ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA MECÁNICA

"DESARROLLO DE UN ENTORNO DE REALIDAD VIRTUAL 3D CON CAPACIDAD DE INMERSIÓN QUE EMULE EL LANZAMIENTO Y RECEPCIÓN DE UN RASPADOR DE TUBERÍA (PIG), ORIENTADO AL ENTRENAMIENTO DENTRO DE LA INDUSTRIA PETROLERA"

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN SISTEMAS DE TRANSPORTE DE PETRÓLEO Y DERIVADOS

JOSÉ LUIS AMAQUIÑA CISNEROS
quayjl1@hotmail.com

DIRECTOR: DR. VÍCTOR HUGO ANDALUZ ORTIZ
vhandaluz1@espe.edu.ec

CODIRECTORA: DRA. MARÍA GABRIELA PÉREZ HERNÁNDEZ
maria.perez@epn.edu.ec

Quito, Marzo 2018

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por el señor José Luis Amaquiña Cisneros, bajo mi supervisión.

Dr. Víctor Hugo Andaluz O.
DIRECTOR DEL PROYECTO

Dra. María Gabriela Pérez H.
CODIRECTORA DEL PROYECTO

DECLARACIÓN

Yo, José Luis Amaquiña Cisneros, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

José Luis Amaquiña Cisneros

AGRADECIMIENTO

A Víctor Andaluz, por tu valioso aporte y guía desde el comienzo hasta el final del trabajo, de seguro no lo hubiese logrado sin tu incondicional ayuda.

A la Dra. María Pérez por su apoyo, esfuerzo y sus consejos.

A mi amada Gaby por su amor infinito, por su soporte para la ejecución de este trabajo, por los buenos y los malos momentos.

A todos quienes formaron parte de este logro: familia, amigos, compañeros.

José Luis.

DEDICATORIA

A mis padres que amo por sobre todas las cosas, con la satisfacción del deber cumplido. ¡Han hecho un buen trabajo!

A la memoria de mis abuelitos.

José Luis

ÍNDICE

.

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ABREVIATURAS

ACIS	Modelador geométrico tridimensional de objetos.
Android	Sistema operativo basado en Linux.
API	Instituto Americano del Petróleo.
AOI	Industria Americana del Petróleo.
AutoCad	Software de diseño asistido por computador para gráficos 2D/3D.
BBL	Barril
BBLS	Barriles.
BMP	Formato de imagen de mapa bits.
BSD	Sistema Operativo derivado de UNIX. Distribución de software Berkeley.
CAD	Diseño asistido por computador.
CAVE	Entorno de RV inmersivo multipersona.
CHAR	Tipo de datos carácter.
CRT	Representación visual por haz de rayos catódicos.
DIN	Organismo Nacional de Estandarización de Alemania
DOUBLE	Tipo de datos dobles.
DWG	Formato de archivo informático de dibujo computarizado.
EP	Empresa Pública
f	Factor de fricción.
FLOAT	Tipo de datos flotantes.
GIF	Formato de intercambio de imágenes.
HCD	tipo de casco estereoscópico inmersivo.
HMD	casco de realidad virtual
HTC	Tech Computer Corporation, fabricante de tecnología.
I/O	Input/Output – Entrada/Salida
I^3	Componente de RV: Inmersión, Interacción e Imaginación.
INT	Tipo de datos enteros.
iOS	Sistema operativo de Apple Inc.
IPC	Forma implícita de comunicación entre procesos.
IR	Infrarrojo
ISO	Organización Internacional de Normalización.
JIS	Estándar Industrial Japonés
JPG/JPEG	Algoritmo de compresión gráfica.
LCD	Representación visual por cristal líquido
LNG	Gas Natural Licuado.
MATLAB	Matrix Laboratory, herramienta de software matemático
MP4/MP3	Formato de compresión digital de audio y video.
msnm	Metros sobre el nivel del mar.
MUVE	Entornos virtuales multiusuario.
P	Presión.
Pa	Unidad de presión en Pascales.
PC	Computador Personal
PDA	Agenda electrónica de bolsillo.

PIG	*Pipeline Inspection Gauges*. Dispositivo empleado para limpieza de tuberías.
PIP	Estándares de Prácticas en Procesos Industriales.
PNG	Formato gráfico para compresión gráfica.
POSIX	Interfaz de Sistema Operativo Portable.
PSIG	Presión relativa a la del ambiente.
P&ID	Diagramación de tuberías e instrumentos.
Q	Flujo
RV	Realidad Virtual
RA	Realidad Aumentada
SOTE	Sistema de Oleoducto Transecuatoriano
UNITY	Motor de videojuego multiplataforma creado por Unity Technologies.
WoW	Visualización de imágenes en 2D o 3D a través de un monitor.
2D	Gráficos 2D, modelos geométricos, texto e imágenes digitales 2D.
3D	Gráficos 3D, geometría y análisis tridimensional.
4D	Cuarta dimensión o espacio euclídeos de más de tres dimensiones.
.FBX	Formato de archivos 3D
ρ	Densidad
ϕ	Diámetro.

RESUMEN

En el presente trabajo se desarrolló una plataforma de realidad virtual 3D con capacidad de inmersión que emula el ambiente y las operaciones de las trampas de lanzamiento y recepción de raspadores de tubería *pipeline inspection gauges* (PIG), con el objetivo de reforzar la capacitación de los operadores de la industria petrolera. Para ello, en primer lugar, se revisó y recopiló la información existente sobre varias trampas lanzadoras y receptoras de las distintas empresas operadoras de ductos del país, logrando definir los parámetros básicos y específicos para la recreación virtual de un modelo típico de trampas. La información recopilada constó de fotografías, P&IDs, procedimientos, especificaciones técnicas, manuales de equipos, etc. A partir de esta información se generaron planos y modelos digitales 3D (archivos CAD), que con ayuda del motor gráfico *Unity 3D* se consiguió un mayor nivel de detalle en la virtualización de todos los elementos involucrados en el proceso, y un alto grado de inmersión dentro del entorno virtual. Esta herramienta permite la integración de dispositivos externos (gafas, controladores y estaciones de rastreo HTC VIVE) con la plataforma, generando una experiencia de alta calidad en cuanto al desarrollo de actividades, familiarización del entorno y capacitación de personal de una manera virtual. Finalmente, para lograr una interacción más realista con la plataforma virtual, se implementaron algoritmos matemáticos y lógicos en Matlab, los cuales se integraron a la aplicación 3D a través de una herramienta de memoria compartida. Al integrar la aplicación con los modelos de Matlab, se consiguió que la interacción entre el usuario y el ambiente virtual se asemeje a un proceso real. Dentro de la plataforma, el operador puede ver el estado del proceso y los cambios que sus acciones generen sobre éste a través de objetos e instrumentos virtuales generados en el entorno 3D.

Palabras clave: Realidad Virtual, Raspador (PIG), entrenamiento

ABSTRACT

This work proposes the development of a 3D virtual reality platform with immersion capability that emulates the launching and receiving traps of a pipeline inspection gauge (PIG), this project aims to reinforce the training methods for oil industry operators.

First, the data was collected on various launching and receiving traps of different pipeline operating companies in Ecuador; this data that would allow a way to define the basic and specific parameters for the virtual re-creation of a traditional trap model. The information collected consists of photographs, P&IDs, procedures, technical specifications, equipment manuals, etc. The information gathered was used to generate plans and 3D digital models (CAD files), and with the help of Unity 3D graphic engine, it was possible to achieve greater detail on the virtualization of the elements that were involved in the process as well as a higher degree of immersion within the virtual environment. This tool allows the integration of external devices with the platform (goggles, controllers, and base stations HTC VIVE), generating a high quality experience in the development of activities, familiarization of the environment, and virtual operator training. To achieve the most realistic interaction with the virtual platform, mathematical and logical algorithms were integrated using Matlab software. A shared memory tool integrates the algorithms with the 3D application, allowing a live response every time there is a change in real-life operating conditions. Platform users can monitor their own decisions and the general progress of the simulation using virtual instruments generated in the 3D environment.

Key words: Virtual reality, pipeline inspection gauge (PIG), training.

INTRODUCCIÓN

Actualmente las operaciones industriales requieren cumplir con nuevos y más exigentes desafíos: eficiencia, seguridad, y entornos amigables con el medio ambiente, lo que ha obligado a las compañías a mejorar las prácticas y estándares de capacitación para su personal operativo. Esta capacitación requiere una planificación precisa para asegurar interrupciones mínimas en la producción (Meza, 2004).

Dentro de las industrias importantes del país y que requieren mayor inversión en facilidades, equipos, talento humano, y capacitación está la industria petrolera en todas sus etapas: exploración y producción; transporte, procesos y almacenamiento; refinación, venta y distribución, siendo el transporte de crudo una de las más críticas en nuestro país.

Las operaciones de transporte de petróleo son continuas durante casi todo el año, las paradas de planta para mantenimiento deben ser programadas y muy bien coordinadas, esto hace que el tiempo destinado para la capacitación con equipos e instalaciones reales se reduzca a unas cuantas semanas al año (Daglas & Coleman, 2012). Este tipo de capacitaciones se vuelven complicadas al momento de querer entrenar personal nuevo o subcontratistas en cualquier época del año, además de que al utilizar las instalaciones en sitio para adiestrar al personal se incrementa el riesgo de accidentes, daños de equipos, paros de planta y posible afectación ambiental.

De acuerdo al Informe Estadístico (EP Petroecuador, 2016, págs. 15-16) el Sistema de Oleoducto Transecuatoriano SOTE paralizó sus operaciones 32 horas y 50 minutos en el año, de las cuales 19 horas fueron como parte del Mantenimiento programado, y las otras por incidentes varios no programados. Este tiempo resulta ínfimo para realizar una capacitación adecuada y con las propias facilidades al personal operativo.

La industria petrolera maneja como política propia distintos programas de capacitación teórico-prácticos, que están siendo orientados a mejorar los procesos de la cadena de producción de hidrocarburos, invirtiendo recursos humanos y económicos para lograr cumplir con estos nuevos desafíos y exigencias. Empero, en los últimos 3 años la industria ha tenido que enfrentar notables adversidades (Pallares, 2016):

a) Disminución del precio del barril de crudo desde la segunda mitad de 2014 (USD 110/barril a +-USD 40/ barril) que se debe a las siguientes causas: exceso de oferta vinculado al crecimiento de la producción no convencional en los Estados Unidos; negativa de grandes productores de reducir el volumen de producción con el objeto de no perder cuotas de mercado; cambios geopolíticos (como el acuerdo con Irán); etc.

b) Amenaza de cambios vinculados al cuidado medioambiental, que no solo se relacionan a mayores costos por endurecimiento regulatorio sino a competencia de otras fuentes de energía renovables.

Para sobrevivir a esta nueva realidad, las empresas deben recurrir a nuevas estrategias, que les permitan ser más competitivas, reducir sus costos y satisfacer las exigencias de un mayor cuidado del medio ambiente.

En cuanto a la capacitación, las empresas han generado alternativas virtuales para sus trabajadores y subcontratistas que les permite acceder a información, inducciones, cursos en línea y certificaciones técnicas utilizando plataformas en red que interactúan directamente con el usuario previamente registrado y muestran su contenido a través de cualquier interfaz tecnológica (computadora, teléfono inteligente, etc.).

Estas prácticas clásicas, si bien han optimizado la manera tradicional de transmitir conocimientos teóricos, no han logrado desarrollar los conocimientos prácticos que la industria requiere. Consolidar las buenas prácticas operativas en los trabajadores demanda mayor planificación, tiempo, inversión económica y seguridad.

Por estas razones se ha desarrollado un módulo de entrenamiento virtual 3D con capacidad de inmersión para preparar de manera complementaria a los operadores y usuarios de las trampas lanzadoras y receptoras dentro de la industria del transporte de hidrocarburos.

Objetivo General

Desarrollar un entorno de realidad virtual 3D con capacidad de inmersión que emule las operaciones de lanzamiento y recepción de un raspador de tubería (PIG) y que esté orientada al entrenamiento dentro de la industria petrolera.

Objetivos Específicos

Definir la secuencia de operaciones para un correcto y normal lanzamiento y recepción de raspadores, considerando el conjunto de válvulas e instrumentos involucrados en cada una de las trampas lanzadora y receptora.

Investigar las fallas, errores comunes, variables y valores a condiciones normales de la secuencia de operación “lanzamiento y recepción” de un raspador de tubería, con el propósito de desarrollar un simulador para entrenamiento virtual.

Desarrollar a través de diagramas P&IDs un entorno virtual inmersivo de un proceso industrial petrolero utilizando como motor gráfico *Unity 3D* considerando las características del proceso de “lanzamiento y recepción” de un raspador de tubería.

Implementar en el entorno virtual información relevante del funcionamiento del proceso, alertas y símbolos de prevención, casos de emergencia que permita entender de manera vivencial los parámetros, riesgos y secuencias involucradas en el lanzamiento y recepción de raspadores.

Identificar puntos de riesgo en el proceso industrial para evaluar el desempeño del operador ante eventos de emergencia en el proceso.

Realizar pruebas de funcionamiento de la aplicación desarrollada, a fin de evaluar el desempeño, grado de inmersión entre el operador y el proceso virtualizado, y método de aprendizaje mediante la plataforma 3D.

1. MARCO TEÓRICO

1.1. Realidad Virtual

El interés del hombre por interpretar la realidad data del siglo XX, realidad que ha sido plasmada y expresada de alguna manera a través de pinturas prehistóricas, libros, películas, etc. En la actualidad se cuenta con herramientas más sofisticadas que junto con el avance computacional, han hecho posible su utilización para realizar las representaciones gráficas necesarias en muchos contextos, convirtiéndose en una herramienta esencial que puede refinar y extender las habilidades de profesionales, como los cirujanos, ingenieros, químicos e incluso psicoterapeutas, así como para favorecer la competitividad en el campo de las empresas y en la educación. En este trabajo se propone usarlo para reforzar la capacitación de los operadores de la industria petrolera.

Aunque el término realidad virtual (RV) es conocido en la actualidad, no se tiene una definición concreta, sobre el mismo. Algunos autores como (Manetta & Blade, 1995) la definen como *"Un sistema de computación usado para crear un mundo artificial en el cual el usuario tiene la impresión de estar y la habilidad de navegar y manipular objetos en él".* Para (Burdea & Coiffet, 1996) es *"como un mundo que a pesar de no tener ninguna realidad física es capaz de darle al usuario, a través de una estimulación adecuada de su sistema sensorial, la impresión perfecta de estar en interacción con un mundo físico"*.

No obstante, se considera que el concepto de 'Realidad Virtual' nació en el año 1965, de la mano de Ivan Sutherland, con su trabajo *"The Ultimate Display"* en el que explicaba el concepto que los científicos llevaban años tratando de acuñar. Dos años más tarde, desarrolló el primer programa diseñado para crear mundos virtuales con imágenes en 3D (Sutherland, 1965). Desde el punto de vista de su función se cita la definición de (Burdea, 1993) que señala *"un sistema de realidad virtual es una interfaz que implica simulación en tiempo real e interacciones mediante múltiples canales sensoriales. Estos canales sensoriales son los sentidos del ser humano: la vista, el oído, el tacto, el olfato y el gusto".*

Por otro lado, es muy común que el término Realidad Virtual se le asocie con Realidad Aumenta (RA), asumiendo un concepto de igualdad, pero son tecnologías muy distintas que diferencian la una de la otra, pues la Realidad Aumentada se dice que es *"aquel sistema que consiste en aumentar la percepción que el usuario tiene de la realidad*

mediante la implementación de elementos virtuales en la misma"; en otras palabras como lo define (Azuma, 1997) *"es un entorno que incluye elementos de Realidad Virtual y elementos del mundo real"*, y se diferencia porque la RV se fundamenta en generación de estímulos por computador sin superponerlos al entorno físico.

Por consiguiente, la Realidad Virtual se considera como una tecnología que está en constante evolución, y que su carácter inmersivo en ambientes participativos artificiales abarca áreas como: simulación por computador, ambiente tridimensional, gráficos, tacto y sonidos; a través del cual, el usuario puede interactuar, convirtiéndolo en una ciencia muy extensa y prometedora que motiva a los investigadores a continuar en el desarrollo de la RV. Además, la RV puede aplicarse en varios ámbitos permitiendo de una u otra manera mejorar la calidad de vida y la manera de percibir y potenciar la imaginación y dar mayor libertad a la mente.

1.1.1. Elementos de la Realidad Virtual

La Realidad virtual considera tres elementos fundamentales, para su representación, los cuales se muestran en la Figura 1.1.

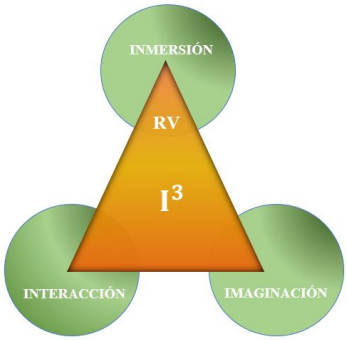


Figura 1.1. Triángulo de los elementos de la Realidad Virtual.
Fuente: Virtual Reality Techonogy (Burdea G, 2003)

a) Inmersión: es clave dentro de los sistemas de RV, para lograr que el usuario pase a ser parte del mundo simulado, en lugar de que el mundo simulado sea una característica del mundo propio del usuario. Éste elemento posee dos propiedades importantes i) habilidad para enfocar la atención del usuario y ii) convertir una base de datos en experiencias, estimulando de esta manera el sistema natural de aprendizaje humano (las experiencias personales). Para

conseguir que el usuario crea que está formando parte de la experiencia virtual, es importante que esté totalmente aislado del entorno físico que le rodea (Kalawsky, 1993). El grado de inmersión dependerá más del grado de atención que el usuario ponga a aquello que se le está presentando, que del sistema utilizado para presentarlo

b) **Interacción:** denominado también como interactividad. La interacción de esta interfaz de usuario es de gama alta, involucra múltiples canales sensoriales para realizar acciones en el sistema que vayan modificando y que el usuario obtenga respuesta a través de sus sentidos, visual, auditivo, tacto, olfato y gusto. La mayor parte de las simulaciones de RV usan modalidades visuales (pantalla estéreo 3-D) y auditivas (interacción o sonidos 3-D), la realimentación háptico (tacto) está empezando a tener reconocimiento y uso en aplicaciones de manipulación intensiva; mientras que, el olfato y el gusto están en la etapa inicial de las investigaciones.

c) **Imaginación:** o presencia, relaciona a la capacidad mental que el usuario tiene para percibir cosas que no existen y crear la ilusión, para que pueda interactuar dentro de la RV, para lo cual emplea diferentes dispositivos de entrada (sensores de movimiento, dimensionamiento, guantes, etc.) (Burdea G, 2003).

1.1.2. Arquitectura de la Realidad Virtual

Los componentes que conforman la arquitectura de los Sistemas de RV, se muestran en la Figura 1.2. (Burdea G, 2003).

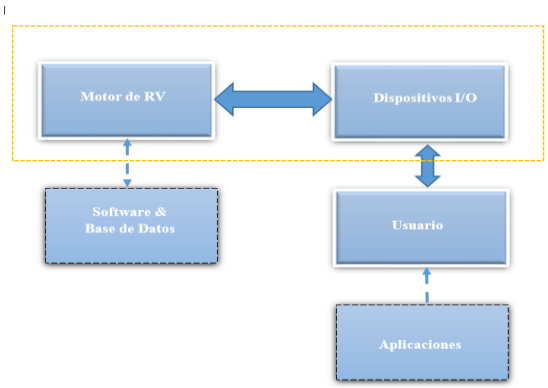


Figura 1.2. Arquitectura de la Realidad Virtual.
Fuente: Virtual Reality Techonogy (Burdea G, 2003)

a) **Dispositivos de salida:** realimentan los dispositivos sensoriales del usuario y proporcionan información del mundo virtual con respuestas visuales, auditivas, táctiles, por ejemplo, Pantalla Montada en la Cabeza (*HMD*, por sus siglas en inglés), guantes de datos o *dataglobes*, etc. (Burdea G, 2003)

b) **Dispositivos de entrada:** realizan un seguimiento del usuario y su interacción con el entorno virtual. Para ello, en el mercado existen dispositivos como guantes de datos o *dataglobes*, ratones 3D, *trakers*, plataformas, etc. (Burdea G, 2003)

c) **Motor de realidad:** se encarga de albergar el software que creará la realidad virtual; se lo puede considerar como un cliente- servidor, donde el *servidor* permitirá la sincronización de las imágenes, el control de colisión de las imágenes, comunicación con los clientes que visualizan el mundo virtual, etc.; mientras que el *cliente* contendrá el mundo virtual, se comunicará con el servidor, cada cliente podrá tener perspectivas distintas del mundo, mientras que el servidor enviará órdenes para que los clientes actúen en consecuencia. (Burdea G, 2003)

d) **Software de Realidad Virtual:** se emplea como software de modelado de objetos virtuales, que involucran características del mundo virtual como geometría, modelado de textura, física de los objetos, dureza, inercia, plasticidad de superficie, etc. Incluye también *drivers* de entrada/salida (I/O), lenguajes de programación, librerías y demás sistemas que se emplean para la implementación de las interfaces del mundo virtual. (Burdea G, 2003)

e) **Base de datos del mundo:** bases que contendrán los objetos del mundo virtual y sus propiedades. (Burdea G, 2003)

1.1.3. Clasificación de la Realidad Virtual

La realidad virtual se puede clasificar en:

Sistema RV de escritorio: conocido también como *Window on World* (WoW), se basa en la visualización de imágenes en 2D o 3D a través de un monitor, “casco” o pantalla de proyección. En ésta sección se encuentran la mayoría de videojuegos para PCs o consolas de los hogares, cuya característica es que el usuario ve la imagen en primera persona sin la necesidad de emplear equipos especiales o específicos para RV.

Figura 1.3. Videojuego.
Fuente: Pria Dan Games (Rizkyyudhis, 2013)

Cabina de simulación: son similares a los sistemas de escritorio, cuya diferencia radica en que el usuario experimenta el mundo virtual dentro de una cabina. Un ejemplo de este tipo son los simuladores de vuelo que se emplean en el entrenamiento de pilotos de aviones.

Figura 1.4. Cabina de simulación de vuelos.
Fuente: Simulador para el entrenamiento de tripulantes de cabina del Superjet 100 (aerotendencias.com, 2010)

Realidad aumentada: el usuario experimenta un mundo real a través de cristales que complementan las imágenes con diagramas, textos o referencias. Un ejemplo de este tipo es el juego pokemon go.

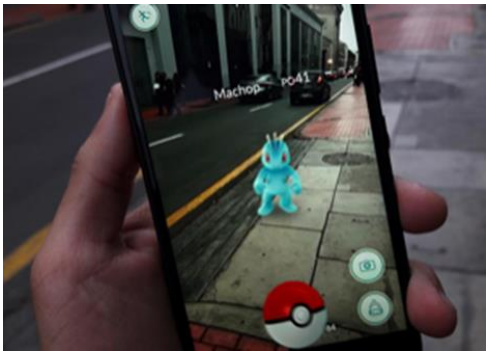


Figura 1.5. Juego Pokemon Go en realidad aumentada.
Fuente: Diario WEB (Perú.com, 2016)

Realidad en segunda persona (o *unencumbered systems*): el usuario sabe que está en el mundo virtual porque se ve a sí mismo dentro de la escena; no es un sistema inmersivo porque el usuario pasa a ser un integrante visible del mundo virtual porque ve la proyección de su imagen en un fondo o ambiente, en éstos sistemas se emplean los HMD e involucran respuestas en tiempo real como consecuencia de las acciones detectadas por los dispositivos que se emplean, bien sean guantes, cascos, etc. (Parra, J.C., et al., 2001).

Figura 1.6. Realidad Virtual aplicado a la Terapia Física.
Fuente: Innovación en la prestación de servicios a domicilio en atención primaria (ENFERMER@invisible, 2013)

Sistemas de telepresencia: se denomina telepresencia, en ciertos casos, al solo hecho de manifestarse en un mundo virtual como un ente, mientras que a su interacción con objetos reales se le conoce como un *"mixed reality*" o realidad mezclada y se emplean para realizar tareas o acciones a distancia. En estos sistemas se emplean cámaras, dispositivos táctiles y de retroalimentaciones ligadas a elementos de control remoto que permiten al usuario manipular robots ubicados a distancia mientras se experimenta de

forma virtual. Ejemplos pueden ser, robots que desarticulan bombas o los que intervienen en una cirugía, etc.

Figura 1.7. Simulación de una intervención quirúrgica.
Fuente: Ventajas, desventajas y usos de simuladores (REDES, 2014)

Sistemas inmersos: sumerge al usuario en el mundo virtual, empleando sistemas visuales de tipo CAVE (Carolina, 1992), con sensores de posición y movimiento quedando el usuario, sumergido realmente en la atmósfera virtual y formando parte de ese mundo (Pérez, 2008).

Figura 1.8. Niño en un ambiente virtual de alta inmersión, empleando casco y rastreador de posición.
Fuente: Realidad Virtual: Un Aporte Real para la Evaluación y el Tratamiento de Personas con Discapacidad Intelectual (Pérez, 2008)

1.1.4. Mundos Virtuales

Se puede definir a internet 3D como un mundo virtual en línea, pues integra una serie de recursos que permiten al usuario la comunicación con otros, dentro de la red, tener acceso también a servicios tradicionales de internet, pero desde otra perspectiva e interacción a través de la personificación, bien sea real o falsa de un avatar que lo represente en este mundo virtual. Se dice que existen actualmente más de 100 sitios que

ofertan mundos virtuales en línea –MUVEs que llegan a diversos grupos: niños, adolescentes, adultos, artistas, acceso telefónico, etc.

Se puede señalar que el tipo de ambientes virtuales creados hasta la fecha, son múltiples y tienen que ver con los distintos propósitos y aplicaciones, es por esto que a continuación se toma como base tres tipos que han sido generalizados.

a) **Mundo muerto:** es aquel en el cual no existe participación del usuario, es decir no hay objetos en movimiento ni partes interactivas, sólo se puede realizar una exploración pues las imágenes están pre calculadas y producen experiencias del tipo pasivo.

b) **Mundo real:** en este mundo los elementos presentan atributos reales, de tal manera que si pulsamos un botón de encendido de iluminación, éstas se encenderán.

c) **Mundo fantástico:** en este mundo se pueden apreciar situaciones irreales, que son producto de la fantasía que no sólo están presentes en videojuegos, sino que pueden estar presentes en aplicaciones educativas como por ejemplo, para entender de mejor manera, el proceso eruptivo de un volcán, adentrándose al interior de su cráter.

1.1.5. Dispositivos empleados en Realidad Virtual

Cascos de visualización

Los cascos estereoscópicos son dispositivos también conocidos como gafas o lentes que se acoplan a la cabeza del usuario y le permiten formar parte del mundo virtual, suelen incorporar un sensor que registra la posición y orientación de la cabeza del usuario, permitiendo actualizar la imagen convenientemente (González P, 1998). Existen básicamente dos tipos de casos:

a) **Inmersivos:** conocidos como Pantalla montada en la cabeza o *head-mounted display* (HMD, por sus siglas en inglés), que es una pantalla montada en la cabeza, son dispositivos que aíslan al usuario del mundo real, permitiendo reproducir imágenes en una pequeña pantalla incorporada con la óptica, necesaria para permitir el enfoque, donde cada ojo está alineado a una pantalla a

fin de recibir una señal de video independiente. El tipo de pantallas que se emplean en estos casos son LCD, CRT, CRT más fibra de vidrio y micro escáner láser. Además, suelen incorporar sistemas propios de audio. Se distinguen dos tipos: los que llevan la pantalla incorporada (ver Figura 1.9a) y los que son esencialmente una carcasa destinada a que el usuario emplee un *Smartphone* (ver Figura 1.9b.).

a)

b)

Figura 1.9. Tipos de Gafas empleados en Realidad Virtual a) Oculus Rifts: gafas que tienen incorporado pantallas, b) HTC: gafas que adaptan un Smartphone.
Fuente: a) *Virtual Reality Still* (MakeUseOf, 2016) b) *Why Does Virtual Reality Make Some People Sick* (Science, 2016)

b) Semi inmersivos: conocidos por sus siglas en inglés como HCD, pantalla acoplada a la cabeza, son dispositivos similares a los HMD pero que incorporan pantallas de tipo CRT, debido a su gran peso van sobre montados en un tipo de soporte mecánico, en el que están acoplados sensores que permiten monitorizar los movimientos del usuario, también se les conoce como sistemas Monitor Binocular de Orientación tipo Omni (BOOM, por sus siglas en inglés) (Figura 1.10.).

