity 3D*.
- Lenguaje de programación en c#.
- Software cad: *Solidworks*.
- Software cad: *Autodesk 3Ds max*.

donde *Unity 3D* cumple la función del motor gráfico, este nos da la opción entre dos lenguajes de programación los cuales son *java* y c#. Aquí se optó por seleccionar la segunda opción, para realizar los modelos en 3D. Se utilizan dos, los cuales son *Solidworks* y *Autodesk* 3*Ds max* esto debido a que el motor gráfico de *Unity* no procesa directamente los archivos generados por *Solidworks*; se vio la opción de exportar un archivo guardado como extensión *step* hacia *Autodesk 3Ds max* donde en este se editaron algunos eslabones del diseño 3D como lo son los puntos unión, uniones entre piezas y puntos de pivoteo para optimizar las articulaciones donde existen los giros.

Una vez seleccionados los paquetes de software se procede a realizar el modelo del brazo articulado. En la figura 1, se puede apreciar el modelado 3D del brazo robótico PUMA 2000 que se modela partiendo del brazo articulado real y del cual se toman las medidas de cada una de las partes mecánicas y electromecánicas de la estructura que conforman al brazo robótico. Este modelo se realiza usando *Solidworks.*

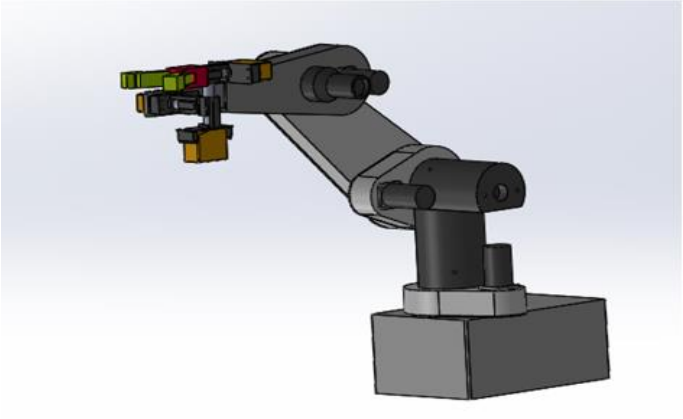

Figura 1: Brazo robótico Puma 2000 modelado en *Solidworks.*

El siguiente paso después de realizar el modelado 3D es exportarlo a *Unity 3D*, pero como se mencionó anteriormente este no es compatible con los archivos creados en *Solidworks* (Aun cuando en la última actualización de *Unity 3D* 2018.1 se crearon librerías para este proceso, estas son de paga y por lo tanto, no se opta por usarla), por lo cual, se investigan paquetes de software capaces de crear modelos 3D en formatos compatibles entre ellos, de los cuales, se encuentran a *MAYA 3D*, *Blender* y *AutoDesk 3Ds Max*. Se toma la decisión de utilizar *Autodesk 3Ds Max* para realizar la compatibilidad del modelo hecho en *Solidworks* con *Unity*. En este caso se utiliza el modelo 3D del brazo articulado desarrollado y con el formato de *Solidworks*, este se exporta al paquete de software *Autodesk* 3D*s Max*. para primeramente visualizarlo, a partir de ahí, ya en el ambiente de *Autodesk* 3D*s Max* se realizan las modificaciones y ajustes necesarios en el modelo y se editan uno por uno los puntos de pivoteo de cada una de las articulaciones para generar el movimiento cinemático entre cada uno de los eslabones que forman la estructura del brazo

articulado. Los resultados de estas operaciones se observan en la figura 2.

