DISEÑO DE HERRAMIENTAS Y PROCESOS PARA EL APOYO AL TRABAJO EN CLUBES DE ROBÓTICA: FASE DISEÑO DE SOFTWARE. (RoboticSAD)

YULI MARCELA MARÍN PEÑA

UNIVERSIDAD PEDAGÓGICA NACIONAL

FACULTAD DE CIENCIA Y TECNOLOGÍA

DPTO. DE TECNOLOGÍA - LIC. EN ELECTRÓNICA

2013

DISEÑO DE HERRAMIENTAS Y PROCESOS PARA EL APOYO AL TRABAJO EN CLUBES DE ROBÓTICA: FASE DISEÑO DE SOFTWARE. (RoboticSAD)

Trabajo de grado presentado por:

YULI MARCELA MARÍN PEÑA

Para obtener el título en

Licenciatura en Electrónica

Director del Trabajo

Diego Mauricio Acero - Ing. Electrónico

UNIVERSIDAD PEDAGÓGICA NACIONAL

FACULTAD DE CIENCIA Y TECNOLOGÍA

DEPTO. DE TECNOLOGÍA – LIC. EN ELECTRÓNICA

BOGOTÁ. D. C

DICIEMBRE DE 2013

Nota de Aceptación

_____

_____

_____

_____

_____

_____

_____

Firma del Director de Trabajo de Grado

_____

Firma del Jurado

_____

Firma del Jurado

Bogotá D.C., Diciembre de 2013

Dedicatorias

Dedico este trabajo a mis padres Miryan Peña y Urbano Marín; a mis hermanos Dibier Marín, Yessika Marín y Neyda Marín; a mi sobrino Santiago Marín y a mi novio Leonardo Murillo; ellos son y serán siempre mi mayor motivación para lograr lo que me propongo y salir adelante con mi proyecto de vida; su amor, comprensión y apoyo son muy importantes para mí. A todos ellos los admiro mucho por lo que son y les agradezco todo lo que han aportado a mi crecimiento personal.

Agradecimientos

Agradezco en primer lugar a Dios por la oportunidad de ingresar a la Universidad y de culminar mi carrera.

A mis padres por todo su apoyo y paciencia ante las situaciones adversas que se han presentado.

A mis hermanos por sus consejos y por su apoyo, en especial a mi hermano, "Mi genio de las matemáticas" quien siempre estuvo dispuesto a colaborarme con mis dilemas matemáticos.

Agradezco a Leonardo, quien más que mi novio, es mi gran amigo, porque siempre estuvo ahí para aconsejarme y darme una voz de aliento cuando me sentía con incapacidad para alcanzar la meta. "Mil gracias por tu amor incondicional".

Agradezco a mi asesor Ing. Diego Mauricio Acero, quien siempre estuvo dispuesto a colaborarme en la resolución de las dificultades que se presentaron en el desarrollo del proyecto.

A Sandra Quintero (amiga y compañera) y a su esposo Ing. Oscar Arboleda por todo el apoyo que me brindaron. También Lorena, que junto con Sandra, me brindaron su amistad sincera y acompañamiento.

A los profesores por todas sus enseñanzas y colaboración durante toda la carrera. Especialmente a Diego Quiroga por sus asesorías de Unity 3D.

Un agradecimiento también para mi sobrino porque ha logrado mostrarme la importancia que tienen las cosas más simples de la vida.

TITULO: Diseño de Herramientas y Procesos Para el Apoyo al Trabajo en Clubes de Robótica: Fase Diseño de Software.

TITLE: Design Tools and Processes To Support Work Robotics Clubs: Phase Software Design.

RESUMEN

Este proyecto consiste en el desarrollo de un software de simulación 3D para un Robot Móvil y un Brazo Robótico; para lograr este objetivo se utiliza el motor de juegos Unity 3D con el lenguaje de programación C# Script y el software de modelado 3D y animación Blender. Además el software se complementa con un recurso de comunicación (mediante puerto USB) con el Brazo Robótico, esto se hace a través de la aplicación Maestro Control Center de Pololu Maestro Servo Controller.

El software está orientado para ser utilizado como recurso en la enseñanza de la Robótica, específicamente en los Clubes de Robótica que están vinculados en el proyecto de la Universidad Pedagógica Nacional, en el programa Licenciatura en Electrónica.

SUMMARY

This project involves the development of a 3D simulation software for a Mobile Robot and Robotic Arm, to achieve this goal using the game engine Unity 3D with the programming language C # Script and software 3D modeling and animation Blender. In addition the software is complemented by a communication resource (through USB port) Robotic Arm with, this is done through the Control Center application of Pololu Maestro Servo Controller.

The software is intended to be used as a resource in the teaching of robotics, specifically in Robotics Clubs are linked in the draft National Pedagogical University in the Bachelor program in Electronics.

1. Información General	
Tipo de documento	Trabajo de grado.
Acceso al documento	Universidad Pedagógica Nacional. Biblioteca Central
Título del documento	Diseño de herramientas y procesos para el apoyo al trabajo en Clubes de Robótica: Fase Diseño de Software. (RoboticSAD)
Autor	**Marín Peña, Yuli Marcela**
Director	**Acero, Diego Mauricio**
Publicación	Bogotá. Universidad Pedagógica Nacional, 2013, 88 p.
Unidad Patrocinante	Universidad Pedagógica Nacional.
Palabras Claves	Software, Robótica, Simulación gráfica, Unity 3D, Blender, C# Script.

2. Descripción
El trabajo de grado tiene como objetivo principal, el diseño y desarrollo de un software para la simulación gráfica (3D) de un brazo robótico y un robot móvil, que sirve como herramienta para el apoyo al trabajo en clubes de robótica, desarrollado por la Universidad Pedagógica Nacional. Este software es la primera de tres fases, por tal motivo se documenta de manera explícita toda la información referente a su diseño, desarrollo e implementación, en consecuencia se dejan como recursos un manual de usuario y un manual de desarrollador para facilitar su desarrollo evolutivo. Este proyecto, a su vez hace parte de un proyecto más ambicioso que se divide en tres partes: Software, Hardware y componente Pedagógico. Para su desarrollo se utilizaron recursos de software tales como Unity 3D, Blender y Maestro Control Center. El Lenguaje de programación utilizado es C# Script.

3. Fuentes

[1] Jacobson, Ivar; Booch, Grady y Rumbaugh, James. *"El Proceso Unificado de Desarrollo de Software".* Editorial Pearson, S.A., Addison Wesley. Madrid, 2000.

[2] Jacobson, Ivar; Booch, Grady y Rumbaugh, James. *"El Lenguaje Unificado de Modelado. Manual de referencia".* Editorial Pearson, S.A., Addison Wesley. Madrid, 2000.

[3] Pressman Roger S. Adaptado por Ince, Darrel. *"Ingeniería del Software Un enfoque práctico"*. Editorial Mc Graw Hill. Quinta Edición.

[4] Barrientos Antonio, Peñin Luis Felipe, Balaguer Carlos, Aracil Rafael. *"Fundamentos de Robótica".* Editorial Mc Graw Hill. Mardid , 1997.

[5] Ferguson, Jeff; Patterson, Brian, Beres, Jason; Boutquin, Pierrre y Gupta, Meeta. *"La biblia de C#"* Ediciones Anaya Multimedia (Grupo Anaya S.A). Madrid, 2003.

[6] Unity 3D®. (s.f). *Manual de Usuario y Manual de Scripting.* Consultado el 28 de abril de 2013, de URL http://unity3d.com/

[7] Blender®. (s.f.). *Documentación*. Consultado el 25 de Febrero de 2013, de URL http://www.blender.org/

[8] Pololu Corporation®. (2001). *"Pololu Maestro Servo Controller User's Guide"*. Consultado el 15 de Junio de 2013, de URL http://www.pololu.com/docs/0J40/all

[9] *"Foro Unity 3D".* (s.f.). Consultado el 28 de Junio de 2013, de URL http://www.unityspain.com/

[10] *"Wiki de Blender".* (s.f.). Consultado el 2 de Marzo de 2013, de URL http://wiki.blender.org/

[11] *"Realidad Virtual".* (s.f.) Consultado el 15 de Marzo de 2013, de URL http://www.realidadvirtual.com/

[12] *"Funciones del Software Educativo".* (s.f.). Consultado el 10 de Marzo de 2013, de URL http://proton.ucting.udg.mx/materias/robotica/r166/r151/r151.htm

4. Contenidos

En el Capítulo 1: Planteamiento del Problema, encontramos lo referente al problema de desarrollo, la justificación, la delimitación y los objetivos del proyecto; finalmente se enuncian algunos antecedentes importantes en el tema.

En el Capítulo 2: Marco Referencia, se encuentran los referentes teóricos que influyeron significativamente en el desarrollo de este proyecto.

En el Capítulo 3: Aspectos metodológicos, se hace una descripción de la metodología escogida para el proyecto en conjunto con la organización del proyecto (que está en concordancia con la metodología elegida).

En el Capítulo 4: Desarrollo del proyecto, se encuentra todo lo concerniente al diseño y desarrollo del Software objetivo de este proyecto, incluyendo desarrollo técnico, análisis, resultados, conclusiones, recomendaciones, dificultades y proyecciones del mismo.

5. Metodología

Este trabajo utiliza la metodología de desarrollo de software RUP (Proceso Unificado Racional), esta consiste en cuatro fases:

Fase de inicio: Se identifican los casos de uso del software, además se establecen los requerimientos y actividades necesarias para el desarrollo del proyecto.

Fase de elaboración: Se realiza la planeación de actividades y se hace un organigrama orientado a la construcción del software teniendo en cuenta los recursos existentes para llevarla a cabo.

Fase de construcción: Se hace la ejecución de las actividades y la documentación del software.

Fase de transición: Se analiza el funcionamiento del software mediante la implementación del mismo, teniendo en cuenta las necesidades del usuario. Si hay errores se procede a corregir.

6. Conclusiones

[1] RoboticSAD cuenta con herramientas que permiten una fácil interacción del usuario, para llevar a cabo la simulación grafica de un Robot Móvil y un Brazo Robótico construidos como HADs para la enseñanza de la robótica.

[2] RoboticSAD puede ser catalogado como un "software educativo" debido a que se pensó y se desarrolló para cumplir las características funcionales definidas para enmarcarlo en esta categoría.

[3] El software realizado tiene como finalidad facilitar el aprendizaje de la robótica, teniendo en cuenta situaciones en que no se cuente con los recursos de hardware necesarios, en cuyo caso se limita la experimentación e implementación física. Además permite que el usuario tome decisiones sin temor a equivocarse y a ocasionar daños en los componentes físicos.

[4] Las instrucciones mínimas para simular el comportamiento del Brazo Robótico están definidas a partir de un motor elegido, su posición en grados y su tipo de velocidad (baja, media, alta). En el caso del Robot Móvil las instrucciones son: Avanzar y Contar, Girar 90° a la derecha y girar 90° a la Izquierda.

[5] Para desarrollar un software de simulación gráfica se requiere un entorno de desarrollo que permita interactuar con modelos 3D así como un software para realizar dichos modelos.

[6] El software desarrollado se puede modificar y complementar fácilmente mediante la creación de nuevas escenas y scripts, gracias a que se diseñó pensando en facilitar el cumplimiento de los objetivos propuestos para las siguientes fases de desarrollo, como el caso de la fase 2 que requiere un componente de comunicación con el Hardware.

Elaborado por:	**Marín Peña, Yuli Marcela**
Revisado por:	**Acero, Diego Mauricio**

Fecha de elaboración del Resumen:	02	09	2013

CONTENIDO

pág.

LISTA DE FIGURAS

pág

LISTA DE TABLAS

pág

LISTA DE ANEXOS

0. INTRODUCCIÓN

La simulación actualmente es una herramienta muy importante en los procesos de enseñanza y aprendizaje en diversas áreas. En el caso particular de la robótica, muchas veces los docentes se encuentran con la dificultad que tienen los estudiantes para comprender las secuencias y los movimientos de los robots, y el cómo se deberían manipular para cumplir ciertas funciones con estos.

Este proyecto consiste en el desarrollo de un software de simulación grafica 3D, que sirve como apoyo al trabajo en clubes de Robótica, realizado por la Universidad Pedagógica Nacional, este se realizó con el fin de simular un robot móvil y un brazo robótico, que a su vez hacen parte de la Fase Diseño de Hardware, realizada por la estudiante Sandra Quintero. Para su ejecución se realizó un análisis de los recursos existentes que pudieran optimizar el desarrollo del mismo, por lo tanto se utilizaron recursos de software tales como *Unity 3D*®, para interfaz GUI y todo lo referente a programación en lenguaje C#; *Blender*® para moldeado en 3D de los robots y *Maestro Control Center*® para la comunicación con el Hardware.

El software de Simulación será utilizado por el docente como una herramienta óptima para el desarrollo de contenidos de robótica, sin embargo este proyecto queda abierto a la "*posibilidad de ser trabajado en otras tesis"*, con el fin de ampliar sus funciones de simulación gráfica 3D a programación, evaluación de contenidos, entre otras; por esto el proyecto se divide inicialmente en tres fases, de las cuales aquí se presenta y se desarrolla la primera: *"Simulación Grafica 3D".*

CAPÍTULO 1

1. PLANTEAMIENTO DEL PROBLEMA

1.1 Problema de desarrollo

En la Universidad Pedagógica Nacional (UPN), desde la práctica pedagógica, se viene trabajando desde hace algunos años con Robótica Escolar y Clubes de Robótica, para llevar a cabo esta labor se necesitan de muchas herramientas como LEGO® (acrónimo de *Leg Godt* que significa jugar bien) y otros dispositivos y/o elementos de robótica que los estudiantes puedan utilizar para su aprendizaje y que puedan aplicar al desarrollo de sus proyectos.

Es importante que este trabajo de la Universidad crezca y se enriquezca cada día más, y para ello es necesario contar con herramientas de los ejes que complementan la robótica escolar, tales como software, hardware, metodologías pedagógicas, entre otras.

Para los estudiantes es muy provechoso tener un software con el cual puedan simular y programar los robots que estén utilizando o que estén desarrollando, por esta razón es adecuado un software de simulación y programación de robots, acorde a las nuevas herramientas de hardware que se desarrollen para apoyar el trabajo de la UPN en robótica escolar y clubes de robótica. Por este motivo se contribuye con el trabajo de la UPN diseñando y desarrollando un software de simulación grafica de robots, que permite a los estudiantes conceptualizar de manera eficiente y optimizar el trabajo que se hace con las herramientas de hardware.

1.2 Justificación

Se escoge este proyecto por gusto personal en el tema de Robótica Escolar y porque éste permite un alto crecimiento profesional e intelectual, además se apoya a la UPN con parte del trabajo que se hace desde la Licenciatura en Electrónica.

El trabajo que realiza la UPN con Robótica Escolar y Clubes de Robótica es un espacio que requiere de muchos recursos didácticos y apoyo por parte de los estudiantes del departamento de tecnología; por este motivo, y como aporte al proyecto general: *"Diseño de herramientas y procesos para el apoyo al trabajo en Clubes de Robótica"* (propuesto por el Ing. Diego Acero), se diseña un software para la simulación de un brazo robótico de mínimo tres grados de libertad y un robot móvil, que también hacen parte del *"Proyecto conjunto que integra Propuesta pedagógica, Software y Hardware".* Adicionalmente se estructura un conjunto de instrucciones que hagan posible la simulación de los robots.

Para el desarrollo del software se tiene en cuenta que éste ayudará a los estudiantes en la comprensión de lo que se está realizando con la parte del hardware y lo que se puede desarrollar con esta; además éste debe acompañar a los estudiantes mediante la didáctica adecuada que les ayude a comprender conceptos de robótica de una forma clara.

Por lo expuesto en los párrafos anteriores se propone un software que permita hacer lo siguiente:

- Simulación gráfica.
- Programar por código.
- Concertar y enviar los programas al hardware mediante un protocolo de comunicación USB.
- Base de Datos para evaluación y seguimiento de los procesos que se llevan a cabo en el software.
- Página web con contenido del proyecto en conjunto y conexión al software.

1.3 Delimitación

Debido al alcance que pretende dársele a la parte del software en el proyecto general, este se divide en tres fases (tres tesis diferentes), de las cuales aquí se desarrolla ***únicamente la Fase 1***.

Fase 1. Esta fase corresponde a la simulación grafica de un brazo robótico de mínimo tres grados de libertad y un robot móvil, que serán desarrollados por Sandra Quintero en la parte de hardware correspondiente al proyecto conjunto.

Fase 2. Aquí se complementará el software agregando herramientas de programación por código, con el fin de que los estudiantes puedan desarrollar habilidades para el desarrollo de programas y además deberá desarrollarse el protocolo de comunicación para que el software se comunique con el hardware.

Fase 3. Esta fase se desarrollará una base de datos que pueda evaluar y hacer seguimiento de los procesos llevados a cabo en el software, esto con el fin de que los estudiantes puedan ver los errores cometidos y puedan ver la evaluación de su trabajo. También se deberá crear una página web que esté conectada al software y que contenga información y material didáctico acerca de las partes que integran el proyecto conjunto.

1.4 Objetivos del proyecto

1.4.1 Objetivo general

Diseñar y desarrollar un software para la simulación gráfica de un brazo robótico de tres grados de libertad y un robot móvil.

1.4.2 Objetivos específicos

- Definir el conjunto mínimo de instrucciones con las cuales sea posible simular el comportamiento de los robots.
- Definir los requerimientos y recursos técnicos necesarios para implementar el software objetivo de este proyecto.
- Implementar el software resultante del diseño y el análisis.

1.5 Antecedentes

[Tesis de Grado] DISEÑO Y CONSTRUCCIÓN DE UN KIT DE ROBÓTICA "ROBOPED" ENFOCADO HACIA LOS ESTÁNDARES EN TECNOLOGÍA E INFORMÁTICA PROPUESTOS POR EL MINISTERIO DE EDUCACIÓN NACIONAL. (Realizada por: Fabián Camilo Nieto Peña, Jaime Alberto Gallego Tovar y Edwin Fernando Reyes Samacá, Universidad Pedagógica Nacional, Facultad de Ciencia y Tecnología, Licenciatura en Electrónica, Bogotá D.C. 2008). Este proyecto incluye el Software RoboPed, el Hardware RoboPed y Unidades Didácticas; en cuanto a la parte del software este se hace únicamente

a partir de iconos debido a que hay dificultad para encontrar los errores del código; en cambio un software de iconos tiene mayor facilidad para encontrar errores debido a que se lleva a cabo a partir del lenguaje máquina. La interfaz gráfica se hizo con Visual Basic 6 y se implementó en él el protocolo RS232 para enviar el programa por el puerto serie del computador a un módulo central con micro controlador.

[Tesis de Grado] DISEÑO DE UN ENTORNO DIGITAL DE SIMULACIÓN PARA LA ENSEÑANZA DE LOS PRINCIPIOS DE MORFOLOGÍA, CONTROL Y COMUNICACIÓN EN LA ROBÓTICA. (Realizada por: Gloria Inés Henao Ortiz y Astrid Piñeros Torres, Universidad Pedagógica Nacional, Facultad de Ciencia y Tecnología, Licenciatura en Diseño Tecnológico con Énfasis en Sistemas Mecánicos, Bogotá D.C. 2008). En este proyecto se realiza un entorno digital de simulación para un Robot Ápodo Modular, teniendo en cuenta los principios básicos de la robótica: Morfología, Control y Comunicación. Este entorno se realizó con el fin de dar respuesta a las inquietudes de los docentes del área de tecnología del CED Don Bosco II, respecto a algunos temas de robótica. El software se desarrolló con Matlab a partir de las herramientas de los módulos Simulink, Reality Virtual Building y Graphic User Interfaz (GUI), para realizar las piezas del robot en 3D se utilizó el software Solid Works 2008. Este proyecto adopta un modelo pedagógico constructivista orientado a la Robótica Escolar para la enseñanza de los principios básicos de la robótica mencionados anteriormente.