Figura 1.10. BOOM, Monitor Binocular con orientación Omni.
Fuente: Estado del Arte de la RV (Mejía N, 2012)

Los cascos o lentes pueden ser monoculares o binoculares. Esta última es la que permite tener una imagen estereoscópica (es decir una imagen del tipo 3D) y por lo general estos dispositivos incorporan sistemas de audio direccional que simula la posición en el espacio de las distintas fuentes de sonido del entorno virtual.

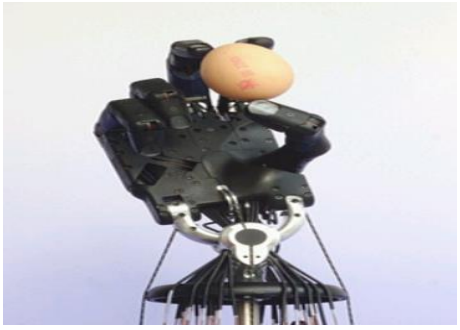
Guantes de Datos (*Dataglove*)

Un guante de datos es un dispositivo interactivo, que es semejante a un guante empleado en la mano, que facilita la detección táctica y el control de movimientos fino en robótica y en realidad virtual. Los guantes de datos están entre los dispositivos electromecánicos más utilizados en aplicaciones hápticas.

La simulación del tacto humano incluye la capacidad de percibir la presión, fuerza lineal, torque, temperatura y textura de la superficie. El control del movimiento involucra el uso de sensores de detección de movimientos del usuario, y transformar estos movimientos en señales que pueden ser utilizados por una mano virtual (Yiannakopoulou E, et al., 2015) o una mano robótica (Santos L, et al, 2015). (Figura 1.11.)

En un mundo de RV un *dataglove* puede permitirle interactuar a un usuario normalmente con objetos (girando perillas de puertas) y recibir retroalimentación háptica para reproducir el tirador de la puerta y sentir el objeto en su mano en lugar de simplemente un gesto en el aire. La retroalimentación háptica es esencial para la inmersión, lo que permite la participación del usuario en entornos virtuales, particularmente para aplicaciones como juegos en realidad virtual o ambientes laborales.

a)



b)

Figura 1.11. Tipos de Data Glove. (a) VMG 30 Plus guante háptico, (b) Shadow Hand, mano robótica.
Fuente: Vrealities (Virtual Realities, 2017)

Head trackers

Head Tracker son dispositivos que se emplean para reproducir los movimientos de la cabeza del usuario en el simulador, y por medio de estos movimientos mirar en todas las direcciones e incluso realizar algunas acciones de movimiento.

Su sistema de posicionamiento para la detección del movimiento corporal lo ejecuta a través de señales infrarrojas del receptor IR y un sistema de puntos de luz infrarroja en el cuerpo del usuario (normalmente en la cabeza) con hasta cinco puntos de luz infrarroja, para transmitir al receptor la ubicación en el espacio tridimensional el movimiento hacia un joystick virtual para la interacción principalmente de simulación aérea, conducción o combate.

Mediante el movimiento de la cabeza se puede realizar movimientos que comprende 6 grados de libertad: cabeceo, guiñado, balanceo, movimiento hacia arriba, abajo derecha e izquierda e inclinación hacia adelante y atrás (Figura 1.12.).

Figura 1.12. TrackIR 5: dispositivo de entrada head tracking.
Fuente: Tackir (PointNatural, 2016)

Dispositivo de fuerza retroalimentada

Equipos que hacen que el usuario experimente los cambios que se están sucediendo en el mundo virtual; las interfaces con fuerza retroalimentada se encuentran en un estado de avance de desarrollo en comparación con las interfaces con retorno táctil, ya que los primeros se vieron beneficiados por los avances realizados en el campo de la teleoperación. Sin embargo, en ambos casos un amplio campo por explorar. Un ejemplo de este tipo de dispositivos es el joystick con fuerza retroalimentada, que permite al usuario sentir el esfuerzo de mover objetos y desarrollar memorias de patrones motores.

1.1.6. Programas empleados en Realidad Virtual

Los sistemas de software conocidos también como herramientas virtuales (en inglés, Virtools), son plataformas o programas de desarrollo para crear contenido 3D orientados a la realidad virtual, simulación, juegos, visualización, etc. Estas herramientas permiten la creación de prototipos en ambientes de inmersión 3D locales o remotos que simulan interactividad con el mundo real; es decir, mediante el empleo de estos programas se puede dar movimiento a los objetos, simular situaciones cotidianas, entre otras, que permiten al usuario interactuar con los elementos que le rodean en el mundo virtual.

Existe un sin número de programas orientados a la programación y simulación, mismos que pueden ser especializadas dependiendo el tipo de aplicación, y que no necesariamente son exclusivas de la realidad virtual ya que no existe un software específico que permita por si solo crear el mundo virtual.

En el desarrollo de un ambiente virtual se involucra la programación de visualización, interacción y simulación; en función de éstas se podrían clasificar en herramientas para *rendering*, interacciones y simulaciones (Mejía N, 2012). A continuación, comercialmente se citan los siguientes programas.

ARcrowd: es una herramienta que permite crear y ver contenido de realidad aumentada o virtual sin necesidad de tener mayor conocimiento o ser experto en programación; puede ser utilizada online sin necesidad de descargarse, pueden emplearse modelos en 3D pre diseñados; sus contenidos pueden ser publicados y compartidos en las redes sociales, tiene un tamaño límite de archivo a reproducirse; pueden añadirse y trabajar con videos (MP4), imágenes (en formatos: png, jpg, gif, bmp, jpeg), archivos de sonido (MP3) y modelos en 3D (Themelsle, 2017).

Accelicad: es una plataforma de tipo CAD, por lo que es compatible con el formato DWG de AutoCad; está basado en la tecnología de IntelliCad, lo que permite que sea un sistema óptimo.

Las herramientas que emplea ésta plataforma permiten crear gráficos en 2D y 3D, emplear funcionalidades de herramientas CAD como cotas, simbologías, creación

de listas, manejar información 3D como superficies y ser compatible con sistemas 3D mediante ACIS.

UNITY: herramienta desarrollada por *UnityTechnologies* en el 2005. Es un motor de desarrollo completamente integrado que proporciona una funcionalidad completa y lista para ensamblar contenido de alta calidad y rendimiento y publicar en múltiples plataformas y compatible con muchas aplicaciones del tipo CAD. Desarrolladores, estudiantes, diseñadores, corporaciones, investigadores, etc., emplean esta plataforma que permite reducir tiempo, esfuerzo y costos, en distintas ramas.

Unity emplea un motor gráfico OpenGL para ser desarrollado en Linux, Mac o Windows; Direct3D para Windows. OpenGL, es para aplicaciones Android e IOs e interfaces propietarias Wii. Además, puede utilizar en aplicativos/software como Adobe Photoshop, Fireworks, Blender, Cinema 4D, Softumage, Allegorithmic Substance, Cheetah3D. Entre las facilidades que presta el motor gráfico utilizando Unity es la importación de librerías y complementos dentro de la aplicación a ser desarrollada (LinLin W, et al, 2017), lo que permite alcanzar mejores resultados por la facilidad de incorporar elementos propios de ésta herramienta (Unity, 2005).

Unity3D es una de las versiones existente que facilita el desarrollo de software para un amplio rango de plataformas (Indraprastha A, et. al, 2009), así como resulta ser muy atractiva por sus prestaciones para desarrolladores e investigadores, emplea un script en lenguaje C para manipular el comportamiento de los objetos simulados (Andaluz VH, et. al, 2016). .

Se debe señalar que los programas mencionados no son únicos, dependiendo el uso se comprueba que existe una infinidad de plataformas abiertas y licenciadas disponibles para desarrolladores, jugadores e investigadores.

1.1.7. Aplicaciones de la Realidad Virtual

Con el paso de los años la realidad virtual ha dejado de ser de uso exclusivo de la ciencia ficción, entretenimiento, investigadores y científicos para llegar a formar parte de la vida cotidiana del ser humano y su entorno.

Entre las aplicaciones y prestaciones que emplean realidad virtual se destacan algunos desarrollos en áreas específicas, tales como medicina, rehabilitación de pacientes, educación, arte, militar, publicación y promoción, entretenimiento, juegos, cine, modelado y creación de entornos virtuales (museos, tiendas, aulas, etc.), industrial, centros de investigación (Mejía N, 2012).

Es preciso indicar que las aplicaciones de esta tecnología están despuntando y día a día se generan mayores áreas en las cuales se ha iniciado el empleo de los mundos virtuales. A continuación, se citan de manera general algunas de las aplicaciones más relevantes de realidad virtual desarrolladas.

a) Medicina

La realidad virtual en el campo de la Medicina, ha desempeñado un rol importante que ha hecho posible a cirujanos, practicantes y personal de diagnóstico compartir espacios virtuales con fines de diagnóstico y consulta. Además de otras posibilidades para los profesionales de este campo, la formación, intervenciones quirúrgicas o tratamientos de enfermedades, etc.

La exploración e intervención médica a nivel celular y genético ha podido ser desarrollada en un entorno virtual, logrando así una disminución y aumento de escalas que permiten a los cirujanos trabajar como si sus áreas fueran expandidas, experiencias que permiten conocer sin comprometerse con casos de alto riesgo, reduciendo también los costos por la inversión en cadáveres y animales en docencia y cirugía, con la posibilidad de tener a disposición el entrenamiento, las veces que sea necesario, algo que no es posible cuando se experimenta con pacientes reales.

Además, la realidad virtual dentro de la medicina se emplea en la rehabilitación (Andaluz V, et.al,, 2016), en fisioterapia (permitiendo al usuario experimentar una terapia desde su casa y en un entorno más interactivo en forma de videojuego) y en personas discapacitadas (Pérez, 2008); siendo éstas el grupo que más podrán experimentar actividades funcionales que por su condición no les permiten producir movimiento, pero con la simulación 3D éstas sensaciones si pueden ser detectadas por el computador y transmitidas; es decir aproximan a la mente del usuario la participación y proporcionan un sentido de control sobre el entorno virtual y facilitar en gran medida los procesos de

aprendizaje y entrenamiento, sin que esto se confunda que a través de ésta tecnología una persona imposibilitada para caminar lo pueda realizar (Mejía N, 2012).

b) Educación

La educación tradicional se ha visto expuesta en los últimos años con la realidad virtual, ya que ésta, en países desarrollados ha ganado un enfoque alternativo a las experiencias del aprendizaje tradicional, debido a que permite al estudiante interactuar con los objetos que se encuentran en el mundo virtual cuyo éxito depende de su capacidad de prever y manipular información abstracta.

La mejora más evidente en el empleo de realidad virtual se ha visto en temas como ciencias, química, física, matemáticas, entre otros, que han permitido hacer de lo abstracto algo más palpable mediante el uso de entornos virtuales ofreciendo así una experiencia interactiva que está complementada con la representación del concepto más allá de la formulación matemática (Flores J, 2014).

La realidad virtual permite imaginar un entorno de aprendizaje, en el cual tiene como actores al profesor y otros compañeros e intercambiar opiniones y material con ellos como si estuvieran sin necesidad de trasladarse de su lugar en el que está ubicado, es así que el rol del profesor cambia también pues ha dejado de ser el transmisor de conocimientos y convertirse en el instructor de estudiantes que aprenden y se involucran con la tecnología, que es la que proporciona recursos interactivos en el aprendizaje.

Existen aplicaciones que se emplean en la educación que van desde laboratorios virtuales hasta aulas virtuales para estudio de comportamiento comercialmente se puede citar, *Aula Nesplora* y *Labster* – laboratorio biológico (Obrist V, et. al,, 2015).

c) Entretenimiento

Los videojuegos se han convertido en el campo más desarrollado de la realidad virtual y se han convertido en un elemento importante dentro de las aplicaciones del entretenimiento, existiendo básicamente dos tipo de videojuegos: de entretenimiento y didácticos. No obstante, como entretenimiento y ocio se pueden encontrar otras aplicaciones fuera de los videojuegos, tales como muse, arte e incluso turismo interactivo.

En este sentido, a través de la realidad virtual se pueden crear ambientes, paisajes reales o imaginarios que permiten contextualizar los objetos de un museo o una exposición, facilitando de esta manera la comunicación del centro con una mejor experiencia en estos lugares, es por ello que en los museos la tecnología ha aportado con dinamismo e interactividad dejando de ser aburrido, estático o anticuado.

La realidad virtual no pretende sustituir, por ejemplo, el turismo, sino ser un complemento que permita mostrar a las personas mediante imágenes producidas incrementar el turismo a través del incentivo que las personas puedan experimentar a través de estas plataformas turísticas en RV (Castro Juan, et. al, 2017).

d) Militar

Quizá la aplicación más conocida en el ámbito militar son los simuladores de vuelo, que permiten a un usuario que está iniciándose en la profesión de piloto reproducir las situaciones con las cuales vaya a interactuar en la vida real. No obstante la realidad virtual en la milicia se puede encontrar para el entrenamiento y simulación de operaciones que involucran armamento y requiere que su personal pueda entrenarse bajo una condición específica y casi real, previo a salir a combate. Además, estas aplicaciones necesitan contar con imágenes muy realistas, pero este es un tema que aún está siendo investigado, por ejemplo, crear campos radioactivos o campos de minas, etc.

En el espacio aéreo también se puede encontrar usos de la realidad virtual como el manejo de aeronaves no tripuladas teledirigidos a distancia, que permiten realizar acciones de inteligencia para monitoreo y control de una zona sin poner en riesgo al tripulante o aplicada al control de tráfico aéreo, desarrollo de aeronaves de uso militar, etc.

e) Industrial

Los procesos industriales desde sus inicios se han visto obligados a incursionar en los avances de la tecnología para ser más competitivos en el mercado, cuya estrategia de mercado se ha enfocado en satisfacer las exigencias de las nuevas generaciones de los consumidores que, entre otras relaciona a la calidad de un producto, sus características, materiales que emplea y también su relación e impacto con el medio ambiente.

La industria se ha visto inmersa en los últimos años con mayor énfasis con el empleo de la robótica, drones, internet de las cosas, impresión 3D y la última tendencia es la realidad digital o virtual (Andaluz V., et. al, 2016) que está relacionada directamente con la vida cotidiana del mundo industrial y sus actores (Pallares, 2016).

En los sectores productivos las ventajas que brinda el empleo de la realidad virtual tanto en optimización de diseño, mantenimiento y control de la planta, training de operaciones y formación de operarios, asistencia y resolución de incidencias, resultan ser varios. No obstante, entre los que más se destacan se tiene: formación más eficaz, reducción de costos y tiempo en el desarrollo del producto, un aliado en el marketing.

f) Petróleo y gas

Las características propias que presenta un entorno industrial son las grandes instalaciones (plantas de procesamiento, plataformas marítimas de refinación, etc.) y complejas (varios procesos y subprocesos) que operan en su máxima capacidad de forma continua durante los 365 días del año, las 24 horas, en condiciones climáticas extremas. Por lo que, este tipo de industria requiere no sólo contar con personal que este constantemente capacitado en el manejo y operación de equipos y sistemas, sino también que conozca y aplique las medidas de seguridad industrial que sean necesarias, esto con la finalidad de evitar situaciones de emergencia que no solo afecte la integridad del individuo y sus compañeros. Además, que produzca daños en la planta y paralice los procesos de producción, lo ~~que~~ cual ocasiona grandes pérdidas económicas (Mons Midttun, et. al, 1998).

En este sentido la realidad virtual en el sector de petróleo y gas puede ser empleado principalmente en los siguientes campos (Pallares, 2016):

Exploración: las diferentes técnicas de realidad virtual han sido empleadas para que los actores dentro del mundo del petróleo y gas (Zhou Zewei, et. al, 2011) como ingenieros, geólogos, científicos, operarios, gerentes, etc., puedan explorar los distintos ambientes tales como los yacimientos; en el cual el empleo de modelos 3D a través de los diferentes recursos disponibles (empleo de herramientas virtuales y programas específicos tipo CAD, etc.) permiten crear una realidad que combine la información proveniente de distintas fuentes como datos sísmicos que revelen las características estructurales, registro de pozos, permeabilidad y otras propiedades

involucradas en el análisis. Además, de la posibilidad de explorar, ésta técnica permite manipular un proceso a fin de investigar alguna situación específica con los datos proporcionados, lo que no es factible cuando la explicación de la exploración de un proceso en años anteriores se lo realizaba mediante el empleo de una representación gráfica no interactiva (Mons Midttun, et. al, 1998).

Operación y mantenimiento: la planificación de mantenimientos en el sector petrolero y gas está presente de forma periódica, esto considerando que este tipo de plantas trabajan de manera continua las 24 horas del día, en esta aplicación la realidad virtual en 3D en la operación o mantenimiento de procedimientos operativos proporciona una manera efectiva para que los se puedan desarrollar escenarios y simulación de escenarios de las evaluaciones y posibles casos de lo que puede pasar con los equipos reales, sin tener que paralizar las actividades en la planta.

También es importante recalcar el beneficio que presta ésta tecnología en el momento de realizar modificaciones o nuevas implementaciones dentro de la planta, que permiten de mejor manera por ejemplo buscar rutas óptimas para extraer o instalar equipos disminuyendo interferencias y la identificación de infraestructura que puede obstruir dichas implementaciones, existiendo para esto también software específico que también proveen de información adicional que resulta útil en el estudio.

Capacitación: en los sectores industriales así como el petrolero y gas, el entrenamiento o capacitación es visto como un proceso educacional a corto plazo que se desarrolla de manera sistémica y organizada, ya sea porque están incorporando nuevo personal o se está llevando una actualización y mejora de capacidades técnicas a su personal, mediante la cual se adquiere aptitudes, conocimientos y habilidades en función de los objetivos definidos del ambiente en el cual se desenvuelven y de manera especial la capacitación en el área de higiene y seguridad industrial, contribuyendo así a la formación integral del trabajador (Marcano Y., 2006); es así que en este campo la realidad virtual ha desempeñado un papel importante con los modelos 3D realistas, simulaciones y visualizaciones con propósitos específicos.

Las características que el entrenamiento permite mediante el empleo de realidad virtual es que los empleados pueden involucrase y ver como es la plataforma, refinería, etc., y sobretodo conocer el trabajo que desempeñarán dentro de ella;

además, dentro del mundo virtual se pueden simular y preparar a los empleados para situaciones eventuales creando una situación crítica que en otra modalidad de entrenamiento no es factible simularla; es por ello que esta opción de capacitación no solo que resulta ser eficaz, rápida, segura y eficiente, sino que permite ahorrar recursos económicos empelado en sus desarrollo y logística.

1.2. Lanzamiento y recepción de raspadores

Una de las partes críticas y más importantes de la cadena de producción hidrocarburífera es transportar el producto desde una locación a otra, cumpliendo un papel estratégico en la explotación, distribución y comercialización de hidrocarburos. El crudo y/o sus derivados son transportados entre estaciones que cumplen con un determinado objetivo, como por ejemplo, separación, almacenamiento, distribución, refinación o incluso exportación.

Los sistemas de transporte de hidrocarburos más utilizados en el país son: sistemas de tuberías (oleoductos, poliductos y gasoductos), tanqueros marítimos, tanqueros terrestres.

En todo el país existe infraestructura de transporte mediante tuberías para petróleo y derivados que interconectan las diferentes estaciones y que pueden atravesar distancias cortas e incluso cientos de kilómetros. Este sistema es utilizado en su gran mayoría debido a su eficiencia, bajo costo de operación (comparado con otros medios de transporte), seguridad y confiabilidad.

Actualmente, el país cuenta con los siguientes sistemas de ductos para el transporte de hidrocarburos (EP Petroecuador, 2016) (EP Petroecuador, 2015):

Tabla 1.1. Ductos para el transporte de hidrocarburos en Ecuador.

Transporte de Crudo	• Sistema de Oleoducto Transecuatoriano SOTE, 498Km. • Oleoducto de Crudos Pesados OCP; 503 Km
Transporte de Derivados	• Esmeraldas – Santo Domingo, 165Km. • Santo Domingo – Quito, 88Km. • Santo Domingo – Pascuales, 277Km. • Quito – Ambato, 110Km. • Ambato – Riobamba, 41Km • Shushufindi – Quito, 305Km • Libertad – Pascuales, 127Km • Libertad – Manta, 170Km • Tres Bocas – Pascuales, 20Km • Tres Bocas – Fuel Oíl, 5.6Km • Monteverde – Chorrillo, 124Km. • Pascuales – La Troncal, 103Km • La Troncal – Cuenca, 112Km.

Gasoductos	• Plataforma Amistad- Bajo alto, 70Km.
Red de Oleoductos del Distrito Amazónico	• Red de tuberías que transportan el crudo producido y lo inyectan a los oleoductos principales, operado por Petroamazonas EP.

Fuente: Informe estadístico 2015, 2016 (EP Petroecuador, 2016) (EP Petroecuador, 2015).

Dependiendo de la locación e importancia del ducto, éste puede operar casi todos los días del año, transportando uno o más tipos de crudo o derivados.

Debido a su continua operación y conforme el tiempo pasa, las paredes internas de estos ductos van acumulando sedimentos y otras formaciones propias del tipo de fluido transportado, produciendo una obstrucción al paso normal del fluido, que hacen que el diámetro interior de la tubería disminuya. Los depósitos acumulados pueden presentarse como una capa dura adherida a las paredes interiores de las tuberías que pueden llegar a tener algunos centímetros de espesor y presentar cristales de 1 cm. o más. El principal problema de la formación de incrustaciones en las tuberías es la reducción de la producción al aumentar la rugosidad de la superficie de la tubería y reducir el área de flujo. Por ello, es de suma importancia utilizar mecanismos de limpieza y mantenimiento de tuberías que permitan garantizar que el sistema de ductos se encuentre en óptimas condiciones y trabajando eficientemente.

Una de los métodos más confiables es el uso de la técnica de "*PIGGING*" que consiste en enviar a través del ducto un artefacto de limpieza conocido como *pipeline inspection gauges* (PIG), raspador, chancho, diablo o *scrapper*. Este dispositivo se encarga de viajar por todo el segmento de tubería que se requiere limpiar, valiéndose de la fuerza de empuje del propio flujo a una velocidad similar a la del fluido, sin afectar las condiciones normales de operación del ducto.

1.2.1. Raspadores o *Pipeline Inspection Gauges*

Los PIGs son dispositivos, herramientas o vehículos independientes que se mueven a través de los ductos y utilizados en operaciones de limpieza, dimensionamiento e inspección que en los últimos años han sido aplicados a otros tipos de líneas de transporte como mineroductos y acueductos. En el caso de poliductos también se usa el "pig" como elemento separador entre productos diferentes (lotes).

Figura 1.13. Raspador de tubería.
Fuente: The Pig that Burst the Keystone Pipeline (gregpalast, 2017).

Un raspador actúa como un pistón móvil libre dentro de la tubería que puede realizar varias tareas incluyendo la limpieza de escombros de la línea, el retiro de los productos residuales internos, detección de fallas en la línea y medición del diámetro interno de la tubería (Fragoso, 2007).

Usualmente, un raspador es un sólido o semisólido, conformado por uno o varios cuerpos unidos. Existen diferentes tipos de chanchos, distintas formas, desde esferas usadas para barrer los líquidos condensados en líneas de flujo de gas, hasta chanchos altamente instrumentados para la inspección de líneas de flujo llamados Raspadores Inteligentes.

Los pigs de limpieza y de separación, no requieren de componentes electrónicos para su aplicación. Actualmente, se les adiciona un componente magnético o electrónico que es detectado por un equipo instalado en el exterior de la tubería y que permite alertar la llegada o salida del raspador (Ramírez & Dutra, 2010).

La corrida o lanzamiento de raspadores es un proceso importante en la limpieza de oleoductos, poliductos y/o gasoductos, debido a los beneficios que se obtienen después de realizar dicha operación, puesto que la producción se incrementa, los productos son más limpios y se requiere una menor presión de bombeo.

Figura 1.14. Tipos de Raspadores.
Fuente: *Girard Industries* (Girard, Product Line)

El principal objetivo de la operación de la corrida de PIGs es que un sistema de tuberías se mantenga eficientemente estable. La eficiencia de la tubería depende de dos aspectos: la operación continua; y la reducción de costos de operación.

Para realizar la corrida de raspadores se requiere de facilidades mecánicas y electrónicas al inicio y al final del segmento de tubería a limpiar. Estas facilidades se llaman trampas de salida o lanzadoras y trampas de llegada o receptoras.

El diseño de las trampas lanzadoras y de recepción varían entre cada ducto y dependen de varios factores como el tipo de fluido, el diámetro de la línea principal, etc. Sin embargo, existen elementos que son básicos en su diseño y posterior operación segura.

Las trampas lanzadora/receptora son fabricadas de manera integral, inspeccionadas y probadas hidrostáticamente con la opción de montarse desde el sitio de su fabricación sobre una base estructural (skid) para montarse sobre una base de concreto en el sitio de instalación definitivo.

Figura 1.15. Trampa lanzadora de raspadores.
Fuente: Estación Lago Agrio SOTE EP Petroecuador.

1.2.2. Trampa lanzadora o de salida

La trampa lanzadora es un arreglo de válvulas y tuberías acoplada a la línea principal de flujo que permite realizar el lanzamiento de raspadores sin la necesidad de parar la operación normal del ducto, y que está delimitada desde un punto anterior a la T de intersección de las línea de bypass y lanzadora hasta un punto ubicado en la línea saliente después de la T principal aguas abajo del indicador de PIG.

La trampa lanzadora está conformada por el Barril mayor, el barril menor, línea de bypass, línea lanzadora, línea de drenaje, línea de venteo, tapa/compuerta sellada, instrumentación y accesorios de conexión (bridas, reducciones, weldolets, etc.). Cada línea tiene una válvula de compuerta o bola que permitirá o bloqueará el paso del fluido.

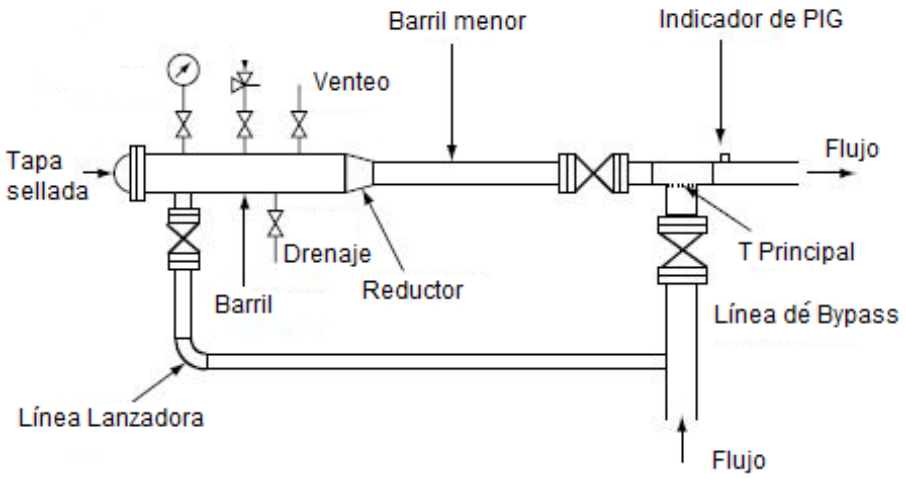


Figura 1.16. Trampa típica lanzadora de raspadores.
Fuente: Autor

Barril Mayor/menor: es la sección de la Trampa que se extiende desde la válvula principal de la trampa hasta la Tapa de la Trampa inclusive, y es requerido para lanzar y recibir los raspadores. El barril mayor se encuentra unido al barril menor a través de la reducción.

En cañerías con diámetro nominal menor a 500 mm (20”), el diámetro nominal del Barril Mayor es de 50 mm (2”) más grande que el diámetro nominal de la Línea Principal.

Para cañerías con diámetro nominal mayor o igual a 500 mm, el diámetro nominal del Barril Mayor es de 100 mm (4”) más grande que el diámetro nominal de la Línea Principal. El diámetro interno del Barril Menor debería ser igual al de la Línea Principal (YPF, 2009).

Línea de Bypass: Es la línea por la cual pasa el fluido en condiciones de operación normal. Conecta la Línea Principal con las instalaciones relacionadas aguas arriba y aguas abajo como por ejemplo la estación de bombeo y el área de tanques de

almacenamiento para la trampa lanzadora y receptora, respectivamente. Esta línea puede ser de menor o igual tamaño que la línea principal del ducto.

Línea lanzadora: Línea que conecta el Barril Mayor con la Línea de *Bypass*, permite el desvío del fluido a través del Barril para lanzar o recibir un chancho. Para una trampa de salida, la Línea Lanzadora debe estar conectada al Barril Mayor tan cerca como sea posible de la Tapa de la Trampa. Para una trampa receptora, la línea ingresa al barril en un punto tan cercano como sea posible del Reductor.

