



UNIVERSIDAD DEL BÍO-BÍO, CHILE

FACULTAD DE CIENCIAS EMPRESARIALES

Departamento de Sistemas de Información

SIMULADOR LABORATORIO CIMUBB

PROYECTO DE TÍTULO PRESENTADO POR PATRICIO ANDRÉS LABRA MEDINA
DE LA CARRERA INGENIERÍA CIVIL INFORMÁTICA
DIRIGIDA POR MÓNICA CANIUPÁN MARILEO

2023

Resumen

La tesis tiene como objetivo abordar la necesidad identificada en el Laboratorio de Sistemas Automatizados de Producción, conocido como CIMUBB, donde se imparte formación en automatización mediante robots y computadoras. La limitación de acceso al laboratorio y la dificultad para impartir clases durante situaciones excepcionales, como la pandemia, han generado la necesidad de encontrar una solución que permita brindar enseñanza sin presencia física.

La propuesta en este informe, consiste en desarrollar un simulador en Unity, un motor de videojuegos, que reproduzca la experiencia de trabajar con los brazos robóticos Scorbot utilizados en el laboratorio. Este simulador permitirá a los estudiantes interactuar virtualmente con los brazos robóticos, realizando ejercicios prácticos, programación y pruebas, de manera similar a como lo harían en el laboratorio real. De esta forma, se busca preservar la calidad de la enseñanza y superar las limitaciones de acceso físico al laboratorio.

El enfoque del proyecto se basará en la recreación de los brazos robóticos Scorbot, incluyendo su funcionamiento y comportamiento. Se implementarán diversas funcionalidades y herramientas que faciliten la comprensión y práctica de los conceptos de automatización, permitiendo que los estudiantes realicen actividades teóricas y prácticas desde cualquier lugar.

El resultado esperado es que este simulador se convierta en una valiosa herramienta para estudiantes y profesores del laboratorio CIMUBB, mejorando la calidad de la enseñanza en sistemas automatizados de producción y brindando la posibilidad de impartir cursos de forma virtual o complementaria a las clases presenciales.

Palabras Clave — Simulador, Brazos Robóticos Scorbot, Unity, Sistemas Automatizados de Producción, Enseñanza Virtual, Automatización, Robótica.

Índice general

1. Introducción	7
2. Justificación del Proyecto	9
2.1. El Proyecto	9
2.2. Otros Programas	10
3. Definición de la institución	13
3.1. Descripción de la institución	13
3.2. Descripción del área de estudio	13
3.3. Descripción de la problemática	14
4. Definición proyecto	17
4.1. Objetivos del proyecto	17
4.1.1. Objetivo General	17
4.1.2. Objetivos Específicos	17
4.1.3. Actividades	17
4.2. Ambiente de ingeniería de software	18
4.2.1. Metodología de Desarrollo	18
4.2.2. Tecnologías	18
4.2.3. Herramientas de apoyo	19
5. Especificación de requerimientos de software	20
5.1. Alcances	20
5.2. Objetivo del software	21
5.3. Descripción global del producto	21
5.3.1. Interfaz de usuario	21
5.3.2. Interfaz de hardware	22
5.3.3. Interfaz software	22
5.4. Requerimientos específicos	22
5.4.1. Requerimientos funcionales del sistema	22

6. Factibilidad	23
6.1. Factibilidad Técnica	23
6.2. Factibilidad Operativa	24
6.2.1. Impacto Positivo	24
6.3. Factibilidad Económica	24
6.4. Conclusión de la factibilidad	26
7. Análisis	27
7.1. Procesos de negocios futuros	27
7.2. Casos de uso	28
7.2.1. Actores	28
7.2.2. Diagrama de casos de uso y descripción	28
7.2.3. Especificación de los caso de uso	28
8. Diseño	30
8.1. Diseño interfaz y navegación	30
9. Código	31
9.1. Lógica Cinta Transportadora	31
9.2. Lógica Brazos Roboticos	36
9.3. Diseño interfaz y navegación	39
10.Pruebas	40
10.1. Elementos de prueba	40
10.2. Especificación de las pruebas	40
10.3. Responsables de las pruebas	40
10.4. Detalle de las pruebas	40
10.5. Conclusiones de prueba	40
11.Plan de capacitación y entrenamiento	41
11.1. Plan de capacitación	41
12.Plan de implantación y puesta en marcha	42
12.1. Plan de implantación	42
13.Trabajo Futuro	43
13.1. Cambios que se deben realizar	43
14.Conclusiones	44
Referencias	45

Índice de figuras

2.1. Software Robocell	10
2.2. Software Scorbace	11
2.3. Software Robocell y Scorbace	12
3.1. Laboratorio CIMUBB	15
3.2. SCORBOT ER V Plus	15
3.3. SCORBOT ER IX	16
6.1. Análisis gasto Personal CIMUBB	26
7.1. Modelo y Notación de Procesos de Negocio	27
7.2. Diagrama de caso de uso: Usuario	28
8.1. Diseño interfaz: Módulo Principal	30
9.1. rigidBody Position 1	33
9.2. rigidBody Position 2	34
9.3. rigidBody MovePosition 1	34
9.4. rigidBody MovePosition 2	35
13.1. Diseño interfaz: Módulo Principal	43

Índice de Tablas

4.1. Tecnologías	18
4.2. Herramientas de apoyo	19
5.1. Requerimientos funcionales	22
6.1. Requerimientos de operación computadores	23
6.2. Requerimientos de operación smartphones	24
6.3. Costos Equipamiento	25
6.4. Resumen de Costos	25
6.5. Resumen de Gasto Mensual CIMUBB	25
7.1. Especificación de casos de uso: Visualizar control de brazo robot	28
7.2. Especificación de casos de uso: Manejar control de brazo robot	29
7.3. Especificación de casos de uso: Visualizar brazo robot	29
7.4. Especificación de casos de uso: Controlar brazo robot	29

Capítulo 1

Introducción

En la actualidad, el mundo se intenta recuperar de una pandemia, la cual afectó de manera significativamente negativa al área de educación [1].

En la Universidad del Bío-Bío, el laboratorio de automatización llamado CIMUBB nace para llegar a preparar a los estudiantes para las nuevas tecnologías de automatización. En este edificio cuentan con distintos tipos de sistemas automatizados, o también se les nombra robots, de estos poseen diferentes tipos, pero en este proyecto se enfoca en el uso de los brazos robóticos de nombre SCORBOT, los cuales son encargados de la manipulación de objetos, su función es mover el objeto para ser depositado en un contenedor o bandeja para ser transportado en una cinta mecánica, o también dejar el objeto en otro robot.

Gracias a este laboratorio, los alumnos pueden salir con el conocimiento para poder llegar a utilizar un sistema automatizado. Pero esta enseñanza fue mermada para cuando se mantuvo una cuarentena, ya que las clases se realizaron de forma on-line, el uso de estos robots fue demasiado limitado, el alumnos debía esperar que el profesor Luis Vera le agendara una videoconferencia, la cual tenía que esperar hasta que el profesor llegara a tener acceso al laboratorio, así dejando una comunicación entre el alumno y el computador central de los robots. Esto llevo a que el alumno el cual, aparte de tener que esperar, no tendría conocimiento directo sobre el uso del robot mas que verlo a través de una cámara.

El profesor se encuentra con la problemática de como lograr realizar la enseñanza de la materia practica en el laboratorio, pero sin tener el acceso a este mismo. Lo que lleva a una solución, que son las tecnologías de la simulación. Hoy existe un programa llamado Unity, en el cual sirve tanto para crear videojuegos como para crear simulaciones realistas que pueden ser utilizadas en cine, presentación de un coche o un edificio totalmente amueblado, etc.

Para este proyecto se mezcla la idea de usar la simulación y los videojuegos, pues gracias a estos últimos se logra realizar el movimiento. La idea en este proyecto es crear una simulación del laboratorio de automatización CIMUBB, en el cual se enfoca en darle vida a los brazos robóticos SCORBOT. Con esto lograr que las personas que deseen aprender sobre el funcionamiento de estos brazos robóticos, o también llegar a probar distintos experimentos en el brazo robótico, el limite de creatividad es el limite del brazo robótico.