[Tesis de Grado] SEGUIMIENTO ADAPTATIVO DE TRAYECTORIAS CON CONVERGENCIA EN TIEMPO FINITO DE UN ROBOT ANTROPOMÓRFICO VIRTUAL DE TRES GRADOS DE LIBERTAD. (Realizada por: Araceli González Rodríguez, Manuel Pineda Ortega y Dely Madai Soberanes Leal, Universidad Autónoma del Estado de Hidalgo, Instituto de Ciencias Básicas e Ingeniería, Centro de Investigación en Tecnologías de la información y Sistemas, Licenciatura en Sistemas Computacionales, Pachuca de Soto – Hidalgo 2007). Este es un trabajo en el cual se propone un software que permita la visualización en línea de un robot antropomórfico de tres grados de libertad, este se hace con el fin de que el usuario pueda experimentar con el robot virtual y un control no lineal por modos deslizantes de segundo grado, permitiendo la manipulación de las ganancias del robot y así mejorar su sintonización. “Las herramientas de desarrollo del software son

Visual C++, OpenGL para el Robot Virtual y para la "Interfaz de Usuario Delphi 7.0", de igual forma se apoyó en Matlab para la realización de la simulación fuera de línea y comprobación de resultados"[1].

[Tesis de especialización] MODELADO Y SIMULACIÓN DE UN ROBOT CAMINADOR BÍPEDO. (Realizada por: Edwin Francis Cárdenas Correa, Universidad Nacional de Colombia, Facultad de Ingeniería, Especialización en automatización Industrial, Bogotá D.C. 2005). En este trabajo se desarrolla un modelo computacional del robot, ya existente, UNROCA II (Universidad Nacional RObot CAminador II), del cual además se realiza una simulación en un entorno virtual de su caminata, con el objetivo de visualizarla en la pantalla de un ordenador; esto se hace para analizar y predecir el comportamiento físico que tendría el robot en un entorno real. Tanto la simulación como el modelo se realizan en el software Matlab 6.5, las piezas del robot en 3D se realizaron con el software Solid Edge versión 14 Profesional.

[1] Fuente: http://www.uaeh.edu.mx/docencia/Tesis/icbi/licenciatura/documentos/robot%20antropomorfico%20virtual.pdf

CAPÍTULO 2

2. MARCO DE REFERENCIA

Este marco de referencia se construye con base en las necesidades conceptuales que se presentaron para el diseño y desarrollo del software objetivo de este proyecto.

2.1 Robótica Escolar y Software

El software se ha convertido en una herramienta muy importante en la Robótica Escolar; tanto para realizar simulaciones, como para programar los algoritmos que van a ser ejecutados por el robot; es decir que un software de robótica puede convertirse en una herramienta propicia para el análisis y el desarrollo de conceptos de dicha disciplina.

2.1.1 Definición de términos

A continuación se presenta la definición formal de algunos términos de interés para el desarrollo de esta sesión, realizados por la *"Real Academia Española"* [2]

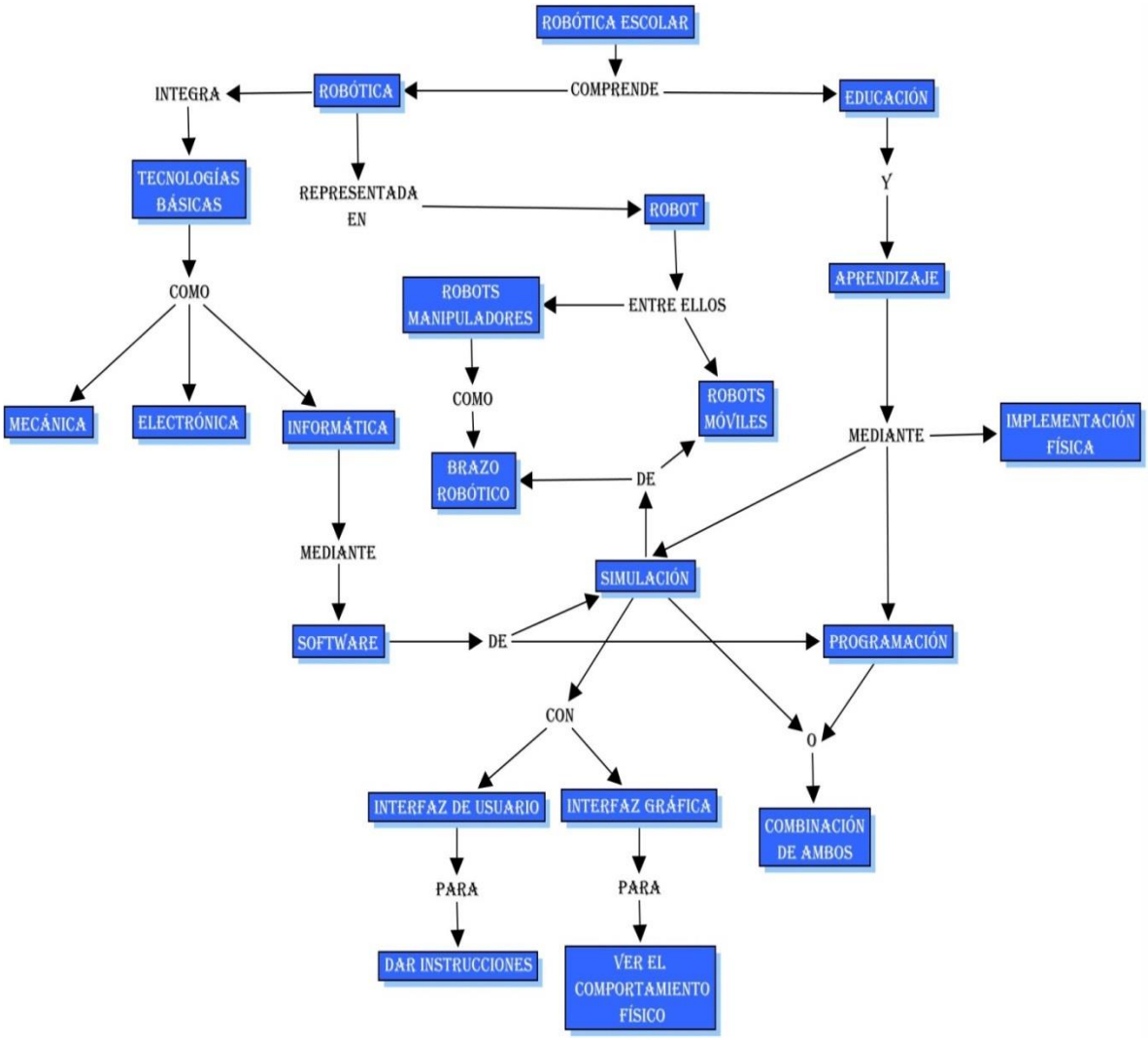
- *Robótica*.1. f. Técnica que aplica la informática al diseño y empleo de aparatos que, en sustitución de personas, realizan operaciones o trabajos, por lo general en instalaciones industriales.
- *Robot.* (Del ingl. robot, y este del checo robota, trabajo, prestación personal). 1. m. Máquina o ingenio electrónico programable, capaz de manipular objetos y realizar operaciones antes reservadas solo a las personas.
- *Software.* (Voz ingl.).1. m. Inform. Conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.
- *Simulador, ra.* 2. m. Tecnol. Aparato que reproduce el comportamiento de un sistema en determinadas condiciones, aplicado generalmente para el entrenamiento de quienes deben manejar dicho sistema.

[2] http://www.rae.es/rae.html

2.1.2 Integración de conceptos

La Robótica escolar integra una amplia gama de conceptos entre los cuales se encuentran la electrónica, la mecánica y la informática; el objetivo principal es la construcción de conceptos en tecnología a partir del desarrollo de proyectos de trabajo conjunto e individual, y representados en un Robot. Es importante recalcar que debe haber una relación coherente entre el Software y el Hardware del Robot[3], esto con el fin de lograr buenos resultados en cuanto a su funcionamiento.

Figura 1. Mapa Conceptual: Robótica Escolar



[3] Fuente: http://es.wikipedia.org/wiki/Rob%C3%B3tica_educativa

2.2 Software Educativo

Un software educativo es un material didáctico que puede acompañar los procesos educativos según convenga el docente, no se puede catalogar un software educativo como bueno o malo debido a que eso depende del uso que le dé el docente en el aula.

2.2.1 Funciones del Software Educativo[4]

- *Función informativa.* El software educativo presenta a los estudiantes de forma ordenada, conceptos que le ayuden a enriquecer su estructura cognitiva, facilitando así el proceso de construcción de conocimiento. Por ejemplo los simuladores y las bases de datos.
- *Función instructiva.* El software educativo se caracteriza por orientar a los estudiantes en su aprendizaje, esto se debe a que promueven determinadas acciones para lograr objetivos preestablecidos. Como ejemplo tenemos el software tutorial.
- *Función motivadora.* Usualmente el software educativo contiene elementos que captan y mantienen el interés de los estudiantes, enfocando así este interés como motivación hacia el aprendizaje. Esta es una característica del software educativo es de gran utilidad para el docente. Por ejemplo video juegos educativos.
- *Función evaluadora.* La evaluación del software puede ser implícita y explícita. La primera es cuando el estudiante cae en la cuenta de sus propios errores, por la respuesta que da el software, y procede a corregirlos. La segunda es cuando el software es el que va presentando información de valoración de procedimientos al estudiante.
- *Función investigadora.* El software educativo ofrece a los estudiantes entornos de investigación donde pueden buscar información, experimentar cambiando los valores de las variables del software y llegar a conclusiones a partir de sus acciones. Ejemplo: simuladores y software de construcción.

[4] Fuente: http://proton.ucting.udg.mx/materias/robotica/r166/r151/r151.htm

- *Función expresiva.* Los ordenadores tienen la capacidad de traducir símbolos en alguna instrucción y esto es lo que nos permite comunicarnos con ellos. El software educativo permite la comunicación con los estudiantes a partir de acciones de ambas partes. Ejemplo: Software que utilice un lenguaje de programación.

2.3 Diseño de software

Un buen proceso de diseño de software requiere que el desarrollador se esfuerce al máximo y dedique bastante tiempo para obtener un producto de calidad, además de esto debe existir comunicación permanente con el usuario, para no desviarse hacia objetivos erróneos. El desarrollo de software se refleja en el ingenio, motivación y orden del desarrollador.

Lo primero que se debe establecer para diseñar un software es el ámbito en el cual se ubica, como por ejemplo ámbito empresarial, educativo, entre otros. Posteriormente se debe hacer un análisis y definición de aspectos como:

- El alcance del proyecto y los antecedentes.
- Las características funcionales del software, los requisitos mínimos para lograr los objetivos y los riesgos de ejecución.
- La metodología que se utilizará para el desarrollo del software, teniendo en cuenta que se debe ajustar a las necesidades del usuario y del desarrollador.
- La planeación de tareas a realizar haciendo estimación del tiempo que tomará cada una de ellas.

2.4 Realidad Virtual

"La realidad virtual es por lo general un mundo virtual generado por ordenador (o sistemas informáticos) en el que el usuario tiene la sensación de estar en el interior de este mundo, y dependiendo del nivel de inmersión puede interactuar con los objetos del mismo en un grado u otro.

No obstante el termino realidad virtual también puede aplicarse a otros mundos virtuales generados por otros medios, como por ejemplo a través de la imaginación (sueños, libros, cine, etc...)

La realidad virtual ideal sería la que desde una inmersión total nos permita una interacción sin límites con el mundo virtual, además de aportarnos como mínimo los mismos sentidos que tenemos en el mundo real (vista, oído, tacto, gusto, olfato). Sin embargo, la mayoría de los sistemas actuales se centran únicamente en dos sentidos (vista y oído), debido a la dificultadas y costes de simular los otros sentidos."[5]

La realidad virtual hoy en día es una herramienta tecnológica que se aplica en muchos campos, como por ejemplo la educación, los videojuegos, la medicina, el entrenamiento militar, procesos industriales, etc.

2.5 Unity 3D

Unity 3D es un motor de aplicaciones 3D, desarrollado por *Unity Technologies*, que permite desarrollar videojuegos y simulaciones; también se pueden realizar aplicaciones 2D, pero su especialidad es el 3D. El tipo de licencias existentes para este motor son: licencia libre, licencia pro y una versión de prueba. El entorno de desarrollo está disponible para Windows y para Mac OS X.

Las aplicaciones pueden ser desarrolladas para plataformas como Windows, Mac, Web, Nintendo Wii, iPhone/iPad, Xbox 360, Play Station 3 y Android. Para su desarrollo Unity 3D utiliza tres lenguajes de script que son: Java Script, C# Script y Boo (derivado de Python); el diseño de las aplicaciones se hace a partir de escenas que interactúan según lo establezca el desarrollador en la programación.

Para la edición de scripts Unity 3D utiliza el software "MonoDevelop" por defecto, el cual viene incluido en el paquete de instalación general, sin embargo se puede usar un editor diferente que trabaje con los lenguajes de script ya mencionados. Cada nuevo script que se realice es considerado como una nueva clase en el proyecto, además de esto se debe tener en cuenta que Unity tiene sus propias funciones que se pueden utilizar para trabajar con las características de los objetos de las escenas.

Aunque Unity posee algunas herramientas para crear objetos tales como esferas, cubos, objetos vacíos, etc… Para realizar modelos 3D más complejos es aconsejable utilizar

[5] http://www.realidadvirtual.com/que-es-la-realidad-virtual.htm

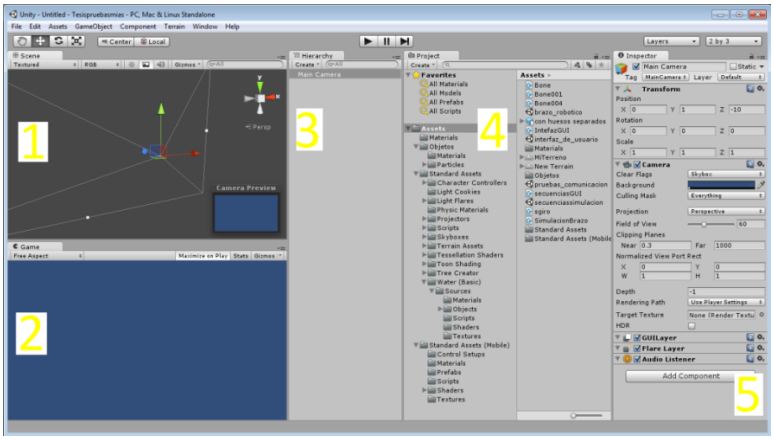
programas de modelado como Blender, Maya, 3Ds Max u otros, y después importarlos a Unity en un formato adecuado (.fbx, .3ds, .obj, entre otras.)

2.5.1 GUI de Unity 3D

La interfaz gráfica de usuario de Unity 3D (GUI de Unity 3D) se divide en cinco áreas principales:

- *[1] Scene.* Aquí, en la vista de escena, podemos organizar de manera visual el diseño del entorno 3D.
- *[2] Game.* Aquí se puede obtener una vista del juego cuando se esté ejecutando de acuerdo con las cámaras que estén interviniendo.
- *[3] Hierarchy.* Aquí se encuentran organizados en jerarquía todos los objetos existentes en la escena que se esté trabajando.
- *[4] Project.* Esta área es muy importante en el desarrollo del juego y se debe organizar muy bien ya que es aquí donde se encuentran los recursos del proyecto (modelos importados, texturas importadas, Scripts, Prefabs, entre otros)
- *[5] Inspector.* En esta área se pueden ver y las propiedades de los objetos y scripts, además sirve para configurar herramientas como la de terrenos.

Figura 2. Ventanas de la GUI de Unity 3D



2.6 Lenguaje C# (C Sharp)

C# es un lenguaje de programación orientado a objetos, incorporado al conjunto de lenguajes de la plataforma Microsoft.NET. La notación C# viene de sus similitudes con C y

C++. El objetivo de C# es conservar las cosas más útiles de dichos lenguajes, agregando unas mejoras para hacer más fácil la programación de un código. Cuando se programa en lenguajes como C y C++, puede tomar demasiado tiempo el acomodar el código a la máquina y esto hace que sea tedioso hacer un código, el lenguaje C# es un lenguaje de alto nivel que facilita en demasía la creación de códigos.

En la Tabla 1se muestra la sintaxis de las declaraciones más usuales en C#.

Tabla 1. Sintaxis en C#

TIPO DE DECLARACIÓN	SINTAXIS
Comentarios	`//para comentar una sola línea;`
	`/*para comentar varias líneas*/`
Declaración de variables	`[modificador de acceso] [tipo de dato] [identificador]; /*Se puede asignar un valor inicial, por ejemplo: "public int velocidad=10;", en caso de no poner un modificador de acceso la variable será private*/`
Declaración de constantes	`[const] [tipo de dato]=[valor de la constante];`
Condicional *if*	`if (condición) {sentencias si se cumple la condición} else {sentencias si no se cumple la condición}`

Condicional *switch … case.*	`switch (expresión){`
	
	`case [valor1de expresión]:`
	`  sentencias`
	`break;`
	
	`case [valor2 de expresión]:`
	`  sentencias`
	`break;`
	`.`
	`.`
	`.`
	`default:`
	`  sentencias cuando no se cumple ningún caso`
	`break; }`

Bucle *while.*	`While (condición) {sentencias}`
Bucle *for.*	`for ([variable inicializada];[condición];[modificación de variable]){ sentencias a ejecutar}` `/*Las sentencias se ejecutaran el número de veces que se indique. Ejemplo: for(i=1;i<10;i++){//sentencias a ejecutar}.*/`
Bucle *do … whie.*	`do{sentencias} while (condición);`

2.6.1 Programación orientada a objetos (POO)

La programación orientada a objetos es un tipo de programación que tiene gran aceptación en la actualidad, debido a que brinda gran efectividad en la resolución de problemas de diseño e implementación de software. Para aproximarse a la comprensión de este lenguaje es indispensable entender dos conceptos fundamentales: clase y objeto; haciendo una analogía se podrían presentar respectivamente como el molde y la galleta.

2.6.1.1 Clase

Una clase es un conjunto de propiedades (o atributos) y métodos que permiten definir objetos. Si un objeto X tiene todas las propiedades y métodos de la clase A, entonces se dice que X es una instancia de A, es decir que X es un objeto que pertenece a la clase A. Sintaxis:

```
[Modificador de acceso] class [identificador] {atributos y métodos de la clase}.
```

2.6.1.2 Objeto

Es un ente de la programación orientada a objetos compuesto esencialmente por tres partes: estado, método (o comportamiento) e identidad. El método puede definirse como un algoritmo que ejecuta un objeto X, en reacción a unas entradas. El estado actual de X será una asignación que depende de la configuración de sus entradas. La identidad de X es un distintivo similar al nombre de las personas, que permite identificarlo de los demás objetos. Sintaxis para crear un objeto de una clase:

```
[Tipo de dato u objeto] [Identificador]= new [nombre del constructor];
```

Ejemplo: `carro Mercede_Benz = new carro();`

Para utilizar los métodos y atributos del objeto la sintaxis es de la siguiente manera:

```
[Identificador del objeto].[atributo a utilizar]=[asignación de un valor];
[Identificador del objeto].[método a utilizar];
```

2.6.1.3 Métodos

Se pueden definir como las funciones de un objeto. Sintaxis:

```
[Tipo de dato que devolverá el método] [Identificador]([Parámetros opcionales]) {Cuerpo de método}.
```

2.6.1.4 Herencia

Dadas clases A y B distintas, se dice que A hereda de B, si toda propiedad de A también es propiedad de B. La herencia permite la reutilización de código ya existente. Su sintaxis:

```
Class [Identificador de la clase]: [identificador de la clase de la que se va a heredar] {Cuerpo de la clase}.
```

Ejemplo: `Class clase_A: clase_B {cuerpo de la clase}`

2.6.1.5 Polimorfismo

Este término se puede desglosar así: poli – muchos y morfismo – formas, es decir muchas formas, o en otras palabras el uso múltiple de métodos. "El polimorfismo implica la posibilidad de tener varias clases que se pueden usar de forma intercambiable, incluso si cada clase implementa las mismas propiedades o métodos de formas distintas. El polimorfismo es esencial para la programación orientada a objetos, ya que permite usar elementos con los mismos nombres, sin importar qué tipo de objeto esté en uso en ese momento."[6]

2.6.1.6 Encapsulación

Se refiere al tipo de protección o visibilidad que tienen los métodos y/o atributos de un objeto. Para indicar esto se utilizan los modificadores de acceso:

Tabla 2. Modificadores de acceso

MODIFICADOR DE ACCESO	DESCRIPCIÓN
Public	Se puede obtener acceso a su contenido desde cualquier otro objeto de cualquier clase.
Private	Solamente puede obtener acceso a su contenido de la misma clase; es e modificador de acceso por defecto.
Protected	Solamente puede obtener acceso a su contenido desde la misma clase o clases derivadas de esta.
Internal	Puede obtener acceso a su contenido clases del mismo ensamblado.
Protectec internal	Se puede obtener acceso a su contenido desde la misma clase, clases del mismo ensamblado o clases derivadas que sean de otro ensamblado.