Línea de drenaje: Permite drenar cualquier líquido que se acumule en el barril tanto para la trampa lanzadora como para la receptora. Se encuentra ubicada cerca de la tapa de la trampa y su tamaño puede variar entre 2 y 4" dependiendo del tamaño de la línea principal. La línea de drenaje conecta a un tanque recolector de fluido.

Línea de venteo: Ubicada en la parte posterior del barril mayor o en la línea lanzadora aguas abajo de la válvula correspondiente. Los diámetros de estas líneas son de al menos 2".

Válvulas: En las trampas, las válvulas son necesarias para bloqueo y por ende son válvulas esféricas (de bola) o válvulas de compuerta. Para cada sistema de Trampas se utilizan al menos las siguientes válvulas:

- 1 válvula de trampa
- 1 válvula de bypass
- 1 válvula lanzadora
- 1 válvula de drenaje
- 1 válvula de venteo

Las 3 primeras válvulas pueden ser operadas de manera manual o a través de un actuador hidráulico, neumático o motorizado, dependiendo del tamaño de las líneas. Las dos últimas válvulas son manuales.

Compuerta: Permite al operador insertar o retirar el raspador del barril, se encuentra soldada al final del barril mayor y se la opera mediante un volante o palanca. La tapa es totalmente hermética cerrada a prueba de fallos, de actuación rápida de tal manera que

permite la apertura y cierre por una persona sin uso de dispositivos adicionales en el menor tiempo posible.

Instrumentación: En la trampa se encuentran distribuidos instrumentos electrónicos y mecánicos que permiten la interacción del operador con las variables del proceso, así como alertar y avisar los parámetros relacionados a la operación de lanzar y recibir los chanchos.

Los instrumentos instalados en la trampa son: medidores de presión, temperatura e indicadores de paso de raspador. Estos instrumentos tienen visualización local y se encuentran conectados al sistema de control de la estación para un continuo monitoreo remoto.

1.2.3. Trampa receptora o de entrada

La trampa receptora es un arreglo de válvulas y tuberías acoplada a la línea principal de flujo que permite recibir los raspadores en la parte final del segmento de tubería limpiada sin la necesidad de parar la operación normal del ducto. La trampa está conformada por: Barril de recepción, línea de bypass de trampa, línea de bypass, línea de drenaje, línea de venteo.

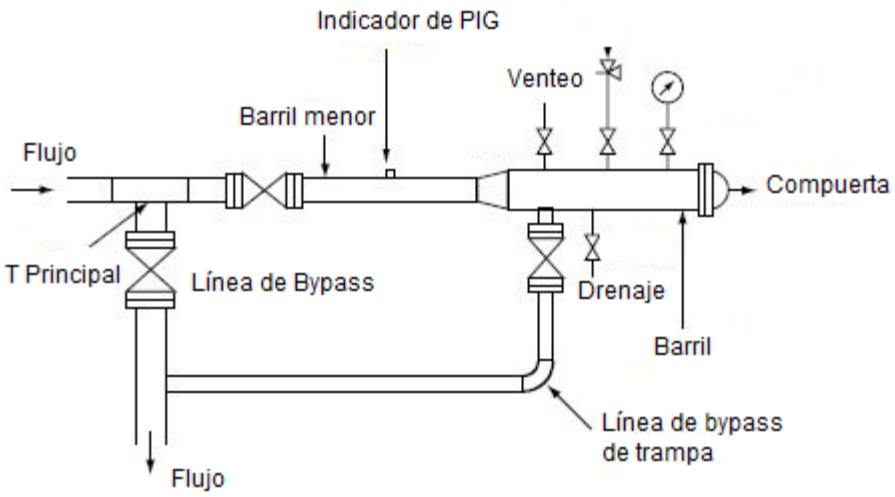


Figura 1.17. Trampa típica receptora de raspadores.
Fuente: Autor

2. MÓDULO DE ENTRENAMIENTO VIRTUAL

El módulo de capacitación virtual 3D con capacidad de inmersión, está orientado a ser un complemento práctico de la instrucción teórica que los operadores reciben antes de manipular los equipos en campo, fortaleciendo las dos dimensiones principales del aprendizaje según (Kolb, 1984): la percepción y el procesamiento; y que de acuerdo al mismo autor el modelo de aprendizaje por experiencia lo describe como *"algunas capacidades de aprender que se destacan por encima de otras como resultado del aparato hereditario de las experiencias vitales propias y de las exigencias del medio ambiente actual"* .

Adicional, el módulo de entrenamiento se vuelve un complemento educativo para cumplir con las políticas de seguridad de las empresas operadoras de ductos. Por ejemplo dentro del Plan Estratégico Empresarial de Petroecuador, en el capítulo de Políticas empresariales se menciona que se debe *"Aplicar buenas prácticas en la industria hidrocarburífera para la prevención de la contaminación y riesgos laborales"*, además de *"Asegurar que todos los empleados y contratistas entienden que el trabajo seguro y la protección del ambiente es un requisito para realizar sus actividades y que cada uno de ellos es responsable de su propia seguridad, de quienes les rodean y la del entorno"*. (EP PETROECUADOR, 2018)

2.1. Aplicación operativa en el transporte de petróleo: Lanzamiento y recepción de raspadores de tubería

Una de las actividades operativas más comunes en el transporte de hidrocarburos es el lanzamiento y recepción de raspadores, operación que implica que los trabajadores manipulen de manera secuencial y coordinada el conjunto de válvulas de las trampas lanzadoras y receptoras, y que verifiquen el estado de las variables involucradas: presión y flujo. Una mala operación en esta tarea podría implicar sobre presurización de tubería, cavitación de unidades de bombeo, daño en la instrumentación e incluso, si no se toman las medidas correctivas de manera rápida, riesgos físicos y ambientales.

Es por esto que para mantener capacitados de manera óptima al personal de estas operaciones, se vuelve necesario generar otras alternativas tecnológicas para impartir

conocimientos y buenas prácticas aprovechando el avance de la era digital y su interacción con el ser humano.

Se plantea una herramienta de capacitación y simulación del lanzamiento y recepción de un raspador, basada en condiciones específicas conocidas, a través de una plataforma de realidad virtual que recree el ambiente de trabajo de un operador, que permita entender de manera visual las respuestas a cambios operacionales (respuesta de variable de presión y flujo), prácticas sistemáticas y posibles riesgos físico ambientales.

En el módulo de capacitación se plantean tres escenarios para el lanzamiento y la recepción del raspador: condición inicial, secuencia de operación correcta, operación no segura.

El usuario podrá interactuar con el entorno virtual de manera directa manipulando las válvulas que componen la trampa lanzadora y receptora, visualizando los distintos instrumentos, señalizaciones y alarmas recreadas en las facilidades de planta virtual.

La aplicación permite alertar al usuario en caso de que la acción realizada no sea segura y represente una emergencia para la estación y su entorno.

2.1.1. Condición Inicial - Lanzamiento de Raspador

De manera didáctica se han identificado las válvulas de la trampa lanzadora de la siguiente manera y de acuerdo a la Figura 2.1:

A: Válvula principal de la trampa

B: Válvula de Bypass de línea principal

C: Válvula de desvío o lanzadora

D: Válvula de drenaje

E: Válvula de venteo

En condiciones normales de operación, cuando no se realizara lanzamiento de raspadores, la trampa no está presurizada ni cargada de líquido. La válvula principal de trampa A, la válvula de desvío o lanzadora C, la válvula de purga/drenaje D y la válvula

de venteo E están cerradas. La válvula de bypass B está abierta y permite el flujo normal del fluido.

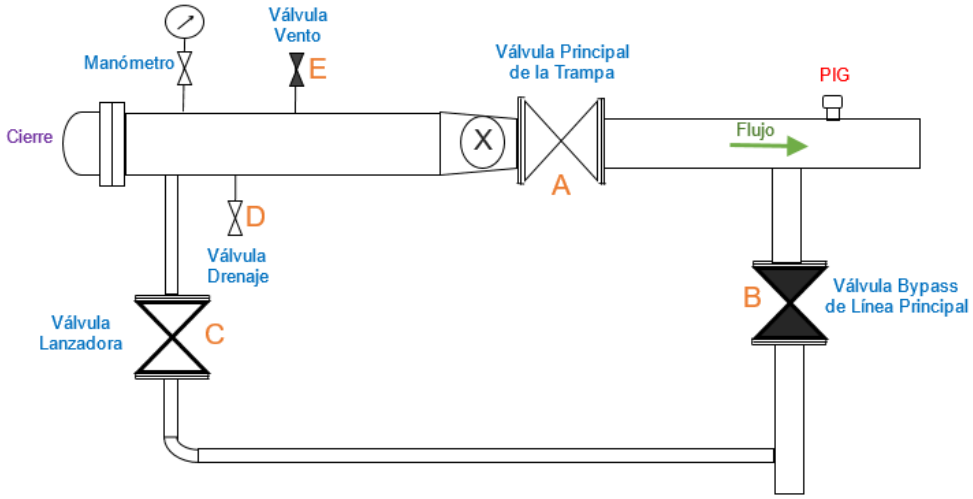


Figura 2.1. Válvulas que conforman la trampa lanzadora.
Fuente: Autor

2.1.2. Secuencia de operación normal - Lanzamiento de Raspador

Para realizar la operación normal de lanzamiento de raspador se deberán seguir los siguientes pasos:

a) Asegurarse que las válvulas A y C se encuentren totalmente cerradas.

b) Drenar la trampa lanzadora abriendo la válvula D y permitir que el aire desplace el líquido abriendo la válvula E.

c) Cuando la trampa esté completamente drenada y a presión atmosférica (0 PSIG) con las válvulas D y E aún abiertas, abrir la tapa de cierre rápido de la trampa e insertar el raspador PIG, empujándolo hasta la reducción anterior a la válvula A.

d) Cerrar y asegurar la tapa de cierre rápido de la trampa. Cerrar la válvula de drenaje D dejando abierta la válvula de venteo E. Purgar el aire dentro de la trampa a través de la válvula E abriendo lenta y completamente la válvula C. Cuando esté cumplido el barrido, cerrar la válvula E para permitir una ecualización de presiones, entonces cerrar la válvula C.

e) Abrir la válvula A primero, luego la válvula C. En este momento el PIG estará listo para ser lanzado.

f) Cerrar parcialmente la válvula B. Esto incrementará el caudal de líquido a través de la válvula C y detrás del cuerpo del Raspador. Continuar cerrando la válvula B hasta que el PIG salga de la trampa introduciéndose en la corriente de la tubería y haciendo saltar la señal del detector de raspador YS-101

g) Cuando el PIG sea despedido de la trampa y se inserte en la tubería principal, abrir la válvula B, totalmente.

h) Cerrar las Válvulas A y C

i) Drenar la trampa a través de la válvula D

En algunos campos petroleros, luego de que el raspador ha sido lanzado y la válvula B ha sido abierta, no realizan los dos últimos pasos.

2.1.3. Condición Inicial – Recepción de Raspador

De manera didáctica se han identificado las válvulas de la trampa receptora de la siguiente manera y de acuerdo a la Figura 2.2.:

A: Válvula principal de la trampa

B: Válvula de Bypass de línea

C: Válvula de bypass de trampa

D: Válvula de drenaje

E: Válvula de venteo

En condiciones normales de operación, antes de que se anuncie el arribo del raspador la trampa está vacía a presión atmosférica. Las Válvulas B, D y E están abiertas; las válvulas A y C están cerradas. La tapa de cierre rápido de la trampa está cerrada y asegurada.

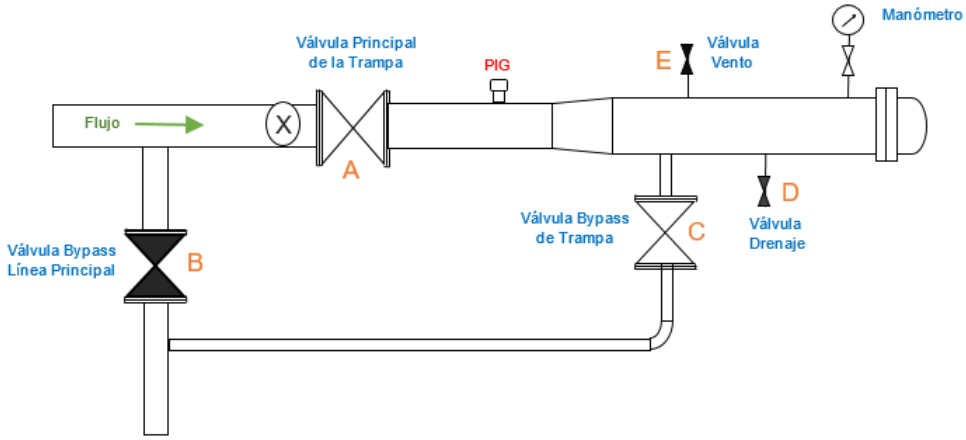


Figura 2.2. Válvulas que conforman la trampa receptora.
Fuente: Autor

2.1.4. Secuencia de operación normal - Recepción de Raspador

Para recibir los raspadores de manera segura se debe seguir y considerar los siguientes pasos:

a) Llenar la trampa cerrando la válvula D, abriendo lentamente la válvula C y venteando a través de la válvula E.

b) Comenzar a ecualizar la presión en la trampa cerrando la válvula E con la válvula C abierta.

c) Con la válvula C todavía abierta, abrir la válvula A. La trampa está ahora en condiciones de recibir el rapador

d) Cuando arribe el raspador, éste se detendrá entre la válvula A y la Tee de entrada a la trampa. El detector de raspador informará que éste se encuentra en el ingreso a la trampa.

e) Parcialmente cerrar la válvula B. Esto forzará al PIG a introducirse en la trampa debido al incremento de flujo de líquido a través de las válvulas A y C.

f) Después que el PIG esté en la trampa (señal emitida por el segundo detector de raspador), abrir completamente la válvula B y cerrar las válvulas A y C.

g) Abrir las válvulas D y E para drenar la trampa hasta que llegue a la presión atmosférica.

h) Después que la trampa esté venteada (0 PSIG) y drenada con las válvulas D y E abiertas, abrir la tapa de cierre rápido de la trampa y sacar el raspador.

i) Cerrar y asegurar la tapa de cierre rápido de la trampa.

Las operaciones de lanzamiento y recepción de raspadores deben ser realizadas por personal capacitado en varias áreas: mecánica, instrumentación, proceso, seguridad y medio ambiente. El no seguir los pasos detallados anteriormente o no conocer de las implicaciones del cambio en las variables involucradas podría ocasionar daños ambientales y afectaciones a la integridad física de los operadores e instalaciones.

2.1.5. Operación no segura

Son operaciones inseguras cualquier maniobra realizada por los operadores que se encuentren fuera de los procedimientos establecidos por la empresa para realizar una determinada actividad y que generen situaciones de riesgo a las instalaciones, a la integridad física o al medio ambiente.

Dentro de las operaciones de riesgo más comunes realizadas en el lanzamiento y recepción de raspadores se encuentran:

- No seguir la secuencia de operación normal de válvulas para el lanzamiento y/o recepción.
- Abrir la compuerta de la trampa sin asegurarse que el compartimiento o barril se encuentre completamente despresurizado y purgado.
- Cerrar las válvulas de lanzamiento y de bypass impidiendo el paso de flujo hacia la siguiente estación.

2.2. Creación de modelos 3D de la aplicación

Al hablar de un sistema de entrenamiento basado en realidad virtual, las características de diseño y funcionalidad del mismo, deben ser lo más cercanas al comportamiento real de cada uno de los componentes de las estaciones representadas.

El conocimiento apropiado de cada uno de los componentes se convierte en un factor fundamental, ya que de ellos dependen todas las características que se puede aportar al sistema de realidad virtual. Por esta razón el primer paso a considerar para la ejecución de este proyecto es el análisis de los diagramas de Instrumentación y Tuberías (P&ID) de cada una de las estaciones consideradas: Lanzamiento y Recepción.

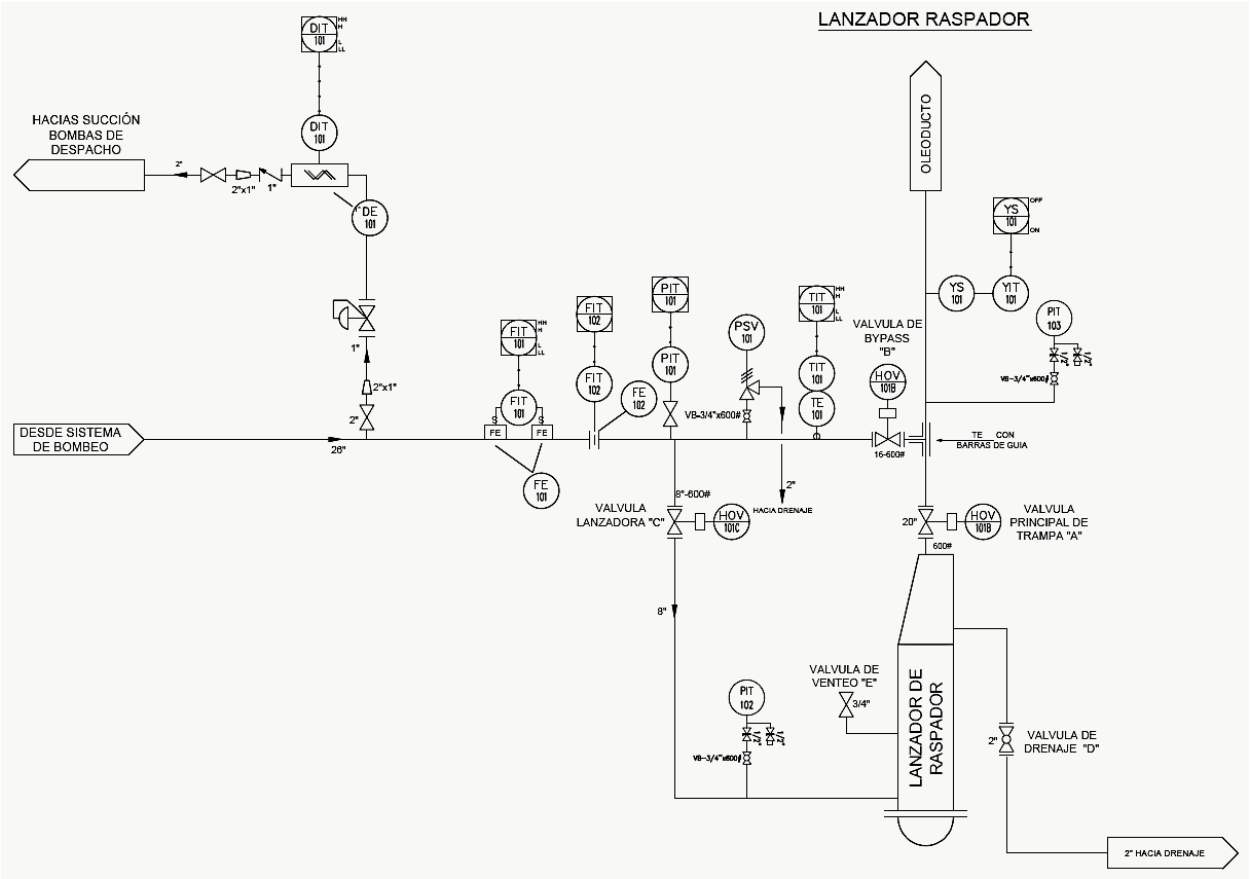


Figura 2.3. P&ID Trampa Lanzadora.

Fuente: Autor

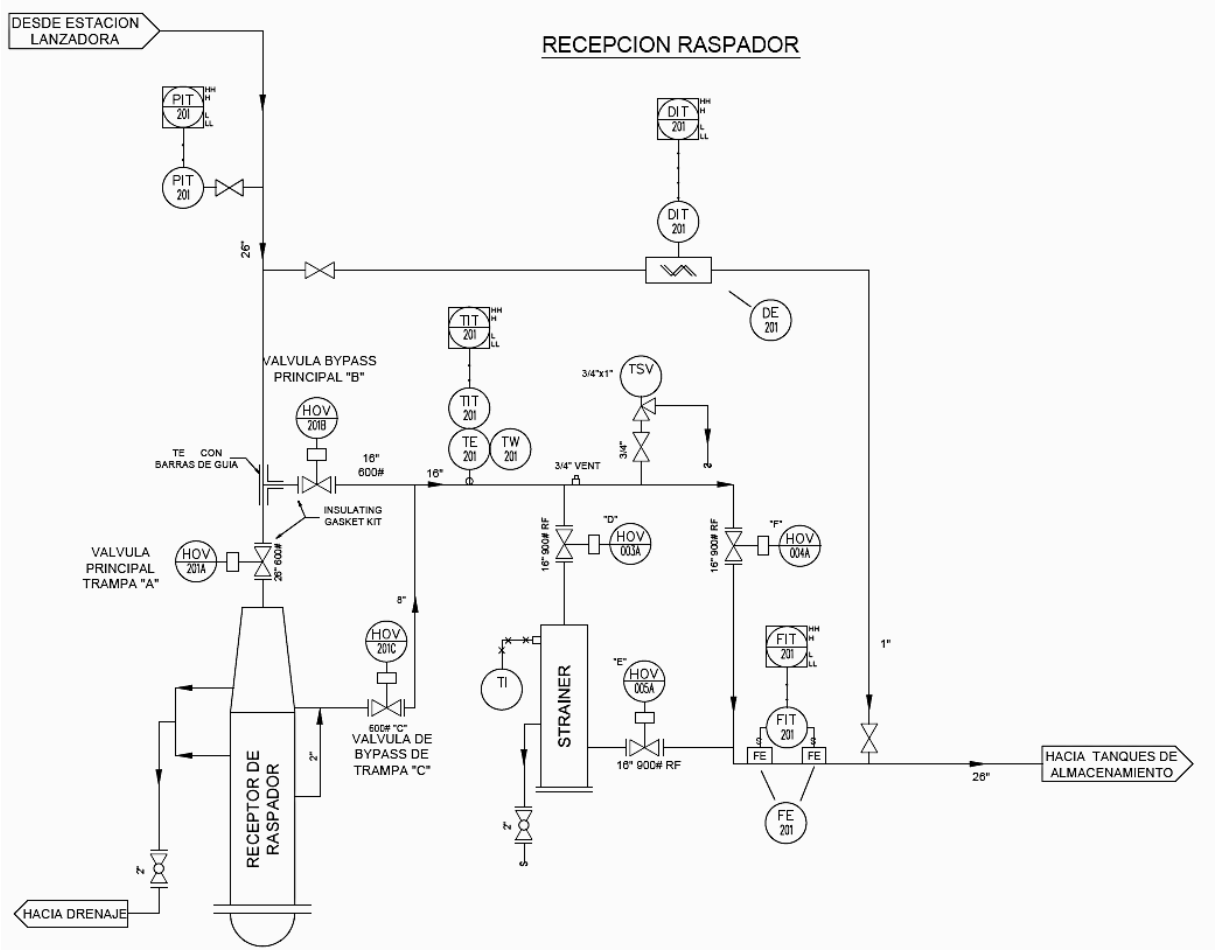


Figura 2.4. P&ID Trampa receptora.
Fuente: Autor

La digitalización de los diagramas P&ID, para una apropiada utilización, se debe realizar mediante un software de diseño CAD especializado.

Existen varios softwares de diseño, en los cuales se puede crear elementos en dos y tres dimensiones, que representan a sistemas simples y complejos. En este caso en particular, se ha escogido *AutoDesk* en su versión 2015, como la herramienta encargada del diseño básico de los diagramas necesitados.

Para el diseño en específico de los diagramas P&ID, existe una herramienta dedicada a la elaboración de diagramas dentro de la gran solución que es *AutoDesk,* la cual se denomina AutoCAD P&ID, de la misma manera enfocándose en la versión 2015.

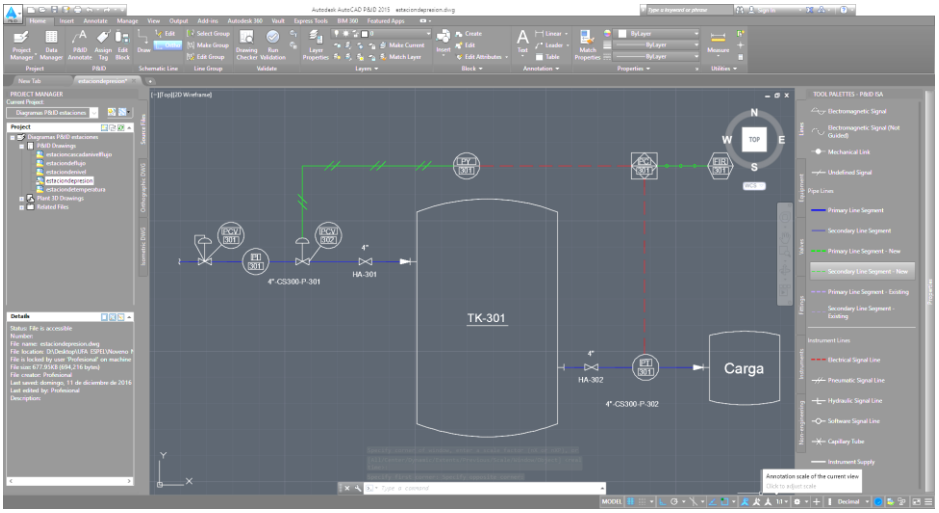


Figura 2.5. Interfaz AutoCAD P&ID.
Fuente: Autor

Una de las principales ventajas que presenta el AutoCAD P&ID es la posibilidad de trabajar bajo normativas internacionales como: *Process Industry Practices* (PIP), *International Organization for Standardization* (ISO), *Instrument Society of America* (ISA), *Deutsches Institut fur Normung* (DIN), *Japanese Industrial Standards* (JIS-ISO); hecho que garantiza la compatibilidad y correcta aplicación de las normas a los ojos del usuario.

2.2.1. Transformación de 2D a 3D

Al tener disponibles los diagramas P&ID como elemento referencial de las estaciones a representar, es importante tener en cuenta que se trata de planos en dos dimensiones, por lo que los archivos generados deben ser procesados para que se puedan utilizar en el motor gráfico que desarrollará el entorno. Por tal razón, se consideran una serie de procesos hasta la obtención del modelo final en tres dimensiones. La Figura 2.6, presenta el diagrama de bloques de la transformación de los diagramas P&ID de dos a tres dimensiones.

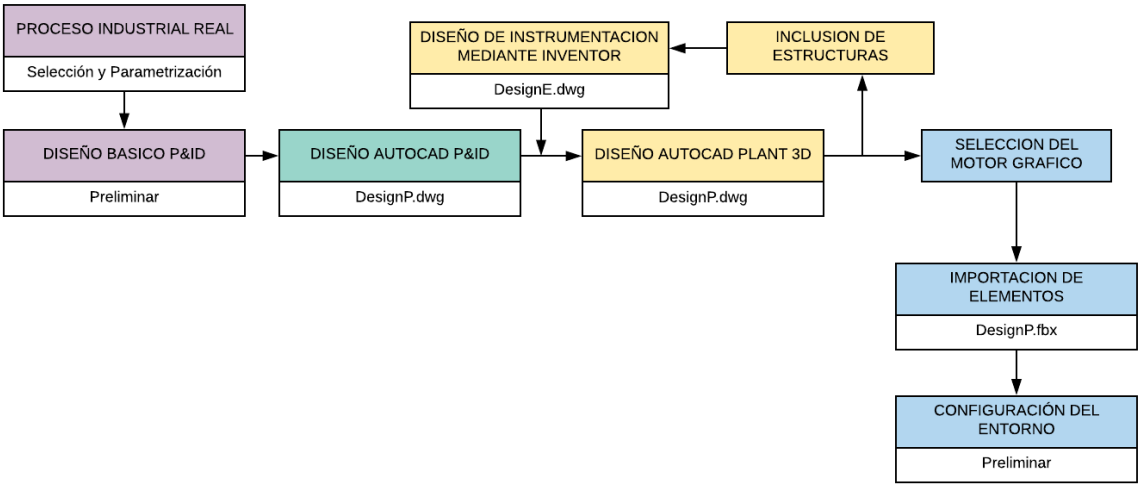


Figura 2.6. Transformación de dos a tres dimensiones de un proceso industrial.
Fuente: Autor

Como se puede observar en la Figura 2.5., la obtención del diagrama P&ID es parte de la etapa inicial del proceso de transformación. Para la siguiente etapa es necesario la utilización de otra de las herramientas con las que cuenta *AutoDesk*, como es AutoCAD *Plant 3D*, el cual a partir de un diagrama desarrollado con toda la normativa correspondiente en *AutoCAD P&ID*, importa una lista con los objetos correspondientes a todos los elementos que componen el diagrama. Además, cuenta con la capacidad de ofrecer al usuario dos formas para implementar la disposición física de las tuberías, siendo estas automáticas o manuales.