En este escrito, se cuenta con ocho capítulos, los cuales están ordenados de la siguiente

manera:

- Capítulo 1: Este capítulo sirve como punto de partida para el trabajo, brindando una visión general del tema y su relevancia. Se establecen los objetivos de la investigación y se presenta la estructura del documento, preparando al lector para adentrarse en el análisis y exploración del tema en cuestión.
- Capítulo 2: Se exponen las razones y motivos detrás de la elección de este trabajo o proyecto en particular. Se describe la importancia del tema y su relevancia en el contexto actual, destacando las problemáticas o necesidades que aborda.
- Capítulo 3: Se proporciona una descripción de la empresa y de la problemática a tratar.
- Capítulo 4: Se establecen los objetivos y alcance del proyecto.
- Capítulo 5: Se detalla los requerimientos y funcionalidades que el software o proyecto debe cumplir.
- Capítulo 6: Se evalúa la viabilidad del proyecto.
- Capítulo 7: Se analiza los actores que participan en el software.
- Capítulo 8: Se describe el diseño del software que se desarrollará para resolver el problema identificado.

Capítulo 2

Justificación del Proyecto

2.1. El Proyecto

El proyecto se enfoca en abordar la necesidad de utilizar los brazos robóticos SCORBOT del laboratorio CIMUBB sin la posibilidad de acceder físicamente a ellos. Ante la situación desafiante presentada por la pandemia, la solución inicial fue el uso remoto de computadoras específicas en el laboratorio para permitir la operación a distancia de las máquinas disponibles. Esta solución fue efectiva para garantizar la continuidad de las actividades educativas y de investigación en un entorno virtual, pero presentó algunas limitaciones.

Una de las principales limitaciones es la dependencia del profesor que debe estar presente físicamente en el laboratorio para realizar el monitoreo y la supervisión de la operación remota de los brazos robóticos. Esto puede ser problemático si el profesor no puede asistir al laboratorio debido a restricciones de viaje, enfermedad u otros compromisos. La falta de presencia física puede aumentar el riesgo de fallas mecánicas o incidentes inesperados, lo que podría resultar en daños considerables en el edificio o los equipos.

La idea central de este proyecto es replicar el laboratorio CIMUBB y el funcionamiento de los brazos robóticos SCORBOT de manera virtual para abordar la necesidad de impartir conocimiento sin requerir la presencialidad física en el laboratorio. Esta solución se presenta como una alternativa efectiva y sostenible ante los desafíos planteados por la pandemia y las limitaciones para acceder a los equipos en persona.

Mediante la simulación, se crea un entorno virtual que intenta recrear fielmente el laboratorio y permite a los estudiantes y profesores interactuar con los brazos robóticos SCORBOT a distancia. Esta garantiza que el conocimiento y las habilidades prácticas relacionadas con el manejo de los brazos robóticos se puedan transmitir de manera efectiva, sin incurrir en costos adicionales de equipamiento o mantenimiento de los equipos reales. Los estudiantes pueden realizar prácticas, experimentos y proyectos de manera segura y eficiente desde sus computadoras, permitiéndoles adquirir experiencia valiosa en el uso de esta tecnología sin exponerse a riesgos físicos o daños en el edificio.

Además, la simulación ofrece una ventaja significativa al permitir la flexibilidad de horarios para los profesores y estudiantes, ya que no están limitados por las restricciones de disponibilidad

del laboratorio físico. Los docentes pueden impartir clases y asesorar a los estudiantes de manera remota, brindando retroalimentación y guiándolos en sus proyectos desde cualquier lugar.

2.2. Otros Programas

- 1.- Robocell de Intelitek Intelitek es una empresa dedicada a la robótica y automatización, en su pagina dicen "brindamos a las instituciones educativas entornos interactivos de aprendizaje tecnológico".

Uno de esos entornos es el software Robocell, un software que le agrega un entorno 3D al usuario, el el cual podra ver un robot con dimensiones reales.

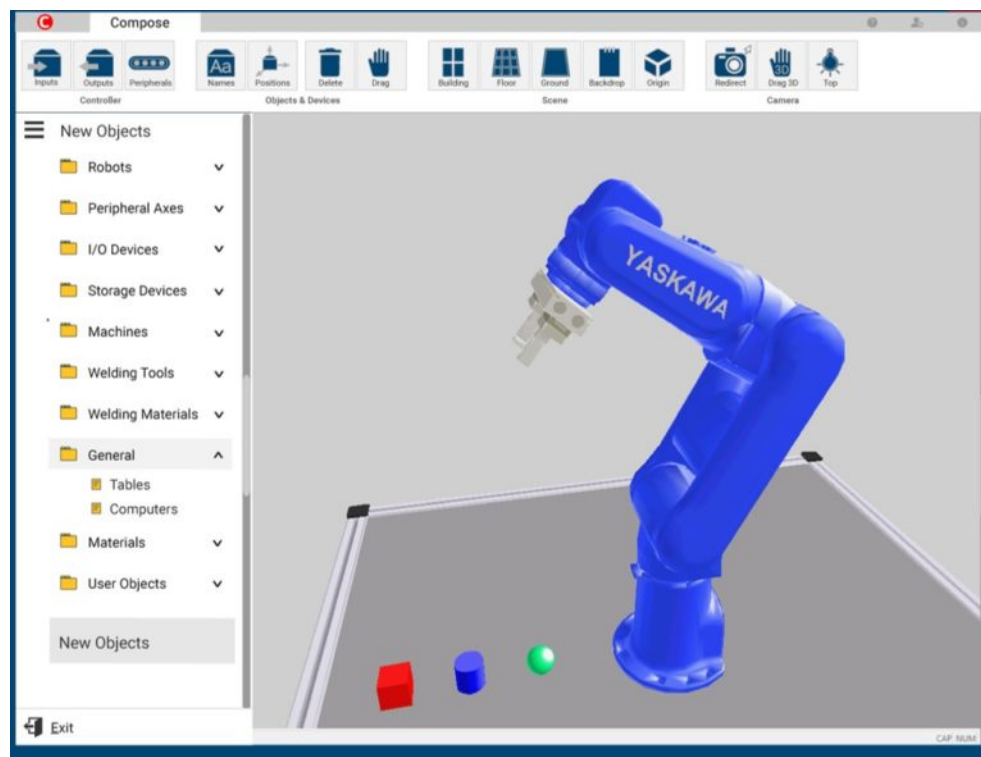


Figura 2.1: Software Robocell

En la figura 2.1 se presenta la interfaz que ofrece la empresa Intelitek, esta interfaz como tal no es util, debido que el software fue desarrollado para tener una conexión al programa Scorbace.

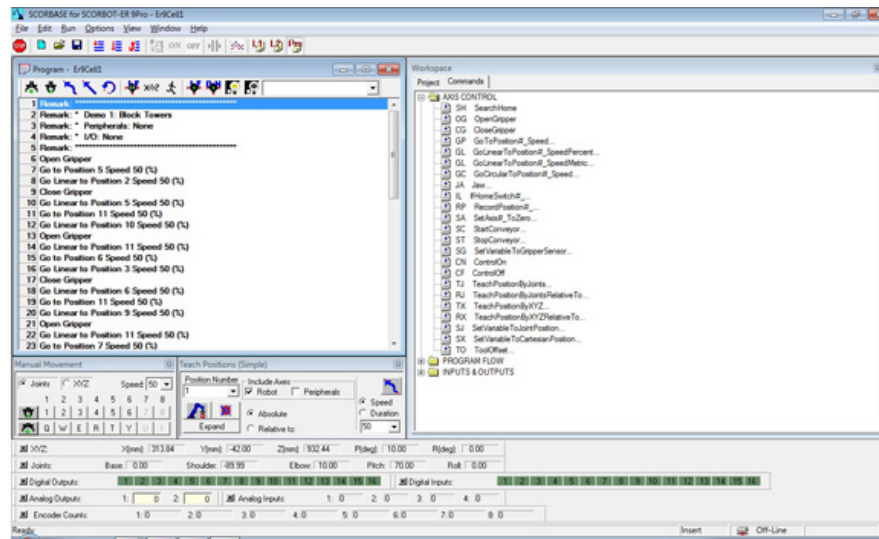


Figura 2.2: Software Scorbaser

En la figura 2.2 se muestra el programa Scorbaser, el cual tiene como finalidad programar y operar a los brazos robóticos, además de ofrecer soporte e integración a componentes externos, por ejemplo para realizar monitoreo o cintas transportadoras. Este programa fue creado especialmente para la operación de maquinaria física, por lo cual carece de una opción de poder visualizar el brazo robótico en uso. En la figura 2.3