[6] Fuente: http://programacionorientadaaobjetos.wordpress.com/tag/polimorfismo/

2.6.1.7 Variables y métodos estáticos (Static)

Una variable static puede ser usada desde otras clases y su valor es el mismo para todas las clases, además puede ser modificable desde cada una de ellas, ejemplo: *"static int num;"*. Un método static, también llamado método de clase, tiene una funcionalidad identica para todos los objetos; por ejemplo: "*public static void Main(string [] args)".* Para acceder a una variable o método static se hace de la siguiente manera: *[nombre de a case].[nombre del método o variable]*.

2.7 Blender

Blender es una aplicación de modelado 3D de código abierto bajo la licencia GNU GPL (Licencia Pública General), incluye funciones de animación, iluminación, desarrollo de videos, entre otras. Se caracteriza por ser una multiplataforma que puede trabajar en Windows, Linux, Mac OS X y FreeBSD, además soporta varios formatos de archivo en 2D y 3D lo que facilita trabajar con archivos importados de otras aplicaciones.

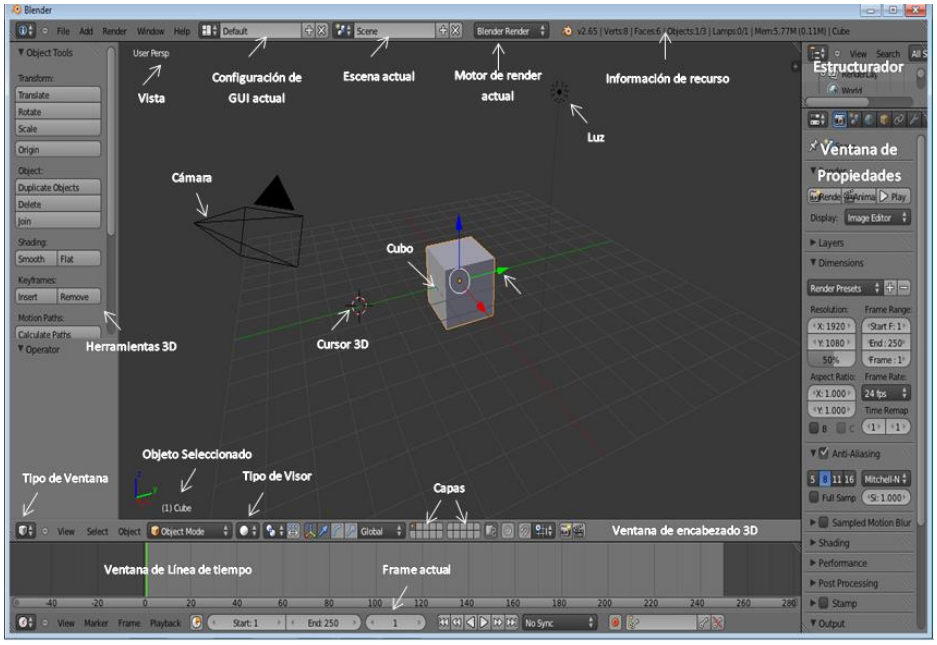
Hoy en día Blender es muy utilizado para el desarrollo de video juegos y simulaciones 3D, por esta razón se ha ido convirtiendo en una herramienta útil en la enseñanza de las disciplinas y en especial en tecnología.

Blender utiliza el lenguaje de Script Python para el desarrollo de aplicaciones, ya sean video juegos, animaciones controladas, etc., e incluye un gran número de funciones para trabajar con ellas.

2.7.1 GUI de Blender

Blender tiene una GUI por defecto que se muestra a continuación, sin embargo esta puede ser modificada por el usuario a su conveniencia, es un poco diferente a la de otros programas.

Figura 3. Ventanas de la GUI de Blender



Fuente: Esta imagen se creó basada en una imagen de http://wiki.blender.org

2.7.2 Modelado 3D

Blender cuenta muchas herramientas para realizar procesos de modelado:

- *Transformaciones.* Estas pueden ser mover, rotar, escalar, espejar, entre otras; también se pueden editar por caras, aristas y vértices.

- *Jerarquía.* Una propiedad importante de los objetos en Blender es que estos se pueden organizar en jerarquía, así que si se traslada, escala o rota un objeto padre, los objetos hijos también se modificaran en relación a su padre.

- *Modificadores.* Los modificadores son unas herramientas de edición muy útiles en el modelado 3D, entre ellos encontramos repetir, construir, operaciones booleanas (unión, intersección, diferencia…), etc

- *Materiales y texturas.* Los materiales modifican de forma uniforme color, especularidad, reflexión, transparencia y otras características de los objetos para

aproximarse a proyecciones del mundo real. Las texturas son imágenes que se utilizan para modificar patrones y agregar detalles a las superficies de los objetos.

2.7.3 Armaduras

Las armaduras se utilizan para el control de movimiento de los objetos, ya sea mediante Scripts o animación manual. El proceso de poner el esqueleto a un objeto es conocido como rigging y el proceso de vincular los huesos con parte del mallado se conoce skinning. Los huesos como cualquier objeto en Blender tienen propiedades de escala, posición y rotación.

Se debe asegurar en el proceso de skining que toda la malla de un objeto este asignada con los huesos para evitar deformaciones en los movimientos. Si queremos que el movimiento de un hueso influya en otro esto se hace mediante jerarquía.

2.8 Pololu Servo - Controlador Maestro

Los Servo - Controlador Maestro se componen de cuatro tarjetas de control de servos: Micro maestro 6, Mini Maestro 12, Mini Maestro 18 y Mini Maestro; las cuales pueden ser controlados de tres maneras: mediante scripts internos para un funcionamiento autónomo, mediante conexión USB directa con el PC y mediante TTL serie para trabajar con otros sistemas.

El software de control, Maestro Control Center, es libre y está disponible tanto para Windows como para Linux (no es compatible con Mac Os), la resolución de ancho de pulso de salida es de 0.25µs que se traducen en 0.025° para un servo típico y sus salidas digitales comprenden 0V y 5V; uno de sus canales puede ser usado como salida PWM con una frecuencia de 2,93 kHz a 12 MHz, hasta 10 bits de resolución. Se puede programar mediante un lenguaje de scripts y pueden ejecutar acciones aun después de desconectarse del puerto USB; el maestro se puede programar desde el PC de dos formas: la primera es mediante el puerto virtual COM, que puede ser manejado desde cualquier software que permita comunicación serie, y la segunda es mediante el kit de desarrollo de software Pololu USB, que puede ser manejado con comandos avanzados USB o mediante los lenguajes C#, Visual Basic.NET y Visual C++.

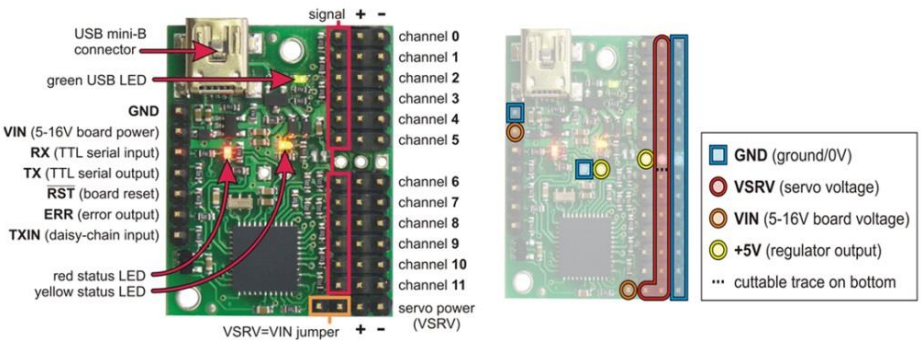
2.8.1 Mini Maestro 12 Servo – controlador USB

Tabla 3. Características técnicas de Mini Maestro 12

	**MINI MAESTRO 12**
Canales	12
Entradas analógicas	12
Entradas digitales	0
Ancho	1.10’’(2.79cm)
Largo	1.42’’(3.61cm)
Peso (sin pines)	4.2g
Frecuencia de pulso	1-333Hz
Rango de pulso	64-4080 µs
Tamaño de Script	8KB

Fuente: http://www.pololu.com/docs/0J40/all

Figura 4. Componentes y pines de Mini Maestro 12



Fuente: http://www.pololu.com/docs/0J40/all

La tarjeta puede ser alimentada desde el PC mediante un cable USB A a mini-B o mediante una batería externa entre 5V y 16V, adicionalmente cuenta con una salida de 5V regulados puesta a disposición del usuario; es posible tener alimentación USB y externa al mismo tiempo, en cuyo caso la tarjeta se alimenta de la fuente externa, pero se debe tener en cuenta que si la alimentación eterna es menor a 5V la calidad en funcionamiento se verá afectada, adicionalmente hay que considerar que la corriente de consumo de un servo es de 1A. Para alimentar los servos solo con la fuente del PC se debe conectar VSRV con VIN, para esto se puede utilizar el Jumper azul incluido en la tarjeta. Es recomendable que la

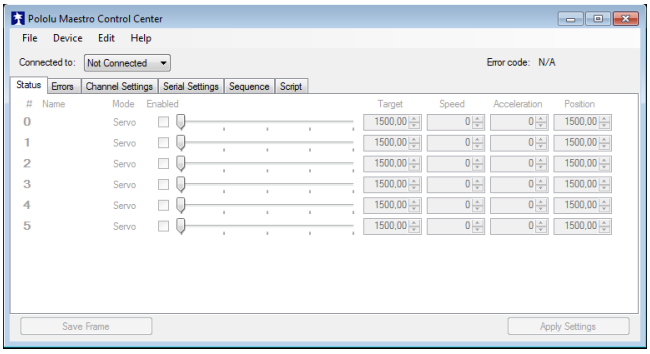
línea TX trabaje como minino con 4V (como bit alto) para un correcto funcionamiento, a su vez la línea TX transmite bits de 0-5V.

2.8.2 Software Maestro Control Center

Este software solo se puede manipular con el Maestro conectado al PC. Tiene las siguientes pestañas de manipulación:

- *Status.* Se pueden hacer ejecuciones en tiempo real para cada uno de los canales de Mini Maestro 12, teniendo como parámetros de control el destino o Target, velocidad, aceleración; El canal 8 de la tarjeta puede usarse como salida PWM.
- *Errors.* Se muestran los problemas que surgen durante el funcionamiento de Maestro.
- *Sequence.* Permite configurar y guardar algunas secuencias para ser ejecutadas por Maestro, con la opción de crear bucles o repeticiones. Estas secuencias se pueden pasar a script con el botón "Copy Sequence to Scrip" o "Copy all Sequences to Script" para añadir todas las subrutinas al script.
- *Script.* La casilla "Run script on startup" activada sirve para que el script se ejecute sin necesidad de conexión al PC. Maestro tiene su propio lenguaje de scripts.
- *Channel Settings.* Es una pestaña para configurar algunos parámetros de los canales utilizados, como por ejemplo el nombre del canal, el modo de trabajo, el pulso de trabajo para cada servo, etc.
- *Serial Settings.* Sirve para ajustar el tipo de comunicación de Maestro.

Figura 5. Software Maestro Control Center



2.8.3 Comunicación por puerto USB

El Maestro control center cuenta con dos puertos virtuales de tipo serial, que permiten la comunicación del PC mediante USB, estos son: *Command Port* y *TTL Port*; una vez instalado el Software, estos se podrán ver en el administrador de dispositivos (para Windows). El tipo de comunicación puede ser: *USB Dual Port*, para enviar *comandos* al Maestro y recibir respuestas del mismo a una velocidad en baudios establecida por el usuario, utilizando el puerto serial virtual Command Port; *USB encadenado* que permite enviar y recibir datos del maestro pero no son interpretados como bytes de comandos, no utiliza el TTL Port y finalmente *UART*, en el cual los datos recibidos llegan directamente al Command Port, pero ignora los que son enviados por este puerto, para este tipo de comunicación el TTL Port no se usa.

2.8.4 Protocolo de comandos

La comunicación con Maestro se hace mediante protocolos de comandos, que pueden ser Compact, Pololu y Mini SSC; los comandos se envían como secuencias de bytes y a su vez cada uno lleva siete bits de información (a excepción de los comandos Mini SSC).

2.8.4.1 Protocolo Compact

Es muy sencillo de utilizar y sirve para casos en que solo haya una tarjeta Maestro conectada. Sus comandos son los siguientes:

- *Set target.* indica el destino del servo y se configura de la siguiente manera: {0x84, numcanal, destinoLB, destinoHB}; donde LB son los 7 bits menos significativos del dato de destino (b0…b6) y HB son los 7 bits más significativos del dato de destino (b7…b13). El dato de destino debe ser un entero positivo.
- *Set Multiple Target.* Se utiliza cuando se deben controlar varios servos configurados en cadena. Configuración: {0x9F, numdestinos, 1numcanal, 1destinoLB, 1destinoHB, 2destinoLB, 2destinoHB,…}.
- *Set Speed.* determina la velocidad a la cual va el servo, cuando se quiere poner un servo en determinado destino y que vaya a una velocidad diferente a la que tiene por defecto el Maestro, primero se debe enviar el comando Set Speed y después el

comando Set Target. Configuración: {0x87, numcanal, velocidadLB, velocidadHB}.

- *Set Acceleration.* Este comando es para establecer un límite en la aceleración del servo para que su velocidad ascienda hasta llegar a la velocidad máxima y luego descienda para llegar a su destino. Configuración: {0x89, numcanal, aceleraciónLB, aceleraciónHB}.
- *Set PWM.* Sirve para ajustar la salida PWM con un pulso y un periodo (unidades de 1/4 µs). Configuración: {0x8A, pulsoLB, pulsoHB, periodoLB, periodoHB}.
- *Get Position.* Sirve para obtener información de la posición del servo, la respuesta se obtiene en dos bytes. Configuración: {0x90, numcanal}.
- *Get Moving State:* si el servo se está moviendo se obtiene como respuesta 0x01, en caso contrario se obtiene 0x00. Configuración: {0x93}.
- *Get Errors.* Permite chequear errores que sean detectados por Maestro y la respuesta se recibe en dos bytes. Configuración: {0xA1}.
- *Go Home.* Sirve para enviar a todos los servos a su posición inicial. Configuración: {0x21}.

CAPÍTULO 3

3. ASPECTOS METODOLÓGICOS

Para el diseño y desarrollo de este proyecto se escoge la metodología RUP (Proceso Unificado Racional), porque es una metodología que se enfoca en los requerimientos y el diseño de cada proyecto en particular.

3.1 Metodología RUP

Esta metodología es presentada como una metodología de desarrollo de software que puede ser adaptable a las necesidades de cada proyecto. Esta metodología define para el proyecto ¿Quién hace qué?, ¿Cómo lo va a hacer? y ¿Cuándo lo va a hacer?

El objetivo principal de esta metodología es asegurar el desarrollo de un proyecto de calidad, que satisfaga las necesidades del usuario y que tenga una distribución adecuada del tiempo y las actividades a realizar, para así cumplir dicho objetivo.

Esta metodología plantea cuatro fases para el desarrollo de los proyectos, las cuales se desarrollan en iteraciones para hacer un seguimiento de la calidad del software.

3.1.1 Fase de Inicio

Esta fase también se conoce como la fase de inspección y concepción, aquí se hace un plan de fases en el cual se identifican los casos de uso para el software; se hace énfasis en las actividades necesarias para la ejecución del proyecto y en los requerimientos y los riesgos de dichas actividades.

3.1.2 Fase de Elaboración

Esta fase se centra en la planeación de actividades a realizar, tratando en lo posible de mitigar los riesgos identificados en la fase de inicio; además de esto se hace un organigrama basado en el análisis, diseño e implementación del software orientada a la construcción del mismo y teniendo en cuenta los recursos existentes para esto.

3.1.3 Fase de Construcción

Se trata de algo operativo, es decir que aquí se ejecutan las actividades provenientes de la planeación para el desarrollo del software, además de esto se realizan documentos como el manual de usuario y el manual para el desarrollador (para mejoras o adecuaciones posteriores del software).

3.1.4 Fase de Transición

En esta fase se hace un análisis mediante la implementación del producto para establecer si éste cumple o no con los requerimientos del usuario en cuanto al funcionamiento, si no se cumplen a cabalidad se procede a corregir los errores para hacer la entrega final.

3.2 Organización del proyecto

Implementando la metodología RUP, las actividades se dividen en cuatro fases:

- *Fase de inicio:* en esta fase se hace una búsqueda de información sobre software, la identificación de casos de uso, el modelamiento del software y la identificación de riesgos.
- *Fase de elaboración:* aquí se establecen los requerimientos de simulación de los robots, se hace el modelado 3D, se diseña la GUI (interfaz gráfica de usuario) y finalmente se hace revisión y ajuste de los modelos y la GUI.
- *Fase de construcción:* aquí se realiza el software con base en el diseño establecido, se hace una revisión y ajuste de este, se realizan los manuales de usuario y desarrollador, por último se documenta el proyecto.
- *Fase de transición:* esta fase es para la realización de pruebas de funcionamiento, ajustes finales y entrega del software.

CAPÍTULO 4

4. DESARROLLO DEL PROYECTO

4.1 Desarrollo Técnico/Educativo

Teniendo en cuenta que este proyecto consiste en un software educativo, que sirve como herramienta para los procesos de enseñanza y aprendizaje llevados a cabo en los clubes de robótica de la UPN, el desarrollo técnico se hace teniendo en cuenta el componente educativo que debe hacer parte del mismo.

4.1.1 Requerimientos del software

4.1.1.1 Requisitos de funcionamiento y presentación

Los integrantes de los clubes de Robótica son estudiantes de noveno, décimo y undécimo de algunos colegios de Bogotá.

En el diseño del proyecto se debe tener en cuenta qué conocimientos tienen los estudiantes acerca de los simuladores y qué requisitos creen que son importantes en un software de simulación enfocado a la robótica. "Un sistema software ve la luz para dar servicio a sus usuarios. Por tanto, para construir un sistema con éxito debemos conocer lo que los futuros usuarios necesitan y desean"[7].

Se realizó un diagnóstico realizando algunas preguntas abiertas a los estudiantes del grado décimo del colegio Claretiano. La elección de este grado es debido a que sus conocimientos en robótica no son muy avanzados pero tampoco son escasos o nulos; además se tiene en cuenta que los dos robots a simular corresponden a los dos últimos HAD (Hardware de Apoyo didáctico) presentados en la en tesis "Diseño de Herramientas y

[7] Jacobso Ivar, Booch Grady, Rumbaugh James. *"El Proceso Unificado de Desarrollo de Software".* Editorial Pearson, S.A., Addison Wesley. Madrid, 2000. Pág. 5.

Procesos Para el Apoyo al Trabajo en Clubes de Robótica (Fase Diseño de Hardware)", estos están encadenados a una secuencia progresiva de herramientas enfocadas al aprendizaje de los conceptos fundamentales de la Robótica.