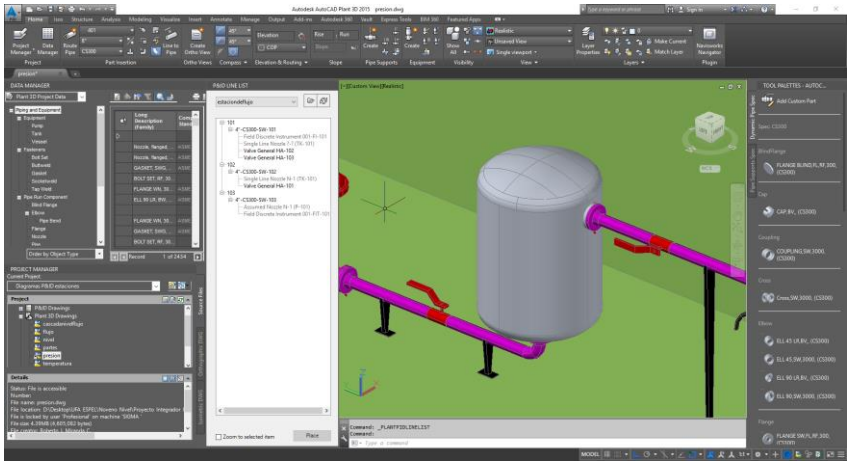


Figura 2.7. Interfaz AutoCAD Plant 3D con lista de elementos.
Fuente: Autor

Es necesario acotar que, AutoCAD Plant 3D posee una herramienta de generación automática de los elementos que componen un sistema de tuberías como son: T's, codos con cualquier tipo de ángulo, reductores, etc. Dentro del modo automático de generación, solo se debe seleccionar el punto inicial y final por donde debe pasar la tubería, así como también el diámetro nominal y el material de la misma; con dichos datos el programa desplegará una lista de opciones, donde el usuario estará en la posibilidad de elegir la disposición que más se acople a su espacio físico.

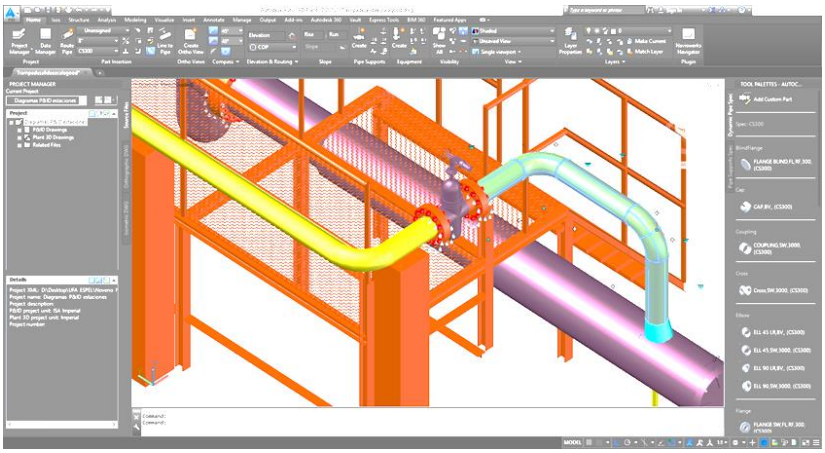


Figura 2.8. Modo automático de conexión de tuberías.
Fuente: Autor

Por otro lado, en el modo manual, la disposición física de las tuberías dependerá del sentido y orientación que el usuario ingrese, tal como se muestra en la Figura 2.9; de esta manera, se brinda al usuario la posibilidad de comparar que opción de generación de tuberías se adapta de mejor forma a sus necesidades. Otra opción que el programa ofrece es el de trazar una serie de líneas dentro del espacio de trabajo tridimensional para después por medio de un comando convertir todas estas líneas en tuberías.

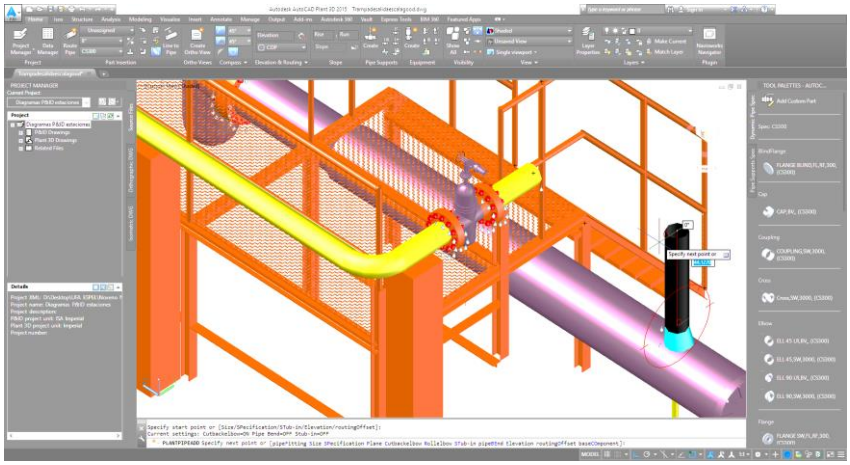


Figura 2.9. Modo manual de conexión tuberías.
Fuente: Autor

Si bien AutoCAD Plant 3D, posee en su base de datos una gran cantidad de los elementos que se encuentran en cada diagrama P&ID, es importante mencionar que no contiene ningún instrumento en absoluto, pero brinda la oportunidad de importarlo, sea al momento de crear el proyecto como tal o durante la ejecución del mismo. Es en esta etapa se debe recurrir a la utilización de otro tipo de software más especializado para la creación de este tipo de elementos como son: transmisores, controladores lógicos programables, sensores, densitómetros, tableros de conexiones, etc.

Para la generación de los instrumentos anteriormente mencionados, se optó por utilizar otro programa de la familia Autodesk como es Inventor. Este programa brinda las características básicas de AutoCAD en cuanto a diseño y modelado de elementos tridimensionales, con la diferencia que incorpora paletas de instrucciones especialmente creadas para la generación de instrumentos; además, está en la capacidad de crear piezas de manera individual, las mismas que posteriormente se ensamblarán en un prototipo final. Las Figuras 2.10a, 2.10b y 2.10c indican el resultado obtenido de un instrumento modelado y ensamblado con el software Inventor frente a un instrumento real.

a)

b)

c)

Figura 2.10. Instrumentos Virtuales vs. Reales a) Transmisor de presión b) Transmisor de temperatura c) Sensor de flujo ultrasónico.

Fuente: Autor

Los instrumentos creados en Inventor van a ser importados en el AutoCAD Plant 3D, para posteriormente ingresarlos en la base de datos y que se encuentren disponibles en la lista de elementos a utilizar para el presente proyecto.

Las estructuras físicas que forman parte del sistema, representado las escaleras, las bases, los soportes de tubería, las barandas, etc., son creadas a partir de elementos más básicos tales como las columnas de acero de diferentes tipos, soportes de concreto, planchas metálicas, mallas y demás, disponibles dentro del mismo programa. Todo esto con el fin de que el modelo virtual se apegue lo más posible a las características físicas del modelo real, tal y como se puede observar en la Figura 2.11.

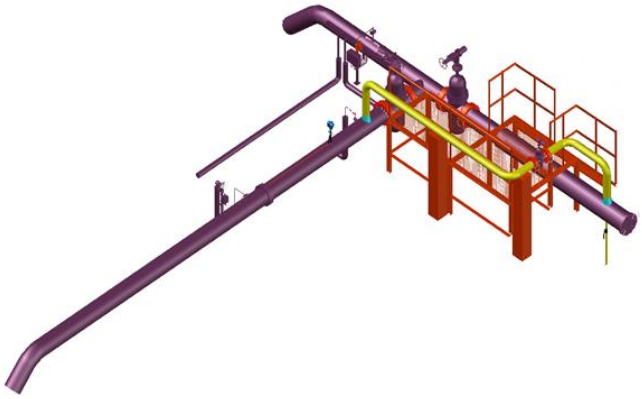


Figura 2.11. Modelo en 3D de una estación añadido estructuras físicas extra.
Fuente: Autor

2.3. Dispositivos de entrada y salida utilizados

El mercado actual ofrece una amplia gama de dispositivos dentro del ámbito de la realidad virtual y realidad aumentada, los cuales presentan características propias que aportan a que el usuario alcance un mayor grado de inmersión e interacción con la aplicación desarrollada.

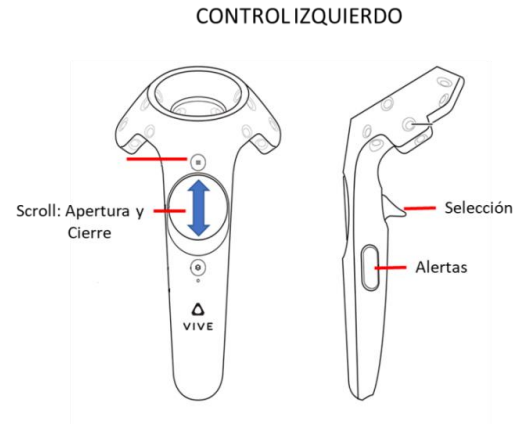
En el caso específico del proyecto desarrollado, la solución seleccionada son las gafas de realidad virtual HTC VIVE, las cuales son fabricadas por las empresas HTC y Valve.

Figura 2.12. Gafas de Realidad Virtual HTC VIVE.
Fuente: VIVE VR System (HTC VIVE, 2018)

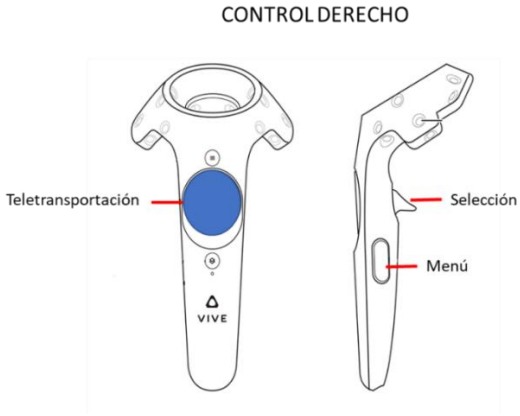
Dentro del sistema de entrenamiento, los dispositivos de HTC funcionan en conjunto como las entradas y salidas del mismo, ya que a través del *Head Mounted Display* (HMD), se visualiza todo el entorno y las interacciones que son propiciadas gracias al operador, mientras que los controles son utilizados como entradas, los cuales a través de la utilización de los botones y sensores gestionan la manipulación tanto de elementos como de datos dependiendo de la programación asignada.

La facilidad de integración que presenta el motor gráfico respecto a la utilización de dispositivos de realidad virtual como son las gafas HTC VIVE es que, existen desarrollados varios demos de la programación básica de los controles, que se puede aplicar al proyecto que se esté desarrollando o bien permite mediante la implementación de librerías y *"prefabs"* (plantillas), realizar la configuración de los controles y su respectiva programación de acuerdo a las necesidades del usuario y del desarrollador.

Para el presente proyecto se ha hecho uso de los distintos métodos de entrada que disponen las gafas de RV, como son: Gatillo, Grip, TouchPanel, sensores de movimiento; para que cada uno de ellos cumpla con una función en específico al momento de realizar la interacción del usuario con el entorno. La Figura 2.13a y la Figura 2.13b muestran las acciones que ejecutarán los controles dentro del entorno de entrenamiento.



a)



b)

Figura 2.13. Configuración de controles a) control izquierdo, b) control derecho.
Fuente: VIVE VR System (HTC VIVE, 2018)

2.4. Ambiente simulado

El desarrollo del entorno de entrenamiento tiene como principal finalidad la recreación de las actividades y procesos que se efectúan en un proceso industrial real. Este motivo conlleva a pensar en un software capaz de brindar todas las facilidades tanto para el desarrollador o programador de generar el mundo virtual; así como al operador, en la participación y manipulación de los elementos para que la experiencia de usabilidad sea correcta.

Unity 3D entre varias de sus ventajas, como la facilidad de uso de varios lenguajes de programación para la creación del entorno también aporta una compatibilidad directa con un sin número de softwares y aplicaciones, de los cuales se pueden importar los distintos

elementos y complementos que llevan al desarrollo virtual a conseguir un mayor realismo y autenticidad.

El sistema de escenas que se dispone dentro de Unity 3D, permite que se desarrollen varios entornos dentro de un mismo proyecto. Sin embargo, se ha decidido utilizar una escena macro que contenga las dos estaciones que se representan por fines de un continuo intercambio de datos y mejor desempeño de las acciones establecidas.

2.4.1. Desarrollo del ambiente geográfico

Para que la inmersión del sistema de entrenamiento sea adecuada, no solo se debe virtualizar la parte operativa de las estaciones de lanzamiento y recepción, sino que se debe incluir todo el ambiente que compone en la vida real a estos sistemas; es decir, se incluirá toda la parte visual necesaria (montañas, árboles, efectos de viento, etc.).

Con este fin Unity permite una personalización completa de un terreno que se cree en esta plataforma, permitiendo adaptar su funcionamiento de acuerdo a las exigencias del usuario o bien del desarrollador.

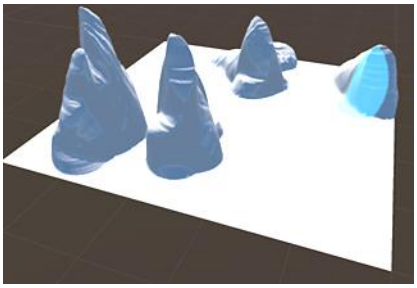


Figura 2.14. Generación de terrenos en Unity 3D.
Fuente: Autor

La herramienta asociada a los terrenos permite configurar las elevaciones que se vayan a encontrar dentro del terreno de manera muy fácil, comparado en funcionalidad a utilizar un programa de dibujo básico como Paint, con una barra de selección de formas, que tiene la posibilidad de configurar su tamaño y su textura, tal y como se puede observar en la Figura 2.15.

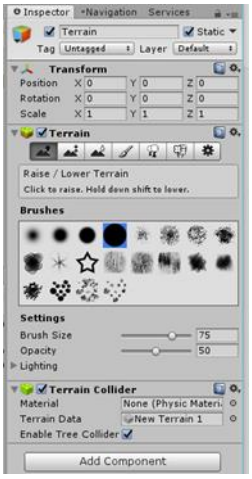


Figura 2.15. Paleta de herramientas para la generación de terrenos.
Fuente: Autor

Con la combinación apropiada de estas herramientas, la colocación de texturas, que se puede realizar análoga a pintar un lienzo y el incluir vegetación característica de acuerdo a la ubicación geográfica que se quiere representar, consiguen un terreno realista que ayuda a la integración del entorno con el usuario e incrementa los niveles de satisfacción en cuanto a la utilización del sistema de entrenamiento. En la Figura 2.16., se puede observar la versión final del terreno creado para la aplicación.

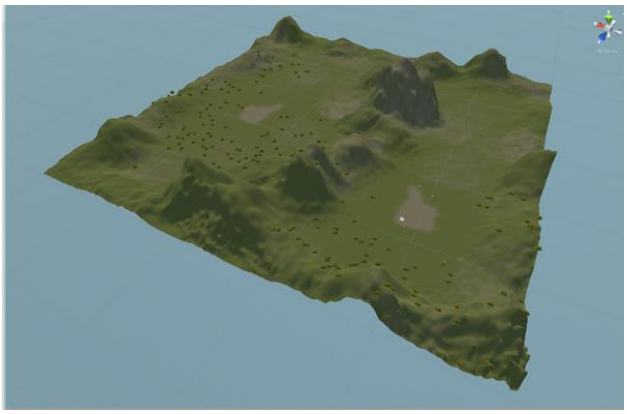


Figura 2.16. Diseño final de terreno dispuesto para el proyecto.
Fuente: Autor

Los modelos en tres dimensiones que fueron desarrollados a partir de los diagramas P&ID, gracias a las características propias que presenta Unity, son importados directamente a través de un .FBX, el cual cuenta con una textura base y queda listo para utilizarlo por el desarrollador, tal y como se indica en la Figura 2.17.

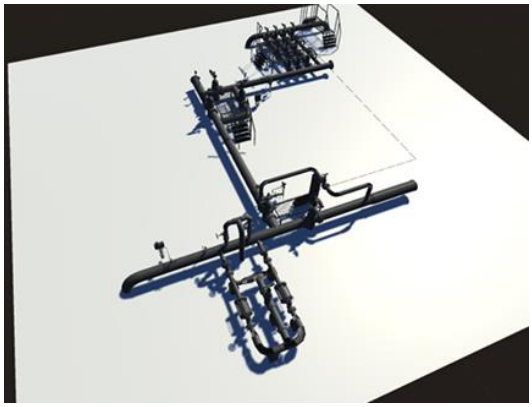


Figura 2.17. Importación de estructuras y modelos 3D en Unity.
Fuente: Autor

Tal y como se ha mencionado, la inmersión es uno de los detalles que mayor relevancia debe tener en el entorno de entrenamiento. Por tal razón, es importante tratar a cada una de las estructuras y modelos importados en Unity 3D, para que contengan las texturas, colores y formas apropiadas, comparadas con el ambiente original de referencia. La Figura 2.18. presenta una de las estaciones importadas, aplicadas las texturas y correcciones de posición finales dentro del entorno de entrenamiento.

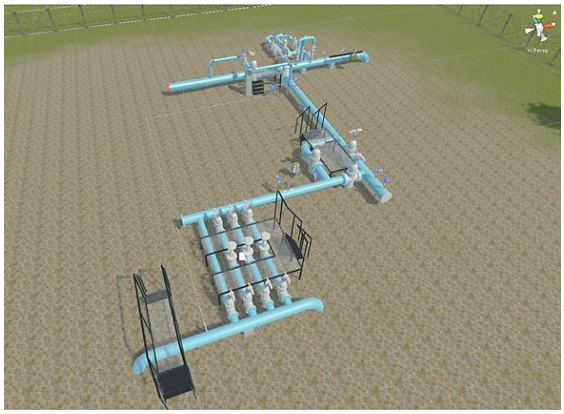


Figura 2.18. Texturas y colores colocados a los modelos en 3D.
Fuente: Autor

Parte del entorno geográfico también se considera a la señalética de seguridad implementada, elementos que en una estación real son normados internacionalmente y permiten un adecuado desempeño de los operadores y generan garantías para el personal, medio ambiente y equipos que componen al sistema en su totalidad.

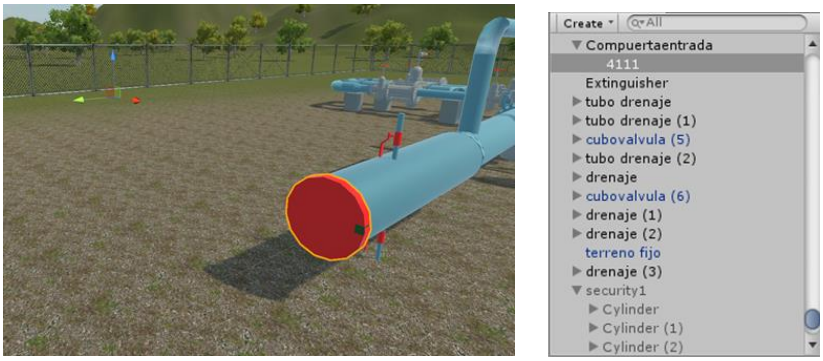
Dentro del entorno de entrenamiento, estos elementos se insertan a manera de complementos *User Interface*, con esto se consigue la incorporación de letreros de

seguridad que brindan información relevante a tener en cuenta por parte de los usuarios en el desarrollo de sus actividades. De igual forma una cerca de seguridad correspondiente se agrega en el contorno de la estación delimitando así las zonas de libre tránsito con los sitios exclusivamente de manipulación por personal autorizado. En la Figura 2.19. se muestra los elementos de seguridad y la señalética incorporada en el entorno de entrenamiento.

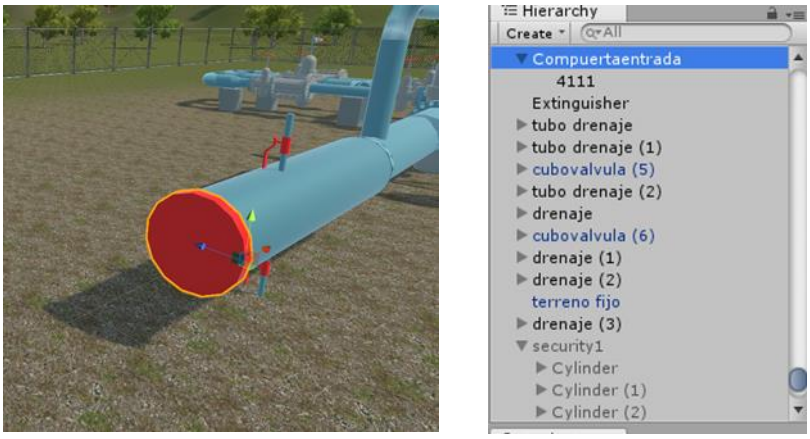
Figura 2.19. Elementos de seguridad dentro del entorno.
Fuente: Autor

2.4.2. Modificación de elementos para manipulación

Dentro de los modelos, existen varios ajustes que se deben realizar a los elementos para que puedan cumplir con las funciones que se les asignara mediante programación de scripts o bien con la ayuda de animaciones; entre ellas una de las más representativas es la corrección de pivote y centro de cada elemento ya que como se ha explicado, el sistema de entrenamiento tendrá interacción directa del usuario con válvulas y compuertas, las mismas que cumplen con ciertos movimientos. Para lograr tal cometido, se debe crear elementos vacíos, que apadrinen a las piezas del modelo original y ellas sean las que determinen el punto de pivote o centro del elemento en estudio, según las necesidades del desarrollador, tal y como se muestra en las Figuras 2.20a y 2.20b.



a)

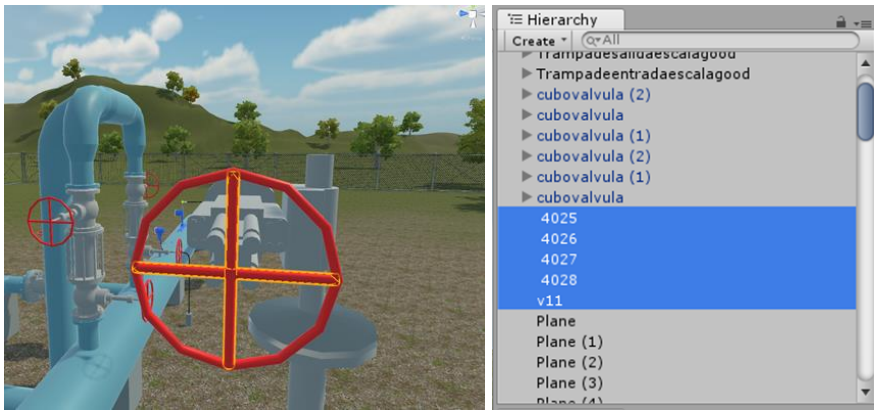


b)

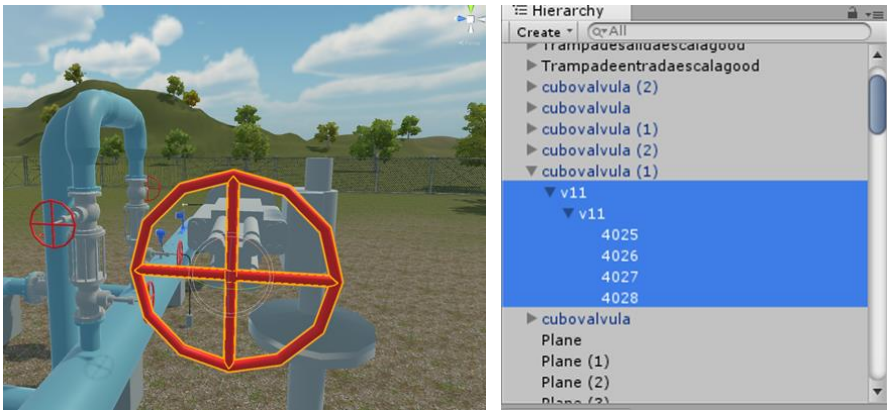
Figuras 2.20. a) Elemento sin corrección de punto de centro y pivote, b) Elemento corregido el punto de centro y pivote.
Fuente: Autor

De manera similar, las válvulas que se obtienen a partir del modelo 3D importado en Unity no son un cuerpo sólido definido, sino que más bien, son varias piezas individuales; motivo por el cual, al realizar una interacción con ellas no trabajan como un conjunto, sino que reaccionan de manera inapropiada.

En este caso la jerarquización de elementos se convierte en el punto a considerar, ya que, mediante este arreglo, se consigue que el componente deseado adquiera las características propias de funcionamiento. Para esto, el agrupamiento de los elementos puede realizarse tal y como se indicó anteriormente, creando un elemento vacío o bien tomando como referencia una de las piezas individuales del modelo original y anclando las restantes para lograr un comportamiento en conjunto, como se muestra en la Figura 2.21ª. y 2.21b.



a)



b)

Figura 2.21. Configuración gráfica del actuador de válvula. a) Válvula representada por piezas individuales, b) Válvula representada por un conjunto jerarquizado.
Fuente: Autor

2.4.3. Incorporación de componentes de inmersión

Tanto la inmersión como la interacción del usuario con el entorno deben ser tomadas en cuenta al momento de desarrollar un entorno de entrenamiento. Por lo tanto, el aspecto visual del ambiente debe estar plenamente complementado con la parte auditiva característica del sistema desarrollado; en este caso, para el transporte de crudo se utiliza una serie de bombas y motores, así como el flujo de crudo propiamente dicho; cuyos sonidos deben ser representados y colocados de manera estratégica para aumentar el grado de realismo al entorno desarrollado.

Por ejemplo, en zonas críticas del entorno, como pueden ser los cuartos de máquinas, se ha dispuesto de un objeto que contiene un archivo de audio, el cual representa el sonido característico de los motores y bombas activadas y funcionando. Los mismos que al tener una configuración de 3D permite la aplicación de un efecto *doppler* al elemento, plenamente configurable. De esta manera, se obtiene un mayor realismo e inmersión en el entorno. La Figura 2.22. muestra la disposición física del elemento de audio y su configuración básica.

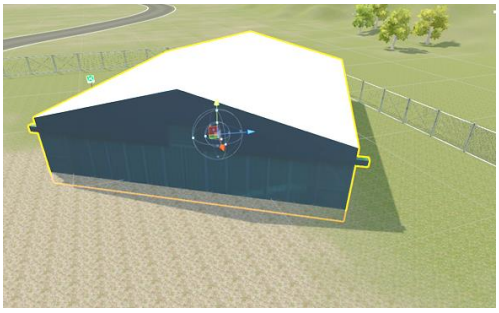


Figura 2.22. Implementación de sonido en cuarto de máquinas.
Fuente: Autor

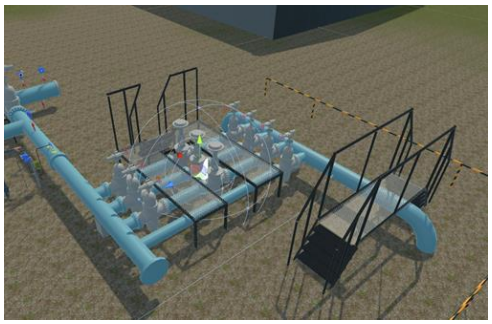
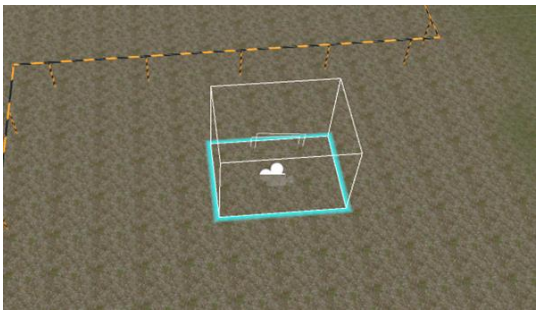


Figura 2.23. Implementación de sonido envolvente y configuración 3D.
Fuente: Autor

La utilización de las gafas de realidad virtual dentro del motor gráfico genera de manera automática una zona de trabajo de 4 x 3 metros que, si bien da un buen desenvolvimiento, al trabajar en una estación industrial es necesario trasladarse y desplazarse a través de todo el entorno. Por lo tanto, como solución de movilidad se implementa un sistema de tele transportación, el mismo que a través de una interacción del usuario con el control en su *"touchpad*", genera un "laser", el cual identifica la zona a donde se redirigirá la persona, tal como se muestra en la Figura 2.24., desplazando la zona inicial de 4 x 3 metros a lo largo de todo el entorno; creando una mejor experiencia.