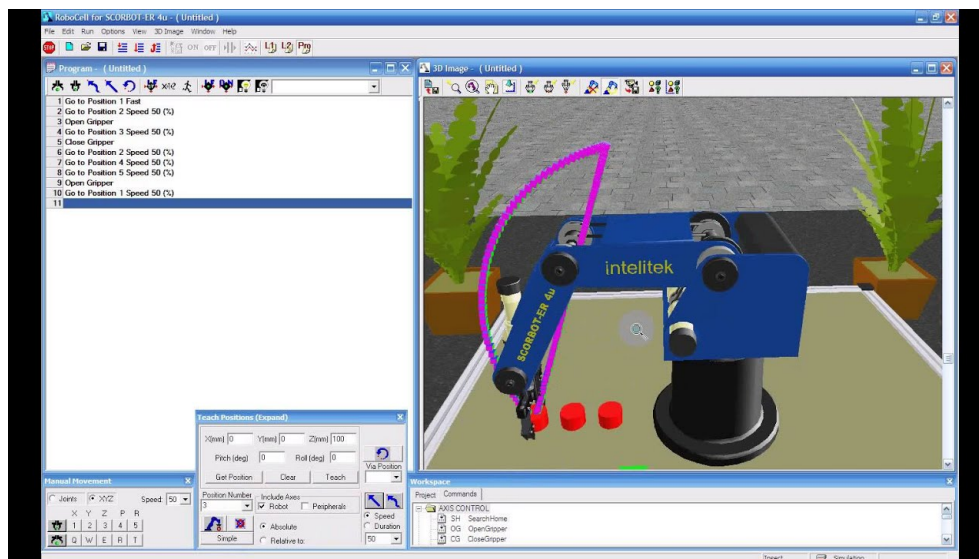


Figura 2.3: Software Robocell y Scorbaser

Capítulo 3

Definición de la institución

En el presente capítulo se presenta la institución y el problema que se aborda en el proyecto.

3.1. Descripción de la institución

- Nombre: Laboratorio de Sistemas Automatizados de Producción
- Dirección: Avenida Collao 1202, Concepción, Chile
- Encargado: Luis Vera
- Rubro: Automatización
- Teléfono: (41) 311 1137
- Descripción: Laboratorio de ensayos para el desarrollo de disciplinas en el ámbito de la Automatización Industrial, Tecnologías Avanzadas de Manufactura, Gestión de la Producción, Robótica, Control Numérico y Visión por Computador.

3.2. Descripción del área de estudio

La presente tesis tiene como objetivo principal abordar el área de estudio relacionada al laboratorio CIMUBB, el cual, por diversas razones, se puede no contar con acceso a este para realizar demostraciones prácticas. La limitación de acceso al laboratorio físico impide brindar una enseñanza completa y práctica a los estudiantes, ya que gran parte del aprendizaje depende de la interacción directa con la maquinaria y los experimentos que se encuentran en su interior.

Con el fin de superar esta limitación, se propone desarrollar un software de simulación que permita a los estudiantes experimentar y aprender de manera virtual dentro del entorno del laboratorio. El software de simulación recreará fielmente los equipos, las configuraciones y las condiciones experimentales presentes en el laboratorio real, brindando una experiencia inmersiva y cercana a la realidad.

La simulación se desarrollará utilizando tecnologías de vanguardia y siguiendo los estándares y protocolos establecidos en el campo de estudio correspondiente. Se diseñará una interfaz intuitiva y amigable para facilitar la interacción de los usuarios con los diferentes elementos del laboratorio

virtual, permitiéndoles realizar experimentos, ajustar parámetros, recopilar datos y observar los resultados de sus acciones.

La creación de este software de simulación tiene como objetivo principal proporcionar una alternativa efectiva y accesible para la enseñanza y el aprendizaje en ausencia del acceso físico al laboratorio. De esta manera, se busca brindar a los estudiantes una plataforma que les permita adquirir conocimientos prácticos, desarrollar habilidades experimentales y comprender los conceptos teóricos en un entorno virtual seguro y controlado.

Se espera que esta iniciativa proporcione una solución innovadora que amplíe las posibilidades de enseñanza en el área de estudio, permitiendo a los educadores complementar las clases teóricas con experiencias prácticas simuladas. Además, se busca fomentar la autonomía y la experimentación activa por parte de los estudiantes, promoviendo el descubrimiento y la resolución de problemas dentro del contexto del laboratorio virtual.

En resumen, esta investigación se enfoca en el desarrollo de un software de simulación de laboratorio para superar la limitación de acceso físico, permitiendo así una enseñanza práctica y completa en el área de estudio correspondiente. Se espera que esta solución contribuya significativamente a la formación académica y profesional de los estudiantes, brindándoles una herramienta virtual valiosa y efectiva para su aprendizaje y desarrollo.

3.3. Descripción de la problemática

A los comienzos de la pandemia, el profesor encargado del laboratorio CIMUBB, Luis Vera, comienza a notar una necesidad para sus alumnos, una necesidad que siempre estuvo presente pero, para en esa fecha, nunca había sido tan clara ni tan vital como llegó a ser.



Figura 3.1: Laboratorio CIMUBB

En la Figura 3.1 se muestra una imagen del laboratorio CIMUBB, en el cual se llevan a cabo la entrega del conocimiento sobre la automatización, para la problemática del proyecto se enfoca en los brazos robóticos SCORBOT, los cuales son tres, siendo dos modelos diferentes.



Figura 3.2: SCORBOT ER V Plus

En la Figura 3.2 se presenta el brazo robótico SCORBOT ER V Plus, es un brazo de antigua generación, teniendo cinco ejes de movimiento para el brazo robótico estático y seis ejes para el que posee una cinta de movimiento lineal.



Figura 3.3: SCORBOT ER IX

En la Figura 3.3 se presenta el brazo robótico SCORBOT ER IX, es una versión mejorada, la cual posee siete ejes de movimiento.

El laboratorio CIMUBB no cuenta con un entorno virtual, lo que dificulta significativamente el aprendizaje y la manipulación de los brazos robóticos SCORBOT cuando los estudiantes no pueden acceder físicamente al edificio. La falta de un entorno virtual restringe el acceso de los alumnos a prácticas y experimentos con los brazos robóticos, especialmente durante situaciones como cuarentenas, cortes de luz u otras fallas que requieren la presencia física de un supervisor o profesor.

La falta de acceso a un entorno virtual también limita la flexibilidad y disponibilidad para los estudiantes que deseen aprender sobre sistemas automatizados y brazos robóticos fuera de los horarios de clases regulares. Si el profesor no tiene acceso físico al laboratorio en un momento dado, los estudiantes se ven obligados a esperar hasta que el profesor pueda estar presente nuevamente en el lugar para realizar cualquier actividad relacionada con los brazos robóticos.

Capítulo 4

Definición proyecto

En este capítulo se detallan los objetivos, la metodología que se utiliza, y las tecnologías en la cual se apoya.

4.1. Objetivos del proyecto

4.1.1. Objetivo General

Desarrollar una aplicación en Unity que permita simular el laboratorio del CIMUBB, con la cual los estudiantes puedan probar el funcionamiento de los brazos robóticos SCORBOT ER V Plus y el brazo robótico SCORBOT ER IX , intentando replicar con el mayor detalle posible las estaciones de los brazos en el laboratorio.

4.1.2. Objetivos Específicos

- (a) Diseñar e implementar los brazos robóticos SCORBOT en Unity.
- (b) Diseñar e implementar interfaz de usuario.
- (c) Diseñar e implementar entorno de la estación.
- (d) Probar e implementar la solución.

4.1.3. Actividades

Las actividades desarrolladas en este proyecto son:

- I) Diseñar e implementar brazos robóticos SCORBOT
 - 1.- Investigación sobre funcionamiento sobre los brazos robóticos
 - 2.- Modelado e implementación de brazo robótico SCORBOT
- II) Diseñar e implementar interfaz de usuario
 - 1.- Investigación de diferentes métodos para desarrollar una interfaz en Unity
 - 2.- Análisis sobre técnicas de interfaces amigables para el usuario
 - 3.- Realización de implementación de la interfaz
- III) Diseñar e implementar entorno de la estación

- 1.- Recopilación de información sobre la estación
- 2.- Desarrollo del ambiente del laboratorio

4.2. Ambiente de ingeniería de software

4.2.1. Metodología de Desarrollo

La metodología de desarrollo a utilizar a lo largo del proyecto es incremental [2], pues se debe a que facilita al desarrollo del proyecto y saber si se encuentra por un buen camino, debido a que se le presenta cada incremento al cliente y obtener una retroalimentación del mismo.