Figura 6. Clubes de Robótica – Colegio Claretiano 10°

Tabla 4. Diagnóstico de requisitos del software según las necesidades de los estudiantes y conocimientos del tema

Pregunta	Intención de la pregunta	Resultados (respuestas y número de mención)	Análisis de resultados
Defina con sus propias palabras ¿Qué es un simulador?	Saber si los estudiantes tienen un concepto claro de qué es un simulador para poder validar las demás respuestas.	Programa para hacer o probar montajes de circuitos (6) Programa para hacer pruebas y no dañar los componentes físicos o ver el funcionamiento de algo (3) Respuestas confusas (2) Programa para indicar aun robot mediante programación que ejerza acciones o que el computador las haga virtualmente. (1) Programa referente a animaciones y videos .(1)	Los estudiantes tienen una idea cercana de lo que es un simulador pero casi siempre lo relacionan con circuitos, dejan de lado en su mayoría otras aplicaciones. También enmarcan los simuladores como programas, no contemplan la posibilidad de simulación física.

¿Cuál es la principal ventaja de usar un simulador?	Saber si los estudiantes comprenden la importancia de utilizar simuladores.	Que se puede prevenir daños físicos en circuitos.(6) Que no se arriesgan vidas humanas.(4) Que se puede saber si un circuito está funcionando bien.(3) Que se puede comprender cómo se realiza el montaje de un circuito.(2) Que se puede saber cómo es y/o qué utilidad tiene algo. (2) Que se pueden obtener datos sobre componentes.(1) Que se puede corregir errores.(1) Que sirve para aprender a manejar una situación. (1)	Enfatizan en que sirve para prevenir daños físicos y riesgos para las vidas humanas. Además hablan de que pueden ver si algo funciona bien, que sirve para aprender sobre lo que se está simulando y sobre la corrección de errores.
¿Ha utilizado simuladores en sus estudios de Robótica? ¿Cuáles?	Saber qué tipos de simuladores ha usado en robótica para lograr hacer algo un poco diferente.	Croclip.(7) No. (4) Protoboard.(2) Proteus.(1) Multisim.(1) Respuestas que no corresponden con lo que se preguntó. (1) Sin respuesta.(1)	Se puede ver que algunos han utilizado Croclip y otros programas de simulación y algunos no recuerdan haber utilizado algún simulador en sus estudios de robótica. Al parecer no han usado simuladores 3D en esta área.
¿Qué característica cree que debe tener un simulador de Robótica?	Esta pregunta es muy importante porque define los requisititos que los estudiantes creen que son importantes en el software.	Contar con todos los dispositivos para simular circuitos. (3) Sin respuesta.(2) No sabe.(2) Sencillo de programar y con efectividad.(1) Con buenos gráficos y/o símbolos. (1) Fácil manejo y bajo costo.(1) Que se pueda ver el funcionamiento de los elementos.(1) Que se pueda ver el diseño.(1) Buen funcionamiento.(1) Que tenga las funciones necesarias para hacer la simulación.(1) Respuestas confusas.(1)	Creen que debe contar con las herramientas necesarias para llevar a cabo la simulación, sea sencilla de utilizar, que tenga buenos gráficos y símbolos, que tenga bajo costo, que permita ver de manera clara el funcionamiento del objeto de simulación.
¿Defina en sus propias palabras qué es programar?	Saber si los estudiantes de décimo comprender la definición de programar, para encaminar un poco el proyecto hacia su segunda fase.	Respuestas confusas.(3) Dar órdenes a un objeto.(2) Dar una caracterización y/o orden a un artefacto para que desarrolle una determinada acción. (2) Dar órdenes con un programa. (2) Definir pasos para Seguir instrucciones.(1) Dar instrucciones y ejecutarlas. (1) No sabe. (1) Respuesta confusa.(1)	Hay una idea cercana de lo que es programar, hablan de instrucciones u órdenes y de su ejecución.

De la anterior tabla podemos definir algunos requisitos importantes para el funcionamiento del software:

- El software debe contar con las herramientas necesarias para llevar a cabo la simulación.
- Debe ser sencillo de utilizar.
- Sus gráficos deben ser llamativos y realistas.
- Que permita ver de manera clara el funcionamiento del objeto de simulación.

Los demás requisitos de funcionamiento se definen a partir de los objetivos del proyecto y de las funciones de un software de educativo definidas en el capítulo 2:

- El software debe presentar conceptos claros y de manera ordenada.
- Debe orientar a los estudiantes en su aprendizaje.
- Debe contener elementos que capten y mantengan el interés de los estudiantes.
- Debe permitir que los estudiantes reconozcan sus propios errores.
- Debe permitir una interacción con los parámetros y variables de simulación.
- Debe permitir una comunicación entre el usuario y este a través de acciones.
- El software debe garantizar la simulación grafica de un robot móvil y un brazo robótico de tres grados de libertad (mínimo).
- Debe tener un conjunto mínimo de instrucciones que permitan simular los robots.

4.1.1.2 Requisitos de recursos técnicos[8]

Para desarrollar este proyecto es necesario utilizar un software de modelado 3D lo suficientemente flexible para permitir simular todas las articulaciones y funcionalidades del brazo robótico. Además se necesita un software que permita diseñar GUIs, y controlar las propiedades de los modelos. En definitiva, el software debe cumplir los siguientes requisitos:

- Facilitar el trabajo con modelos 3D.
- Optimización para crear interfaces de usuario.
- Permitir control del movimiento de los modelos 3D.
- Permitir el desarrollo de simulaciones de objetos con propiedades físicas.

[8] Objetivo específico No. 2

- Admitir comunicación serial (para su continuación).

Por esta razón se realizó una tabla comparativa de algunos programas, con el fin de tomar la decisión de cuál es la mejor opción para ser el software de desarrollo central de este proyecto.

Tabla 5. Comparación de software para el desarrollo del proyecto

CARACTERÍSTICA	UNITY 3D	BLENDER	VISUAL STUDIO
Licencia.	Tiene tres tipos de licencia: Pro, Free con algunas limitaciones y Versión de prueba por 30 días con todas las cualidades Pro.	Licencia Free de código abierto.	Cuenta con Microsoft Visual Studio Express que tiene licencia free y otras versiones con licencia pro.
Modelado 3D.	Permite creación de piezas básicas como planos, cubos, esferas, etc. Además cuenta con recursos ya existentes para crear entornos de juego rápidos, como por ejemplo árboles, cielos, herramientas de creación y edición de terrenos, etc. incluye recursos de texturas y ambientación como iluminación, sonido, entre otras. Implementa la librería OpenGL (*Open Graphics Library*) para la representación de gráficos 2D y 3D.	Se puede desarrollar cualquier tipo de modelo 3D a partir de figuras primitivas como planos, esferas, cilindros, entre otros. Cuenta con herramientas para renderizado y ambientación como materiales, texturas, iluminación, entre otras. Para la representación de gráficos 2D y 3D utiliza OpenGL.	Visual Studio tiene un editor de modelos 3D, que permite crear modelos desde cero y editar modelos importados. Permite trabajar con texturas, materiales y sombreados. Para la representación de gráficos utiliza DirectX.
Flexibilidad para importar formatos de modelos 3D.	Se pueden importar formatos de modelo 3D como .fbx, .3ds, .obj, .blend, entre otros.	Permite trabajar con modelos 3D .dae, .3ds, .fbx, .obj y otros, incluyendo la extensión local .blend.	Es compatible con modelo de extensión .dae, .fbx, .obj.
Herramientas de animación y efectos físicos.	La animación de modelos 3D se hace mediante el sistema Mecanim, el cual viene integrado en el motor de Unity; aquí se hace todo el proceso de rigging y skinning. Unity tiene un componente de físicas bastante amplio, que incluye manejo de colisiones, variables como gravedad, fuerzas esternas, y otras.	Cuenta con herramientas para realizar los procesos de rigging y skinning. Permite el integrar cualidades físicas de estática y dinámica en la realización de simulaciones.	No se encontró información al respecto.

Lenguajes de Programación.	Cuenta con tres lenguajes de programación Java Script, C# Script y Boo Script (Derivado de Python). Tiene integrado el entorno de desarrollo MonoDevelop (software libre), exclusivamente para programación, aunque se puede programar con otros entornos (por ejemplo Visual Studio).	Utiliza el lenguaje de programación Python y el entorno de programación viene integrado con Blender .	Se pueden desarrollar aplicaciones con lenguajes de la plataforma Microsoft .NET: Visual C++, Visual C#, Visual J#, Visual Basic .NET, entre otros.
Comunicación serial.	Si se puede realizar.	Si se puede realizar.	Si se puede realizar.
Documentación.	En su página oficial http://udn.unity3d.com/, se encuentra el manual de usuario, manual de Scripting, además de ejemplos, foros, chat, entre otros. Adicionalmente en la web se encuentran algunos foros y tutoriales.	Se puede encontrar documentación en la página oficial http://www.blender.org/, junto el foro oficial. También hay tutoriales y foros.	MSDN[9] library es la biblioteca de desarrollo de Microsoft, y se puede encontrar toda su documentación en la página http://msdn.microsoft.com/library. También se pueden encontrar libros e información general en la web.

Se escogió Unity 3D como el software central del proyecto debido a que cumple con los requisitos primordiales para el desarrollo del software y por familiaridad con el lenguaje de scripting C#; también se tuvo en cuenta el hecho de ser un software con versión libre con buena documentación, la calidad gráfica y el rendimiento. En cuanto al modelado 3D le faltan algunas herramientas, por esta razón el proceso de modelado se realiza en Blender, debido a que es un software libre y que se enfoca principalmente en el modelado y animación 3D.

4.1.2 Casos de uso

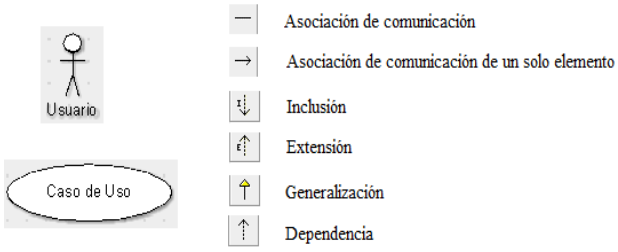
La metodología RUP utiliza el Lenguaje de Moldeamiento Unificado (UML) para la identificación de casos de uso. Los casos de uso posibilitan la delimitación de los requisitos del software, lo cual conlleva a un mejor desarrollo de este; por lo tanto se puede afirmar que los casos de uso son los requisitos de funcionamiento del sistema, teniendo en cuenta

[9] Microsoft Developer Network (Red de Desarrolladores de Microsoft)

que cada uno proporciona un servicio específico al usuario y que el usuario, a su vez, puede ser una persona u otro sistema.

Los diagramas de casos de uso se componen de tres partes fundamentales: los actores, los casos de uso y las relaciones de casos de uso. Los actores son los usuarios, los casos de uso las funciones y las relaciones es el tipo de vínculo entre los casos de uso, estas pueden ser extensión, inclusión, generalización, asociación de comunicación, dependencia, entre otras.

Figura 7. Componentes de un diagrama de casos de uso



Fuente: Esta imagen se creó basada en una imagen de http://es.wikipedia.org/wiki/Casos_de_uso

A partir de la anterior información se realizaron dos diagramas de casos de uso: un diagrama general que involucra las tres fases del proyecto y un diagrama particular para la "Fase 1"[10].

El *Diagrama General de Casos de Uso* consta de cuatro casos de uso principales:

- *Simular Robots:* corresponde a la Fase 1 del proyecto, incluye los casos de uso Simular Brazo Robótico y Simular Robot Móvil.
- *Programar Robots:* corresponde a la Fase 2 del proyecto, incluye los casos de uso Programar Brazo Robótico y Programar Robot Móvil.
- *Acceder a Base de Datos:* hace parte de la Fase 3 del proyecto, los casos de uso Consignar Datos de Evaluación y Consultar Resultados de Evaluación dependen de este caso.

[10] La Fase 1 corresponde al objetivo general del proyecto.

- *Acceder a Página Web:* hace parte de la Fase 3 del proyecto, los casos de uso Consultar Información sobre el Software, Consultar Información sobre el Hardware y Acceder a Unidades Didácticas dependen de este caso.

El usuario principal integra al tutor, el estudiante y el administrador. Adicionalmente hay dos usuarios secundarios: UART (Universal Asynchronous Receiver-Transmitter) que es el puente de comunicación entre el software y el Hardware, que representa los robots físicos.

Para los usuarios secundarios existen dos casos de uso Comunicarse con Software y Comunicarse con Hardware.

Figura 8. Diagrama General de casos de Uso

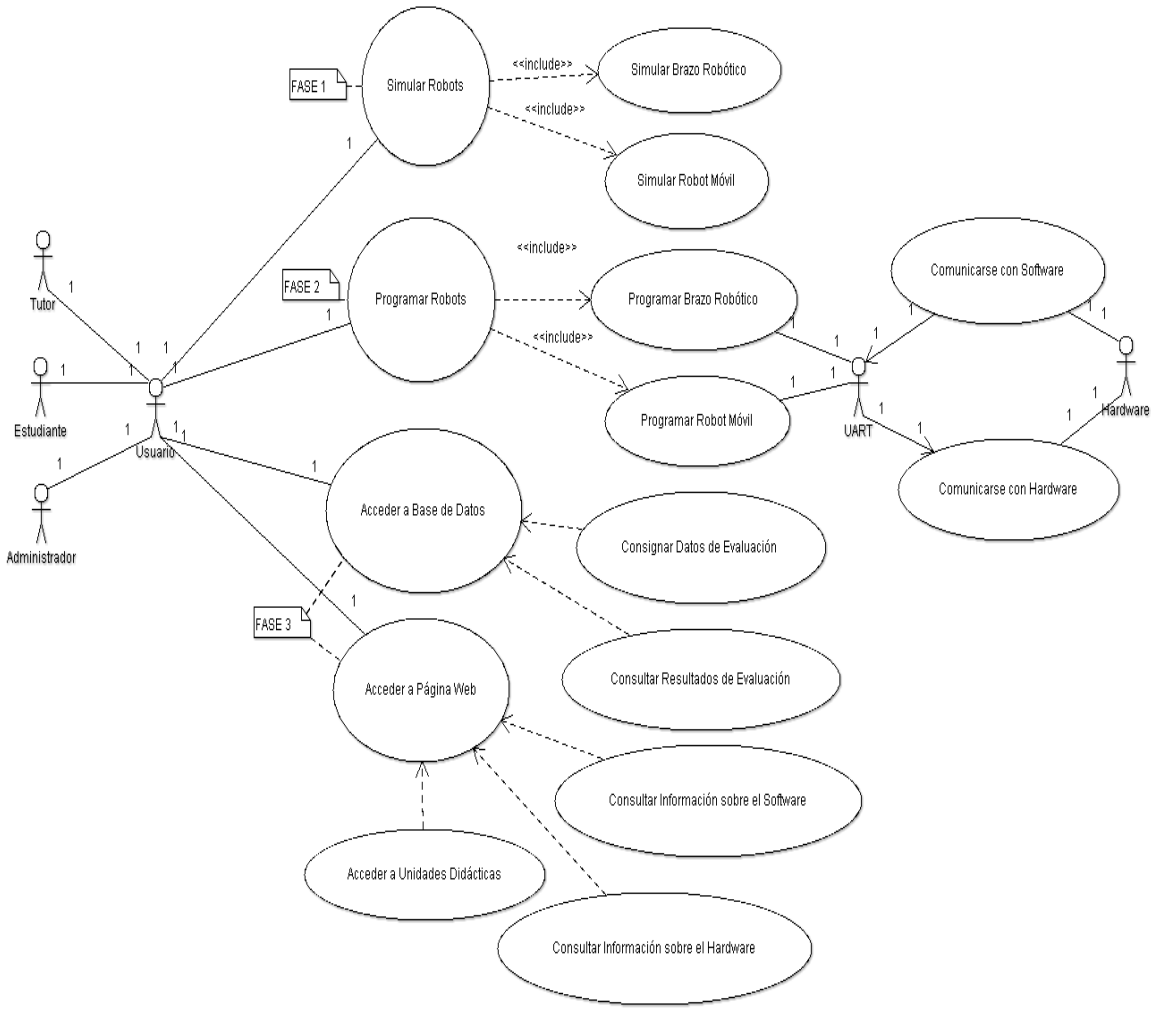


Figura 9. Diagrama de Casos de Uso para la Fase 1

En el *Diagrama de Casos de Uso para la Fase 1* se establece un caso de uso principal llamado Simular Robots, dos casos de uso secundarios denominados Simular Brazo Robótico y Simular Robot Móvil y cinco terciarios que se distribuyen en los secundarios y se explican a continuación:

- *Establecer Instrucción*: Permite al usuario escoger una instrucción para el Robot.

- *Enviar Instrucción:* Permite enviar esa instrucción para ser almacenada en una memoria del software.
- *Imprimir Instrucción:* Después de ser almacenada se imprime la instrucción en una consola y se procede a establecer una nueva instrucción.
- *Ejecutar Secuencia de Instrucciones:* Una vez almacenadas las instrucciones se procede a ejecutarlas en el orden de envió.
- *Resetear Instrucciones y Variables de Simulación:* En caso de equivocación en el envío de instrucciones o de finalización de ejecución de una secuencia se permite al usuario empezar a simulación desde cero.

Debido a que Unity trabaja por escenas se organiza el software de manera que cada robot se simule en una escena diferente, por esta razón los mismos casos de uso terciarios aplican a los dos secundarios. Además estos siguen un orden lógico en su funcionamiento, de tal manera que se logre hacer una simulación satisfactoria, por esto están organizados de paso 1 al paso 5 como se puede ver en el diagrama.

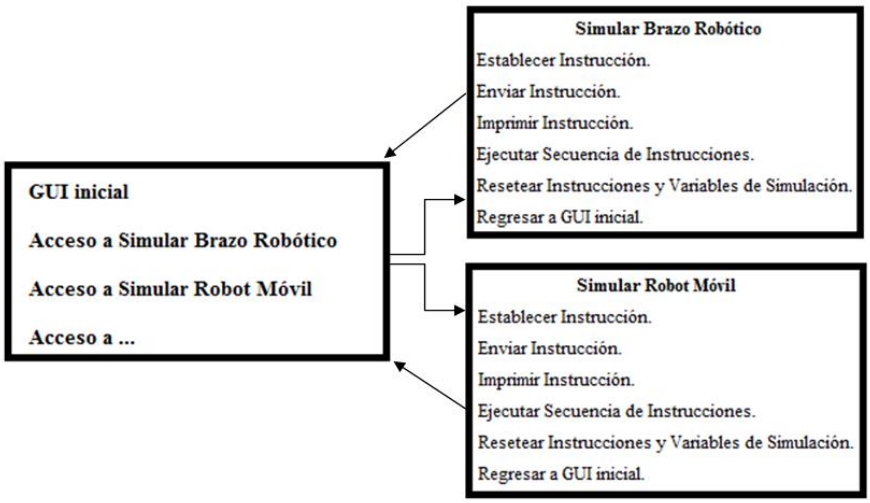
El usuario como se muestra en el diagrama de casos de uso general puede ser el tutor, el administrador o el estudiante.

4.1.3 Modelado del Software

A partir del diagrama UML para la Fase 1 del software, se estructura el software de la siguiente manera: se realizan tres escenas en Unity una corresponde a la interfaz de usuario inicial y las otras dos para simulación de los robots respectivamente, con esto los nuevos desarrolladores podrán crear nuevas escenas para complementar el proyecto; dichas escenas serán accesibles desde la interfaz inicial y así mismo esta será accesible desde las escenas existentes.

Cada escena de simulación debe tener acceso a los casos de uso Establecer Instrucción, Enviar Instrucción, Imprimir Instrucción, Ejecutar Secuencia de Instrucciones, Resetear Instrucciones y Variables de Simulación y adicionalmente tendrán el caso de uso Regresar a GUI inicial.