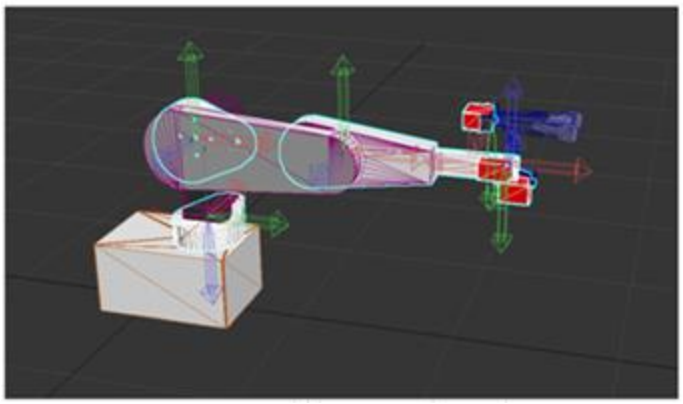

Figura 2: Brazo robótico puma 2000 exportado a 3DS MAX

Una vez que se lleva a cabo la virtualización y la edición del brazo robótico dentro de *Autodesk 3D Max* se prosigue ahora con la exportación a un formato FBX, ya que este formato es compatible con *Unity 3D*. De esa forma es posible transferir los diseños desarrollados en *Solidworks* a *Unity 3D*, Los resultados que se obtienen se muestran en la figura 3, donde además se observa el ambiente virtual en el que se encuentra el brazo articulado.

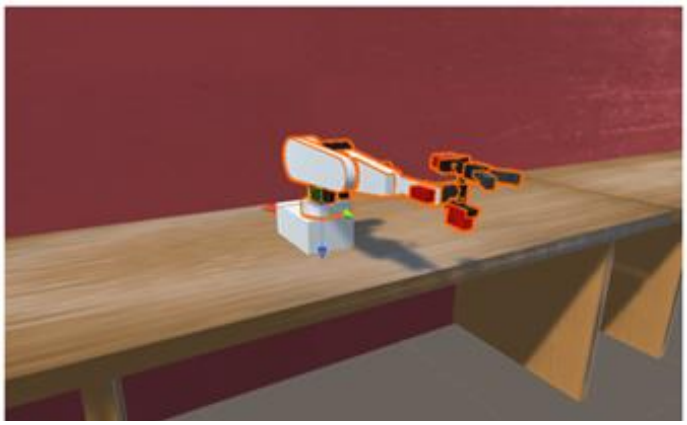

Figura 3: Brazo robótico Puma 2000 exportado a *Unity* 3D

El ambiente virtual del laboratorio en donde se encuentra el brazo articulado también es diseñado en *Unity 3D*, creando de esa forma el ambiente en donde el brazo articulado realiza las tareas programadas. Esos escenarios se pueden modificar y crear el ambiente industrial en donde el robot desarrollará sus tareas de manufactura y ensamble. Estos resultados se muestran en la figura 4.

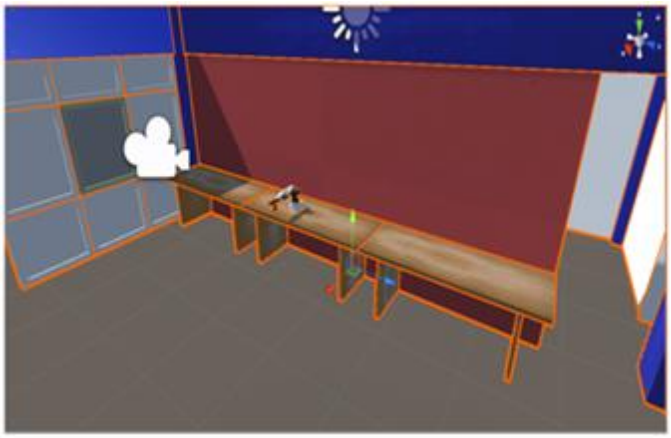

Figura 4: Virtualización y exportación del ambiente del robot a Unity 3D

4. DESARROLLO DEL MODELO CINEMÁTICO Y SU SIMULACIÓN.

Al proceso de digitalización de un brazo robótico le sigue el paso de desarrollar un modelo cinemático del brazo articulado por medio de la metodología de Denavit-Hartenberg, a través de la cual obtenemos los valores de los parámetros de la matriz DH. Lo anterior permite realizar la validación de los movimientos cinemáticos del brazo para diversas posiciones y movimientos angulares de cada una de las articulaciones del robot. La tabla 4 y las Figuras 5, 6 y 7 muestran los resultados que se obtienen con el análisis cinemático del brazo articulado..

Tabla 4: Resultados de la matriz DH.