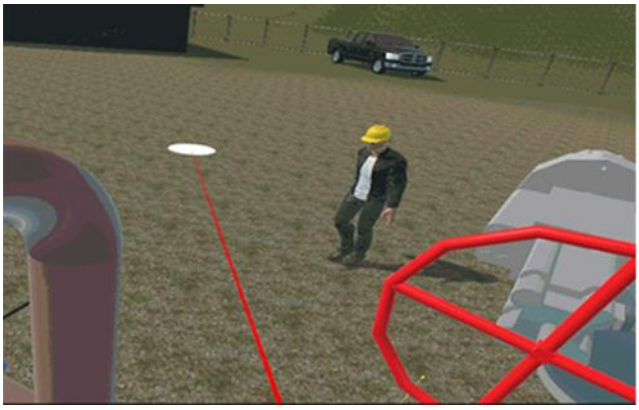


Figura 2.24. Sistema de movilidad implementado.
Fuente: Autor

2.4.4. Interacción generada entre usuario y entorno

El sistema de entrenamiento desarrollado tiene como principal característica la interacción directa del usuario con los elementos primarios como son válvulas y compuertas, pero el análisis principal debe realizarse en la manipulación del raspador de tuberías. Este elemento no tiene un modelo estandarizado dentro de la industria. Pero su construcción tiene un nivel de importancia elevado de acuerdo al grado de interacción que se requiera.

Para la creación del raspador de tubería se utiliza elementos propios generados en Unity, agrupados y jerarquizados para formar un cuerpo uniforme, dando como resultado una estructura como la presentada en la Figura 2.25.

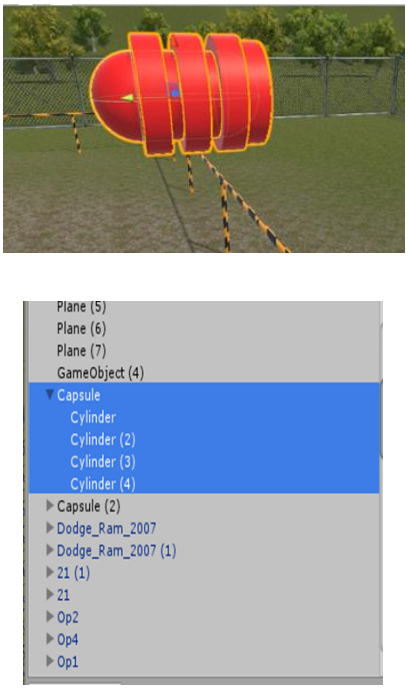
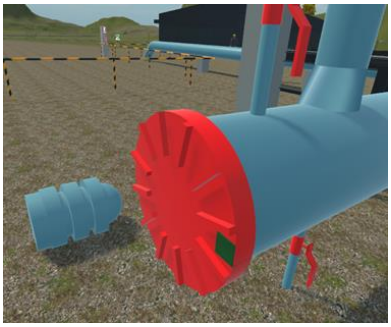


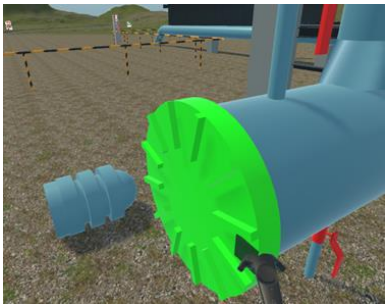
Figura 2.25. Construcción de raspador de tubería.
Fuente: Autor

Para que el desarrollo del entrenamiento se realice de manera intuitiva, como parte de la programación implementada, se ha establecido un cambio de color que indica la interacción del usuario con un componente del entorno, en este caso, las válvulas principales de cada estación, así como, las compuertas hacia las cámaras de lanzamiento y recepción del raspador.

El script que controla la manipulación de objetos permite que en el instante que se realiza una colisión del control con el objeto, se cambie el material dispuesto en las válvulas o las compuertas; por lo que su color original que es rojo, cambiará a verde mientras se realiza la interacción con dicho elemento. Una vez terminada la manipulación del objeto, se restablece el color original del elemento, tal como se muestra en la Figura 2.26a y 2.26b.



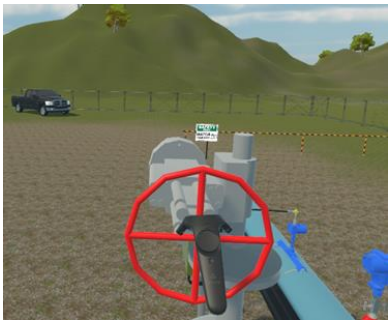
a)



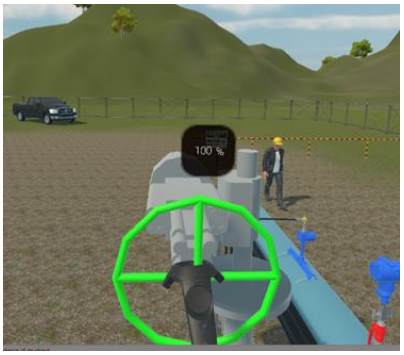
b)

Figura 2.26. Configuración de manipulación de elementos a) condiciones normales, b) cambio de color de elemento a manipular.
Fuente: Autor

Una correcta interpretación del funcionamiento es vital en el entrenamiento del proceso, por tal motivo visualizar los porcentajes de apertura de las válvulas manipuladas ayuda a entender mejor el comportamiento del proceso y garantiza que el protocolo de capacitación se realice de manera adecuado. Con este fin se coloca en cada válvula una pantalla indicativa del porcentaje de apertura de dicha válvula, la misma que se mostrará únicamente en el instante en que se tome contacto del control con la válvula. Esta acción se indica en la Figura 2.27a y 2.27b.



a)

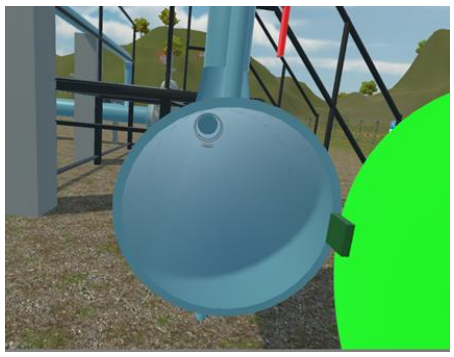


b)

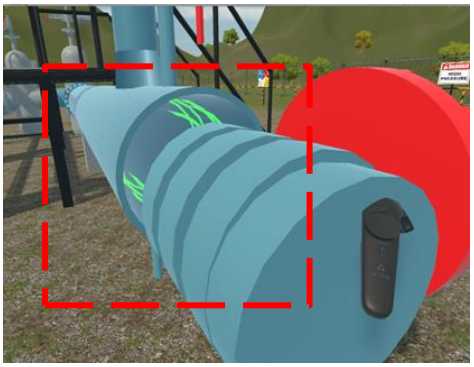
Figuras 2.27. a) Válvula sin interacción de usuario, b) Información de estado de las válvulas.
Fuente: Autor

El sistema de entrenamiento basa su funcionamiento en la preparación y manipulación del raspador de tuberías para su lanzamiento y posterior recepción. Este motivo genera la necesidad de representar la tarea de ingreso y extracción del raspador en las estaciones de lanzamiento y recepción respectivamente.

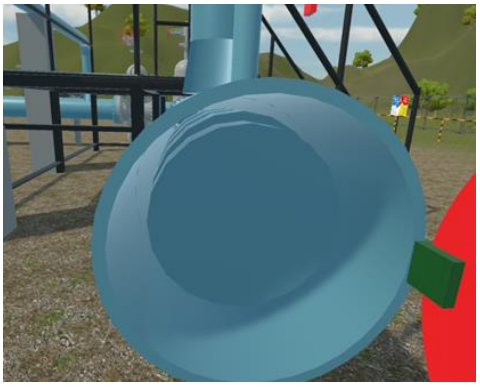
Con el fin de generar la experiencia adecuada en el usuario, se ha creado dos raspadores, los mismos que tienen la siguiente funcionalidad: el primero se encuentra fuera de la estación de lanzamiento, listo para su manipulación; el segundo raspador se mantiene de forma original invisible, una vez que se interactúa con el primer raspador, se muestra el segundo dentro de la tubería, representado con un material distinto y vistoso, demostrando así el sitio correcto de colocación del raspador que se manipula. Una vez que el primer raspador es colocado en su destino final, éste desaparece y el segundo raspador cambia de textura, generando de esta manera la animación deseada, como se indica en la Figura 2.28.



a)



b)



c)

Figura 2.28. Inserción de Raspador en la trampa a) apertura de compuerta, b) inserción de raspador, c) raspador dentro de la trampa.
Fuente: Autor

2.4.5. Consideraciones adicionales para desarrollo del entorno

Como uno de los aspectos de inmersión finales se toma en cuenta que la operación que un operador realiza dentro de una planta industrial no es la única tarea que se ejecuta en ese instante y que otros operadores desempeñan otras actividades de manera simultánea. Por tal motivo, y con la intención de incrementar el grado de inmersión del usuario, se establece la inclusión de avatares, los cuales han sido modelados e importados en Unity obteniendo como resultado lo que se muestra en la Figura 2.29.

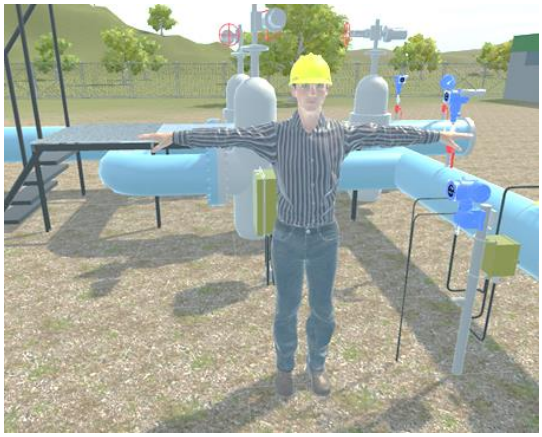


Figura 2.29. Avatar de operador industrial importado a Unity 3D.
Fuente: Autor

Estos operadores virtuales cumplirán la función de patrullaje dentro del entorno, generando en el usuario una mejor experiencia y aumentando el realismo en cuanto a las tareas cotidianas y el común funcionamiento de todo el sistema. Para ejecutar las rutas de patrullaje, es necesario colocar marcas a lo largo del terreno que representan los puntos principales a seguir por el avatar, como se indica en la Figura 2.30.

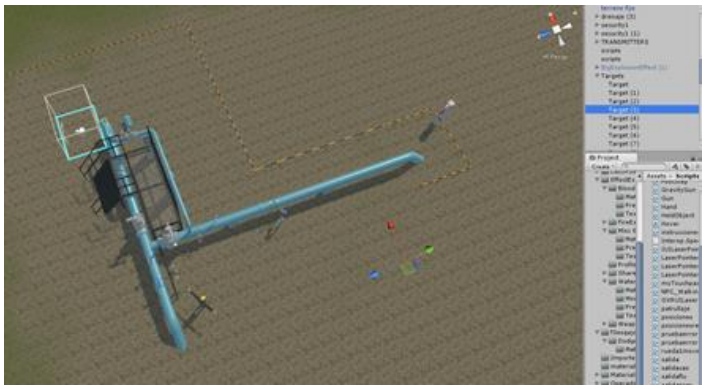


Figura 2.30. Colocación de marcas para el patrullaje de operadores extra
Fuente: Autor

Las marcas colocadas serán referenciadas dentro de un script de programación de movimiento aplicado a cada uno de los avatares en donde se reconoce la posición actual y se envía la información del siguiente punto para su avance, convirtiendo a la ruta creada en un circuito que será recorrido y repetido por el avatar, dejando como resultado lo que se plasma en la Figura 2.31.

Figura 2.31. Movimiento de avatares
Fuente: Autor

Se considera realismo a todos los detalles que se pueden incorporar dentro del entorno en comparación al funcionamiento normal de las estaciones. Por este motivo no solo se deben incorporar los efectos de un correcto procedimiento, sino que, además, es importante establecer los puntos de riesgo que se presentan en la operatividad y que llegan a ser el punto de inflexión entre un desempeño óptimo y una catástrofe.

Bajo este análisis se establece una situación de emergencia como parte de todas las opciones disponibles dentro del entorno de entrenamiento. Estas situaciones se activan bajo ciertos parámetros de mal funcionamiento por parte del usuario, que de no ser controlado puede alcanzar una repercusión incluso mortal para todo el entorno. Las principales emergencias se representan de manera visual para un mejor entendimiento del usuario y se conforman por cambios de color en las tuberías al existir elevada presión concentrada en un punto, o efectos visuales de derrame en otro tipo de evento. La Figura 2.32 muestra la secuencia en una situación de emergencia.

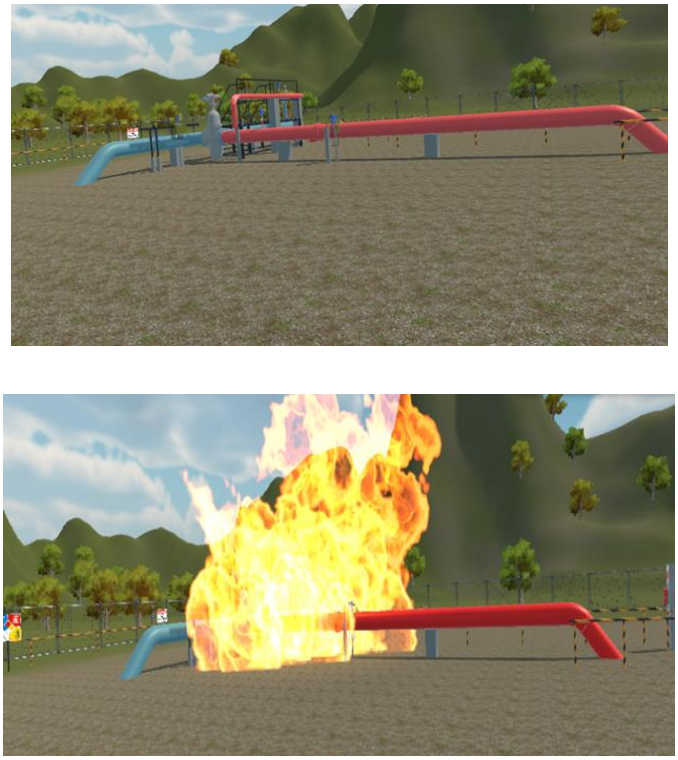


Figura 2.32. Simulación de emergencias dentro del entorno.
Fuente: Autor

3. HIDRODINÁMICA DE LA APLICACIÓN

Para ver el comportamiento de los componentes de la trampa lanzadora y receptora de raspadores antes, durante y después de las secuencias de operación correspondientes, se partirá de parámetros conocidos tomados de las estaciones de bombeo Lago Agrio (Trampa lanzadora) y Lumbaqui (Trampa receptora) pertenecientes al SOTE de EP PETROECUADOR:

- Línea principal de 20” API 5L X60
- Línea de Bypass de 8” API 5L X60
- Línea de Trampa de 20” API 5L X60
- Espesor de tubería: 0,334”
- Rugosidad Relativa $\frac{\varepsilon}{D}$: 0,00007
- Ubicación de la Trampa lanzadora (Estación A) 00.00 Km @296 msnm
- Ubicación de la Trampa receptora (Estación B) 63.00 Km @628 msnm
- Presión de salida de la estación A: 1800 PSI
- Presión de entrada a la estación B: 150 PSI
- Temperatura del crudo en ambas estaciones: 50ºC
- Densidad: 24 API, 910 Kg/m3 aprox.
- Flujo: 367000 BBLS/día (Capacidad de bombeo del SOTE) (EP PETROECUADOR, 2018)

3.1. Análisis matemático de las trampas lanzadoras y receptoras

Los cambios de las variables de proceso y de la señalización visual (presión, flujo, alarmas) serán una respuesta a la manipulación de las diferentes válvulas que componen las trampas. Para caracterizar matemáticamente el comportamiento de cada una de las válvulas junto con sus respectivas respuestas se asume lo siguiente para ambas estaciones (lanzadora y receptora):

- El flujo será ininterrumpido siempre que se cumpla con la secuencia normal de lanzamiento y recepción, por lo que la variable del medidor de flujo instalado en la línea principal antes de la trampa no se verá afectada en condiciones normales de operación.
- No hay diferencia de altura entre la entrada y la salida de la misma trampa.
- El flujo másico que entra es igual al que sale.
- No existe variación inmediata de densidad del producto entre la entrada y la salida.

De acuerdo a estas condiciones, el cambio visible para el operador dentro de un procedimiento normal será el de la variable de presión. Si el lanzamiento de raspador no es realizado en apego al procedimiento correcto, las variables afectadas y visibles para el operador serán la presión, el flujo y la situación de emergencia.

Para lograr que las respuestas estén apegadas a la realidad, la caracterización matemática deberá incluir todas las maniobras posibles (escenarios) que el operador pueda realizar, las operaciones correctas que no generarán situaciones de riesgo y las operaciones no seguras que podrán generar comportamientos atípicos e inseguros.

3.1.1. Escenario Inicial: Válvula B totalmente abierta (100% apertura), válvulas A y C cerradas (0% de apertura)

Esta condición es la inicial y de normal funcionamiento para ambas estaciones, es decir, se cumple cuando no se requiere lanzar o no se espera recibir el PIG. En este caso el flujo a través de las trampas tendrá un solo camino correspondiente a la línea principal y asociado a la válvula B de Bypass, de acuerdo a la Figura 3.1 y la Figura 3.2.

Tanto para la trampa lanzadora y receptora se considerará un flujo igual a Q=255BBL/min

DATOS:

$$Q_1 = 255\frac{BBL}{min} \times \frac{1\ min}{60s} \times \frac{1\ m^3}{6{,}29\ BBL} = 0{,}6757\left[\frac{m^3}{s}\right]$$

$$\emptyset_B = D_1 = 25.66" \rightarrow 25.66" \times \frac{1m}{39{,}37"} = 0{,}65\ [m]$$

$API = \frac{141}{GE} - 131.5$; *API=24 por lo tanto*

$$\rho = 910 \frac{Kg}{m^3}$$

$P_1 = 1800\ [PSI] \ \rightarrow \ 1800\ PSI \times \frac{6894{,}757\ Pa}{1\ PSI} = 12410562{,}6\ Pa$ (Lanzadora)

$P_{11} = 150\ [PSI]$ (Receptora)

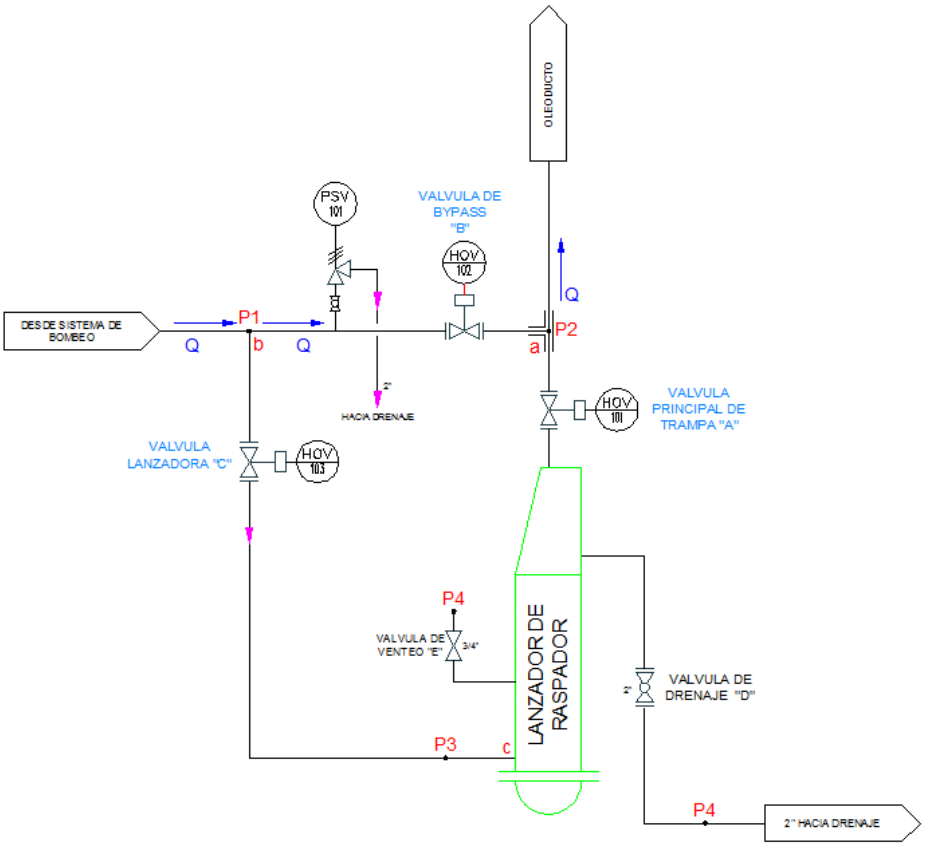


Figura 3.1. Válvulas C y A cerradas completamente- Trampa lanzadora.
Fuente: Autor

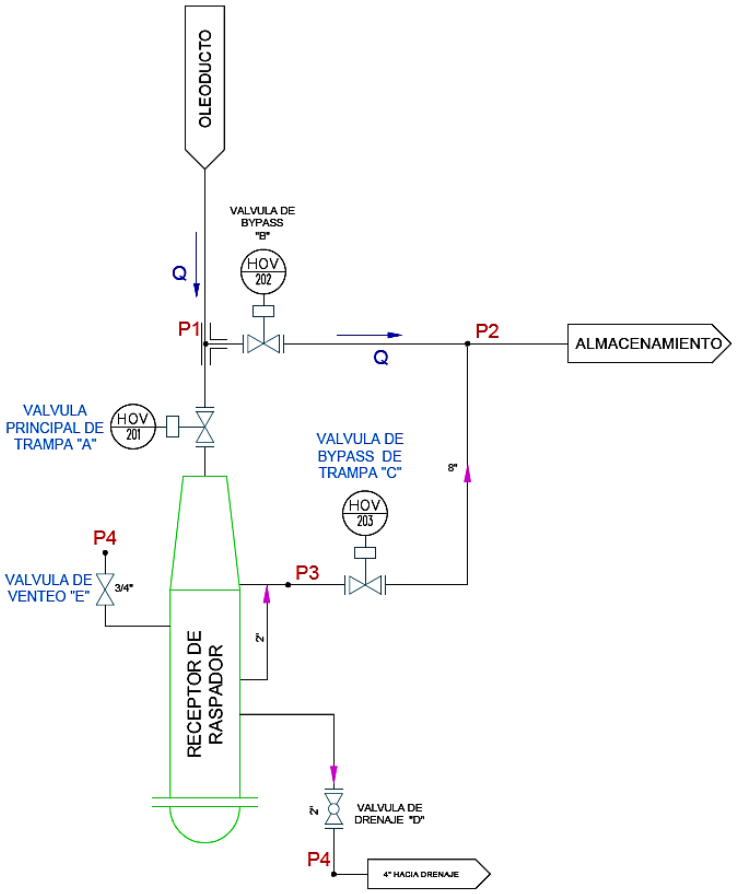


Figura 3.2. Válvulas C y A cerradas completamente- Trampa Receptora.
Fuente: Autor

Para esta condición, aplicando la ecuación de continuidad se tiene:

$$\rho_1 A_1 V_1 = \rho_2 A_2 V_2$$

Considerando que $\rho_1 = \rho_2$, la expresión se reduce a “El caudal que ingresa es igual al caudal sale”

$$A_1 V_1 = A_2 V_2$$

$$Q_1 = Q_2 = Q$$

Aplicando la ecuación de la conservación de la energía, que tiene tres componentes: energía de flujo, energía cinética y energía potencial gravitatoria se obtiene:

$$\frac{p1}{\rho} + \frac{{V_1}^2}{2} + gz1 = \frac{p2}{\rho} + \frac{{V_2}^2}{2} + gz2 + gH$$

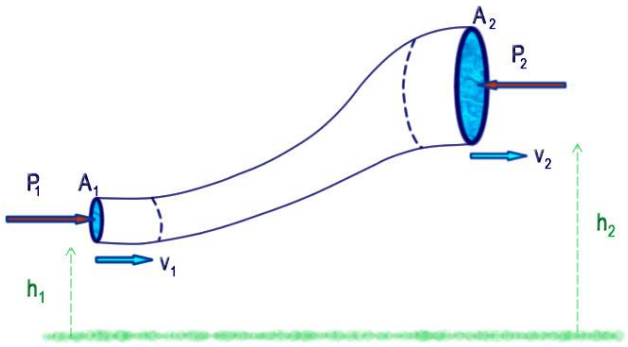


Figura 3.3. Sistema para un flujo sin derivaciones.
Fuente: Autor

- No hay variación de diámetros en el ramal, por tanto las velocidades son iguales
- No hay variación de altura, z1=z2

$$\frac{p1}{\rho} = \frac{p2}{\rho} + gH$$

$$P2 = P1 - \rho gH \quad \boldsymbol{Ec.1\ (General)}$$

Donde H es la pérdida de carga propia del tramo y sus accesorios, pérdidas regulares más pérdidas singulares, y está definida como el producto de la altura cinética por un coeficiente de pérdidas k_T, que para este escenario H=H1

$$H_1 = k_{1T}\frac{{V_1}^2}{2g} = \left(f\frac{L_B}{D_B} + k_B\right)\frac{8Q^2}{\pi^2 g {D_B}^4} \quad \boldsymbol{Ec.2}$$

Dónde:

$$k_{1T} = \left(f\frac{L_B}{D_B} + k_B\right)$$

D es el diámetro interno de la tubería

f es el factor de fricción (adimensional) que se le puede calcular con la ecuación de Colebrook-White:

$$\frac{1}{\sqrt{f}} = -2Log\left(\frac{\frac{\varepsilon}{D}}{3.7} + \frac{2.51}{Re.\sqrt{f}}\right) \quad \boldsymbol{Ec.3}$$

$f\frac{L}{D}$ es el coeficiente de pérdida por fricción de acuerdo a la ecuación de D'Arcy-Weisbach.

Y k_B es el coeficiente de resistencia de la válvula B que define las perdidas por fricción en términos de altura o presión (Smith, 2004)

Al cerrar parcial o totalmente cualquiera de las válvulas de la trampa se generará una diferencia de presión entre la entrada y la salida debido el estrangulamiento del fluido y estará relacionada al porcentaje de apertura/cierre de la válvula.

Para las válvulas de compuerta totalmente abiertas y en condiciones de flujo turbulento, se tiene un valor de k=0.1 – 0.3. (Smith, 2004)

Para obtener los valores de K_B cuando la válvula está parcialmente abierta se debe aplicar un factor de corrección **k1** de acuerdo a la siguiente gráfica (Smith, 2004)

$$K_B = Kk_1$$

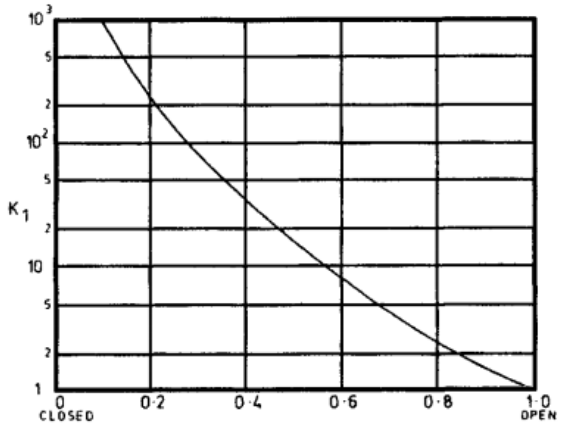


Figura 3.4. Gráfica para encontrar el valor k1 en válvulas de compuerta.
Fuente: Valve selection Handbook (Smith, 2004)

Por lo tanto:

$$k_B = 0.2k_{1B}$$

K1=1 cuando la válvula está totalmente abierta, y conforme se vaya cerrando K1 irá incrementando o viceversa.

Reemplazando estos términos en la Ec.2, se puede calcular las pérdidas de carga H1 para este escenario (en metros)

$$H_1 = \left(f\frac{L_B}{D_B} + 0{,}2k_{1B}\right)\frac{8Q^2}{\pi^2 g D^4} \quad \boldsymbol{Ec.4}$$

3.1.2. Escenario 2: las 3 válvulas de las trampas se encuentren con un porcentaje de apertura diferente de cero (0%)

En este caso, se tendrá una configuración de tuberías en serie y en paralelo entre el punto b y a de acuerdo la figura 3.5. Cuyo ramal correspondiente a la válvula C (Lbc) estará en serie con el ramal correspondiente a la válvula A (Lca), y juntos estarán en paralelo con el ramal correspondiente a la válvula B (Lba). Esta configuración podrá darse tanto en la trampa de salida como en la de entrada.