Figura 10. Modelo del software por escenas



4.1.4 Identificación de Riesgos

Los riegos de un proyecto tienen que ver con las inseguridades que se tienen respecto al desarrollo de este y la especulación sobre algunos factores que podrían amenazar con el éxito del mismo.

Para este proyecto se plantearon los siguientes riesgos:

- *Riesgo1:* Dado que Unity es una tecnología joven, entonces existe la posibilidad de que presente algunas fallas de funcionamiento.
- *Riesgo2:* La programación se puede tornar compleja y se puede extender más del tiempo esperado.
- *Riesgo3:* El tiempo que se puede tardar en aprender el manejo de Blender y/o Unity no se puede establecer a ciencia cierta, en el caso de que sobrepase el estimado puede afectar con un alargue del proyecto.
- *Riesgo4:* Desarrollar un producto que no cumpla con las funciones esperadas por el usuario.

Para mitigar el impacto de estos riesgos en el desarrollo de proyecto, se debe hacer un seguimiento constante que permita llevar a cabo una acción que impida que estos riesgos perjudiquen el proceso.

El riego 1 se puede constatar haciendo pruebas frecuentes en el software y remitiendo preguntas en foros, adelantándose un poco en lo que se necesite para el desarrollo del software. Para mitigar el impacto del riesgo 2, constantemente se debe documentar de forma organizada y estructurada cada avance en la programación, esto con el fin de facilitar el acceso a variables y partes del software. En caso de que el riesgo 3 se esté convirtiendo en un problema, se debe seguir el aprendizaje del software desarrollado enfocándolo a los procesos que den cumplimiento a los objetivos propuestos (enfocarlo hacia la robótica). Finalmente el riesgo 4 es posible remitiéndose a los objetivos del proyecto y a los datos arrojados en la encuesta[11].

4.1.5 Requerimientos para simulación de robots

Los principales requisitos para llevar a cabo la simulación de los robots son:

- Se debe ejecutar gráficamente el mismo tipo de instrucciones que implemente el hardware.
- Los modelo 3D se deben asemejar lo más posible a los robots físicos.
- Debe implementar los casos de uso que se plantearon para las simulaciones.
- El entorno de simulación no debe distraer la atención del usuario, esta se debe enfocar en el comportamiento de los robots.
- Debe permitir que el usuario participe en la configuración de las variables de simulación.

4.1.6 Modelado 3D

4.1.6.1 Modelado 3D de entornos y prototipos de prueba

En Primer lugar se modelaron los entornos para llevar la simulación del brazo robótico, el entorno para la simulación del robot móvil y el entorno para la comunicación del brazo. Estos entornos básicamente se construyeron a partir de planos, algunos como fondo y otros formando cuadriculas.

[11] Tabla 4. Diagnóstico de requisitos del software según las necesidades de los estudiantes y conocimientos del tema.

Figura 11. Modelo 3D: Entorno para Simulación del Brazo Robótico

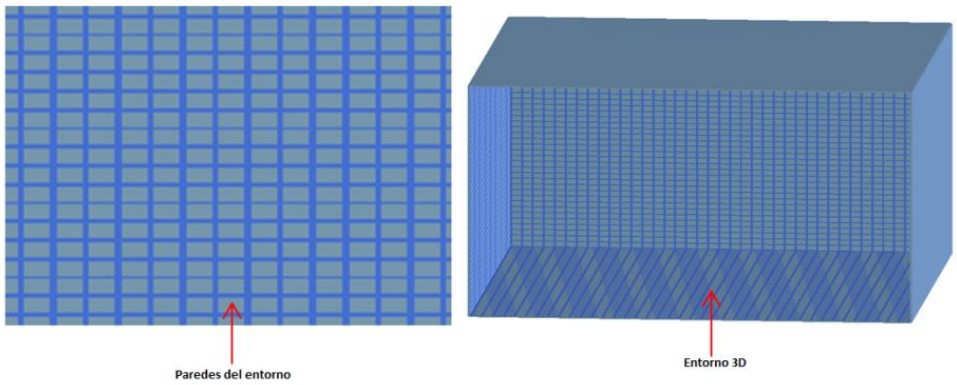


Figura 12. Modelo 3D: Entorno para Comunicación del Brazo Robótico

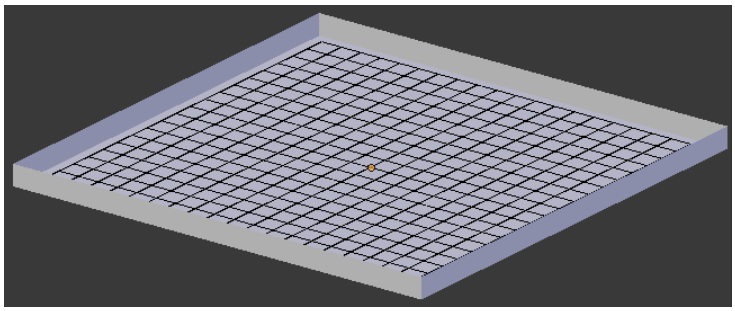
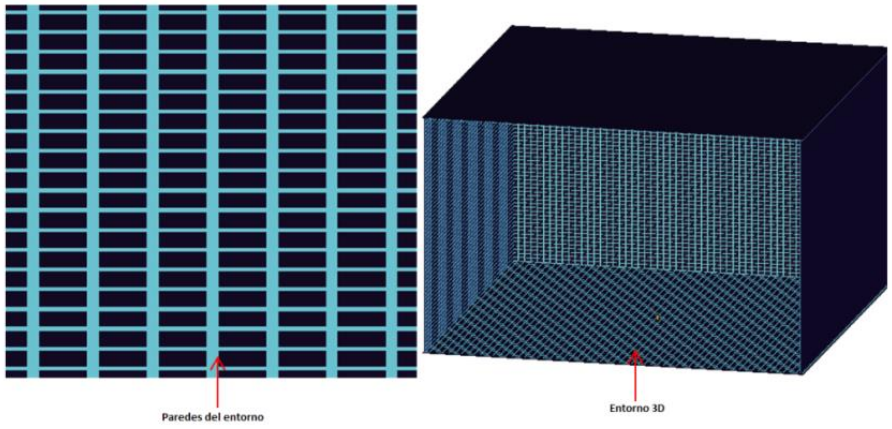


Figura 13. Modelo 3D: Entorno para Comunicación del Brazo Robótico



Posteriormente se realizó un prototipo de brazo robótico y un prototipo de robot móvil para realizar pruebas en Unity, puesto que los robots físicos aún no se habían construido. Esto

sirvió también para ir viendo posibles inconvenientes en la construcción de los modelos reales, teniendo en cuenta que estos son modelos con un grado de complejidad mucho mayor.

Figura 14. Modelo 3D: Prototipo de Brazo Robótico de Prueba

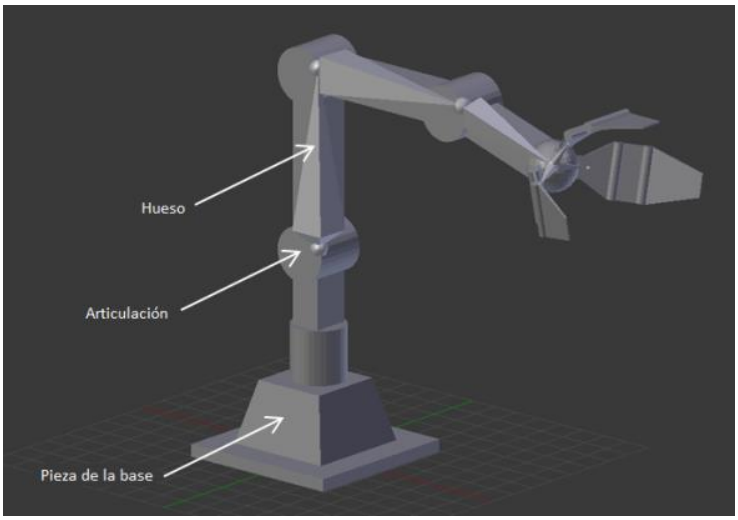


Figura 15. Modelo 3D: Prototipo de Robot Móvil de Prueba

A partir del desarrollo de los modelos de entornos de simulación y los modelos de prueba, se establecieron los siguientes pasos para el modelado 3D:

Paso 1. Realizar piezas básicas como lo son esferas, cilindros y cubos; esto teniendo en cuenta propiedades como Escala, Dimensiones, Rotación y Posición (Centroide).

Paso 2. Construir partes del modelo a partir de la aplicación de operaciones booleanas, entre las piezas básicas, las operaciones más usadas fueron diferencia, intersección y unión.

Paso 3. Diseño y asignación de materiales a cada parte del modelo.

Paso 4. Reducir líneas de las mallas generadas a partir de la aplicación de pruebas booleanas. Esto se hace disolviendo líneas, fusionando caras, colapsando, etc.

*Paso 5.*Unir todas las piezas para formar una sola malla en el modelo

Paso 6. Asignación de la armadura del modelo, teniendo en cuenta las articulaciones que influyen en el movimiento del mismo. Para no tener conflicto se realizan los huesos por separado pero pertenecientes a la misma armadura. La herencia de padres a hijos (si la hay) preferiblemente hacerla dentro de la jerarquía de Unity.

Paso 7. Asignación de la malla que mueve cada hueso.

Paso 8. Realizar pruebas de rotaciones y demás parámetros del modelo, teniendo en cuenta la armadura.

Paso 9. Finalmente verificar parámetros del modelo para poder exportarlo a otros programas, en este caso Unity 3D.

Estos pasos no necesariamente son aplicables en todos los casos, se establecieron con el fin de organizar la secuencia de construcción de los modelos 3D de este proyecto; además pueden variar en su orden según las necesidades de desarrollo de los modelos.

4.1.6.2 Modelado 3D del Brazo Robótico Real

Para la construcción del modelo 3D del brazo se realizó pieza por pieza: caja de la base, chumacera; pieza de unión de la chumacera con articulación 1, servo- motores, piezas de articulaciones 1 y 2, articulación 3 con motor de muñeca incluido, soportes transversales de las articulaciones y finalmente la pinza.

A continuación se muestran figuras con algunas vistas para cada pieza.

Figura 16. Modelo 3D: Base

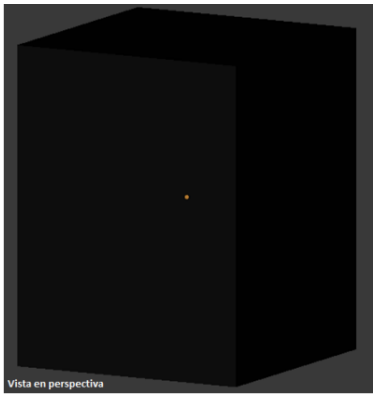


Figura 17. Modelo 3D: Chumacera

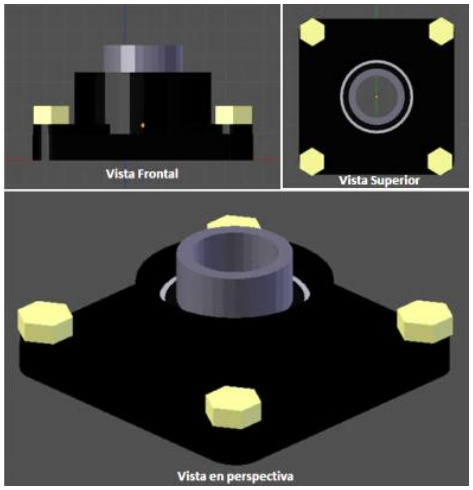


Figura 18. Modelo 3D: Pieza de Unión de la Chumacera con Articulación 1

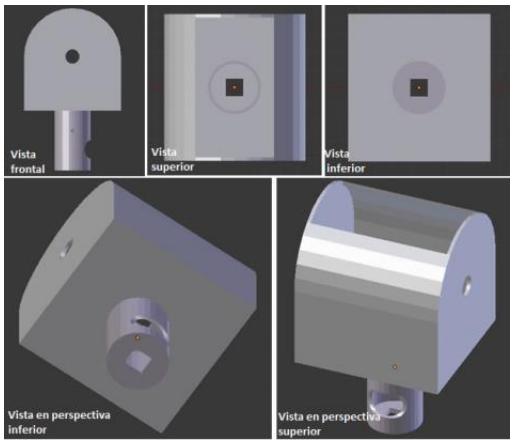


Figura 19. Modelo 3D: Servo Motor

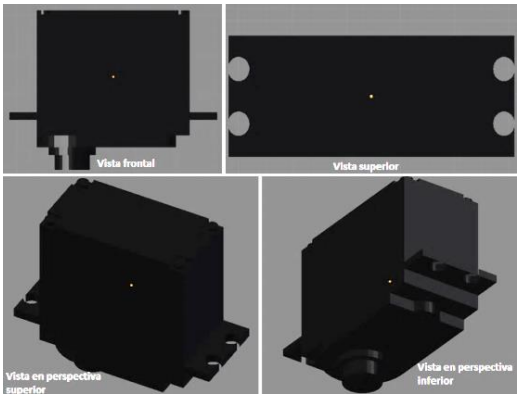


Figura 20. Modelo 3D: Articulaciones 1 (pieza grande) y 2 (pieza pequeña)

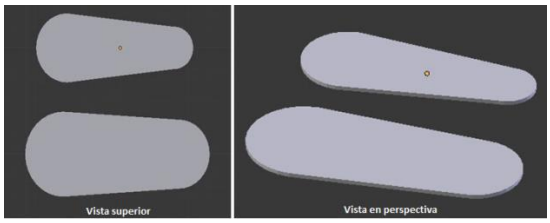


Figura 21. Modelo 3D: Articulación 3 con Motor de Muñeca Incluido

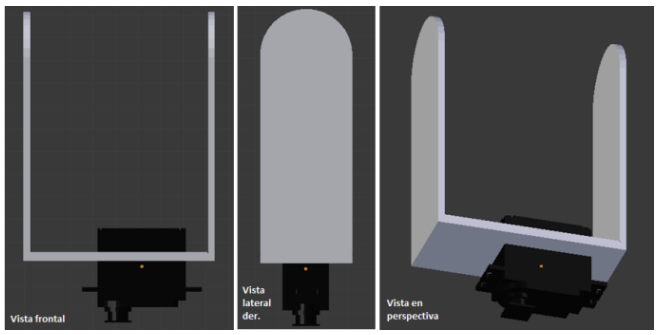


Figura 22. Modelo 3D: Soportes Transversales de las Articulaciones

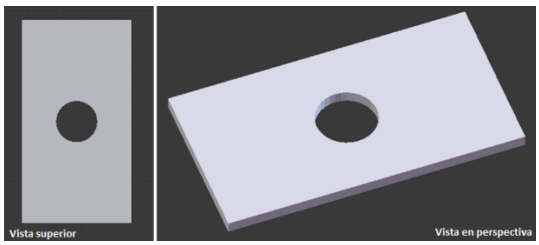
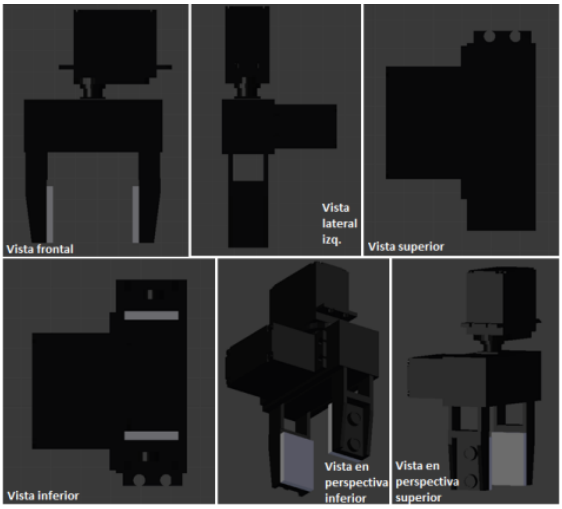


Figura 23. Modelo 3D: Pinza



Una vez terminadas las piezas se procedió a ensamblarlas para unirlas formando una sola malla. La base y la chumacera se unieron formando una sola pieza y las demás partes del brazo forman otra, esto se hizo con el fin de no tener conflictos en la asignación de las mallas a los huesos, además algunas piezas se dejaron separadas un poco para que sus mallas quedaran separadas con el mismo fin.

Figura 24. Brazo Robótico Ensamblado

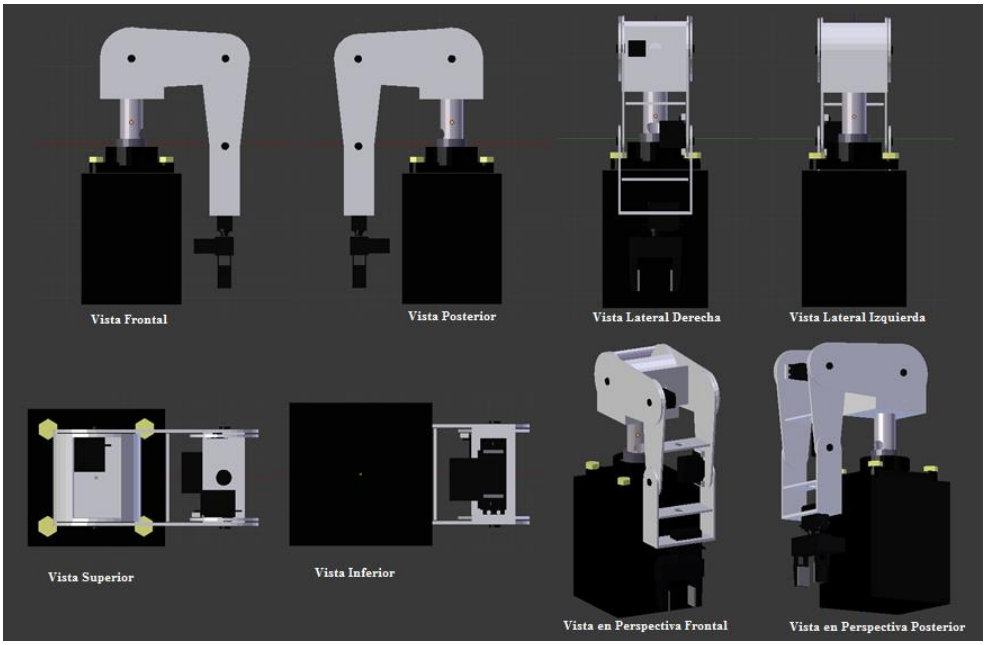
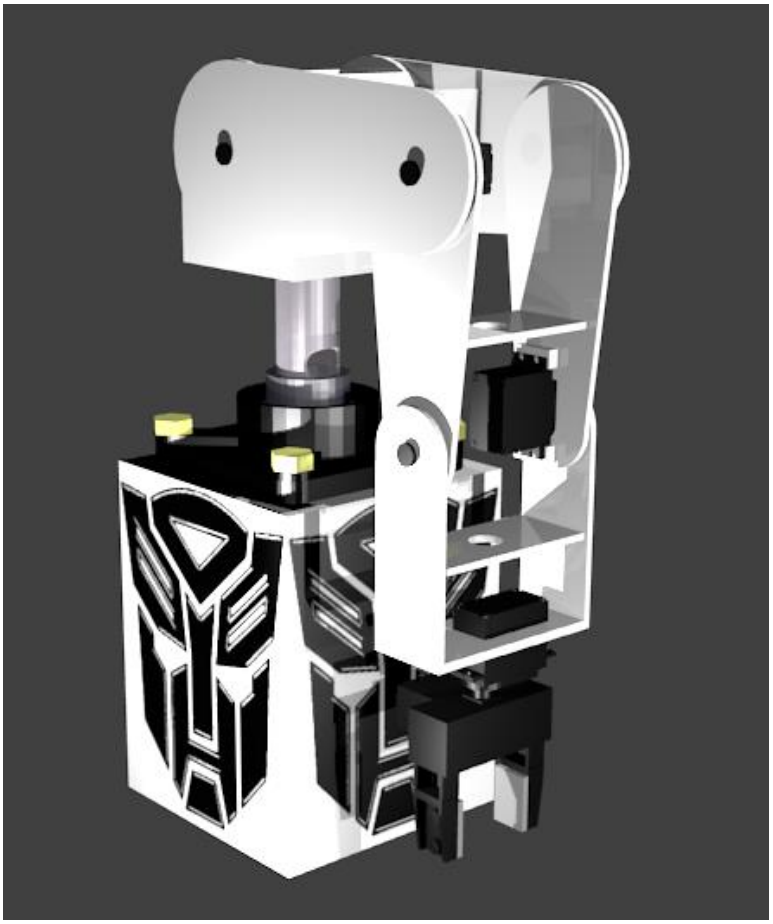


Figura 25. Brazo Robótico Renderizado en Blender



4.1.6.3 Modelado 3D del Robot Móvil Real

Al igual que con el modelo del Brazo Robótico primero se realizaron las piezas principales: parte superior e inferior de la carcasa, ruedas y piezas adicionales.