Art	θi	di	ai	i
1	θ1	L1	0	90
2	θ2	a2	-L3	0
3	θ3	0	-L4	0
4	θ4+90	-a5	0	90
5	θ5+90	L6	0	90
6	θ6	-L7	0	0

Las simulaciones se realizaron desarrollando el algoritmo Denavit-Hartenberg en Matlab. Una vez obtenidos los desplazamientos angulares, se alimenta la interfaz de Unity para llevar acabo la simulación dinámica del brazo articulado y comprobar si los puntos de pivote se establecieron de forma correcta. Se comprueba que las posiciones y rotaciones adquiridas por cada uno de los eslabones de la estructura del brazo, así como la posición del efector final se corresponden con las obtenidas en Unity. Estos resultados se pueden validar observando las posiciones que se muestran en las figuras 5, 6 y

7, comparándolas con las posiciones de las figuras 8, 9 y 10 obtenidas en *Unity*.

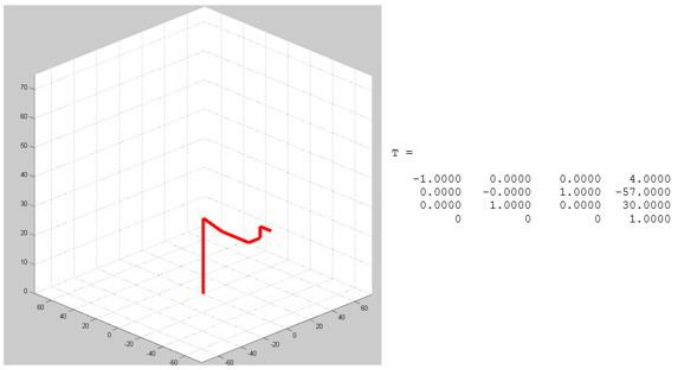

Figura 5: Simulación de movimiento articular del Brazo robótico en Matlab (Posición de "*home*")

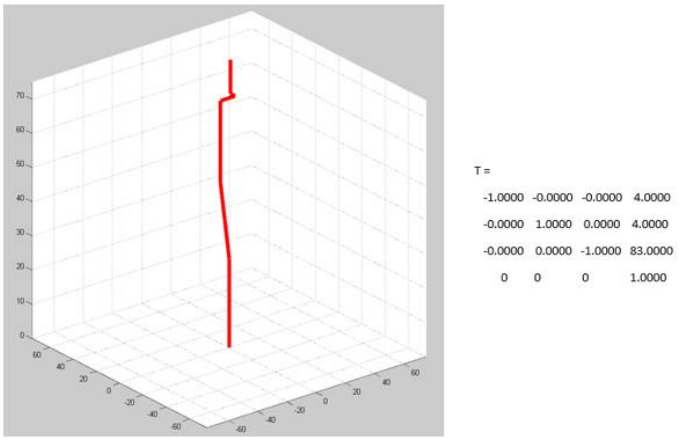

Figura 6: Simulación de movimiento articular del Brazo robótico en Matlab (Rotación en: A2 = 90°)

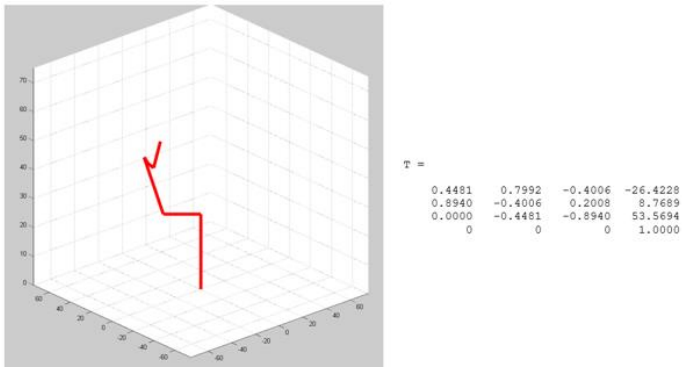

Figura 7: Simulación de movimiento articular del Brazo robótico en Matlab (Rotación en: A1 = -90 °, A3 = 45 ° y A4 = 45 °)