4.2.2. Tecnologías


Nombre	Logo	Descripción
C#		C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común . Su sintaxis básica deriva de C / C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Tabla 4.1: Tecnologías

4.2.3. Herramientas de apoyo





Nombre	Logo	Descripción
Unity		Unity es un motor de desarrollo en tiempo real que te permite crear experiencias interactivas en el Editor de Unity que se utiliza para la creación de videojuegos. Estos se pueden publicar en diversas plataformas como PC, videoconsolas, móviles, etc. Gracias a su flexibilidad es una herramienta que también se usa en diferentes industrias como arquitectura, ingeniería, automotriz y de entretenimiento.
Blender		Blender es un programa informático multiplataforma, dedicado especialmente al modelado, iluminación, renderizado, la animación y creación de gráficos tridimensionales. También de composición digital utilizando la técnica procesal de nodos, edición de vídeo, escultura (incluye topología dinámica) y pintura digital.
Visual Studio Code		Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS. Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código.
Github		Es un servicio en la nube que ayuda a los desarrolladores a almacenar y administrar su código, al igual que llevar un registro y control de cualquier cambio sobre este código.

Tabla 4.2: Herramientas de apoyo

Capítulo 5

Especificación de requerimientos de software

En el siguiente capítulo se presentan los requerimientos de la aplicación.

5.1. Alcances

El alcance de la simulación abarca la recreación de los brazos Scorbot y otros componentes y maquinarias presentes en el laboratorio, aunque estos últimos no serán funcionales. Entre estos elementos se incluye una cinta transportadora y un objeto específico que el brazo Scorbot podrá tomar y manipular.

La simulación se centra en replicar el comportamiento y las características de los brazos Scorbot de manera precisa, garantizando un rango de movimiento y una capacidad de manipulación de objetos lo más realista posible. Además, se incluye una representación visual detallada de los brazos Scorbot y de los demás componentes y maquinarias presentes en el laboratorio.

El brazo robótico Scorbot puede realizar su movimiento en modo Joints, el cual hace girar en función de los motores. Por otra parte, el movimiento en modo XYZ no está disponible debido a fallos de diseño del modelado del robot. El brazo tampoco es capaz de realizar movimientos guardados, sin embargo en el código se está planteando la lógica de esta.

La cinta transportadora está disponible en la simulación, pero no funciona como en el entorno físico. Su presencia permite a los estudiantes experimentar con la interacción del brazo Scorbot en relación con la transferencia de objetos a lo largo de la cinta, aunque su movimiento y operación real no estarán habilitados del todo. Esto debido a darle más importancia al brazo como tal, sin embargo, se implementó un código rústico, el cual sirve pero no está pulido y tiene varias fallas.

Asimismo, se proporcionará un objeto específico en la simulación que el brazo Scorbot podrá tomar y manipular. Esto permite a los usuarios practicar las habilidades de agarre y manipulación del brazo, familiarizándose con los comandos y las técnicas necesarias para realizar estas tareas.

Es importante tener en cuenta que, aunque los componentes y otras maquinarias estarán presentes en la simulación, su funcionalidad real no estará activa.

El enfoque principal será brindar a los estudiantes una experiencia interactiva y visualmente representativa del entorno del laboratorio, centrándose en la operación y el uso del brazo Scrobot. El programa no está diseñado para personas no videntes, tampoco se contempla que el usuario no sepa usar el computador

5.2. Objetivo del software

El objetivo del software de simulación desarrollado es proporcionar una herramienta virtual que permita a los estudiantes experimentar, aprender y practicar el funcionamiento de los brazos Scrobot y otros componentes del laboratorio, en ausencia del acceso físico al mismo.

Los principales objetivos del software de simulación son:

- 1.- Brindar una experiencia realista: El software se diseñará con el objetivo de recrear de manera precisa y realista el comportamiento y las capacidades de los brazos Scrobot, así como otros componentes y maquinarias presentes en el laboratorio. Se buscará proporcionar una representación visual y funcionalidad detalladas, para que los estudiantes puedan interactuar con ellos de manera similar a como lo harían en el entorno físico.
- 2.- Facilitar la práctica y el aprendizaje: El software permitirá a los estudiantes practicar y adquirir habilidades en el uso y manejo de los brazos Scrobot, así como explorar la interacción con otros componentes del laboratorio. Los usuarios podrán realizar diferentes tareas y experimentos, ajustar parámetros y recopilar datos, todo dentro de un entorno virtual controlado y seguro.
- 3.- Fomentar la comprensión teórica y práctica: El software de simulación ayudará a los estudiantes a comprender los conceptos teóricos relacionados con el funcionamiento de los brazos Scrobot y su aplicación en diferentes situaciones. Podrán experimentar directamente los efectos de las acciones realizadas y observar los resultados de sus interacciones, lo que fortalecerá su comprensión de los principios subyacentes.
- 4.- Superar limitaciones de acceso: Al proporcionar una alternativa virtual al laboratorio físico, el software de simulación permitirá a los estudiantes acceder al aprendizaje práctico y experiencial en cualquier momento y lugar, sin restricciones de horarios o limitaciones de espacio físico. Esto ampliará las oportunidades de aprendizaje y brindará una opción adicional para aquellos que no tienen acceso directo al laboratorio.

5.3. Descripción global del producto

5.3.1. Interfaz de usuario

La interfaz de usuario es diseñada con un enfoque en la amigabilidad y familiaridad para minimizar la resistencia al cambio y maximizar la adaptabilidad. Al crear una interfaz que los usuarios encuentren intuitiva, se facilita su aceptación y comodidad al utilizar el software. Además, la adaptabilidad es crucial para garantizar que la interfaz funcione de manera eficiente en diferentes contextos y dispositivos, permitiendo a los usuarios acceder y utilizar el programa

de manera óptima en diversas situaciones. Un diseño cuidadoso y una atención especial a la experiencia del usuario pueden fomentar una transición más fluida y exitosa hacia el software.

5.3.2. Interfaz de hardware

En el contexto de computadoras, las interfaces de hardware típicas son el teclado y el ratón para la entrada de datos, y el monitor para la salida visual. Mientras que en el caso de teléfonos y tabletas, las interfaces de hardware se basan en pantallas táctiles, que permiten a los usuarios interactuar directamente con el dispositivo mediante toques y gestos en la pantalla.

5.3.3. Interfaz software

El programa no requiere la instalación de software externo adicional, ya que funciona directamente en sistemas operativos como Windows, Linux o Android. Esto implica que los usuarios pueden ejecutar el programa sin necesidad de instalar bibliotecas o programas adicionales, lo que simplifica su implementación y uso en diferentes plataformas. La compatibilidad con estos sistemas operativos amplía la accesibilidad del programa a una variedad de dispositivos y entornos, facilitando su distribución y adopción.

5.4. Requerimientos específicos

5.4.1. Requerimientos funcionales del sistema

Se presentan los requerimientos funcionales de la aplicación, donde se destaca las características que tiene.

ID	Nombre	Descripción
RF-01	Controlar el brazo robótico	La aplicación debe permitir al usuario hacer uso del brazo robótico
RF-02	Controlar botonera	La aplicación debe permitir al usuario hacer uso de la botonera del brazo robótico

Tabla 5.1: Requerimientos funcionales

Capítulo 6

Factibilidad

En el capítulo actual se mostrará la factibilidad que tiene el proyecto de ser realizado.

6.1. Factibilidad Técnica

Identificar necesidades del programa y documentarlas para cumplir objetivos y satisfacer a los usuarios.

Requerimiento de uso del software en computadores:

Requisitos mínimos	Windows	macOS	Linux
Versión sistema operativo	Windows 7 (Service Pack 1+), Windows 10 y Windows 11	High Sierra 10.13+	Ubuntu 20.04, Ubuntu 18.04, y CentOS 7
Procesador	Arquitectura x86 o x64 con soporte de instrucción SSE2	Arquitectura x64 con soporte de instrucción SSE2 para CPU Intel, Apple M1 o superior para CPU Apple	Arquitectura x64 con soporte de instrucción SSE2
API Gráfico	GPU compatible con DirectX versión 10, 11 o 12	GPU Intel, AMD o Apple compatible con Metal	GPU compatible con OpenGL 3.2+ o Vulkan

Tabla 6.1: Requerimientos de operación computadores

El programa al tener la disponibilidad de poder desarrollar una versión para celulares y derivados que posean el sistema operativo o sean capaz de ejecutar el sistema, se investiga sobre el requerimiento que se necesita en caso de ser ejecutado en alguno de estos dispositivos

Requisitos mínimos	Android	iOS
Versión sistema operativo	Android 5.1 (API 22)	iOS 12
Procesador	ARMV7 compatible con Neon o ARM64	A7 SoC
API Gráfico	GPU OpenGL ES 2.0 o Vulkan	GPU compatible con Metal

Tabla 6.2: Requerimientos de operación smartphones

Debido a que el proyecto tiene enfoque en ser utilizado en el lugar donde reside el usuario, se tomara de base los componentes que poseen los equipos que disponen en la biblioteca, pues se mantiene la idea del contexto en el cual los integrantes de la Universidad no poseen el acceso a el establecimiento, por lo cual se debía realizar el trabajo remoto desde el lugar de residencia.