DATOS:

$L_C = L_{bc} = 11{,}50\ [m]$

$L_A = L_{ca} = 7{,}0\ [m]$

$L_2 = L_A + L_C = 18{,}5\ [m]$

$L_B = 4{,}0\ [m]$

$\emptyset_C = 7.66" \rightarrow 7.66" \times \frac{1m}{39{,}37"} = 0{,}194[m]$

$$\emptyset_A = 19.66" \rightarrow 19.66" \times \frac{1m}{39{,}37"} = 0{,}499\ [m]$$

Para analizar lo que sucede entre el punto b y a (Volumen de control) se aplica la ecuación de la continuidad (conservación de masa):

$$\oint_{SC} \rho V dA + \frac{\delta}{\delta t} \oint_{VC} \rho dV = 0$$

$$\rho Q - \rho Q_1 - \rho Q_2 = 0$$

$$Q = Q_1 + Q_2 \quad \boldsymbol{Ec.5}$$

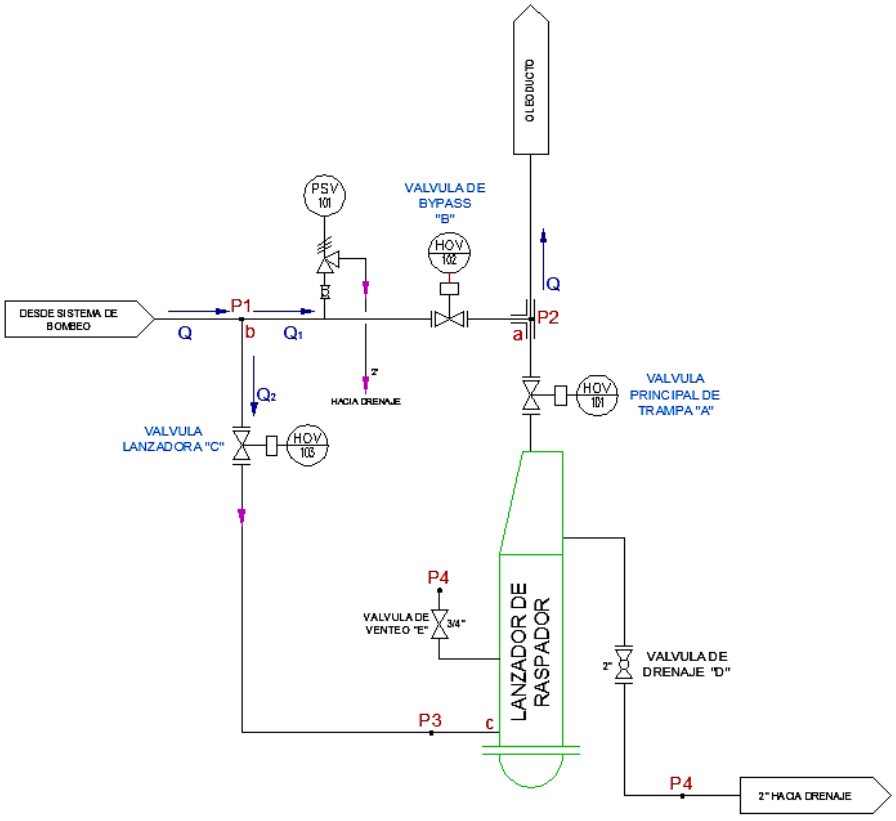


Figura 3.5. Válvulas A, B y C parcial o totalmente abiertas.
Fuente: Autor

Por lo tanto, cuando dos o más tuberías se colocan en paralelo, el caudal circulante total es la suma de los caudales individuales, y la pérdida de carga entre los extremos es la misma para todas las tuberías.

Se definirá un sistema equivalente al arreglo de tuberías, de tal manera que se pueda encontrar la pérdida de carga en metros del sistema en conjunto, para lo cual se conoce el caudal total que ingresa y sale del sistema, y se deducirá la pérdida H y diámetro D equivalente.

Se analizará las pérdidas de carga en cada ramal H1 y H2, y se igualará a una perdida general equivalente H

$$H_1 = H_2 = H = \frac{k_T V^2}{2g} = \frac{8 k_T Q^2}{\pi^2 D^4 g} \quad \boldsymbol{Ec.6}$$

$H_1 = \frac{k_{1T} {V_1}^2}{2g}$; $k_{1T} = coeficiente\ de\ pérdida\ equivalente\ para\ el\ ramal\ 1.$

$H_2 = \frac{k_{2T} {V_2}^2}{2g}$; $k_{2T} = coeficiente\ de\ pérdida\ equivalente\ para\ el\ ramal\ 2.$

$$k_T = f\frac{L}{D} + k_v$$

Donde

H1 son las pérdidas de carga (m) del tramo *ba*

H2 son las pérdidas de carga (m) del tramo *bca,* confomado por las perdidas de carga del tramo *bc* y *ca* (Hc y Ha respectivamente)

D es el diámetro equivalente del sistema

$f\frac{L}{D}$ son las pérdidas de fricción y k_v son pérdidas por válvulas.

Para la configuración de las dos tuberías en serie del tramo **bca (Lbc+Lca)** se tiene que:

- El caudal es el mismo a lo largo de la conducción: Q2

- La pérdida de carga de la conducción es igual a la suma de las pérdidas de carga de cada tramo de tubería: $H_2 = H_C + H_A$

$$\left(f\frac{L_2}{D_2}+k_2\right)\frac{{V_2}^2}{2g}=\left(f\frac{L_C}{D_C}+k_C\right)\frac{{V_C}^2}{2g}+\left(f\frac{L_A}{D_A}+k_A\right)\frac{{V_A}^2}{2g}$$

Donde kc y ka representan el coeficiente de resistencia de las válvulas de compuerta C y A respectivamente

$$k_C = 0.2k_{1C}$$

$$k_A = 0.2k_{1A}$$

Ley de continuidad

$$Q_2 = Q_C = Q_A$$

$$\left(f\frac{L_2}{D_2}+k_2\right)\frac{8{Q_2}^2}{g\pi^2{D_2}^4}=\left(f\frac{L_C}{D_C}+k_C\right)\frac{8{Q_2}^2}{g\pi^2{D_C}^4}+\left(f\frac{L_A}{D_A}+k_A\right)\frac{8{Q_2}^2}{g\pi^2{D_A}^4}$$

$$\left(f\frac{L_2}{{D_2}^5}+\frac{k_2}{{D_2}^4}\right)=f\left(\frac{L_C}{{D_C}^5}+\frac{L_A}{{D_A}^5}\right)+\frac{k_c}{{D_C}^4}+\frac{k_A}{{D_A}^4}\quad \boldsymbol{Ec.7}$$

$$\frac{L_2}{{D_2}^5}=\frac{L_C}{{D_C}^5}+\frac{L_A}{{D_A}^5}\quad \boldsymbol{Ec.8}$$

De la Ec.7 se puede encontrar el diámetro equivalente D2 en función de los datos conocidos: diámetros y longitudes de los tramos *bc* y *ca.*

De la Ec.6 se deduce y calcula *k2* que es el coeficiente de pérdidas equivalente para el tramo *bca*:

$$k_2 = {D_2}^4\left(\frac{k_c}{{D_C}^4}+\frac{k_A}{{D_A}^4}\right)\quad \boldsymbol{Ec.9}$$

Partiendo de la Ec.5 se deduce la pérdida equivalente H del sistema de esta manera:

$$AV = A_1V_1 + A_2V_2$$

Considerando que el Área es igual a:

$$A = \pi r^2 = \frac{\pi D^2}{4}$$

y que la velocidades se las puede expresar en función de las pérdidas de carga así:

$$V = \sqrt{\frac{2gH}{k_T}}; \quad V_1 = \sqrt{\frac{2gH_1}{k_{1T}}}; \quad V_2 = \sqrt{\frac{2gH_2}{k_{2T}}}$$

Se tiene que:

$$\frac{\pi D^2}{4}\sqrt{\frac{2gH}{k_T}} = \frac{\pi {D_1}^2}{4}\sqrt{\frac{2gH_1}{k_{1T}}} + \frac{\pi {D_2}^2}{4}\sqrt{\frac{2gH_2}{k_{2T}}}$$

Como las pérdidas de carga son iguales H=H1=H2

$$\frac{\pi}{4}\sqrt{2gH}\left(\frac{D^2}{\sqrt{k_T}}\right) = \frac{\pi}{4}\sqrt{2gH}\left(\frac{{D_1}^2}{\sqrt{k_{1T}}} + \frac{{D_2}^2}{\sqrt{k_{2T}}}\right)$$

$$\frac{D^2}{\sqrt{k_T}} = \frac{{D_1}^2}{\sqrt{k_{1T}}} + \frac{{D_2}^2}{\sqrt{k_{2T}}}$$

$$\frac{k_T}{D^4} = \frac{1}{\left(\frac{{D_1}^2}{\sqrt{k_{1T}}} + \frac{{D_2}^2}{\sqrt{k_{2T}}}\right)^2}$$

Reemplazando esta última expresión en la Ec.6 se tiene:

$$H = \frac{8\mathrm{Q}^2}{\pi^2 g\left(\frac{{D_B}^2}{\sqrt{k_{1T}}} + \frac{{D_2}^2}{\sqrt{k_{2T}}}\right)^2} \quad \boldsymbol{Ec.10}$$

Si no existe flujo por el ramal $\overline{bca}$ se regresa al escenario1 de la Ec.4 eliminando el término correspondiente a este ramal:

$$\frac{{D_2}^2}{\sqrt{k_{2T}}} = 0$$

Donde

$$k_{1T} = f\frac{L_B}{D_B} + k_B$$

$$k_{2T} = f\frac{L_2}{D_2} + k_2$$

Reemplazando la Ec. 9 en la última expresión se tiene:

$$k_{2T} = f\frac{L_2}{D_2} + {D_2}^4\left(\frac{k_C}{{D_C}^4} + \frac{k_A}{{D_A}^4}\right) \quad \boldsymbol{Ec.11}$$

Con los términos definidos se puede aplicar la Ec.1 general para encontrar la P2 de salida de la trampa lanzadora o P12 de la trampa receptora (en Pascales Pa).

$$P_2 = P_1 - \rho g H \ [Pa]$$

Para trabajar las presiones en unidades del sistema inglés (PSI) se debe aplicar el multiplicador de conversión $\frac{1}{6894{,}757}$

$$P_2 = P_1 - \rho g H \times \frac{1}{6894{,}757} \ [PSI]$$

Para calcular la presión P3 se debe calcular inicialmente Hc para la trampa lanzadora y Ha para la trampa receptora

$$P_3 = P_1 - \rho g H_C$$

$$H_C = \left(f\frac{L_C}{D_C} + k_C\right)\frac{8{Q_2}^2}{\pi^2 g {D_C}^4} \quad \boldsymbol{Ec.12}$$

$$H_A = \left(f\frac{L_A}{D_A} + k_A\right)\frac{8{Q_2}^2}{\pi^2 g {D_A}^4} \quad \boldsymbol{Ec.13}$$

Si se desea conocer la cantidad de flujo (en m3/s) que circula por cada ramal se deberá aplicar las siguientes ecuaciones:

$$Q_1 = \frac{\pi {D_B}^2}{4}\sqrt{\frac{2gH}{k_{1T}}} \quad \boldsymbol{Ec.14}$$

$$Q_2 = \frac{\pi {D_2}^2}{4}\sqrt{\frac{2gH}{k_{2T}}} \quad \boldsymbol{Ec.15}$$

3.1.3. Escenario 3: Válvula B totalmente cerrada (0% apertura), válvulas A y C abiertas.

Este caso puede darse al momento de lanzar el raspador, generando que el flujo sea direccionado totalmente por una sola vía, a través del barril de la trampa (lanzadora y receptora), con lo que se tendrá el mismo caso del escenario 1:

$$P2 = P1 - \rho g H \quad \boldsymbol{Ec.1}\ (\boldsymbol{General})$$

Salvo que H dependerá del coeficiente de pérdida del ramal 2, k_{2T}:

$$H = \frac{8k_{2T}\mathrm{Q}^2}{\pi^2 g {D_2}^4} \quad \boldsymbol{Ec.16}$$

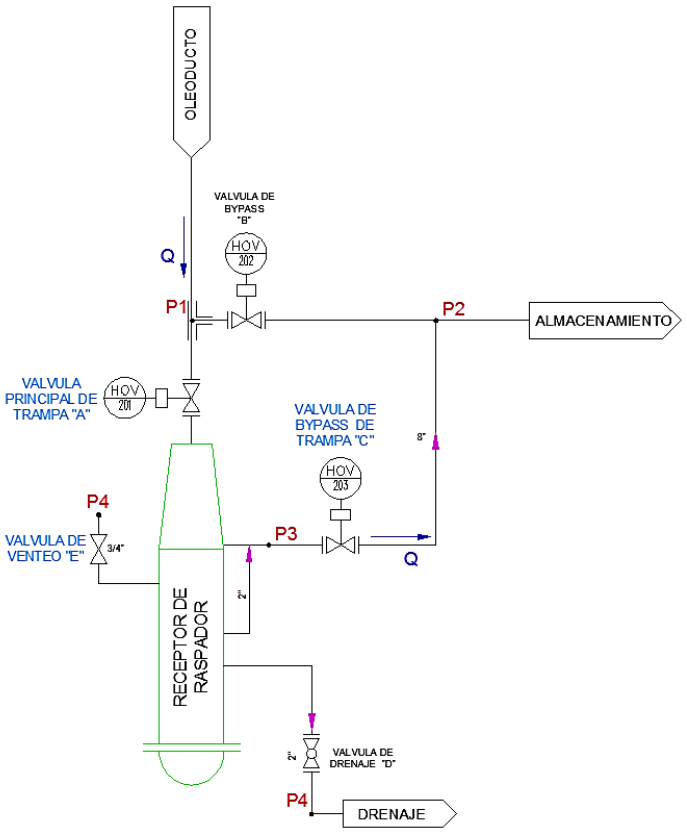


Figura 3.6. Válvulas A y C parcial o totalmente abiertas, válvula B cerrada.
Fuente: Autor

3.1.4. Escenario 4: Válvula de venteo y/o drenaje (D o E) tengan un porcentaje de apertura mayor a cero (0%)

Las líneas de venteo y purga son tuberías de 2” adaptadas al barril de la trampa, que permiten eliminar los residuos de producto, despresurizar y equilibrar la línea antes de proceder al lanzamiento y recepción de raspador. La línea de venteo desfoga los gases que pueden producirse a la atmósfera, y la línea de purga desfoga el producto hacia un tanque sumidero a través de un canal abierto.

En caso de que la línea lanzadora se encuentre presurizada y sin flujo al momento de abrir la válvula de venteo y/o purga, la presión P3 comenzará a caer hasta el valor de la presión atmosférica (0 PSIG).

En caso de que exista flujo a través de la línea lanzadora y se abran las válvulas de venteo o purga, se considerará el evento como si existiera una fuga (o robo) de producto a través de una línea de “pinchazo” u “ordeño” tal cual como se representa en la figura 3.6.

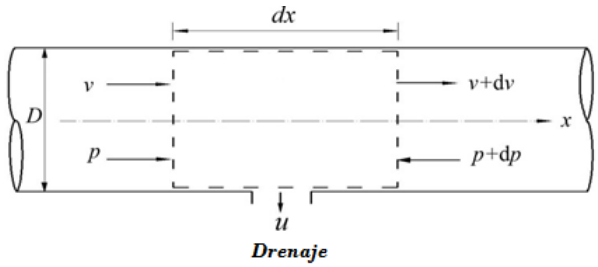


Figura 3.7. Equivalencia a un pinchazo de tubería.
Fuente: Autor

Al abrir la válvula de drenaje la presión en el barril tratará de igualarse a la presión atmosférica provocando una caída de carga que se verá reflejada en el transmisor correspondiente aguas arriba y aguas abajo. Esta onda de presión viajará en ambos sentidos de la tubería a la velocidad del sonido correspondiente al medio por la que viaja (crudo) y se irá atenuando conforme se aleja de la toma de purga, tal cual lo muestra la Figura 3.8a

Por otro lado, al momento de abrir la válvula de purga el flujo de producto antes de la toma aumentará levemente, mientras que el flujo después de la toma disminuirá de acuerdo a las Figura 3.8b.

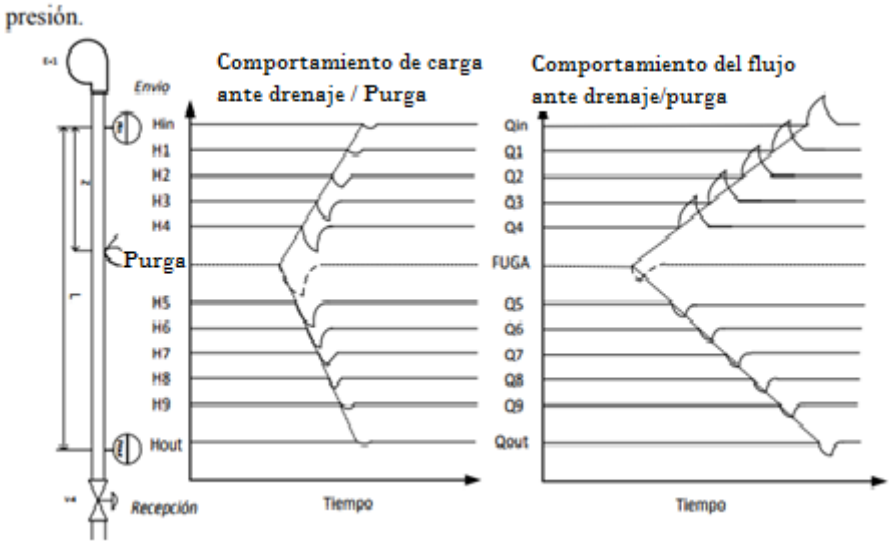


Figura 3.8. Comportamiento de carga y flujo ante la purga o drenaje.
Fuente: Autor

El sistema volverá a estabilizarse al momento que la válvula de drenaje se vuelva a cerrar.

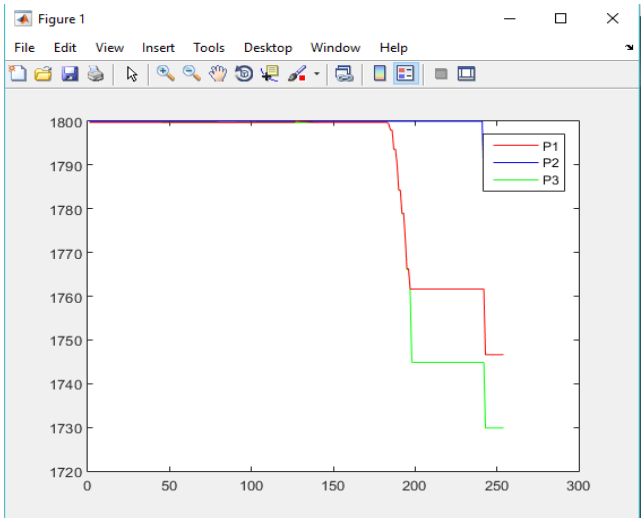


Figura 3.9. Comportamiento de carga simulado en Matlab al momento de drenaje.
Fuente: Autor

3.1.5. Escenario 4: Flujo=0 m3/s

Esta condición se da si el operador cierra la válvula B y cualquiera de las válvulas A o C, obstruyendo el paso de fluido hacia la siguiente estación. Al ocurrir este evento el sistema

alarmará al operador debido a la sobrepresión generada aguas arriba para que tome acción inmediata de abrir la válvula correspondiente y restablecer el flujo de producto.

Al no existir flujo la presión P2 en la trampa lanzadora será la resultante de toda la columna correspondiente al empaquetado del tubo desde la estación A hasta la estación B. Por lo tanto aplicando la ecuación de la conservación de la energía:

$$\frac{p1}{\rho} + \frac{{V_1}^2}{2} + gz1 = \frac{p2}{\rho} + \frac{{V_2}^2}{2} + gz2$$

P1=0 PSI (Presión atmosféricas en la estación B)

V1=V2=0 m/s

$$P_2 = g\rho(z1 - z2)$$

Donde z1= 628m y z2=296m

4. INTEGRACIÓN DE DATOS: AMBIENTE VIRTUAL Y CONDICIONES DE PROCESO

Para que la aplicación diseñada se apegue a la realidad se deben integrar en tiempo real los datos generados en el ambiente virtual 3D (gráficos, sonidos, objetos de interacción, etc) con los datos generados en la herramienta de procesamiento matemático.

A partir de las condiciones operacionales, variables involucradas y cálculos hidrodinámicos se recreará el entorno de operación, y la respuesta de variables a las acciones tomadas por el individuo inmerso en el ambiente virtual.

La integración e intercambio de datos funciona de tal manera que, si el operador acciona un equipo en el ambiente simulado, automáticamente las condiciones cambien y la respuesta pueda ser percibida a través de los instrumentos virtuales de la plataforma.

Las acciones que el usuario realiza para la puesta en marcha de la operación que se está recreando se traducen en escritura de datos dentro del procesador matemático (Matlab), quien será el encargado de realizar las compensaciones, cambios y efectos que se reflejarán en el funcionamiento del entorno de entrenamiento virtual.

4.1. Intercambio de datos en tiempo real

Para realizar el intercambio de datos en tiempo real se utiliza la herramienta de memoria compartida.

En sus inicios las memorias compartidas eran utilizadas para procesar grandes cantidades de información de manera distribuida, lanzando varios hilos de ejecución en varios nodos de una red, potenciando el poder de procesamiento de una forma significativa. Esta solución fue ampliamente utilizada en sistemas operativos Linux y/o BSD, uno de los motivos por los que se aprecia es que estos sistemas operativos empiezan a sobresalir en el área de *clustering*, servidores, control de paralelismo y soporte de concurrencia. En la actualidad estos sistemas operativos abarcan un gran porcentaje de servicios en internet y supercomputación.

Con el acceso a computadores con mayor poder de procesamiento, el uso de memorias compartidas enfocadas al desplazamiento y análisis de información en red ha sido

desplazado, dando lugar a la creación de localidades de memorias accesibles desde múltiples procesos, hilos y aplicaciones. Esto permite realizar proyectos complejos que requieren de análisis de una única información compartida bajo el manejo específico de cada aplicación. Entre los múltiples métodos de IPC que se encuentran en la actualidad hay que tener en cuenta que, desde el inicio de la informática, la memoria de un computador ha sido el medio de acceso más rápido, ya sean computadores de escritorio, portátil, mainframe, servidor y supercomputadores e.t, por eso podemos decir que la memoria compartida es el IPC más rápido disponible.

En sistemas distribuidos, múltiples procesos pueden interactuar, ya sean estos locales o entre nodos de una red, los métodos estándar POSIX de IPC se agrupan en tres mecanismos de intercambio de información y sincronización, (memoria compartida, cola de mensajes y banderas), además cuando se refiere a comunicación entre procesos se debe considerar las siguientes características: medio de comunicación, acceso de datos y sincronización.

4.1.1. Características de la memoria compartida

En la figura 4.1 se puede apreciar la interfaz de usuario en la cual se pueden agregar tantas memorias como se desee (1), en cada memoria a ser creada se agregan las características de: etiqueta (nombre) (2), cantidad (3) que hace referencia al número de localidades disponibles en los cuales se alojaran los datos en esa memoria y así mismo el tipo de dato que se va a almacenar (4).

Figura 4.1. Administrador de Memorias.
Fuente: Autor

Internamente con el barrido del *StringGrid* de la interfaz se obtiene la etiqueta de la memoria compartida, la cantidad de localidades de dicha memoria y el tipo de dato deseado, con lo cual se gestiona dinámicamente la creación de cada una de ellas a través de arreglo de punteros enlazados tipo HANDLE, como se muestra en la Figura 4.2, con esto se logra la característica de poder crear tantas memorias compartidas como el usuario necesite, siempre y cuando su tamaño total no exceda el tamaño de la memoria física libre.

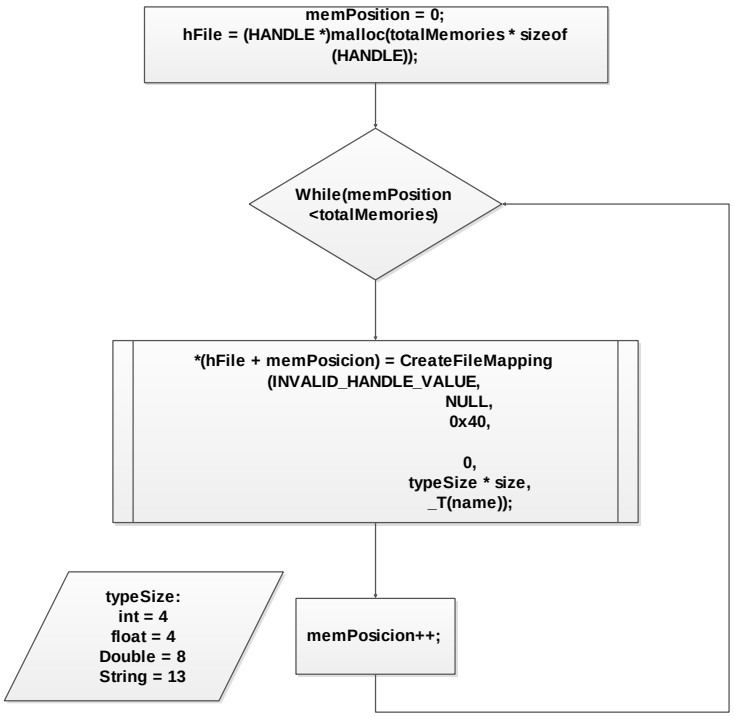


Figura 4.2. Diagrama de flujo del ciclo de creación de memorias.
Fuente: Autor

Acceso a la o las memorias creadas: La aplicación cliente puede obtener acceso a una o varias memorias compartidas que desee a través de las funciones alojadas en la librería de enlace dinámico **openMemory** (char *memoryName, int memoryType) en donde: *memoryName*: corresponde a la etiqueta especificada en el administrador gráfico; *memoryType:* se especifíca 1 = Integer, 2 = Floats, 3 = Doubles, 4 = String.

Internamente se instancia un nodo con la estructura de la Figura 4.3a para acceder a la memoria con la etiqueta indicada, luego de crear el apuntador a la memoria compartida,

se crea un manejador correspondiente al tipo de variables indicado (int, float, double, char*) para lograr el acceso a lectura y escritura. Figura 4.3b.

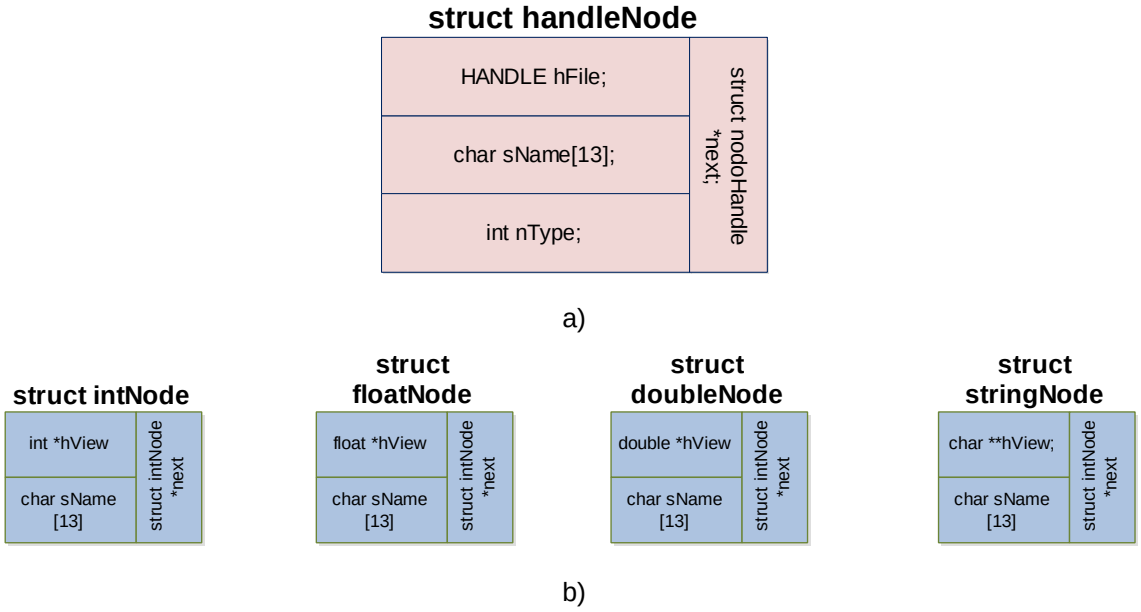


Figura 4.3. Acceso a la memoria creada- tipo de variables. a) Nodo tipo HANDLE, b) Nodos tipos de dato (entero, flotante, doublé y char *).
Fuente: Autor

Métodos de accesibilidad a las memorias creadas: La librería de enlace dinámico ofrece funciones de lectura y escritura para el acceso de cada memoria compartida abierta - enlazada como se muestra en la Figura 4.4, se agrega una función adicional para lectura de cadenas desde Matlab, específicamente para el manejo de datos tipo char*.