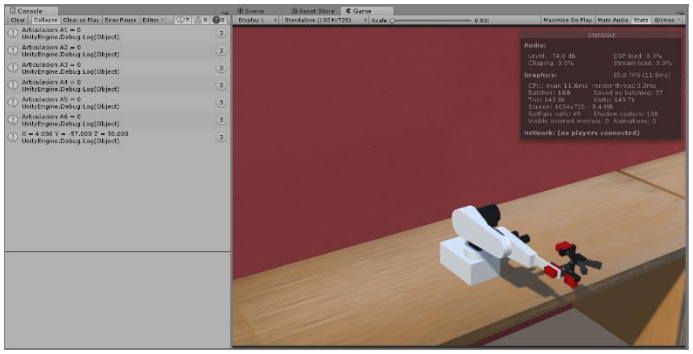

Figura 8: Simulación de movimiento articular del Brazo robótico en *3D* (Posición de "*home*")

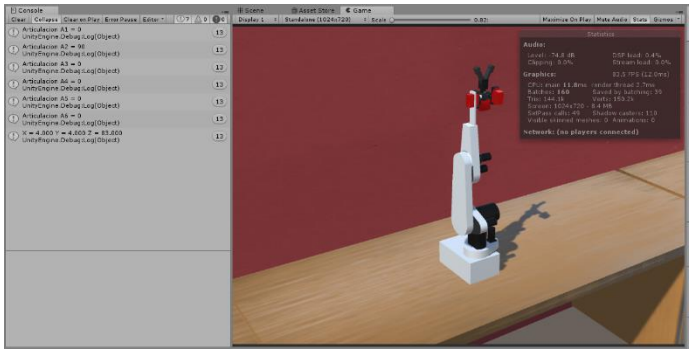

Figura 9: Simulación de movimiento articular del Brazo robótico en *Unity 3D* (Rotación en: A2 = 90°)

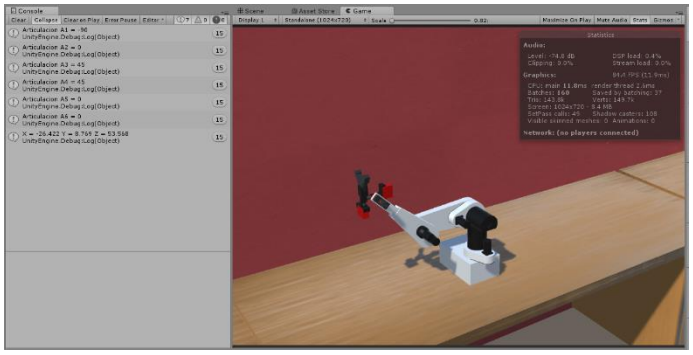

Figura 10: Simulación de movimiento articular del Brazo robótico en *Unity* 3D (Rotación en: A1 = -90°, A3 = 45 ° y A4 = 45 °)

La demostración matemática obtenida con la matriz DH y Matlab fue validada con la simulación dentro de *Unity 3D* y obteniendo resultados similares en tolerancias de +/- 0.001

5. IMPLEMENTACIÓN Y DESARROLLO DE LA INTERFACE DE USUARIO Y SIMULACIÓN.

Después de validar los cálculos de la cinemática de la estructura mecánica del brazo, por medio de la metodología Denavit-Hartenberg así como la simulación, y la emulación del brazo robótico en el entorno virtual desarrollado en *Unity 3D*, se procedió con el desarrollo de los primeros prototipos de interfaces de usuarios. Por medio de construcción de escenas, templetes y otras herramientas que *Unity 3D* proporciona, se obtuvo un avance significativo en la investigación donde se lograron los siguientes puntos:

- Desarrollo de una HMI
- Comunicación entre ordenador y tarjeta adquisitoria de datos
- Monitoreo de sensores de posición
- Simulación de cinemática directa.
- Simulación de posición por medio de sensores externos.

Como se puede observar en la Figura 11 Se muestra el primer prototipo de la HMI, donde se muestra el panel principal mostrando las funciones que se pretenden desarrollar como lo es la configuración del puerto serial y simulación de cinemática inversa. Por el momento el prototipo solo realiza cálculos de cinemática directa como se muestra en la figura 12 y 13 donde se puede observar que la simulación corre entre 70 a 110 FPS sin problema.

Manejando la estructura de control que se muestra en la figura 14 se puede apreciar que se están manejando 3 *scripts* principales dentro del compilador de *Unity 3D*. Los cuales se encargan del control, monitoreo, simulación del sistema y el encargado de la comunicación y gestión de datos. Donde este último es el encargado de manejar la información entre la tarjeta de adquisición de datos y la computadora.