Equipos portátiles:

- Procesador: Intel i3 8va gen (Por confirmar)
- GPU: Intel HD Graphics (Por confirmar)
- Sistema Operativo: Windows 10 (Por confirmar)

Utilizando los equipos que se encuentran disponibles en la biblioteca se puede hacer uso de la aplicación correctamente.

6.2. Factibilidad Operativa

El perfil de usuario al que va dirigido principalmente son los alumnos de la Universidad del Bío-Bío, los cuales deseen ocupar o aprender sobre el laboratorio del CIMUBB. Por otra parte, esta puede ser utilizada por cualquier persona.

6.2.1. Impacto Positivo

El impacto positivo de este proyecto es que entrega la posibilidad de poder ocupar virtualmente un laboratorio, sin necesidad de equipamiento, libre de riesgo de romper un equipo costoso.

6.3. Factibilidad Económica

Con lo visto en la factibilidad técnica, la Universidad cuenta con ordenadores que pueden ser utilizados, estos no genera gasto pero son contabilizados de igual manera.

Equipamiento	Costo Por Unidad
Equipo Portátil Biblioteca	\$314.990

Tabla 6.3: Costos Equipamiento

Para los costos del software:

- Unity: Se puede utilizar la versión gratuita o también la estudiante, posee versiones de pago pero no afecta el resultado.
- Blender: Es gratuito, posee modelos ya creados de pago y gratuitos que se pueden utilizar.
- Visual Studio Code: Es gratuito.
- Github: Es gratuito, posee versiones de pago pero para el desarrollo solo influye la mejora de almacenamiento en la nube.

Para el desarrollo, al tomar el coste promedio por hora de un Ingeniero Civil Informático en Chile, el cual esta rodeando los \$6.000 por hora. Con este dato se multiplica por las horas de trabajo e investigación, las cuales en al rededor de 13 semanas, tomando en cuenta la ley chilena de no trabajar más de 45 horas a la semana, serian 585 horas en total, lo que daría unos \$3.510.000

Al tomar en cuenta que se esta realizando un proyecto de título, este costo de desarrollo es despreciable. Además de que los equipos ya se encuentran disponibles, estos tampoco generan costo alguno.

Variable	Costo
Hardware	\$314.990
Software	\$0
Mano de obra	\$3.510.000
Total	\$3.824.990

Tabla 6.4: Resumen de Costos

Se analiza los gastos que se genera por la necesidad de mantenciones a los brazos robóticos y los gastos de mantener al personal del laboratorio

Variable	Costo Mensual
Personal	\$1.500.000
Mantención Brazo Robótico	\$165.000
Total	\$1.665.000

Tabla 6.5: Resumen de Gasto Mensual CIMUBB

Al tener en cuenta la finalidad del proyecto, el cual no busca reemplazar el laboratorio como tal, sino dar una alternativa para poder acceder a la simulación del laboratorio desde el lugar

de residencia, se podría reducir las horas de trabajo del personal del laboratorio, el cual trabaja 45 horas a la semana, al contar el mes como cuatro semanas, se obtiene que semanalmente es un gasto de \$375.000. Al ser 45 horas, y contando la semana desde lunes a viernes, el personal trabaja 9 horas diarias, por lo cual diariamente se tiene un gasto de \$75.000. Con esto se analiza que:

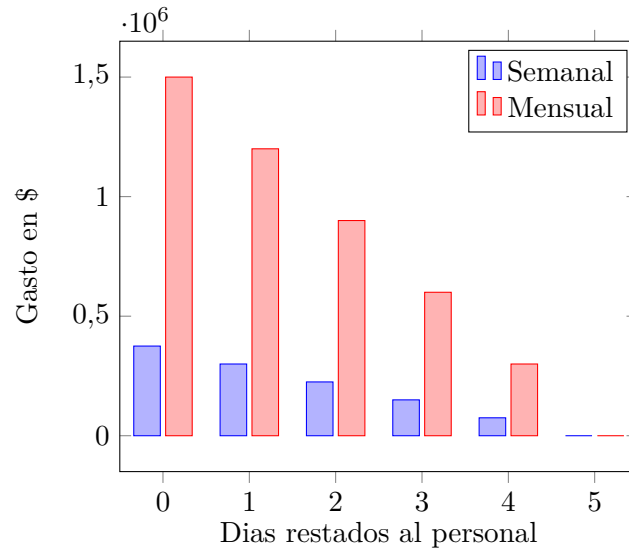


Figura 6.1: Análisis gasto Personal CIMUBB

En la Figura 6.1 se logra apreciar la reducción de gasto al reducir los días de trabajo del personal. Sin embargo, para las mantenciones no se pueden reducir, estas son contabilizadas anualmente y son de manera correctiva y no preventiva, siendo esta necesaria por el mal uso de los usuarios y nunca por el desgaste natural.

6.4. Conclusión de la factibilidad

En cuanto a la factibilidad técnica, la universidad dispone de las herramientas necesarias para el desarrollo y para su uso, se puede llegar a realizar y ocupar sin tener mayores gastos. Por otra parte, al revisar los gastos mensuales, se llega a concluir que la aplicación reduciría un gasto considerable.

Capítulo 7

Análisis

En este capítulo que nos conlleva a analizar los actores que ejecutarán la aplicación, los flujos que esta tendrá.

7.1. Procesos de negocios futuros

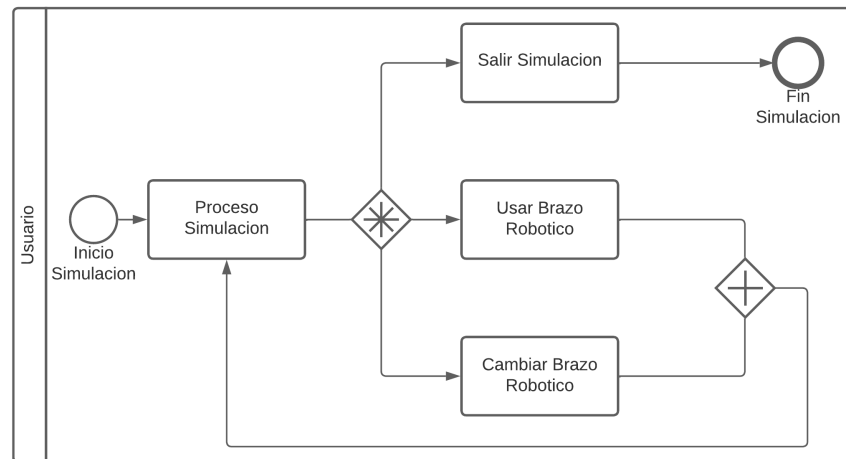


Figura 7.1: Modelo y Notación de Procesos de Negocio

En la Figura 7.1 se muestra el proceso de como el usuario interactúa con la simulación, el proceso inicia con el usuario ejecutando la simulación, una vez dentro de esta, el usuario puede elegir entre usar el brazo robótico, cambiar de brazo robótico, con las cuales lo lleva a mantenerse en el proceso de la simulación, por ultimo esta la opción de salir de la simulación que finaliza este proceso

7.2. Casos de uso

7.2.1. Actores

Usuario: Persona que utilizara la aplicación, ya sea estudiante o profesor de la Universidad del Bío-Bío, como también personas externas a esta. El usuario puede utilizar la aplicación completamente.

7.2.2. Diagrama de casos de uso y descripción

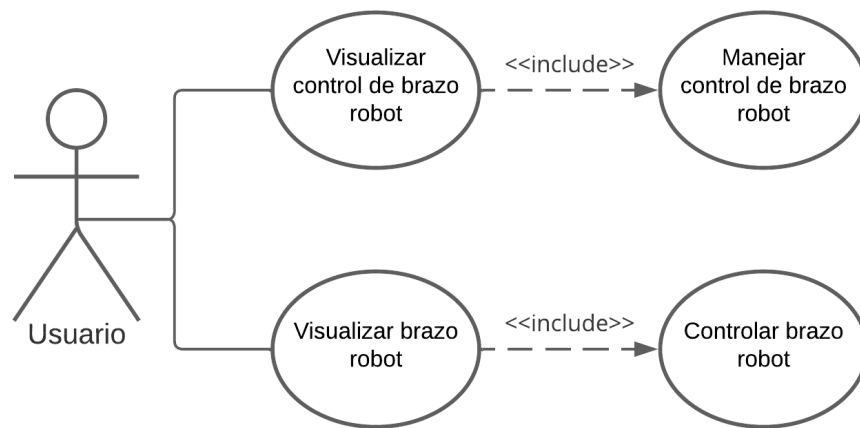


Figura 7.2: Diagrama de caso de uso: Usuario

Como se logra apreciar en la Figura 7.2, se muestran las interacciones que tiene el usuario en la aplicación. El actor es simplemente un usuario, ya que la aplicación es de uso personal.