Figura 26. Modelo 3D: Parte Superior de la Carcasa

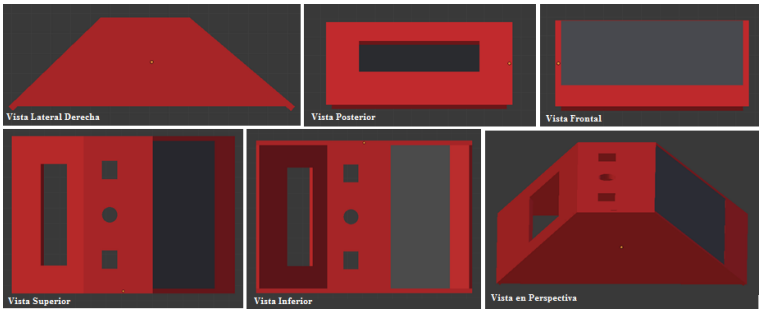


Figura 27. Modelo 3D: Parte Inferior de la Carcasa

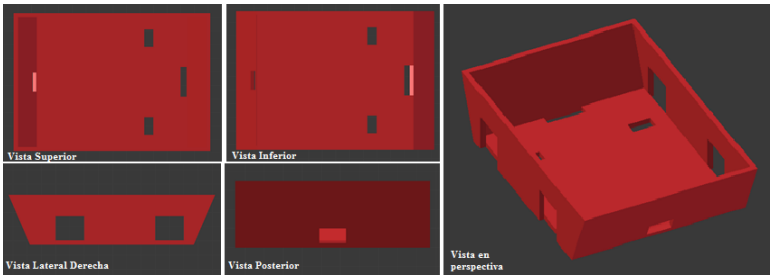


Figura 28. Modelo 3D: Ruedas

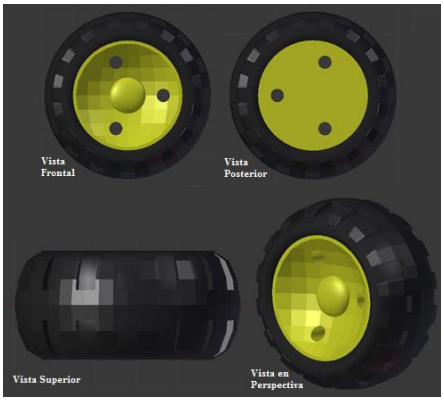
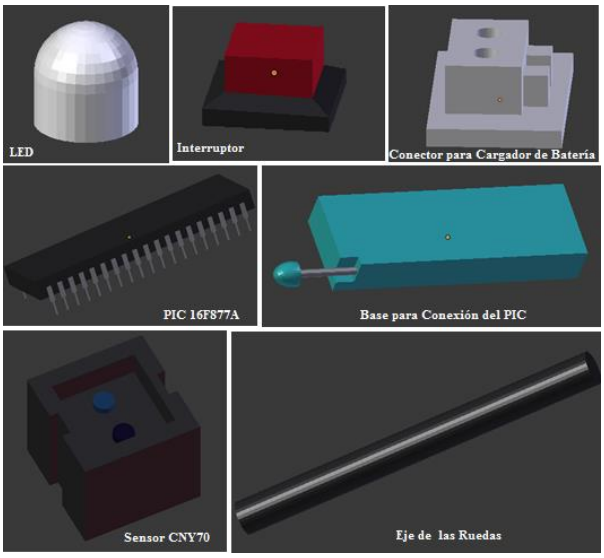


Figura 29. Modelo 3D: Piezas Adicionales



Finalmente se ensamblan las piezas formando una sola malla.

Figura 30. Robot Móvil Ensamblado

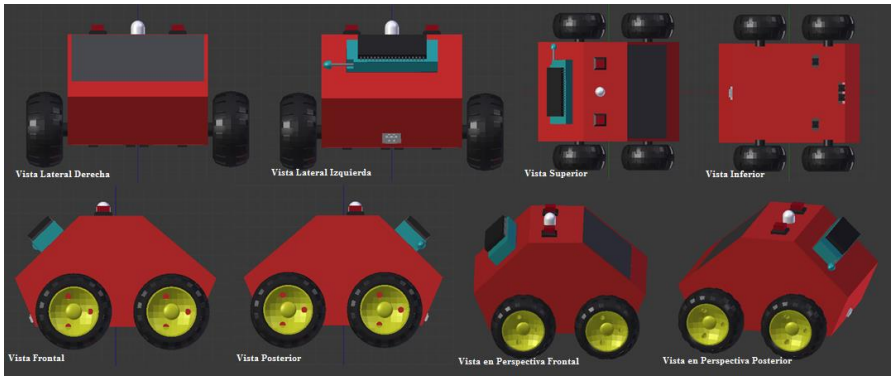
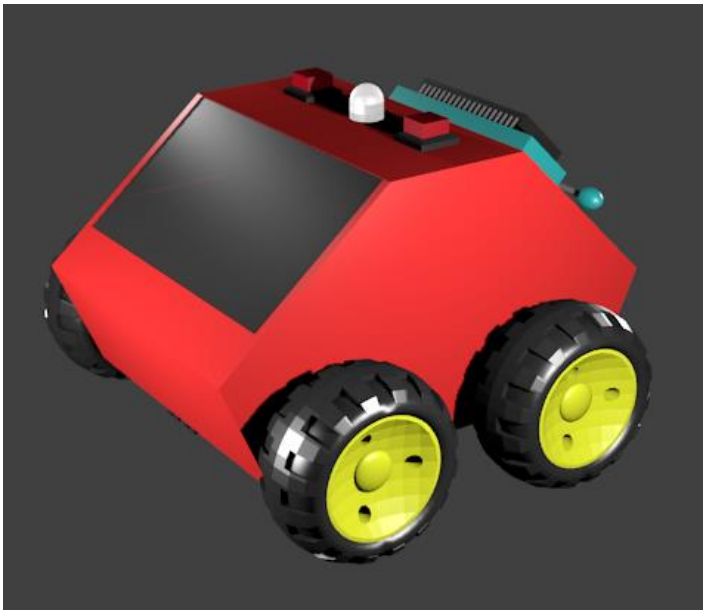


Figura 31. Robot Móvil Renderizado en Blender



En la asignación de armaduras, en el caso del brazo robótico, se asignó un hueso a cada articulación del brazo y en el caso del robot móvil se asignó uno a cada rueda y uno central que es el padre de los demás huesos.

4.1.7 Interfaz de Usuario

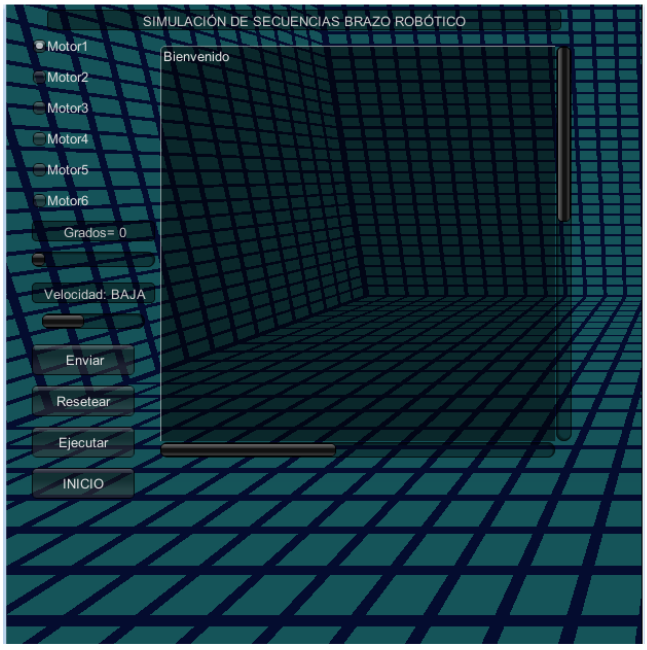
GUI Principal. La GUI principal se realizó teniendo en cuenta que el usuario debe inferir que este software es un producto de la UPN y también que debe servir como acceso a las demás escenas del proyecto. Además durante su creación se estableció como nombre para

el software ***"RoboticSAD"***, teniendo en cuenta que es un Software de Apoyo Didáctico para la enseñanza de Robótica y que este debería estar plasmado en la ventana de inicio del software.

Figura 32. GUI Principal

GUI Simulación Brazo Robótico y GUI Simulación Robot Móvil. Estas dos interfaces deben cumplir las mismas funcionalidades pero en diferentes escenarios y para diferentes instrucciones, así que se asemejan mucho en la estructura. Para el caso del brazo robótico se facilitan las funciones de elegir un motor, definir la posición en grados, elegir velocidad, enviar la instrucción a la consola, resetear las instrucciones, ejecutar las instrucciones y volver a la GUI principal. En el caso del robot móvil se permite elegir la instrucción a simular, cuenta con un slider que manipule el número de intersecciones en caso de que la instrucción sea Avanzar y Contar (esta se explica más adelante) y finalmente tiene las mismas funciones enviar a la consola, resetear, ejecutar y volver a la GUI inicial.

Figura 33. GUI Simulación Brazo Robótico y GUI Simulación Robot Móvil

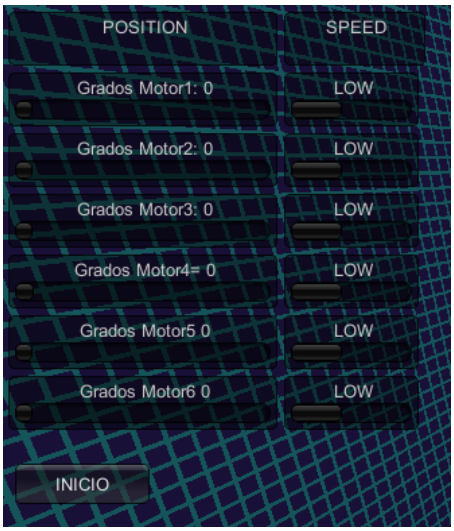


GUI Simulación Brazo Robótico

GUI Simulación Robot Móvil

GUI Comunicación Brazo Robótico. En esta interfaz básicamente se encuentran los sliders individuales para cada motor y su respectiva velocidad, adicionalmente está el botón para regresar a la GUI principal.

Figura 34. GUI Comunicación Brazo Robótico

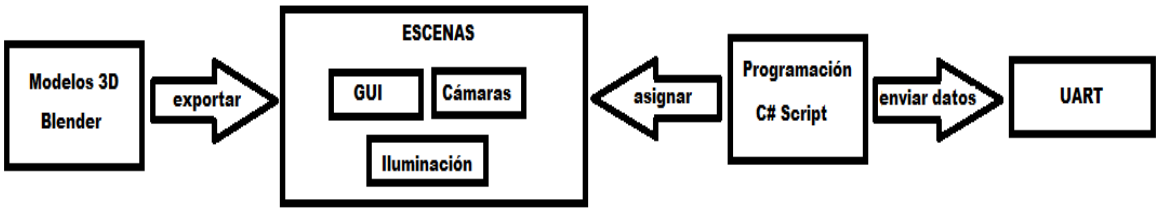


4.1.8 Integración de componentes del software

La integración se lleva a cabo mediante la exportación de los modelos 3D a Unity, el diseño y organización de las escenas, y finalmente el desarrollo de los scripts que van a influir en el comportamiento virtual de los robots y en el comportamiento físico del brazo robótico.

Por lo anterior se pueden establecer cuatro componentes principales en software: los modelos 3D, las escenas, los scripts y la UART; en la Figura 35 se puede ver como se integran dichos componentes.

Figura 35. Integración de componentes



4.1.9 Implementación del software[12]

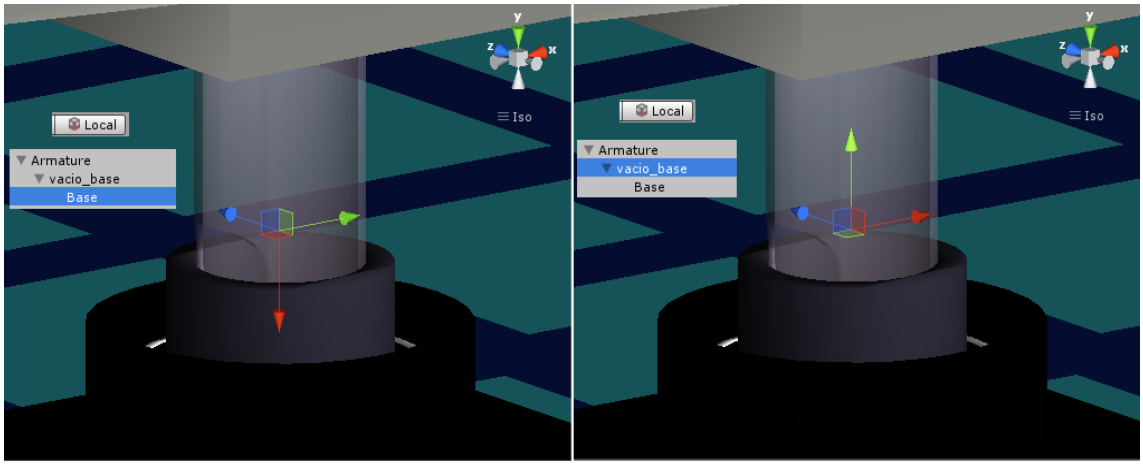
4.1.9.1 importación y adecuación de modelos 3D en Unity

Para importar los modelos de Blender a Unity basta con arrastrar el archivo y soltarlo en la ventana *Proyect – Assets* (recursos del sistema), una vez hecho esto podemos arrastrar el modelo hacia la ventana de jerarquía o directamente en la ventana de escena y soltarlo, en cualquiera de los dos casos ya se podrá ver como un objeto o *GameObject* en la escena, con sus respectivas propiedades *transform*. Otra forma de hacerlo es en la pestaña *Assets*, seleccionando la opción *Import New Asset* y buscando la ruta en que fue guardado el archivo .blend; este no es el único formato que admite Unity y tampoco el único formato de exportación de Blender, sin embargo no presenta conflictos debido a que estos dos software son compatibles.

[12] Objetivo específico No. 3

Una vez que se importan los modelos se deben posicionar, rotar y escalar de acuerdo a las necesidades de simulación. Posteriormente se deben jerarquizar los huesos del objeto, si no se ha realizado esto desde Blender, esto con el fin de que el movimiento del hueso principal incida en el movimiento de los demás huesos de la jerarquía; además se adicionan unos objetos vacíos (sin render pero con las mismas propiedades básicas de un GameObject) que va a hacer las veces de padre para cada hueso, esto para no generar conflicto con los ejes locales de Unity y Blender.

Figura 36. Ejes locales desde Blender a Unity



Hueso de la Base **Objeto Vacío - Padre del Hueso de la Base**

4.1.9.2 Funcionamiento de los robots

Para los dos robots se establecen un conjunto de instrucciones mínimo para la simulación, esto se hace de acuerdo al funcionamiento del hardware[13]:

- *Instrucciones para Brazo Robótico.* Este HAD realiza los movimientos de cada articulación utilizando un servo motor, por esta razón las instrucciones de simulación son: mover Motor1 a la posición "n" a "m" velocidad, mover Motor2 a la posición "p" a "q" velocidad, etc.

- *Instrucciones para Robot Móvil.* Las instrucciones del robot móvil son: Avanzar y contar intersecciones, girar 90° hacia la derecha y girar 90° hacia la izquierda.

[13] Objetivo específico No. 1

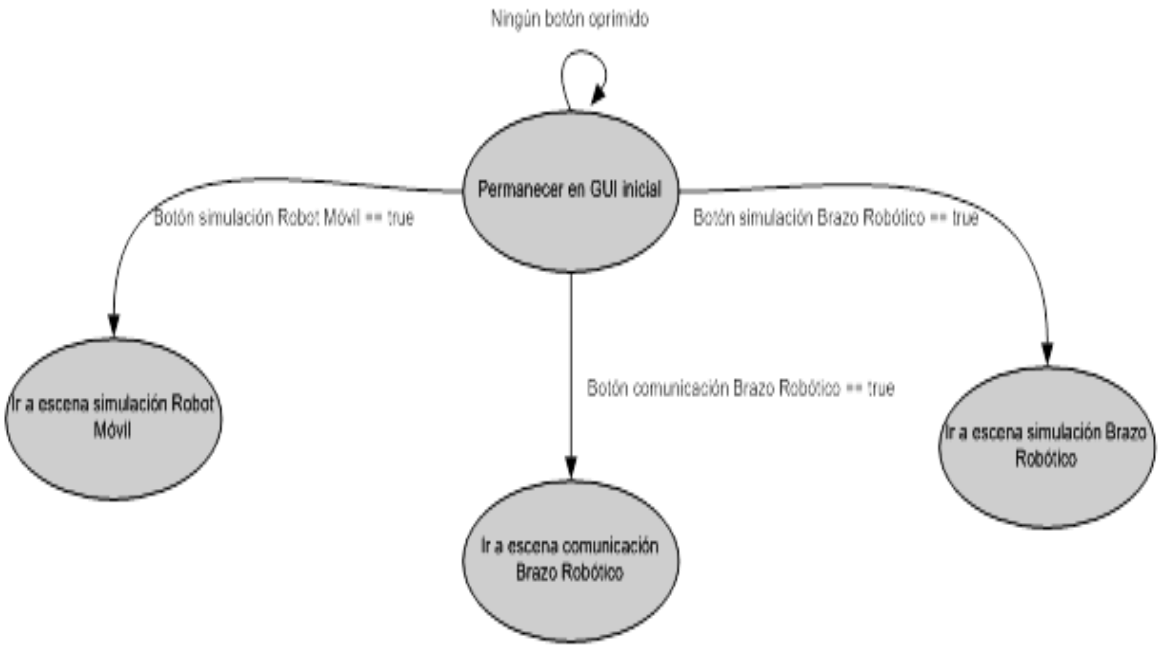
Físicamente este HAD cuenta con cuatro sensores CNY70 en su parte interior, dos que funcionan como seguidor de línea y dos que cuentan las intersecciones sobre línea negra (Ver Figura 30. Modelo 3D: Robot Móvil Ensamblado).