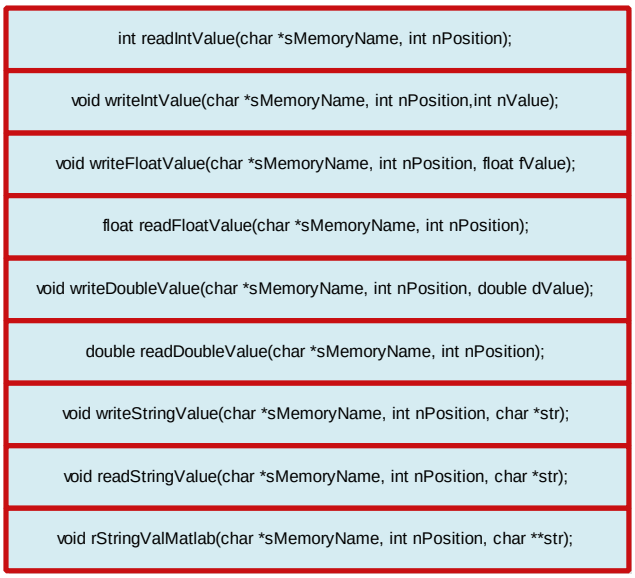


Figura 4.4. Funciones de lectura y escritura sobre memorias compartidas.
Fuente: Autor

Debido a que la memoria compartida puede presentar la posibilidad de crear *n* repositorios o memorias propiamente dichas, y a su vez, estas poder contener un número infinito de localidades, se ha dispuesto la creación de dos memorias principales, en las que se almacenarán los datos necesarios para su procesamiento y posterior utilización. Cada una de estas memorias contendrá los valores definidos en la siguiente Tabla 4.1

Tabla 4.1. Distribución de datos en la memoria.

LOCALIDAD	MEMORIA 0	MEMORIA2
0	**VAS**	**P1S**
1	**VBS**	**P2S**
2	**VCS**	**P3S**
3	**VAE**	**QS**
4	**VBE**	**P1E**
5	**VCE**	**P2E**
6	**SPT**	**P3E**
7	**SPF**	**QE**
8	**SPN**	**API**
9	**SPP**	**PIG**
10	**VDS**	**PIG2**
11	**VES**	
12	**VDE**	
13	**VEE**	

Fuente: Autor

En donde, la Memoria 0 se encarga de recibir todos los porcentajes de apertura de las válvulas que intervienen en los procesos de lanzamiento y recepción del raspador; cada uno situado en una localidad distinta para su fácil acceso y utilización. Mientras que la Memoria 2 se ha dispuesto para el almacenamiento de la información correspondiente a las variables del proceso en sí; es decir, los valores de presiones, temperaturas y caudales en distintos puntos de las estaciones representadas.

Por último, en la Memoria 2 también se incluyen dos localidades en las que se asignan valores binarios para conocer el estado del raspador en cada una de las estaciones. Con esto se indica si el raspador está presente ya sea en la estación de lanzamiento, o a su vez, ya se encuentra en la estación de recepción.

4.2. Integración de equipos de entrada y salida

Tal como se indicó anteriormente, el desarrollo del entorno de entrenamiento está asociado directamente a las gafas HTC VIVE. Son estas las encargadas de activar los componentes que permitirán el intercambio de información con la herramienta de procesamiento matemático según las acciones que se ejecuten.

El motor gráfico realiza automáticamente el reconocimiento de los dispositivos una vez que se ejecuta la aplicación. Por tal razón, es necesario que se configure y programe ciertas condiciones iniciales para su correcto funcionamiento, las mismas que deben cumplirse siguiendo las consideraciones reales que presentan las dos estaciones; lanzamiento y recepción, tal y como son, el flujo de crudo que pasa por las tuberías, la presión a condición normal de trabajo y la temperatura a lo largo del sistema. Para que suceda esto, se establecen en los scripts desarrollados ciertos comandos de escrituras a localidades de la memoria compartida, que permitirán la ejecución adecuada del sistema.

Comandos de escritura/lectura de memoria compartida

```
abrirMemoria("Memoria2", 2);
abrirMemoria("Memoria0", 2);
dato.text = p1s.ToString();
writeFloatValue("Memoria2", 0, p1s);
dato2.text = p1e.ToString();
writeFloatValue("Memoria2", 1, p1e);
dato1.text = q.ToString();
writeFloatValue("Memoria2", 2, q);
dato3.text = api.ToString();
writeFloatValue("Memoria2", 3, api);
```

```
dato4.text = q2.ToString();
writeFloatValue("Memoria2", 8, q2);
ras.text = "OFF";
writeFloatValue("Memoria2", 9, 0.0f);
rae.text = "OFF";
writeFloatValue("Memoria2", 10, 0.0f);
writeFloatValue("Memoria0", 0, 100.0f);
writeFloatValue("Memoria0", 5, 100.0f);
writeFloatValue("Memoria0", 5, 100.0f);
writeFloatValue("Memoria0", 1, 0.0f);
writeFloatValue("Memoria0", 2, 0.0f);
writeFloatValue("Memoria0", 3, 0.0f);
writeFloatValue("Memoria0", 4, 0.0f);
writeFloatValue("Memoria0", 6, 0.0f);
writeFloatValue("Memoria0", 7, 0.0f);
writeFloatValue("Memoria0", 8, 0.0f);
writeFloatValue("Memoria0", 9, 0.0f);
writeFloatValue("Memoria0", 10, 0.0f);
writeFloatValue("Memoria0", 11, 0.0f);
writeFloatValue("Memoria0", 12, 0.0f);
writeFloatValue("Memoria0", 13, 0.0f);
```

De esta manera, se garantiza que los valores de apertura de las válvulas, presiones y flujo sean los correctos al momento de ejecutar la aplicación, y los mismos se encuentren disponibles en la memoria compartida para que la herramienta de procesamiento matemático se encargue de utilizarlos y efectuar los cálculos correspondientes.

Cada una de las acciones que ejecuta el usuario tienen efecto gracias a las entradas del sistema y por lo tanto sus resultados son mostrados en las gafas de realidad virtual. Para ello, cada uno de los scripts que se han desarrollado, interactúan directamente con los componentes físicos y la memoria compartida; en este caso, al entrar en contacto con una de las válvulas, el ángulo de giro que se ejecuta en cada una de ellas representa al porcentaje de apertura con el que trabajará dicha válvula, este valor es obtenido y escrito a su vez en la memoria compartida, para que se encuentre disponible por el procesador matemático, tal como se indica en la Figura 4.5.

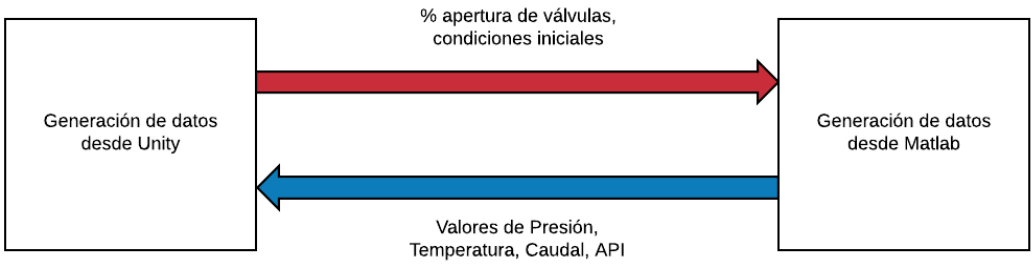


Figura 4.5. Diagrama integración entrada y salida de datos.
Fuente: Autor

Para evitar sobre escritura de datos, a cada válvula se le asigna una localidad, tal como se mencionó anteriormente, y de esta manera se puede realizar los cálculos y representaciones de funcionamiento que realiza el procesador matemático y a su vez también los escribe en la memoria compartida, pero la diferencia en este proceso es que los datos generados serán utilizados por el motor gráfico.

Las pantallas de cada transmisor del entorno de entrenamiento, reciben los datos provenientes del procesador matemático y con ello se evidencia que el entorno reacciona de manera adecuada, tal cual su comportamiento real, pudiendo evidenciarse a través del HMD. De esta manera, consiguiendo la interacción completa del usuario con los dispositivos de entrada y salida que conforman a las HTC VIVE.

5. RESULTADOS Y CONCLUSIONES

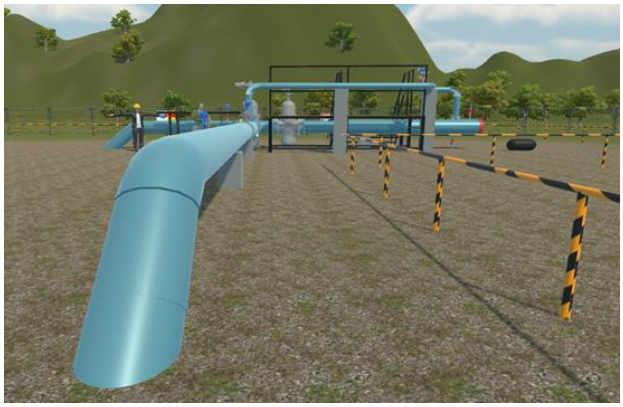
En este capítulo se presentan los resultados y las conclusiones que se han obtenido luego de la implementación y prueba de la plataforma virtual 3D de entrenamiento. Además, se presenta el diseño de un instrumento de evaluación del sistema, para verificar y garantizar las funcionalidades, y el cumplimiento de los objetivos planteados. Asimismo, al final de este capítulo se presentan las referencias bibliográficas que soportan este trabajo y los Anexos respectivos del código fuente implementado.

5.1. Resultados

Uno de los principales resultados del proyecto desarrollado tiene relación directa con grado de semejanza que presenta el ambiente real, con las facilidades, objetos y animaciones presentadas de manera virtual.

5.1.1. Semejanza

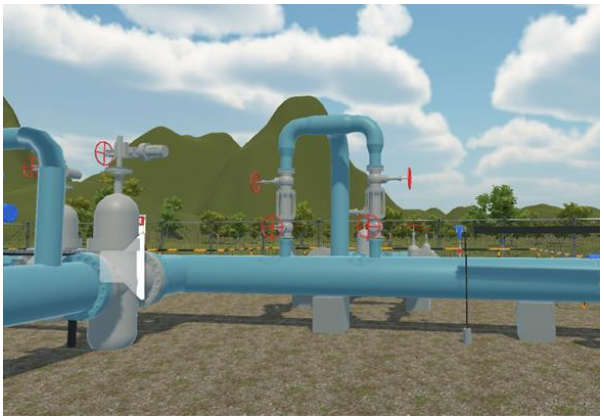
Como primer objetivo el proyecto pretende representar de manera fiel un sistema real de dos estaciones para el entrenamiento de las tareas de lanzamiento y recepción de un raspador de tuberías PIG. Las Figura 5.1. y 5.2. muestran la comparación del entorno virtual creado con respecto al sistema real tanto de la trampa lanzadora como de la trampa receptora.



a)

b)

Figura 5.1. Estación de lanzamiento a) Virtual, b) Real.
Fuente: Autor



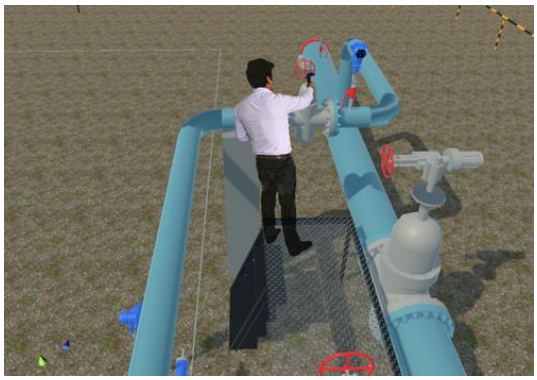
a)

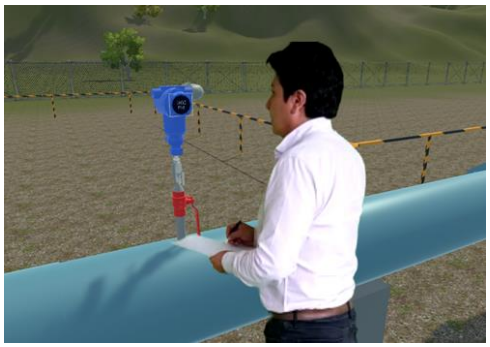
b)

Figura 5.2. Estación de recepción a) Virtual, b) Real.
Fuente: Autor

5.1.2. Inmersión

De manera análoga la implementación de un sistema de entrenamiento virtual debe generar en el usuario un alto nivel de inmersión, de esta manera, puede experimentar por sí mismo, las consecuencias favorables y adversas de aplicar el protocolo de entrenamiento de forma adecuada o no, respectivamente. En las Figuras 5.3., se muestra la representación de la inmersión generada en el usuario al utilizar el sistema de entrenamiento.





Figuras 5.3. Representación de inmersión de usuario.
Fuente: Autor

5.1.3. Casos de Estudio

Como parte de los resultados, se han planteado cuatro casos de estudio con distintas condiciones de operación, en los que se presentan los valores de las variables involucradas en los principales puntos de la trampa, los cambios que se producen en el sistema global bajo ciertas condiciones y, las situaciones de riesgo provocadas por una mala práctica dentro del entorno de entrenamiento, que incluso desemboquen en una emergencia.

a) Caso de estudio 1

El sistema se encuentra en funcionamiento normal, a condiciones iniciales, en donde las válvulas A, C y D se encuentran totalmente cerradas mientras la válvula B trabaja con un 100% de apertura. Los resultados obtenidos en los puntos de medición de presión se muestran en las siguientes figuras (figura 5.4. a la figura 5.9.).

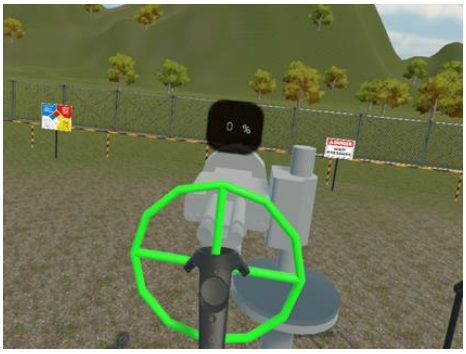


Figura 5.4. Válvula A colocada a 0%.

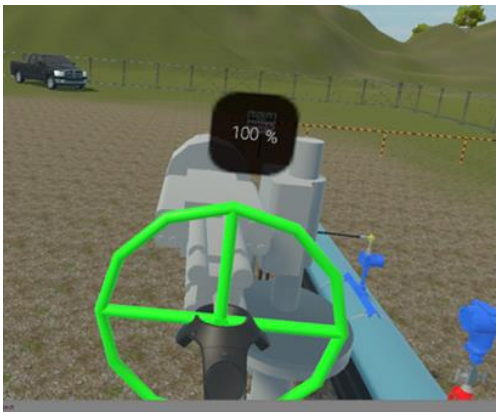


Figura 5.5. Válvula B colocada a 100%.

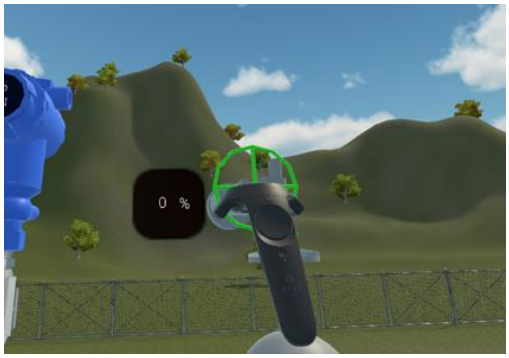


Figura 5.6. Válvula C colocada a 0%.

Figura 5.7. Presión 1.

Figura 5.8. Presión 2.

Figura 5.9. Presión 3.

Para ver el cambio en el comportamiento del sistema las figuras a continuación muestran los resultados de las presiones obtenidas al modificar la apertura de la válvula B hasta un 50%, mientras que el resto de las válvulas se mantienen cerradas.

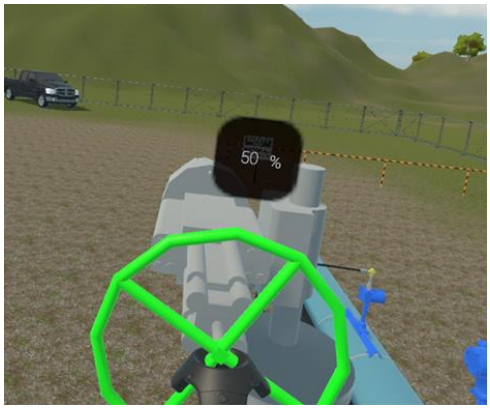


Figura 5.10. Válvula B colocada a 50%.

Figura 5.11. Presión 1 a 50% de apertura de válvula B.

Figura 5.12. Presión 2 a 50% de apertura de válvula B.

Figura 5.13. Presión 3 a 50% de apertura de válvula B.
Fuente: Autor

b) Caso de estudio 2

Uno de los puntos críticos de la operatividad del sistema es la ecualización de presiones que se lleva a cabo dentro de la cámara contenedora del raspador (barril). Las acciones requeridas para que se efectúe este proceso son abrir al 100% las válvulas A, B y C; generándose nuevos valores en los puntos de medición referencial, los mismos que se plasman en las figuras 5.14. a la 5.19.

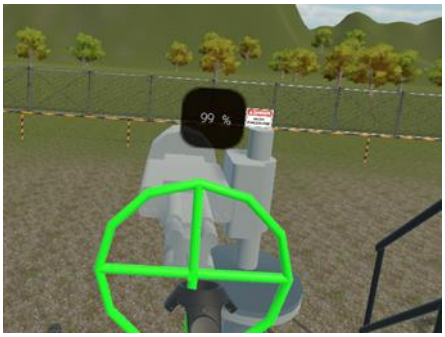


Figura 5.14. Válvula A colocada a 99%.

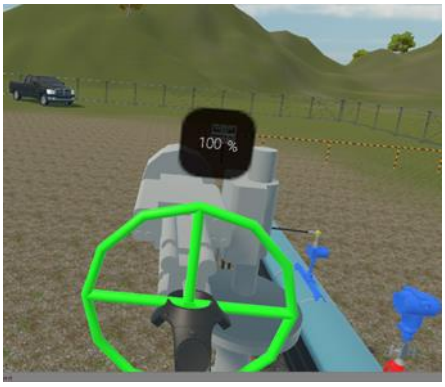


Figura 5.15. Válvula B colocada a 100%.

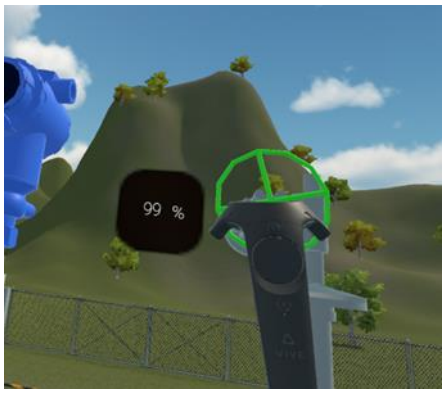


Figura 5.16. Válvula C colocada a 99%.

Figura 5.17. Presión 1 a 100% de apertura de todas las válvulas.

Figura 5.18. Presión 2 a 100% de apertura de todas las válvulas.

Figura 5.19. Presión 3 a 100% de apertura de todas las válvulas.
Fuente: Autor

Como punto de análisis se muestra en las figuras a continuación, una modificación de la operación anterior, en donde las válvulas A y C se mantienen abiertas, mientras se empieza a cerrar la válvula B (operación necesaria para lanzar el raspador), obteniendo las siguientes presiones (figura 5.20. a la figura 5.25.)

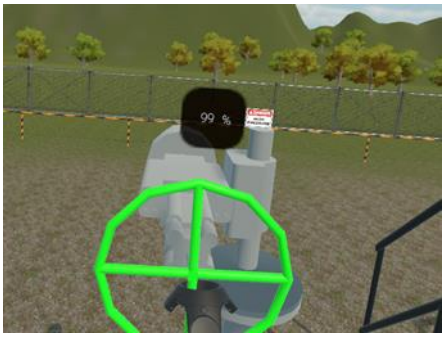


Figura 5.20. Válvula A colocada a 99%.

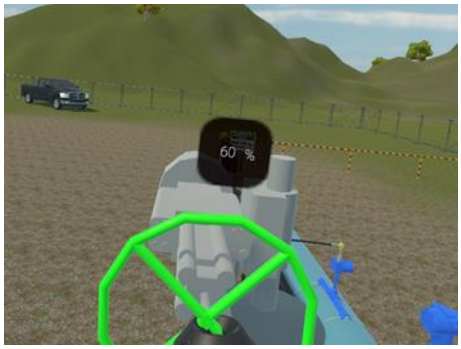


Figura 5.21. Válvula B colocada a 60%.

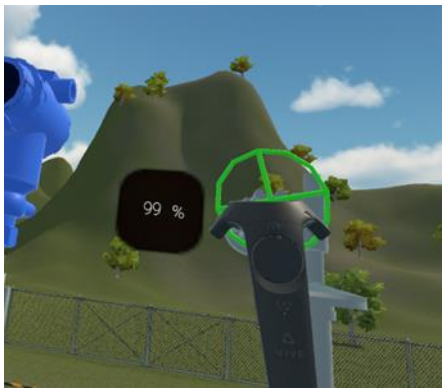


Figura 5.22. Válvula C colocada a 99%.

Figura 5.23. Presión 1 a 100% de apertura de válvulas A y C, B a 60%.

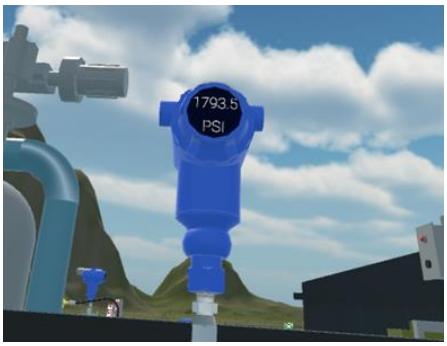


Figura 5.24. Presión 2 a 100% de apertura de válvulas A y C, B a 60%.

Figura 5.25. Presión 3 a 100% de apertura de válvulas A y C, B a 60%.
Fuente: Autor

c) Caso de estudio 3

Una de las posibles malas prácticas que se pueden encontrar en el desarrollo del entrenamiento propuesto es la apertura de la válvula D (drenaje) cuando las tres válvulas principales se encuentran abiertas a la vez, en el instante de ecualización de presiones. Esta mala práctica produce una descompensación del sistema, cuyos resultados en los valores de las presiones se muestran en las figuras 5.26 a la 5.32

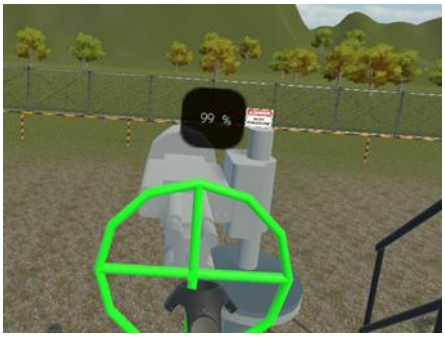


Figura 5.26. Válvula A colocada a 100%.

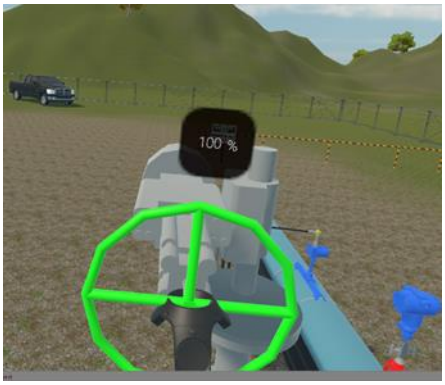


Figura 5.27. Válvula B colocada a 100%.

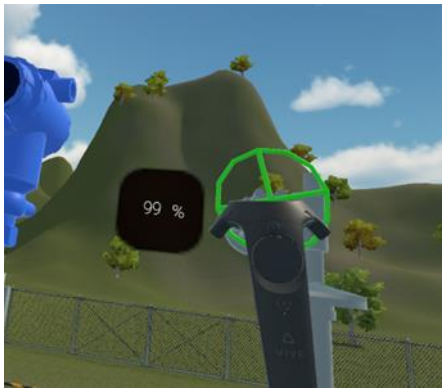


Figura 5.28. Válvula C colocada a 100%.

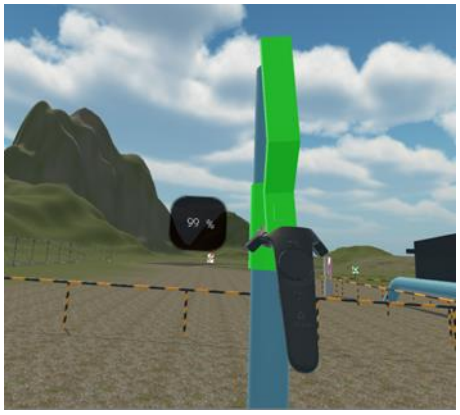


Figura 5.29. Válvula D colocada a 100%.

Figura 5.30. Presión 1 con las válvulas A, B, C y D abiertas.

Figura 5.31. Presión 2 con las válvulas A, B, C y D abiertas.

Figura 5.32. Presión 3 con las válvulas A, B, C y D abiertas.
Fuente: Autor

d) Caso de estudio 4

Otra situación de emergencia que puede tomar lugar en el desarrollo del entrenamiento es el aumento de la presión 1, en cualquiera de las dos estaciones; esta acción se llega a producir cuando en el funcionamiento normal del sistema, por error se cierra el paso de flujo mediante la válvula B y válvula A o C de la trampa; donde de manera virtual se produce un cambio de material de la tubería correspondiente a la sección sobre presurizada (cambio de color) y se brinda al operador un tiempo determinado para la corrección del error. Conforme el tiempo va disminuyendo, la tubería tendrá otro cambio de material, lo que visualmente indica que la tubería está sometida cada vez a una mayor presión. La representación de esta situación se muestra en las figuras 5.33. a la 5.38.

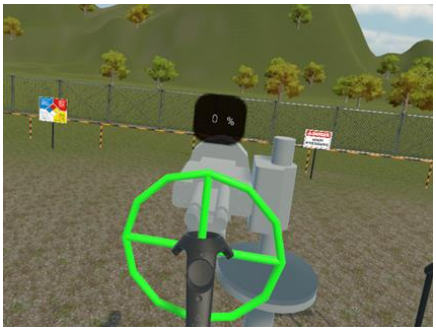


Figura 5.33. Válvula A colocada a 0%.

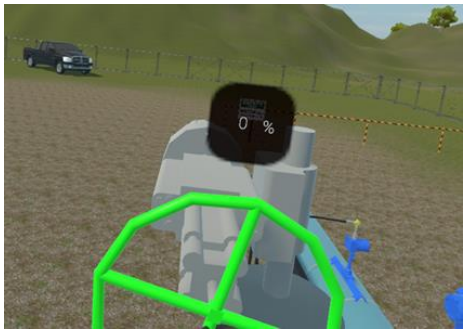


Figura 5.34. Válvula B colocada a 100%.

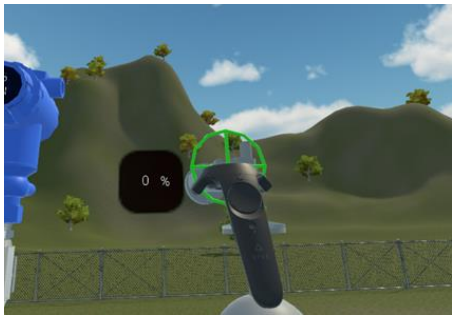


Figura 5.35. Válvula C colocada a 0%.

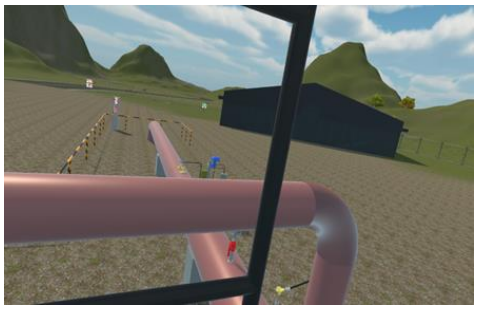


Figura 5.36. Primer cambio de color de tubería como símbolo de presurización.

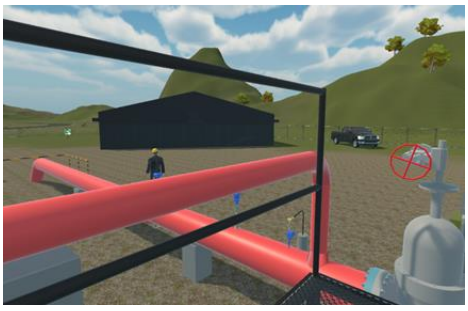


Figura 5.37. Segundo cambio de color de tubería en situación de emergencia.