7.2.3. Especificación de los caso de uso

Caso de Uso: Visualizar control de brazo robot	
ID	CU-01
Descripción	El actor puede visualizar el control de brazo robot.
Actores	Usuario
Flujo principal	1.- El actor debe haberse dirigido al menú desplegable 2.- El actor debe seleccionar la opción de "Visualizar Control"

Tabla 7.1: Especificación de casos de uso: Visualizar control de brazo robot

Caso de Uso: Manejar control de brazo robot	
ID	CU-02
Descripción	El actor puede manejar el control de brazo robot.
Actores	Usuario
Flujo principal	1.- El actor debe haberse dirigido al menú desplegable 2.- El actor debe seleccionar la opción de "Visualizar control de brazo robot"

Tabla 7.2: Especificación de casos de uso: Manejar control de brazo robot

Caso de Uso: Visualizar brazo robot	
ID	CU-03
Descripción	El actor puede visualizar el brazo robot.
Actores	Usuario
Flujo principal	1.- El actor debe haber abierto la aplicación

Tabla 7.3: Especificación de casos de uso: Visualizar brazo robot

Caso de Uso: Controlar brazo robot	
ID	CU-04
Descripción	El actor puede controlar el brazo robot.
Actores	Usuario
Flujo principal	1.- El actor debe haberse dirigido al menú desplegable 2.- El actor debe seleccionar el robot a utilizar 3.- El actor controla a través de teclado el robot

Tabla 7.4: Especificación de casos de uso: Controlar brazo robot

Capítulo 8

Diseño

En este capítulo, se presenta el diseño de interfaz de la aplicación desarrollada, y las diferentes opciones que se presentan.

8.1. Diseño interfaz y navegación

En la Figura 8.1 se presenta un bosquejo de la interfaz del usuario, la cual muestra la simulación principalmente, y además un control flotante. También se muestran los menús desplegables.

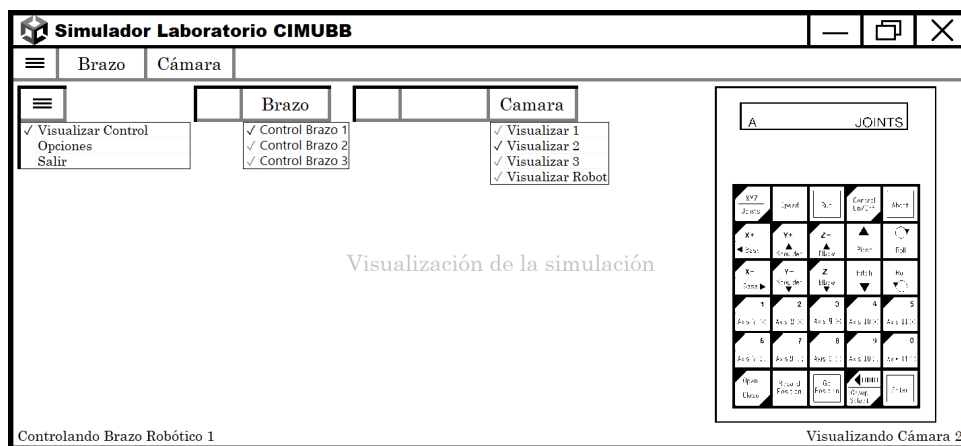


Figura 8.1: Diseño interfaz: Módulo Principal

Las distintas opciones se detallaran a continuación

- Menú
 - Visualizar Control: Mostrar u ocultar el control flotante
 - Opciones: Despliega las opciones de programa
- Brazo: Selecciona el brazo robótico a utilizar
- Cámara: Selecciona la cámara a visualizar

Capítulo 9

Código

En este capítulo, se presentan explicaciones sobre el código fuente de la aplicación desarrollada.

9.1. Lógica Cinta Transportadora

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Belt : MonoBehaviour
{
    public float speed = 5f;
    Rigidbody rBody;
    public Vector3 direccion;

    void Start()
    {
        rBody = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        Vector3 pos = rBody.position;
        rBody.position += direccion * speed * Time.fixedDeltaTime;
        rBody.MovePosition(pos);
    }
}
```

Este código proporciona la lógica simple de movimiento de objetos en la cinta transportadora, sin embargo este no llega a replicar su funcionamiento real debido a limitaciones que se explican en los detalles siguientes:

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
```

Estas líneas son declaraciones de uso de espacio de nombres (namespace) que indican qué bibliotecas se están utilizando en el código. Están importando las bibliotecas necesarias para trabajar con Unity y otros espacios de nombres estándar de C#.

```
public class Belt : MonoBehaviour
{
    ...
}
```

Aquí comienza la definición de la clase Belt, que hereda de la clase MonoBehaviour. Esto significa que este script puede ser adjuntado a objetos en Unity y ejecutado como parte del comportamiento de esos objetos.

```
public float speed =5f;
Rigidbody rBody;
public Vector3 direccion;
```

En esta parte se declaran las variables, se parte con una variable pública llamada speed (velocidad) que tiene un valor predeterminado de 5. Esta variable representa la velocidad a la que se moverá la cinta transportadora. Después se declara una variable llamada rBody de tipo Rigidbody. Esta variable se utilizará para almacenar una referencia al componente Rigidbody adjunto al objeto al que se adjunte este script. Un Rigidbody es un componente utilizado en el desarrollo de videojuegos y simulaciones físicas para representar objetos que tienen propiedades físicas, como masa, velocidad, y fuerzas que actúan sobre ellos. Por último se declara una variable pública llamada direccion (dirección) de tipo Vector3. Vector3 es una estructura de datos utilizada para representar vectores tridimensionales en un espacio tridimensional. Esta variable se utiliza para definir la dirección en la que se moverá la cinta transportadora. La dirección se establece desde el Inspector de Unity cuando adjuntes este script a un objeto en el juego.

```
void Start()
{
    rBody = GetComponent<Rigidbody>();
}
```

El método Start() es uno de los métodos especiales en Unity que se llama automáticamente cuando se inicia un objeto al que se adjunta un script, en este caso se obtiene una referencia al componente Rigidbody del objeto al que se adjunta este script. Esto se hace utilizando la función GetComponent<Rigidbody>(), y la referencia se almacena en la variable rBody.


```
void FixedUpdate()  
{  
    ...  
}
```

FixedUpdate es un método especial que se encuentra en Unity y es utilizado para realizar cálculos y actualizaciones relacionadas con la física en un juego. A diferencia de Update, que se llama una vez por cada fotograma (frame) renderizado, FixedUpdate se llama en intervalos de tiempo fijos y regulares, lo que lo hace especialmente adecuado para manejar la simulación de física y movimiento en un juego.

```
Vector3 pos = rBody.position;
```

Aquí se crea una variable local llamada pos que almacena la posición actual del objeto asociado al Rigidbody.

```
rBody.position += direccion * speed * Time.fixedDeltaTime;
```

Esta línea actualiza la posición del objeto con el componente Rigidbody (rBody) al agregarle un desplazamiento basado en la dirección (direccion), la velocidad (speed), y el tiempo transcurrido (Time.fixedDeltaTime). Esto aun no simula el movimiento de la cinta transportadora y se explica con las siguientes figuras.

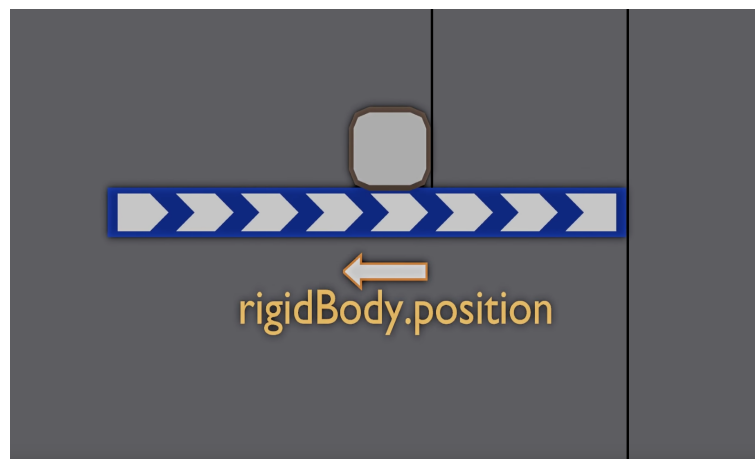


Figura 9.1: rigidBody Position 1

En la Figura 9.1 se presenta el movimiento de la cinta, este movimiento solo aplica para la cinta, a diferencia del método MovePosition que se presenta más adelante.