4.1.9.3 Máquinas de estados

Se hace un diagrama de flujo para cada escena del software:

- *Escena GUI Inicial.* Como se pudo apreciar en la GUI de esta escena hay tres botones para acceder a las demás escenas, por lo tanto el diagrama de máquina de estados es el siguiente:

Figura 37. Máquina de estados GUI Inicial

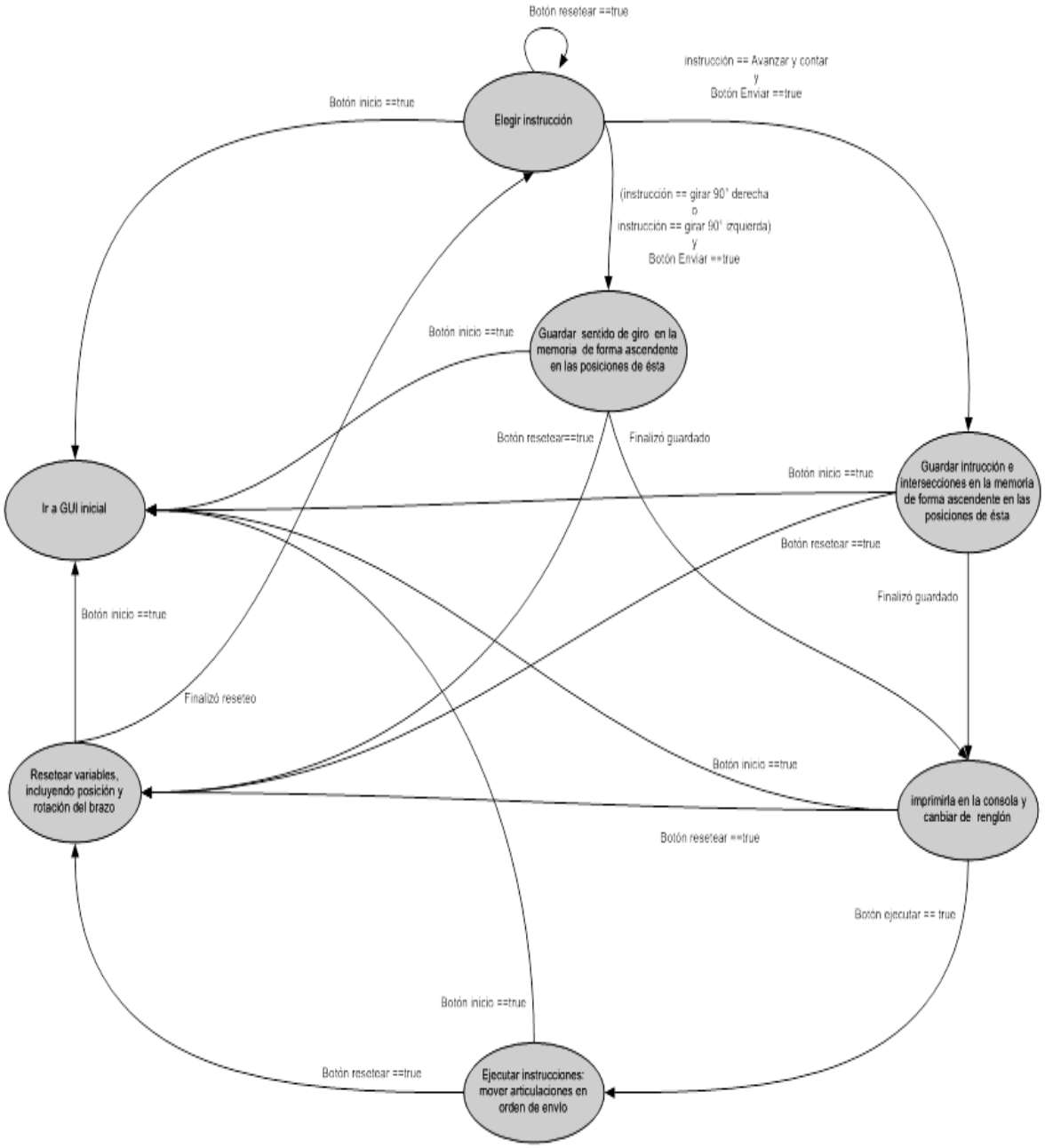


- *Escena Simulación Brazo Robótico.* En esta máquina de estados se tiene en cuenta que en cualquier momento se puede resetear la simulación y en cualquier momento se puede regresar a la escena GUI inicial, por esta razón la mayoría de los estados están verificando que ninguno haya sido oprimido.

Figura 38. Máquina de estados Simulación Brazo Robótico

- *Escena Simulación Robot Móvil.* Esta máquina de estados es similar a la del brazo robótico, con la variante de que en esta se plantean un estado para la instrucción avanzar y contar, y un estado para la instrucción girar 90° a la derecha o a la izquierda, esto debido a que a diferencia del brazo las instrucciones tienen parámetros diferentes.

Figura 39. Máquina de estados Simulación Robot Móvil



- *Escena Comunicación Brazo Robótico.* Para esta escena no se realiza máquina de estados, debido a que a medida que se van cambiando los sliders de velocidad y grados de cada motor, también se van ejecutando las instrucciones en el modelo virtual y se van enviando al robot físico para que también las ejecute.

4.1.9.4 Funciones importantes en Unity 3D

Para el desarrollo de los scripts se contemplaron las siguientes funciones de la clase *Monobehaviour:*

- *Start ().* Se ejecuta únicamente al inicio de la escena en la que se esté interactuando.
- *Update ().* El código que se encuentre dentro de esta función se está ejecutando todo el tiempo siempre y cuando el script que lo invoca este activo (tener en cuenta que la ejecución de los scripts en Unity se hacen por *frame*).
- *OnGUI ().* Esta función se ejecuta por *frame* y sirve para utilizar comandos GUI, si algún tipo de comando GUI se pone dentro de la función *Update* no funcionará.

De la clase *Rigibody* se contemplaron las siguientes funciones:

- *OnCollisionEnter ().* Se ejecuta cuando el objeto al que se asignó el script colisiona con otro objeto, siempre y cuando ambos cuenten con un componente de colisión y al menos uno de ellos tenga el componente r*igidbody*.
- *OnCollisionExit ().* Se ejecuta cuando el objeto al que se asignó el script sale de colisión con otro objeto, siempre y cuando ambos cuenten con un componente de colisión y al menos uno de ellos tenga el componente *rigidbody*.

De la clase *Transform* se contemplaron las siguientes funciones:

- *Translate ().* Permite mover el *transform* en la dirección y distancia que se le indiquen, también se le puede indicar si se moverá según las coordenadas locales o globales.
- *Rotate ().* Sirve para rotar un *transform* indicándole los grados de rotación en ángulos Euler, se le puede indicar si la rotación es respecto a sus coordenadas locales o globales.
- *Find ().* Esta función sirve para buscar por nombre un *transform* hijo del *transform* al que hemos asignado el script. Si hay varios niveles en la jerarquía se utiliza *slash* "/" para dar la ruta (Ejemplo: *Find* ("*Art1/Art2/pinza*")).

4.1.9.5 Realización y asignación de Scripts

Cada script es una clase en Unity y cada una de ellas hereda de la clase base MonoBehaviour, esto se puede ver en los scripts que se presentan como anexo de este documento.

- *Script para GUI inicial.* Para este script se crearon tres botones y una caja de texto dentro de la función *OnGUI ()*, cada botón carga la escena correspondiente en caso de que sea activado y la casa de texto muestra el nombre del software *"RoboticSAD"* como título. Este script está asignado al *GameObject "Main Camera"* o cámara principal.

- *Script para simulación Brazo Robótico.* Aquí se creó una matriz de 50 * 3 (filas * columnas) que hace las veces de memoria del software (esto limita el número de instrucciones que se pueden enviar a 50), aquí se guardan las instrucciones para la simulación en orden de envío, cada vez que se presione el botón "Enviar" se guarda la instrucción en una fila y se incrementa un contador que hace que al volver a presionar ese botón la nueva instrucción se guarde en la siguiente fila. La cada fila tiene los siguientes datos [motor elegido, grados, velocidad], inicialmente tienen los siguientes valores guardados en cada fila [1, 0, 1], que se traducen en [motor1, 0°, velocidad baja], la velocidad puede ser 1: baja, 2: media y 3: alta.

  El botón enviar solo funciona si no se está llevando a cabo una ejecución de instrucciones, para parar la ejecución se debe oprimir el botón resetear, el cual vuelve todas las variables a su valor inicial incluyendo rotación y posición del brazo.

  Dentro de la función *Update ()* se ejecutan las instrucciones, cada vez que se ejecute una instrucción se incrementa un contador para pasar a la siguiente, cuando este sea igual al número del contador que se incrementó en el proceso de guardado no se realizan más movimientos, por esto para volver a crear una secuencia se debe resetear la simulación. En el caso de las pinzas los grados de 0-180 se traducen en un desplazamiento mediante la siguiente ecuación:

$$d = m - \frac{cx}{n}$$

Donde,

m = posición local inicial de la pinza derecha (pinza abierta) = 5.4

n = número de intervalos en que se va a traducir el desplazamiento = 180

c = distancia que hay entre la posición de la pinza derecha cuando está cerrada a cuando está totalmente abierta = 5.4 - 1.6 = 3.8

x = valor en grados enviados desde el slider = 0 < x < 180

Se debe tener en cuenta que para que el valor "d" no cambie se debe conservar en el modelo 3D de brazo la escala, rotación y posición iniciales para esta escena.

Remplazando los valores para d tenemos:

$$d = 5.4 - \frac{3.8x}{180}$$

Todo lo que tiene que ver la GUI de esta escena se realizó dentro de la función *OnGUI ()*. Este script se le asignó al *GameObject* "brazo_real1_aristas reducidas_7" que es el mismo modelo 3D del brazo Robótico de esta escena.

- *Script para simulación Robot Móvil.* Al igual que en la escena de simulación del Brazo robótico se creó una matriz de 50 * 2 para guardar las instrucciones, nuevamente se limita el número de instrucciones a 50, el envío, ejecución y reseteo se llevan a cabo de la misma manera. Inicialmente las 50 filas de la matriz tienen los siguientes valores guardados [1, 0], que se traducen en [instrucción 1, 0 intersecciones], el número de intersecciones solo se puede cambiar en caso de que elija la instrucción 1 *"Avanzar y Contar"* las demás instrucciones solo tienen en cuenta la casilla uno de la fila.

  Los elementos GUI de esta escena se crean y se configuran dentro de la función *OnGUI ()*.

En la función *Update ()* se lleva a cabo la ejecución de las instrucciones, cuando se termine de ejecutar una secuencia el robot móvil se detiene, para generar una nueva secuencia primero se debe resetear la simulación. Las instrucciones básicamente se rigen según la orientación del robot móvil, para esto se utilizaron las siguientes variables:

```
int ulto = 0;

int orientación =0;
```

Cada vez que se gire el móvil a la derecha $ulto = ulto + 90$, cuando se gire a la izquierda $ulto = ulto - 90$.

Si $ulto \geq 0$, $orientación = ulto \,\%\, 360$ (orientación = residuo de ulto/360).

Si $ulto < 0$, $orientación = 360 - \Big(\big(ulto * (-1)\big)\% 360\Big)$.

Esto se hace con el fin de que la *orientación* del móvil de un resultado entre cinco valores posibles: 0, 90, 180, 270 y 360, luego, este valor influye en el la dirección de desplazamiento del móvil.

Para el conteo de las intersecciones se tiene en cuenta la posición inicial del móvil y que dichas intersecciones están ubicadas a 20 unidades una de la otra; por ejemplo para el caso de que la orientación sea 0°, si se da la instrucción de avanzar y contar 2 intersecciones se definirá la posición en $ultx = ultx - 20 * 2$, donde *ultx* es la última posición que tenga en el eje x.

Cuando se ejecuta la instrucción de "Avanzar y contar" las cuatro ruedas del robot móvil giran hacia adelante con la misma velocidad, si se ejecuta la instrucción de *"Girar a la Derecha"* las ruedas del costado derecho del móvil reducen su velocidad de giro y finalmente, si la instrucción es *"Girar a la Izquierda"* las ruedas del costado izquierdo reducen su velocidad (Según especificaciones funcionales del HAD).

El script está asignado al *GameObject* "movil2" o modelo 3D del Robot Móvil de esta escena. Adicionalmente existe un script asignado a la armadura del móvil, la cual cuenta con un componente *Box Collider* y un componente *Rigibody*, este script lo que hace es que detiene el móvil cuando colisiona con alguna de las paredes del entorno (emulando la situación cuando el HAD termina la detección de línea negra), para esto se utilizó la función *OnTriggerEnter (),* que se ejecuta cuando el objeto colisiona con otro que tenga un componente *Collider* de tipo *Ttrigger* (área de colisión, al ocurrir esto se activa una variable estática de *pausa*, que indica al script principal del móvil que éste se debe detener y esperar a que se resetee la simulación.

- *Script para comunicación Brazo Robótico.* Aquí se realizan dos script uno que traduce las instrucciones, establecidas a partir de sliders, en movimientos del modelo 3D instantáneamente y otro que realiza la comunicación con el Brazo Robótico. El primer script está asignado al modelo 3D del brazo llamado "*brazo corregido*" y contiene la parte de la GUI, y el segundo está asignado al modelo 3D *"entorno_com"* (entorno de comunicación).

  Para la comunicación hubo la necesidad de procesar el ángulo en grados asignado mediante el slider, de tal manera que concordara con la posición de los servo del hardware. Los pasos que se realizaron para esto fueron:

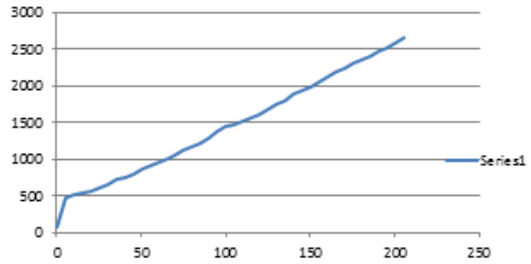
  1. Se hicieron pruebas para saber qué valor en µs se debía enviar al *Maestro Control Center* para que lo representara en una posición en grados para el servo; luego se constató que el comportamiento era lineal.

Tabla 6. Datos de prueba Grados vs µs

Grados	µs	Grados	µs
0	64	95	1380
5	476	100	1446
10	512	105	1480
15	543	110	1513
20	565	115	1556
25	619	120	1623
30	663	125	1687
35	715	130	1747
40	760	135	1801

45	799	140	1881
50	863	145	1940
55	909	150	1990
60	957	155	2063
65	1006	160	2126
70	1055	165	2186
75	1118	170	2236
80	1173	175	2302
85	1224	180	2349
90	1276		

Figura 40. Grafica de prueba Grados vs µs



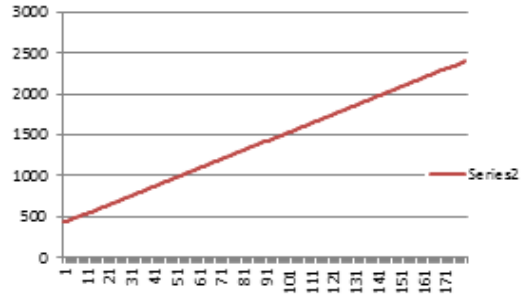
2. Al verificar que el comportamiento es casi lineal, se procedió a encontrar la ecuación de la recta que se acercara a este comportamiento y se obtuvo:

$$y = 11{,}666667x + 450$$

O de forma más clara:

$$valor\ en\ \mu s = 11{,}666667 * (valor\ en\ grados) + 450$$

Figura 41. Grafica de "y"



3. Teniendo la ecuación se procede a multiplicar el valor de "*y*" cuatro veces y se convierte a entero, esto se hace porque *Pololu* trabaja con cuartos de µs:

$$Us1 = [y * 4]$$

Los corchetes "[]" se utilizan para obtener la pate entera del número real, es decir el mayor número entero que no lo sobrepase.

4. Convertir Us1 a binario y separarlo en los 7 bits más significativos (7HB) y los 7 menos significativos (7LB). Para esto se utilizaron las siguiente ecuaciones:

$$HB = \left[\frac{Us1}{2^7}\right]$$

$$LB = [Us1 - (2^7 * HB)]$$

Como se puede apreciar estos valores están en enteros, ahora se convierten a byte así:

$$8HB = (byte)(7LB)\ ;\ 8LB = (byte)(7LB)$$

5. Después de hecho esto se envía la instrucción correspondiente a los motores mediante el puerto serie. Para la instrucción *Set target* se bebe enviar la cadena de bytes como sigue:

$$[0x84, numcanal, 8LB, 8HB]$$

numcanal es el número de canal del servo, por ejemplo el canal 1 se pondría como 0x01.

Para el comando *Set speed* estos valores se calcularon de forma manual debido a que son tres opciones posibles:

[0x87, numcanal, 0x24, 0x00]; velocidad alta 90grad en 1s = 0.9µs/ms

[0x87, numcanal, 0x12, 0x00]; velocidad media 90grad en 2s = 0.45µs/ms

[0x87, numcanal, 0x9, 0x00]; velocidad baja 90grad en 4s = 0.225µs/ms

Se debe enviar el comando *Set speed* y luego el comando *Set target.*

4.1.10 Ajustes finales

Los ajustes que se hacen finalmente al software son retirar todos los modelos de prueba importados en las escenas y los que se crearon directamente desde Unity, también configurar las cámaras de tal manera que se pueda visualizar de manera efectiva la ejecución de las instrucciones y se facilite el aprendizaje de los estudiantes, se eliminaron los scripts de prueba que se crearon durante el desarrollo del proyecto; todo esto con el fin de alivianar un poco la aplicación y utilizar menos recursos de PC.

El software debe cumplir con los requisitos planteados en la sesión 4.1.1 de este documento por esta razón se procede a hacer los ajustes pertinentes para lograrlo.

4.2 Presentación de resultados

Se verificó el cumplimiento de los requisitos de funcionamiento del software, sustentándolo y consignándolo en la siguiente tabla.

Tabla 7. Cumplimiento de los requisitos de funcionamiento del software

Requisito	Cumplimiento	Sustento
El software debe contar con las herramientas necesarias para llevar a cabo la simulación.	Se cumplió	Los usuarios podrán crear y ejecutar secuencias con las instrucciones de los robots.
Debe ser sencillo de utilizar.	Se cumplió	Las GUI se presentan en español y son fáciles de manipular.
Sus gráficos deben ser llamativos y realistas.	Se cumplió	Se realizaron los modelos 3D muy similares a los HAD y se utilizó Blender para esto, que es un software famoso por permitir realismo en los modelos.
Que permita ver de manera clara el funcionamiento del objeto de simulación.	Se cumplió	Se organizan las cámaras de tal manera que el usuario pueda ver con detalle la ejecución de los

		movimientos.
El software debe presentar conceptos claros y de manera ordenada.	Se cumplió	Permite conceptualizar la posición de un motor en grados, velocidad, programar, ejecutar, manipular, entre otros.
Debe orientar a los estudiantes en su aprendizaje.	Se cumplió	Cada acción que se realice dentro del programa permite al estudiante encaminarse en el paso a seguir.
Debe contener elementos que capten y mantengan el interés de los estudiantes.	Se cumplió	El realismo de los modelos, el escenario y la GUI se asemejan a un juego 3D, por esta razón el software despierta interés en el usuario.
Debe permitir que los estudiantes reconozcan sus propios errores.	Se cumplió	La ejecución de una secuencia o instrucción que puede generar conflictos en los HAD se puede visualizar en el software.
Debe permitir una interacción con los parámetros y variables de simulación.	Se cumplió	Todo el tiempo el usuario está controlando las variables y parámetros de simulación.
Debe permitir una comunicación entre el usuario y este a través de acciones.	Se cumplió	Las GUI y las escenas en si cumplen esta función.

4.3 CONCLUSIONES

En el desarrollo del proyecto “RoboticSAD” hubo construcción de conocimiento de manera constante y las principales conclusiones obtenidas son:

- RoboticSAD cuenta con herramientas que permiten una fácil interacción del usuario, para llevar a cabo la simulación grafica de un Robot Móvil y un Brazo Robótico construidos como HADs para la enseñanza de la robótica.

- RoboticSAD puede ser catalogado como un “software educativo” debido a que se pensó y se desarrolló para cumplir las características funcionales definidas para enmarcarlo en esta categoría.

- El software realizado tiene como finalidad facilitar el aprendizaje de la robótica, teniendo en cuenta situaciones en que no se cuente con los recursos de hardware necesarios, en cuyo caso se limita la experimentación e implementación física. Además permite que el usuario tome decisiones sin temor a equivocarse y a ocasionar daños en los componentes físicos.

- Las instrucciones mínimas para simular el comportamiento del Brazo Robótico están definidas a partir de un motor elegido, su posición en grados y su tipo de velocidad (baja, media, alta). En el caso del Robot Móvil las instrucciones son: Avanzar y Contar, Girar 90° a la derecha y girar 90° a la Izquierda.