El *script* de control es el encargado de sincronizar las funciones de la HMI tales como habilitar el módulo de simulación, monitoreo, configuración del puerto serial y seleccionar el método de cálculo de posición ya sea de la cinemática inversa o directa, además de realizar los cálculos de trayectoria y posición para el brazo robótico tanto en el ambiente virtual como en el real.

El *script* de Monitoreo es el módulo encargado de procesar los datos que provienen de las señales obtenidas de los sensores de posición de cada una de las articulaciones del brazo robótico, para determinar la posición y orientación de cada uno de los de los sistemas de coordenadas $\{S_i\}$ que posicionan y orientan al efector final.

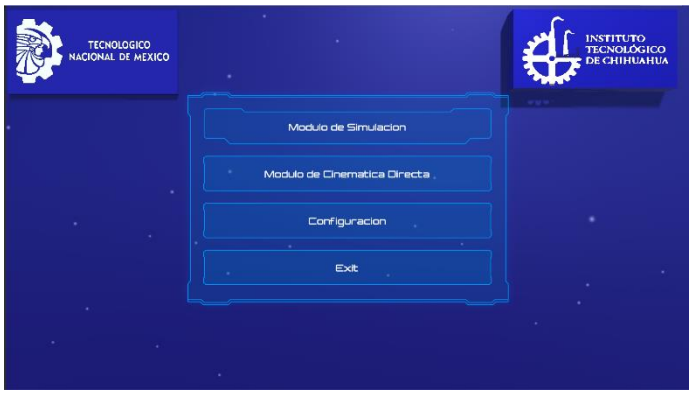

Figura 11: Prototipo visual de la HMI.

Figura 12: Prueba del módulo de cinemática directa.

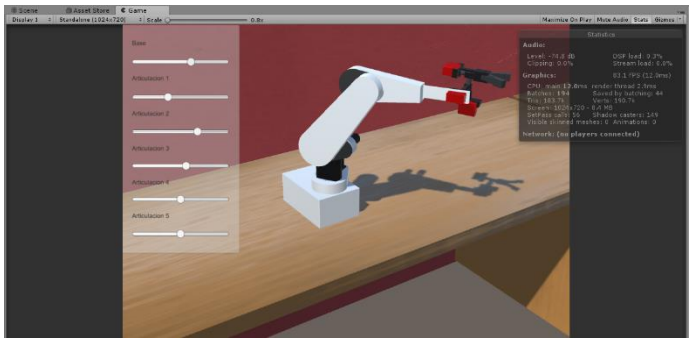

Figura 13: Prueba del módulo de cinemática directa.

El *script* de simulación es el encargado de simular los movimientos y trayectorias del brazo robótico de manera *off-line* (sin recibir ni enviar señales al *script* de comunicación) de tal manera que el usuario pueda emular el comportamiento del brazo robótico para evitar colisiones, dentro de la simulación al

momento de colisionar los eslabones del brazo robótico del mundo virtual cambian de color y envían una señal de alerta para el usuario.

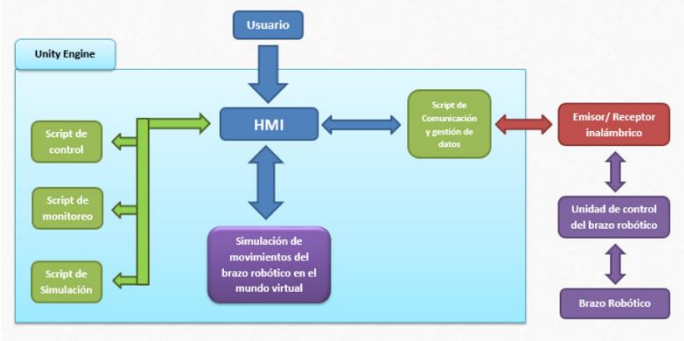

Figura 14: Diagrama a bloques de la estructura de control.

Otros resultados obtenidos fue el desarrollo de la simulación donde con la ayuda del monitor de estadísticas dentro de *Unity* como se muestra en las figura 8, 9, 10, 12 y 13 donde lanzaron resultados del consumo del sistema como:

- CPU load : entre 1.8 a 12ms por operación.
- Renderización: 2.0 a 3.5 ms por tarea.
- Fotogramas por segundo (FPS): De 70 a 235 Fps.