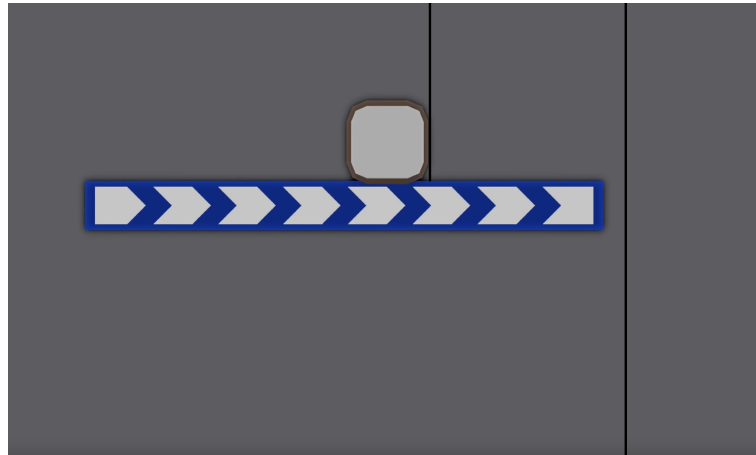


Figura 9.2: rigidBody Position 2

En la Figura 9.2 se muestra donde finaliza el movimiento, como se aprecia este solo mueve la cinta y el objeto sobre esta no posee movimiento

```
rBody.MovePosition(pos);
```

En este método, se devuelve la cinta a su posición original, la diferencia que con el método de movimiento anterior es que si se mueve el objeto, tal como se presenta en las siguientes figuras.

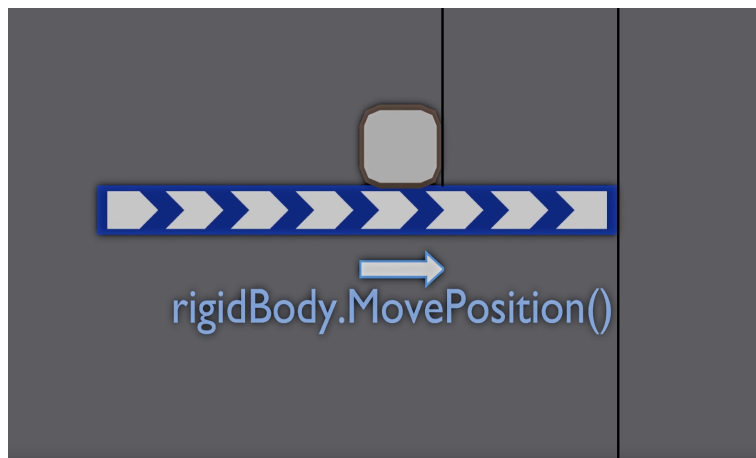


Figura 9.3: rigidBody MovePosition 1

En la Figura 9.3 se presenta el movimiento de la cinta, este movimiento aplica tanto para la cinta como para el objeto.

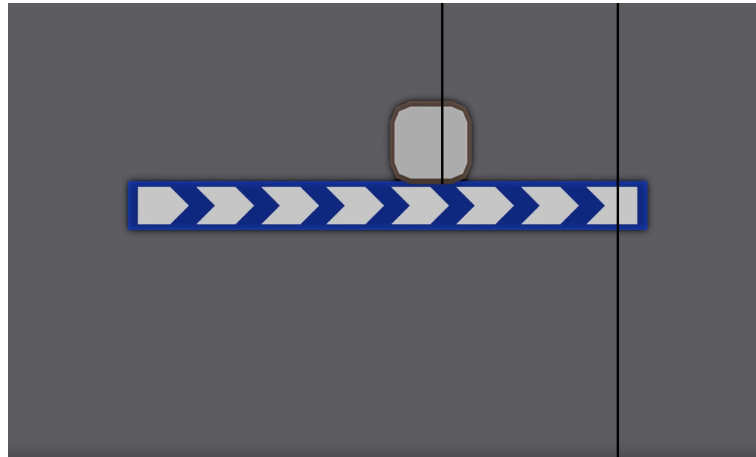


Figura 9.4: rigidBody MovePosition 2

En la Figura 9.4 se muestra donde finaliza el movimiento, como se aprecia este mueve tanto la cinta y como el objeto.

Con esta sucesión de movimientos, la cinta al ir para atrás "sin mover el objeto, y después "volver a su posición" moviendo el objeto, repitiéndolo una y otra vez se logra el efecto de una cinta transportadora. El problema es que el objeto solo se mueve y en las esquinas no rota como en el laboratorio.

9.2. Lógica Brazos Roboticos

Para esta lógica, se explicara los detalles mas importantes debido a que el código es demasiado extenso.

```
//Base
public float velocidadRotacionBase = 50.0f; // Velocidad de rotacion
public KeyCode teclaRotacionPositivaBase = KeyCode.Q; // Tecla para la ←
    rotacion en direccion positiva
public KeyCode teclaRotacionNegativaBase = KeyCode.A; // Tecla para la ←
    rotacion en direccion negativa
public Transform Base; // Transform del objeto que se va a rotar
private float LimitePositivoBase = 155.0f; // Angulo limite del objeto
private float LimiteNegativoBase = -155.0f; // Angulo limite del objeto
private float sumaRotacionBase = 0.0f; // Suma de la rotacion del objeto ←
    para su uso en limites

//Shoulder
public float velocidadRotacionShoulder = 50.0f; // Velocidad de rotacion
public KeyCode teclaRotacionPositivaShoulder = KeyCode.W; // Tecla para la ←
    rotacion en direccion positiva
public KeyCode teclaRotacionNegativaShoulder = KeyCode.S; // Tecla para la ←
    rotacion en direccion negativa
public Transform Shoulder; // Transform del objeto a rotar
public Transform puntoFijoShoulder; // Transform del punto fijo alrededor ←
    del cual se realizara la rotacion
private float LimitePositivoShoulder = 10.0f; // Angulo limite del objeto
private float LimiteNegativoShoulder = -130.0f; // Angulo limite del objeto
private float sumaRotacionShoulder = 0.0f; // Suma de la rotacion del objeto ←
    para su uso en limites
```

Para explicar las declaraciones de las partes de los brazos, se toma en cuenta dos partes distintas pero aplica para cualquier otra parte similar, la base es un objeto que tiene rotación sobre su propio eje, es decir no necesita otro punto para rotar o rota sobre su propio centro, a diferencia del Shoulder o un brazo cualquiera, para rotar este tiene un punto fijo en uno de sus extremos y no en el centro. Se empieza con declarar la velocidad de la rotación del objeto en una variable float. Después se declaran variables tipo KeyCode que sirven para almacenar la información de una tecla en el teclado en particular, esto se realiza para realizar pruebas de movimiento. Al declarar una variable de tipo Transform, en Unity, un "Transform" se refiere a un componente fundamental que se encuentra en la mayoría de los objetos en un escenario 3D o 2D. El componente Transform está asociado con la posición, rotación y escala de un objeto en el espacio de juego. Básicamente, controla la ubicación y la orientación de un objeto en el mundo virtual creado en Unity. Con esta variable se puede guardar toda la información previamente descrita, con la finalidad de realizar movimientos. En casos de rotación sobre el mismo objeto como la Base, solo se pide una variable, para objetos con centro en su extremo se piden dos variables, mas adelante se detallara como funciona el movimiento y el motivo de pedir uno o dos variables. Y por ultimo se tienen los limites, consisten en dos variables que marcan el limite de operación de cada parte, para

este caso se utilizan los limites establecidos mecánicamente por la misma unidad descrita en su manual. Y también la suma de rotación, la cual almacenara como tal el movimiento realizado, esto para ser utilizado en la comprobación de limites

```
private string nombreObjeto = "";
```

En esta linea se declara una variable para identificar la parte que se moverá con la botonera.

```
private bool positivoboton = false;
private bool negativoboton = false;
```

Con estas variables sirven para saber si el botón presionado es positivo o negativo

```
private bool isControlPressed = false;
private bool isAltPressed = false;
```