- Para desarrollar un software de simulación gráfica se requiere un entorno de desarrollo que permita interactuar con modelos 3D así como un software para realizar dichos modelos.

- El software desarrollado se puede modificar y complementar fácilmente mediante la creación de nuevas escenas y scripts, gracias a que se diseñó pensando en facilitar el cumplimiento de los objetivos propuestos para las siguientes fases de desarrollo, como el caso de la fase 2 que requiere un componente de comunicación con el Hardware.

4.4 Recomendaciones

- Cuando se esté en el proceso de modelado 3D y luego se necesite exportar esos modelos a otro software, se recomienda exportar los modelos cada vez que se les modifique algo de manera significativa; esto con el fin de no encontrar sorpresas con la manera en que el otro software los pueda renderizar y conlleve a un retroceso en el proceso de modelado.
- Se recomienda que al hacer una fase del proyecto se tengan en cuenta los requisitos que debe tener para dar lugar a su continuación.
- Antes de empezar a utilizar un software de desarrollo es importante conocer las posibilidades que brinda según las necesidades que se tengan y realizar algunas pruebas de diagnóstico, esto brindara seguridad en la implementación de software.
- Se recomienda tener siempre presentes las necesidades del usuario para tomar decisiones en la implementación del software, esto ayudara a que el producto entregado sea óptimo.
- Desde el inicio se pensó en la continuidad del proyecto, por esto se les recomienda a los futuros desarrolladores que contemplen todas las posibilidades que ofrece Unity como motor de videojuegos y simulaciones, para así lograr que este software se convierta en una herramienta educativa de largo alcance en la robótica.

4.5 Dificultades

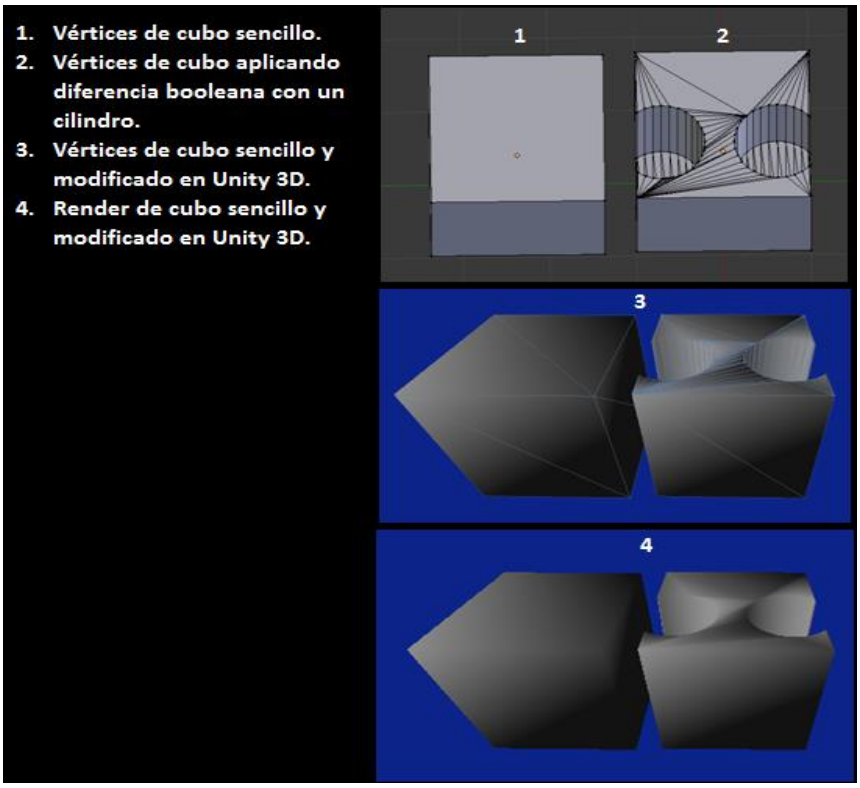
En el desarrollo del proyecto es importante documentar los inconvenientes que se fueron presentando, ya que es de gran utilidad para futuros desarrolladores. La idea es tratar de dar luz para quien quiera continuar el proceso y que sepan de antemano que problemas se pueden presentar.

Así mismo las dificultades presentadas fueron muy importantes en la construcción de conocimiento personal, ya que al tratar de solucionarlos, se fueron afianzando más los conceptos.

Las dificultades más relevantes en el desarrollo del proyecto fueron:

- En la realización del modelado 3D al unir piezas para formar una sola se generaban muchas aristas en una misma figura, lo cual generó un problema en la importación de modelo a Unity. En Blender el render era bastante bueno pero al importar los modelos a Unity se veían unas deformaciones en los objetos, inicialmente se pensó que era al importar el modelo con huesos pero después de buscar información acerca de esto, se concluyó que al trabajar con operaciones booleanas en Blender se forman varios vértices sobre todo en las áreas de corte (aplicación de diferencia booleana); al importar estos modelos, Unity le adicionaba más vértices y además los proyectaba al exterior del modelo causando su deformación visual.

Figura 42. Vértices de figuras en Blender y Unity 3D



- Una dificultad que se presentó con el maestro control center es que los datos que vienen en la guía de usuario respecto a la posición en grados para el giro del servo motor, no concordaban con su posición en la realidad. Debido a esto se realizó una aproximación de datos haciendo pruebas entre la posición del motor real y el dato que se le estaba enviando, luego se realizó una gráfica de los resultados y se

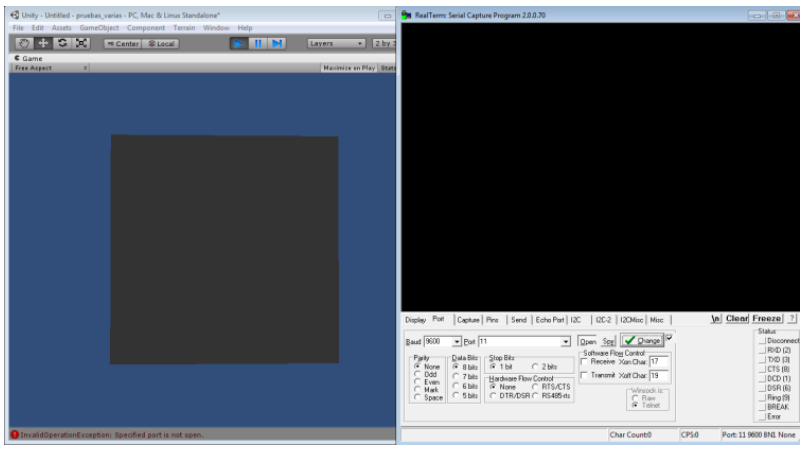
asemejaba bastante a una recta; así que se calculó la ecuación que definiera los datos a enviar desde el Maestro Control Center.

- Inicialmente se estaban realizando pruebas para la comunicación con el puerto serial virtual, con puertos mayores o iguales a 10 (COM10, COM11, COM12…), pero se presentaba un error (el error se puede observar en la Figura 44, en la parte inferior izquierda), enunciando que el puerto no estaba abierto; se llegó a pensar que dicha comunicación no era posible. Teniendo en cuenta que C# incluye librerías para comunicación serie, no podría ser problema de lenguaje sino de permisibilidad de Unity 3D, quizás porque era la versión libre del software.

Figura 43. Código de prueba de comunicación con puerto COM10

```
using UnityEngine;
using System.Collections;
using System;
using System.IO.Ports;

public class puerto : MonoBehaviour {
SerialPort serial=new SerialPort(); // variable serial de tipo SerialPort

    // Use this for initialization
void Start () {
serial = new SerialPort("COM10",9600); // Puerto 10 a 9600 Baudios
serial.Open(); // abrir el puerto

                    }

void Update () {
serial.Write("Yuli Marin "); // Enviar caracteres a través del puerto
                        }
                                                                }
```

Figura 44. Error de compilación con puerto COM10

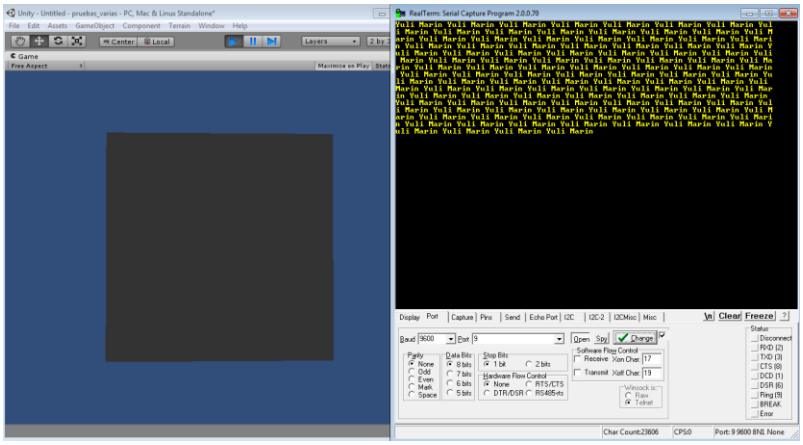


Después de varios intentos se realizó la misma prueba pero con puertos menores a 10, COM8 y COM 9, y los resultados obtenidos eran los que se estaban esperando.

Figura 45. Código de prueba de comunicación con puerto COM8

```
using UnityEngine;
using System.Collections;
using System;
using System.IO.Ports;

public class puerto : MonoBehaviour {
SerialPort serial=new SerialPort(); // variable serial de tipo SerialPort

    // Use this for initialization
void Start () {
serial = new SerialPort("COM8",9600); // Puerto 8 a 9600 Baudios
serial.Open(); // abrir el puerto

                    }

void Update () {
serial.Write("Yuli Marin "); // Enviar caracteres a través del puerto
                        }
                                                    }
```

Figura 46. Prueba de comunicación exitosa con puerto COM8



- Hubo un inconveniente que en un momento se pensó iba a obligar un cambio de software de desarrollo, por fortuna se solucionó buscando información acerca de esto: cuando se estaba trabajando en el movimiento de las articulaciones con el brazo de prueba en Unity, a pesar de la existencia de la jerarquía entre huesos si se rotaba uno de estos, sus huesos hijos no seguían este movimiento. Ya se habían realizado pruebas con algunas animaciones en Blender y los movimientos concordaban con la estructura jerárquica, se hicieron varios intentos en Unity hasta que se encontró en algunos foros que para lograr que los objetos hijos sigan el movimiento de sus padres, se deben dar las instrucciones respecto a los ejes de coordenadas locales y no globales, además de esto se deben poner objetos vacíos sin

rotación a cada hueso de la jerarquía, debido a que desde Blender ya pueden traer algún tipo de rotación, que interfieren con lo que se quiera realizar.

- Para este proyecto el mayor ángulo de rotación que se trabajó en Unity fue 180°. En la parte de simulación del brazo robótico, cuando se le daba la instrucción a cualquiera de los huesos que rotara un ángulo menor a 180° lo hacía en el sentido que se le indicara, pero cuando la instrucción era girar 180° Unity cambiaba el sentido de giro, así que se hizo para el caso de que el ángulo de giro fuese 180° dividiera el giro en dos partes es decir primero lo moviera a una posición de un ángulo menor y de allí si se desplazara hasta los 180°.

4.6 Proyecciones

Este software se desarrolla con el fin de que futuros proyectos de grado puedan dar continuidad a sus siguientes fases, sin embargo también está dispuesto para su mejora y/o evolución a partir de nuevas propuestas que promuevan su carácter pedagógico y que den atención a las necesidades que se vayan identificando en el campo de la Robótica Escolar. RoboticSAD también puede servir como base de inspiración hacia el desarrollo de nuevos proyectos enfocados en otros campos de estudio.

En cuanto a la siguiente fase de programación, se dispuso un tipo de comunicación entre el software y el brazo robótico a modo de prueba, sin embargo es importante trabajar nociones de programación más avanzadas, por esta razón se recomienda trabajar la programación de ciclos y secuencias tanto para el brazo robótico como para el robot móvil, además se debe definir la UART para el robot móvil.

En caso de querer complementar el software, algunas ideas diferentes a las fases ya propuestas y que pueden contribuir a su evolución pueden ser:

- Simulación y programación de nuevos HADs
- Desarrollo de entornos didácticos que lleven a cabo procesos de evaluación de competencias.
- Desarrollo de simulación de componentes básicos integrados al estudio de la Robótica, como por ejemplo transistor, LEDs, motores, capacitores, entre otros.

- Crear entornos donde el usuario pueda construir sus propios robots.
- Modificar los entornos de simulación de tal manera que los robots puedan interactuar de distintas formas con elementos de dichos entornos.
- Debido a que Unity es un motor de juegos 3D y brinda un gran número de posibilidades en este campo, se podrían crear entornos en los cuales puedan interactuar dos usuarios y realizar competencias sobre simulaciones y/o programación, o también que se planteen retos de trabajo conjunto entre dos usuarios.

No olvidar que el proyecto general se divide en tres fases: Hardware, Software y Pedagogía, por tanto se pueden hacer aportes en estas áreas, ya sea de manera integral o específica; por esta razón el proyecto tiene muchos campos de trabajo, los cuales se traducen en oportunidades para desarrollar un sinnúmero de proyectos.

BIBLIOGRAFÍA

Libros y artículos

[1] Jacobso, Ivar; Booch, Grady y Rumbaugh, James. *"El Proceso Unificado de Desarrollo de Software".* Editorial Pearson, S.A., Addison Wesley. Madrid, 2000.

[2] Jacobson, Ivar; Booch, Grady y Rumbaugh, James. *"El Lenguaje Unificado de Modelado. Manual de referencia".* Editorial Pearson, S.A., Addison Wesley. Madrid, 2000.

[3] Pressman, Roger S. Adaptado por Ince, Darrel. "*Ingeniería del Software* Un enfoque práctico". Editorial Mc Graw Hill. Quinta Edición.

[4] Barrientos, Antonio; Peñin, Luis Felipe; Balaguer, Carlos y Aracil, Rafael. *"Fundamentos de Robótica".* Editorial Mc Graw Hill. Mardid , 1997.

[5] Ferguson, Jeff; Patterson, Brian, Beres, Jason; Boutquin, Pierrre y Gupta, Meeta. *"La biblia de C#"* Ediciones Anaya Multimedia (Grupo Anaya S.A). Madrid, 2003.

[6] Nieto Peña, Fabián Camilo; Gallego Tovar, Jaime Alberto y Reyes Samacá, Edwin Fernando. Tesis de Grado*: "Diseño y construcción de un kit de robótica "RoboPed" enfocado hacia los estándares en tecnología e informática propuestos por el ministerio de educación nacional"*. Universidad Pedagógica Nacional, Facultad de Ciencia y Tecnología, Licenciatura en Electrónica, Bogotá D.C. 2008.

[7] Henao Ortiz, Gloria Inés y Piñeros Torres, Astrid. Tesis de grado *"Diseño de un entorno digital de simulación para la enseñanza de los principios de morfología, control y comunicación en la robótica"*. Universidad Pedagógica Nacional, Facultad de Ciencia y Tecnología, Licenciatura en Diseño Tecnológico con Énfasis en Sistemas Mecánicos, Bogotá D.C. 2008.

[8] González Rodríguez, Araceli; Pineda Ortega, Manuel y Soberanes Leal, Dely Madai. Tesis de Grado: *"Seguimiento adaptativo de trayectorias con convergencia en tiempo finito de un robot antropomórfico virtual de tres grados de libertad"*. Universidad Autónoma del

Estado de Hidalgo, Instituto de Ciencias Básicas e Ingeniería, Centro de Investigación en Tecnologías de la información y Sistemas, Licenciatura en Sistemas Computacionales, Pachuca de Soto – Hidalgo 2007.

[9] Cárdenas Correa, Edwin Francis. Tesis de Especialización: *"Modelado y simulación de un robot caminador bípedo"*. Universidad Nacional de Colombia, Facultad de Ingeniería, Especialización en automatización Industrial, Bogotá D.C. 2005.

Cibergrafía y Bibliografía electrónica

Robótica y pedagogía.

[1] *"Robótica Educativa"* (s.f.). Consultado el 10 de Marzo de 2013, de URL http://es.wikipedia.org/wiki/Rob%C3%B3tica_educativa

[2] *"Funciones del Software Educativo"*. (s.f.). Consultado el 10 de Marzo de 2013, de URL http://proton.ucting.udg.mx/materias/robotica/r166/r151/r151.htm

Unity 3D y Blender

[3] Unity 3D®. (s.f). *Manual de Usuario y Manual de Scripting.* Consultado el 28 de abril de 2013, de URL http://unity3d.com/

[4] Blender®. (s.f.) *Documentación*. Consultado el 25 de Febrero de 2013, de URL http://www.blender.org/

[5] *"Curso de Unity"*. (2011). Consultado el 20 de Febrero de 2013, de URL http://www.youtube.com/user/FenixDiscom/

[6] *"Foro Unity 3D"*. (s.f.). Consultado el 28 de Junio de 2013, de URL http://www.unityspain.com/

[7] *"Wiki de Blender"*. (s.f.). Consultado el 2 de Marzo de 2013, de URL http://wiki.blender.org/

[8] *"Tutorial #1 de Unity 3D"*. (s.f.). Consultado el 10 de Julio de 2013, de URL http://www.widget-101.com/juegos/unity-3d/introduccion-a-unity-3d/

C#

[9] Microsoft®. (s.f.). *"Visual C#"*. Consultado el 30 de Marzo de 2013, de URL http://msdn.microsoft.com/es-es/library/kx37x362(v=vs.90).aspx

[10] *"Programación orientada a objetos"*. (s.f.). Consultado el 15 de Abril de 2013, de URL http://es.wikipedia.org/wiki/Programaci%C3%B3n_orientada_a_objetos

[11] *"C Sharp"*. (s.f.). Consultado el 6 de Junio de 2013, de URL http://es.wikipedia.org/wiki/C_Sharp

[12] *"Descubriendo Lenguaje C#"*. (2011). Consultado el 18 de Agosto de 2013, de URL http://www.youtube.com/watch?v=ZpGFfjxuAek&list=PLGlfxrSj1dRc1zOopVZqoTarQT1P-jqIE

[13] Espinosa, Hector. *"Herencia y Polimorfismo con C# .NET"*. (2011). Consultado el 15 de Agosto de 2013, de URL http://programacionorientadaaobjetos.wordpress.com/tag/polimorfismo/

Pololu

[14] Pololu Corporation®. (2001). *"Pololu Maestro Servo Controller User's Guide"*. Consultado el 15 de Junio de 2013, de URL http://www.pololu.com/docs/0J40/all

Otros

[15] *"Realidad Virtual"*. (s.f.) Consultado el 15 de Marzo de 2013, de URL http://www.realidadvirtual.com/

GLOSARIO.

LEGO®: Empresa que produce juguetes educativos en forma de piezas interconectables.

Acrónimo: Palabra que se forma al unir elementos palabras diferentes.

GUI: Siglas para Interfaz Gráfica de Usuario

Unity 3D®: Motor y de videojuegos utilizado como software de desarrollo principal para este proyecto.

Blender®: Software de modelado y animación 3D, utilizado para modelar los objetos 3D del proyecto.

C#: Lenguaje de programación orientado a objetos desarrollado por Microsoft.

Script: Conjunto de instrucciones que permite la automatización de tareas.

MonoDevelop: IDE (Entorno de Desarrollo Integrado) que utiliza Unity 3D para el desarrollo de scripts.

RoboticSAD: Nombre que se le da a este proyecto. Significa Software de Apoyo didáctico enfocado a temas de Robótica.

HAD: Siglas para Hardware de Apoyo Didáctico.

Chumacera: Pieza sobre la que descansa y gira el eje de una máquina.

Espejar: Reflejarse en algo.

Especularidad: Propiedad para reflejar una imagen.

UART: Siglas para Transmisor - Receptor Asíncrono Universal. Utilizado para comunicación serial.

GameObject: Objetos de Unity3D.

Render: En programas que trabajan con modelos 3D se refiere al proceso de generar una imagen desde un modelo.

Rigibody: Cuerpo con propiedades físicas.

Collider: Espacio 3D que representa los límites físicos de un GameObject.