6. CONCLUSIONES

En este artículo se presentó el análisis de seis plataformas para la simulación en 3D de sistemas robóticos que actualmente se utilizan en el medio de desarrolladores de video juegos y en robótica virtual, así como los criterios de selección final de tres paquetes de *software* que permiten desarrollar un modelo virtual de un brazo articulado. Se desarrolló el proceso para llevar a cabo el modelado del brazo articulado real. Se obtuvo el modelado cinemático del brazo robótico y se implementó el algoritmo Denavit-Hartenberg en *Matlab* y se validó en *Unity* programando los respectivos movimientos angulares (θ_i) de cada una de las articulaciones del brazo robótico y generar la posición y orientación final de la herramienta ($x,y,z,\alpha,\beta,\gamma$). Se desarrolló la interfaz de usuario prototipo HMI, la cual se implementó con tres funciones básicas en el menú de opciones. La idea clave de la propuesta es la utilización de tres paquetes de diseño los cuales son Solidworks, *Autodesk 3Ds Max* y *Unity3D* de tal manera de separar por fases el proyecto para hacer una integración y tener el

La ventaja que proporciona la siguiente combinación de paquetes de *software* Solidworks, *Autodesk 3Ds Max* y *Unity 3D* nos permite desarrollar modelos de alta precisión gracias a Solidworks que pesar que *Autodesk 3Ds Max* o *Unity 3D* son capaces de lograr estos requieren habilidades avanzadas para desarrollarlo, bajo este criterio de operación se basó el desarrollo del proyecto, de ir dividiéndolo por secciones utilizando las herramientas que satisfagan cada necesidad para poderlas integrar de manera gradual para así obtener el avance y los resultados planeados.

Para trabajo futuro se pretende mejorar y ampliar las capacidades de interacción entre el usuario y el brazo articulado por medio de la HMI, desde el mundo virtual al mundo real realizando las siguientes mejoras:

- Aumentar las funciones en el menú de inicio.
- Incorporar un sistema de administrados de cuentas de usuario.
- Incorporar funciones que permitan detectar colisión entre elementos.
- Integrar el módulo de cinemática inversa
- Aumentar la librería de brazos robóticos disponibles para simulación.
- Comunicación y control incorporando algún *middleware* robótico.

6. REFERENCIAS.

[1] S. Ivaldi, J. Peters, V. Padois, y F. Nori, «Tools for simulating humanoid robot dynamics: a survey based on user feedback», en *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, 2014, pp. 842–849.

[2] E. Einhorn, T. Langner, R. Stricker, C. Martin, y H.-M. Gross, «MIRA - middleware for robotic applications», en 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Vilamoura-Algarve, Portugal, 2012, pp. 2591-2598.

[3] R. F. T. Alen y M. F. Silva, «Development and simulation on V-REP of an algorithm for the RoboCup@Work BNT», en 2014 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Espinho, Portugal, 2014, pp. 315-320

[4] C. Bartneck, M. Soucy, K. Fleuret, y E. B. Sandoval, «The robot engine - Making the unity 3D game engine work for HRI», en 2015 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Kobe, Japan, 2015, pp. 431-437.

[5] Y. Mizuchi y T. Inamura, «Cloud-based multimodal human-robot interaction simulator utilizing ROS and unity frameworks», en 2017 IEEE/SICE International Symposium on System Integration (SII), Taipei, 2017, pp. 948-955.

[6] Y. Hu y W. Meng, «ROSUnitySim: Development and experimentation of a real-time simulator for multi-unmanned aerial vehicle local planning», SIMULATION, Transactions of the Society for Modeling and Simulation International, vol. 92, n.o 10, pp. 931-944, oct. 2016.

[7] Enrico Sita, Csongor M´ark Horv´ath, Trygve Thomessen, P´eter Korondi2, Anthony G. Pipe, «ROS-Unity3D Based System for Monitoring of an Industrial Robotic Process», Proceedings of the 2017 IEEE/SICE International Symposium on System Integration, Taipei, Taiwan, December 11-14, 2017.