Para estas lineas se utilizo para realizar pruebas para guardar posiciones y realizar el movimiento a esa posición, estas servían para saber si Control fue presionado o el Alt fue presionado

```
private bool EmpezarGuardar = false;
private bool TerminarGuardar = false;
private bool EmpezarMover = false;
private bool TerminarMover = false;
```

Acá los booleanos son para lo descrito anteriormente, sirven para marcar los inicios y finales de guardar posiciones y mover a las posiciones

```
private int NumeroUsar = -1;
```

Este numero es el numero usado en la botonera, en el cual se guardara la información.

```
private Dictionary<int, float> GuardarBase = new Dictionary<int, float>();
private Dictionary<int, float> GuardarShoulder = new Dictionary<int, float>();
private Dictionary<int, float> GuardarElbow = new Dictionary<int, float>();
private Dictionary<int, float> GuardarWrist = new Dictionary<int, float>();
private Dictionary<int, float> GuardarEndEffector = new Dictionary<int, float>();
```

Estas variables son diccionarios, los cuales con una variable devuelve una segunda variable, con esto poder guardar por ejemplo, la posición dependiendo del numero requerido, función que ocupa la botonera. Esta declarada para que con un numero entero devuelva un flotante.

```
private void Start()
```

```

{
    GuardarBase.Add(0, 0f);
    GuardarShoulder.Add(0,0f);
    GuardarElbow.Add(0,0f);
    GuardarWrist.Add(0,0f);
    GuardarEndEffector.Add(0,0f);
}

```

Al ejecutar el programa, en su inicio ejecutara esta secuencia de funciones, las cuales son para agregar la posición cero a los diccionarios.

```

private void Update()
{
    ...
}

```

```

private void Update()
{
    Movimiento(Base, teclaRotacionPositivaBase, teclaRotacionNegativaBase, ←
        velocidadRotacionBase, LimitePositivoBase, LimiteNegativoBase, Base, ←
        Base.up, ref sumaRotacionBase);
    Movimiento(Shoulder, teclaRotacionPositivaShoulder, ←
        teclaRotacionNegativaShoulder, velocidadRotacionShoulder, ←
        LimitePositivoShoulder, LimiteNegativoShoulder, puntoFijoShoulder, ←
        puntoFijoShoulder.forward, ref sumaRotacionShoulder);
    Movimiento(Elbow, teclaRotacionPositivaElbow, teclaRotacionNegativaElbow ←
        , velocidadRotacionElbow, LimitePositivoElbow, LimiteNegativoElbow, ←
        puntoFijoElbow, puntoFijoElbow.forward, ref sumaRotacionElbow);
    Movimiento(Wrist, teclaRotacionPositivaWrist, teclaRotacionNegativaWrist ←
        , velocidadRotacionWrist, LimitePositivoWrist, LimiteNegativoWrist, ←
        puntoFijoWrist, Wrist.forward, ref sumaRotacionWrist);
    Movimiento(EndEffector, teclaRotacionPositivaEndEffector, ←
        teclaRotacionNegativaEndEffector, velocidadRotacionEndEffector, ←
        LimitePositivoEndEffector, LimiteNegativoEndEffector, ←
        puntoFijoEndEffector, EndEffector.right, ref sumaRotacionEndEffector ←
        );
    MovimientoBoton(nombreObjeto);
    Guardar();
    Usar();
    GuardarBoton();
    UsarBoton();
    MoverEje();
    MoverEjeBoton();
}

```

```

rBody.MovePosition(pos);

```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

```
rBody.MovePosition(pos);
```

9.3. Diseño interfaz y navegación

Capítulo 10

Pruebas

Este capítulo se enfoca en las diferentes pruebas del software desarrollado. Se destaca su importancia para garantizar la calidad y funcionalidad del producto final, asegurando que cumpla con los requisitos establecidos.

- 10.1. Elementos de prueba
- 10.2. Especificación de las pruebas
- 10.3. Responsables de las pruebas
- 10.4. Detalle de las pruebas
- 10.5. Conclusiones de prueba

Capítulo 11

Plan de capacitación y entrenamiento

Este capítulo presenta el plan para capacitar y entrenar al equipo del proyecto. El objetivo es brindar las habilidades y conocimientos necesarios para lograr un desarrollo exitoso del software.

11.1. Plan de capacitación

Capítulo 12

Plan de implantación y puesta en marcha

En este capítulo, se detalla el plan para implementar y poner en funcionamiento el software del brazo robótico en el laboratorio.

12.1. Plan de implantación

Capítulo 13

Trabajo Futuro

En este capítulo, se presentan los cambios que se pueden generar para tener un programa mas fiel en su representación.

13.1. Cambios que se deben realizar

- Mejorar modelos: Uno de los cambios mas fundamentales que se debe realizar para mejorar la experiencia e intentar lograr ser lo mas fiel posible en la representación del laboratorio. Se puede empezar por realizar un modelado fiel y a escala real de los objetos presentes en el laboratorio, para esto se presenta el programa que provee Intelitek para su brazo Scorbot ER-4U. En la Figura 13.1 se presenta la interfaz del programa RoboCell, el cual sirve para darle instrucciones al brazo robotico, ya sea real o simulado, en este ultimo se ve el detallado del modelo que da mejor sensación de realismo.

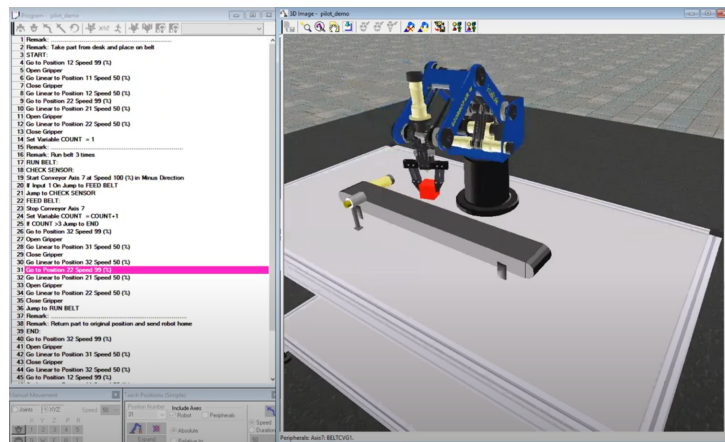


Figura 13.1: Diseño interfaz: Módulo Principal

- Opciones: Despliega las opciones de programa

Capítulo 14

Conclusiones

A lo largo de este proyecto, he podido comprender la importancia fundamental de mejorar la educación a través de la simulación [3]. Aunque la simulación no puede replicar completamente la experiencia real, he observado que puede acercarse lo suficiente como para tener un impacto significativo en el proceso de aprendizaje. La capacidad de crear situaciones virtualmente similares a la realidad resulta especialmente relevante para eliminar barreras educativas, proporcionando alternativas accesibles y asequibles para personas con diversas limitaciones, como problemas de movilidad, foto-sensibilidad, discapacidad y otros obstáculos.

El enfoque de la simulación va más allá de simplemente recrear situaciones en un entorno virtual. Su verdadero potencial radica en la creación de un ambiente inclusivo donde todos puedan acceder a oportunidades educativas de manera efectiva, independientemente de sus capacidades físicas o cognitivas. Al adoptar esta herramienta en el ámbito educativo, se abren puertas para aquellas personas que, de otro modo, enfrentarían dificultades para participar en experiencias de aprendizaje costosas o restringidas.

En resumen, la simulación en la educación tiene el poder de promover una sociedad más equitativa e inclusiva, permitiendo que cada individuo alcance su máximo potencial. Aunque es un campo en desarrollo, espero que esta tesis inspire a educadores, instituciones académicas y desarrolladores tecnológicos a utilizar la simulación como una herramienta transformadora en la búsqueda de una educación para todos, sin barreras ni limitaciones. Al hacerlo, construiremos un futuro en el que la adquisición de conocimientos y habilidades sea una posibilidad accesible para cada persona, contribuyendo así al avance y prosperidad de la sociedad en su conjunto.

Referencias

- [1] G. Salas, P. Santander, A. Precht, H. Scholten, R. Moretti, and W. López-López, “Covid-19: impacto psicosocial en la escuela en Chile. desigualdades y desafíos para Latinoamérica,” *Avances En Psicología Latinoamericana*, vol. 38, no. 2, 2020.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*. Boston, MA, USA: Addison-Wesley Professional, 2000.
- [3] Z. Cataldi, F. J. Lage, and C. Dominighini, “Fundamentos para el uso de simulaciones en la enseñanza,” *Revista de informática educativa y medios audiovisuales*, vol. 10, no. 17, pp. 8–16, 2013.