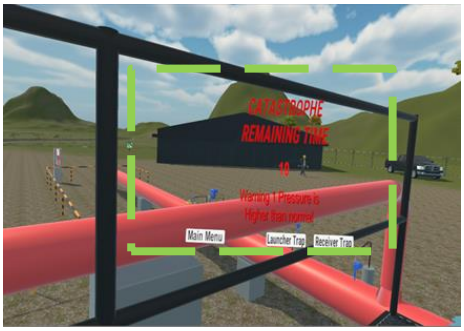


Figura 5.38. Presentación del menú indicativo de la situación de emergencia.
Fuente: Autor

5.2. Evaluación de funcionalidad de la herramienta

Uno de los instrumentos de evaluación que ayudan conocer la calidad de las herramientas tecnológicas desarrolladas es el uso de cuestionarios, mediante los cuales se recoge información que permita evaluar cuantitativamente las reacciones de los usuarios al utilizar la plataforma de entrenamiento virtual 3D para la operación de trampas de lanzamiento y recepción de raspadores.

Se diseñó un cuestionario que consta de 12 preguntas cerradas con una escala de Likert limitadas a responder con los siguientes valores 1= nunca/nada, 2= casi nunca/poco, 3= algo, 4= casi siempre/bastante, 5= siempre/mucho según la pregunta que corresponda (Tabla 5.1.).

El cuestionario tiene como objetivo medir la funcionalidad, desempeño y nivel de inmersión, además de conocer cómo influye la herramienta 3D desarrollada en la calidad de aprendizaje de los usuarios.

El instrumento diseñado se aplicó a 20 usuarios de distintas edades, estudiantes de últimos semestres de carreras técnicas, a los cuales inicialmente se les explicó la secuencia operativa de lanzamiento y recepción de raspadores.

Las 6 primeras preguntas se les realizó antes de utilizar el módulo de entrenamiento virtual, mientras que las otras 6 preguntas fueron contestadas luego de probar la plataforma virtual

Tabla 5.1. Cuestionario de evaluación de la plataforma 3D

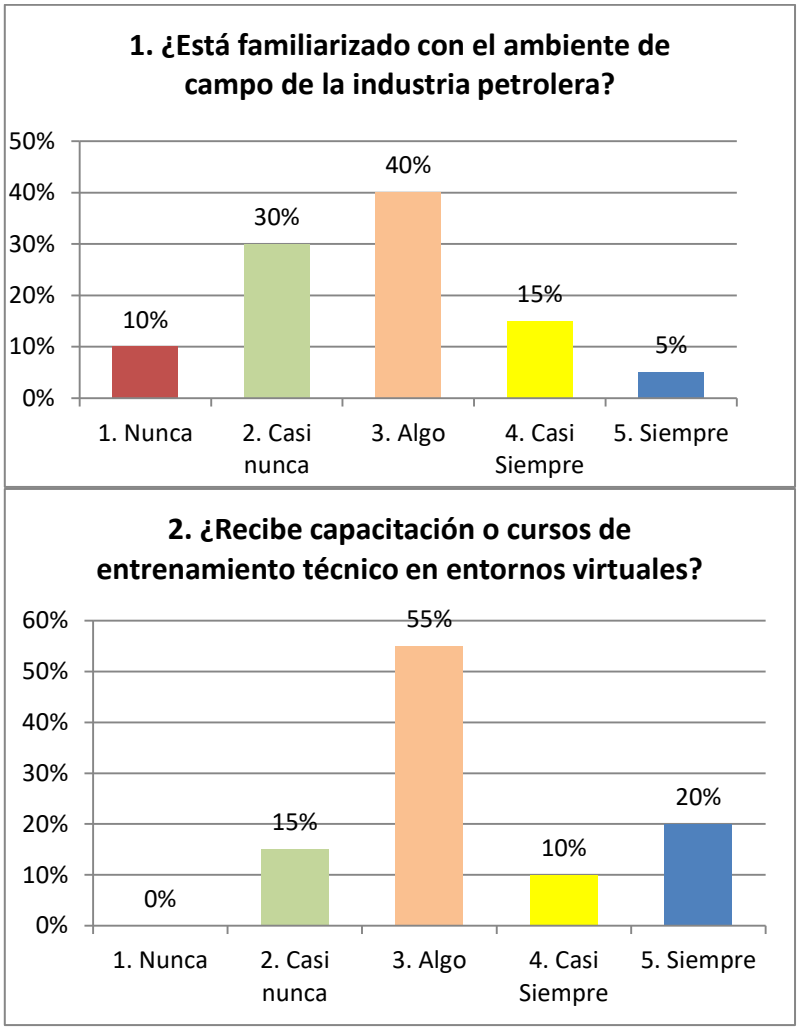
PREGUNTAS	**RESPUESTAS**
1) ¿Está familiarizado con el ambiente de campo de la industria petrolera?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
2) ¿Recibe capacitación o cursos de entrenamiento técnico en entornos virtuales?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
3) ¿Tiene conocimiento de los dispositivos de realidad virtual que existen en el mercado?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

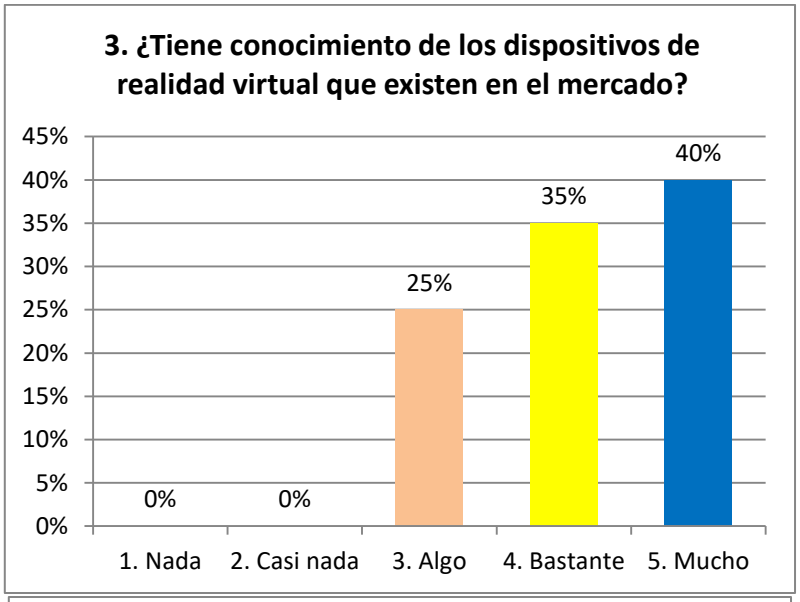
4) ¿Ha utilizado dispositivos de realidad virtual para visualizar entornos simulados?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
5) ¿Utiliza frecuentemente gafas de realidad virtual para: entretenimiento, comunicación, trabajo, juegos, etc.?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
6) ¿Podría identificar los objetos involucrados en el procedimiento de operación impartido previamente, y aplicarlo de manera correcta en un entorno real?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
7) ¿Le resultó fácil el manejo de los dispositivos de inmersión en el entorno virtual?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
8) ¿Cuál fue el grado de inmersión que alcanzó al utilizar el ambiente simulado al campo de operaciones?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
9) ¿Podría identificar los objetos involucrados en el procedimiento de operación impartido previamente, y aplicarlo de manera correcta en un entorno real?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
10) ¿Podría identificar las situaciones o condiciones que generen riesgo y situaciones de emergencia en un entorno real?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
11) ¿Recomendaría el módulo de entrenamiento como una herramienta complementaria a la capacitación teórica que se imparte en la industria?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐
12) ¿La utilización de la plataforma de realidad virtual le ayudó a comprender y retener de mejor manera la información recibida?	1 ☐ 2 ☐ 3 ☐ 4 ☐ 5 ☐

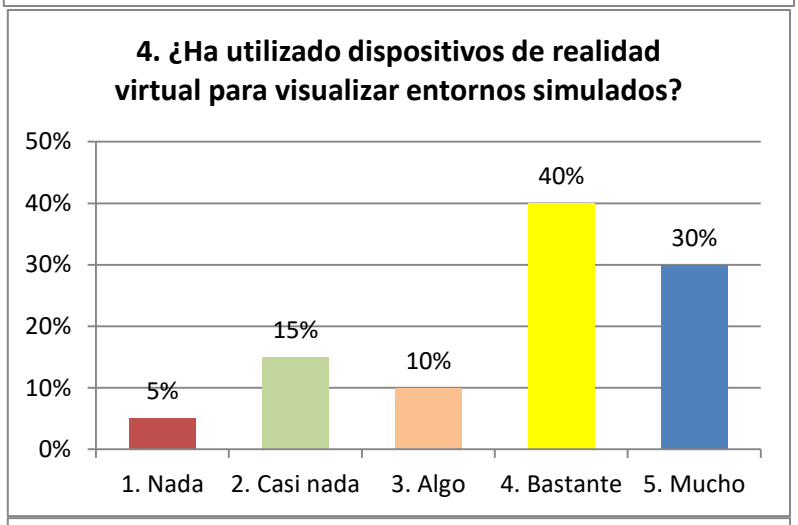
Fuente: Autor

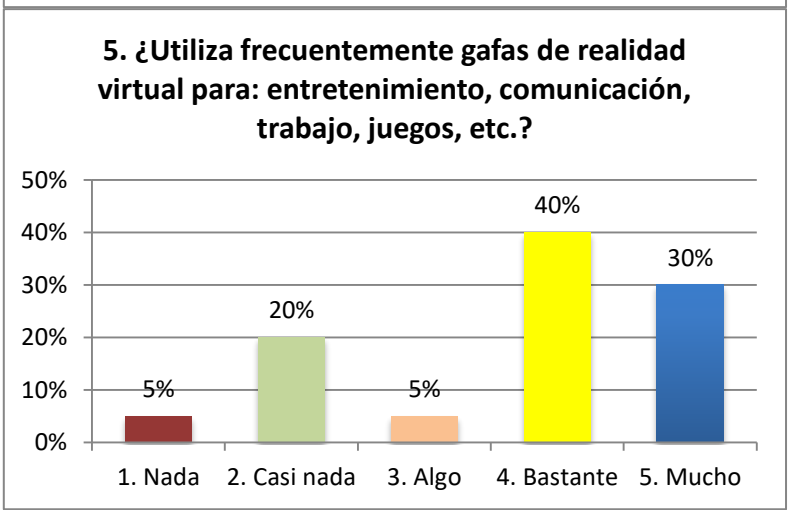
Una vez tabuladas las respuestas de los usuarios se presenta los resultados obtenidos por cada pregunta, tal como se puede observar en las figuras que se presentan en la Tabla 5.2.

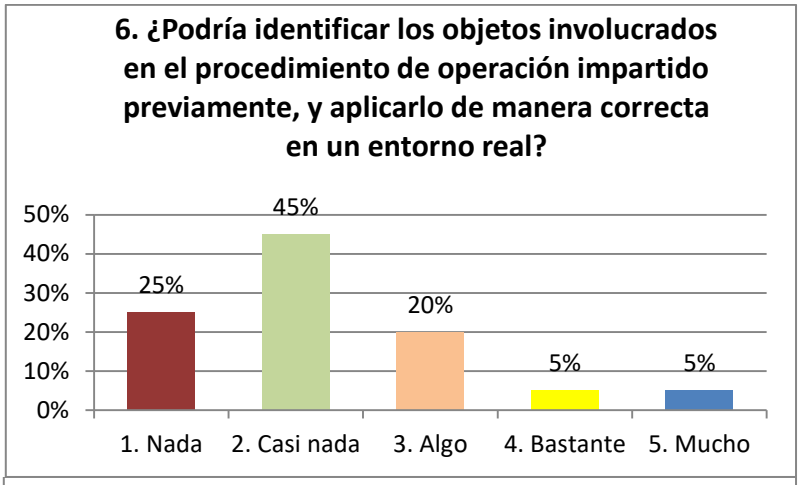
Tabla 5.2. Resultados obtenidos

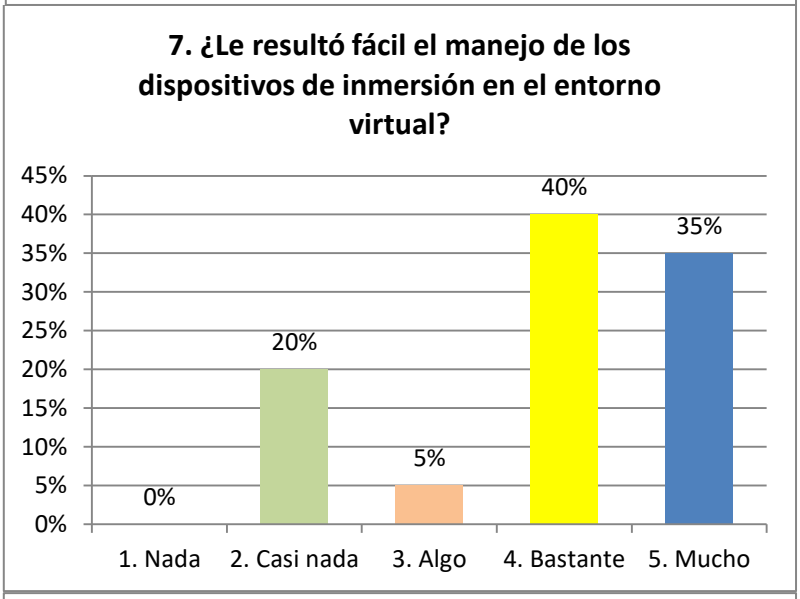


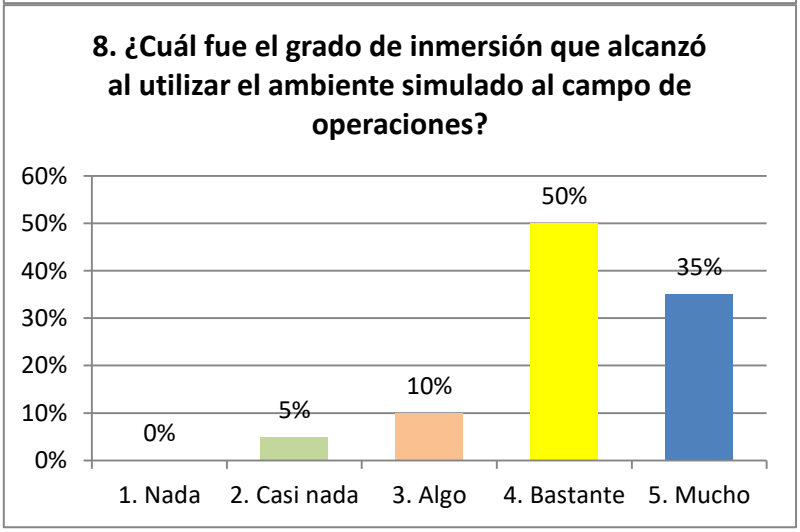

3. ¿Tiene conocimiento de los dispositivos de realidad virtual que existen en el mercado?
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
0%
0%
25%
35%
40%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho

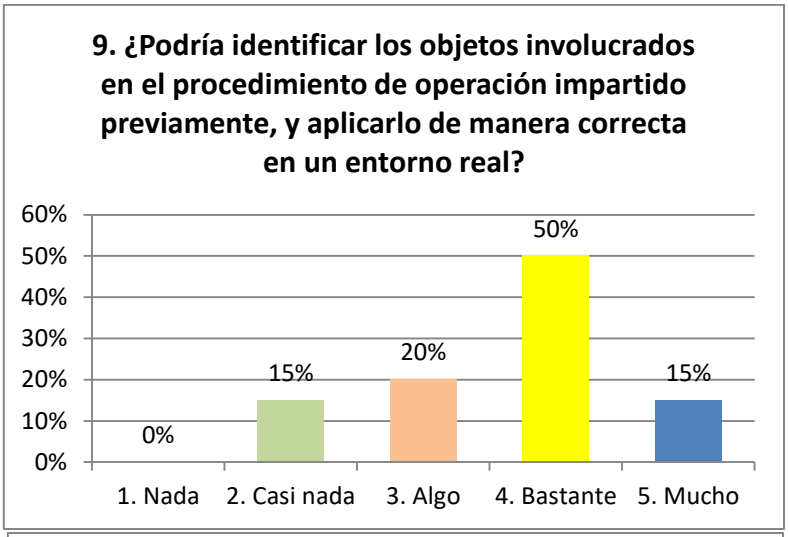

4. ¿Ha utilizado dispositivos de realidad virtual para visualizar entornos simulados?
50%
40%
30%
20%
10%
0%
5%
15%
10%
40%
30%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho

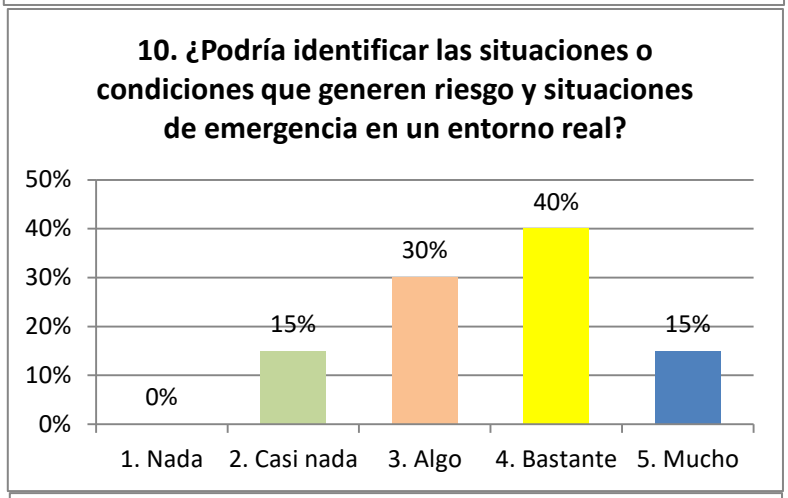

5. ¿Utiliza frecuentemente gafas de realidad virtual para: entretenimiento, comunicación, trabajo, juegos, etc.?
50%
40%
30%
20%
10%
0%
5%
20%
5%
40%
30%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho

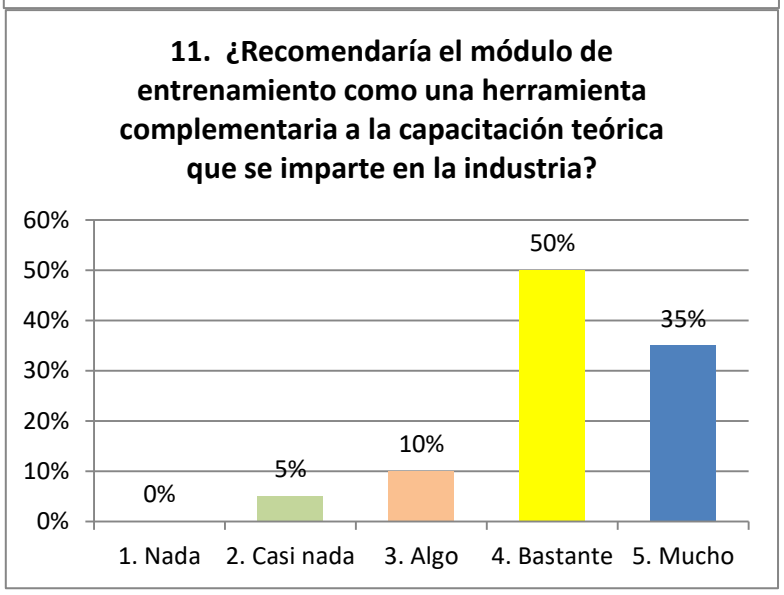

6. ¿Podría identificar los objetos involucrados en el procedimiento de operación impartido previamente, y aplicarlo de manera correcta en un entorno real?
50%
40%
30%
20%
10%
0%
25%
45%
20%
5%
5%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho

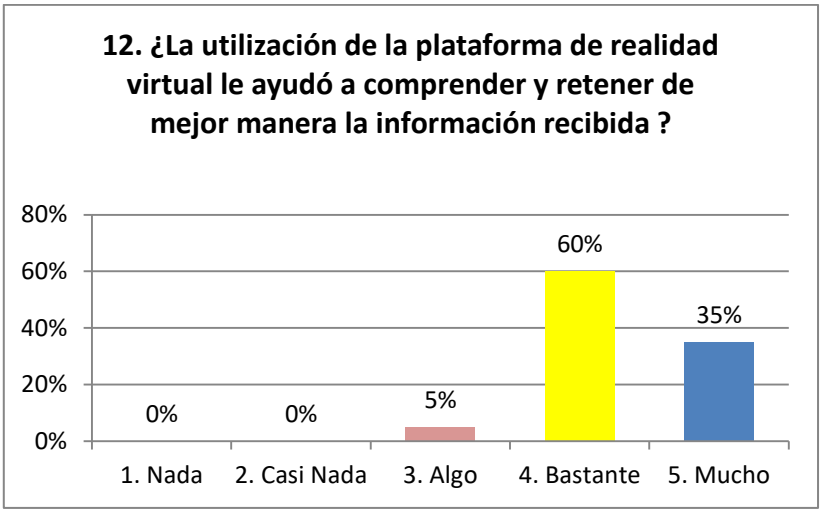

7. ¿Le resultó fácil el manejo de los dispositivos de inmersión en el entorno virtual?
45%
40%
35%
30%
25%
20%
15%
10%
5%
0%
0%
20%
5%
40%
35%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho


8. ¿Cuál fue el grado de inmersión que alcanzó al utilizar el ambiente simulado al campo de operaciones?
60%
50%
40%
30%
20%
10%
0%
0%
5%
10%
50%
35%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho


9. ¿Podría identificar los objetos involucrados en el procedimiento de operación impartido previamente, y aplicarlo de manera correcta en un entorno real?
60%
50%
40%
30%
20%
10%
0%
0%
15%
20%
50%
15%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho


10. ¿Podría identificar las situaciones o condiciones que generen riesgo y situaciones de emergencia en un entorno real?
50%
40%
30%
20%
10%
0%
0%
15%
30%
40%
15%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho


11. ¿Recomendaría el módulo de entrenamiento como una herramienta complementaria a la capacitación teórica que se imparte en la industria?
60%
50%
40%
30%
20%
10%
0%
0%
5%
10%
50%
35%
1. Nada
2. Casi nada
3. Algo
4. Bastante
5. Mucho



Fuente: Autor

5.3. Conclusiones

En la industria del petróleo y sus derivados, más concretamente en el sector del transporte a través de tuberías, es importante conocer, respetar y aplicar todos los procedimientos operativos propios de cada estación, con el objetivo de mantener la planta libre de riesgos físicos y ambientales, garantizando la seguridad de las instalaciones y del personal que trabaja en ellas.

Por ello, el desarrollo de esta plataforma de capacitación y entrenamiento de realidad virtual que permita emular las operaciones necesarias que rodean a la actividad de lanzamiento y recepción de un raspador de tubería, es un complemento educativo importante para el personal de operación de la industria del transporte hidrocarburífero, ya que genera habilidades adicionales tanto operativas y reactivas, utilizando métodos de aprendizaje basados en la experiencia.

La implementación de este sistema de realidad virtual enfocado al entrenamiento y capacitación de operarios generará grandes beneficios a las empresas, debido a que se optimizan recursos monetarios, de infraestructura y personal que se encargue de las capacitaciones; además se brinda la mayor seguridad a los usuarios y una experiencia completamente basada en el desempeño real de la planta.

Una capacitación de manera virtual no reemplaza completamente a una realizada de forma tradicional, pero da la oportunidad de conocer mejor al entorno y a todos los componentes con más detalle y sin premura de tiempo, ganando una mayor experticia para el desenvolvimiento de las tareas planificadas para los operadores en cualquiera de las dos estaciones representadas. Además de que la plataforma virtual de entrenamiento genera satisfacción en los estudiantes, aumentando la probabilidad de éxito en el proceso de aprendizaje.

Cualquier módulo de entrenamiento virtual con capacidad de inmersión deberá ser intuitivo, de fácil manejo y lo más cercano posible al ambiente real, tanto en la semejanza de los objetos, el modo de operación, la reacciones a estímulos generados y todo lo que los sentidos humanos puedan percibir.

Para el desarrollo e implementación de la plataforma virtual se tomó como punto de partida la información recolectada de dos estaciones reales de Petroecuador y se centró únicamente en las operaciones dentro de la trampa lanzadora y receptora, sin que esto sea un limitante para extender el alcance en futuros proyectos de la plataforma virtual hacia otras operaciones de la planta: unidades de bombeo, válvulas reductoras, tanques de almacenamiento, etc.

Se recolectó información acerca de las secuencias normales de operación al momento de lanzar y recibir el raspador, con lo cual se implementó las secuencias lógicas y algoritmos matemáticos en Matlab para que cualquier operación atípica se reporte como una emergencia, logrando que el usuario perciba el error y lo pueda corregir de manera inmediata.

Con los datos iniciales, se logra un ambiente más real con condiciones y respuestas apegadas a las operaciones que se realizan en una estación verdadera, incluso llegando a recrear escenarios de riesgo o emergencia en caso de realizar alguna operación que esté fuera del procedimiento normal de trabajo.

Si bien ambas estaciones presentan un comportamiento similar, al momento de implementar la plataforma se consideró los factores físicos, matemáticos y operativos que las diferencian: alturas geométricas, pérdidas, secuencia de operación, factores de riesgo, etc. Estos valores y condiciones fueron caracterizados e ingresados de manera

independiente en el procesador matemático para que el resultado se apegue a la realidad.

El intercambio de datos a través de una herramienta de memoria compartida, permitió el desempeño en tiempo real de la aplicación, ya que, en su elevada velocidad de procesamiento se ejecutan varios ciclos de lectura y escritura de datos en las distintas localidades, a los cuales se puede acceder y utilizarlos mediante bucles de programación obteniendo resultados que emulen al proceso real

La factibilidad que presenta el motor gráfico al momento de escribir la programación de los distintos componentes gráficos y auditivos permite que se pueda experimentar entre el lenguaje C y Java para encontrar el mejor desempeño de la aplicación de acuerdo a los estándares planteados por el desarrollador.

El presente trabajo representa una base sobre la cual se puede desarrollar varias aplicaciones industriales de manera virtual, no solo con el objetivo de capacitar al personal, sino también de poder recrear las condiciones de proceso de planta con el objetivo de conocer las respuestas del proceso ante determinada situación, por ejemplo: simuladores de sitios confinados, plantas de refinación, plataformas marinas, sistemas meteorológicos, etc.

Actualmente, en el país no se ha desarrollado una cultura de uso de aplicaciones que permitan mejorar el rendimiento de la industria mediante herramientas proactivas o simuladas que se puedan adelantar a los posibles hechos que afecten al normal desarrollo de la operación. Al contrario, muchas de la industrias actúan de manera reactiva aumentando las probabilidades de situaciones de riesgo.

Finalmente, de los resultados obtenidos del instrumento de evaluación (sección 5.2) se puede concluir lo siguiente:

- Las personas evaluadas en su gran mayoría no tenían solidas experiencias dentro de la industria petrolera, a pesar de eso, luego de utilizada la herramienta pudieron entender claramente el proceso de lanzamiento y recepción de raspadores.
- El 85% de las personas evaluadas considera que el grado de inmersión de la plataforma es alto.

- Luego de la utilización de la plataforma, los usuarios están capacitados para identificar los componentes del proceso, realizar las maniobras de manera segura e identificar los posibles riesgos de operación.

- El 95% de las personas evaluadas considera que la utilización del sistema de realidad virtual le ayudó a entender y captar de mejor manera la información recibida, incluso las personas que no tienen experiencia en la industria del transporte de petróleo.

- Se ha podido constatar que la plataforma de realidad virtual desarrollada cumple con lo inicialmente propuesto en este trabajo. Se logró que los usuarios se capaciten de manera teórica y vivencial acerca de los componentes, operación y riesgos envueltos en el lanzamiento y recepción de raspadores.

Bibliografía

Andaluz V., et. al. (2016). Inmerssive Industrial Process Enviroment from a P&ID Diagram. *Internationaal Symposium on Visual Computing*, 701-712.

Andaluz VH, et. al. (2016). Unity·d - MatLab Simulator in Real Time for Robotics Applications. *International Conferense on Augmented Reality, Virtual Reality and Computer Graphics*, 246-263.

Andaluz VH, et.al,. (2016). Virtual Reality Integration with Force Feedback in Upper Limb Rehabilitation. *International Symposium on Visual Computing*, 259-268.

Azuma, R. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 355-385.

Burdea G, P. C. (2003). *Virtual Reality Technology* (Segunda ed.). USA: Wiley.

Burdea, G. (1993). Virtual reality systems and applications. *Electro'93 International Conference*.

Burdea, G., & Coiffet, P. (1996). *Tecnologías de la realidad virtual.* Barcelona: Paidos.

Carolina, C. (1992). The cave, Audio Visual Experience Automatic Virtual Enviroment. *Tesis doctoral*. Universidad de Illinois-Chicago.

Castro Juan, et. al. (2017). Virtual Reality on e-Tourism. *IT Convergence and Security 2017*, 86-97.

Daglas, H., & Coleman, G. (2012). Virtual Reality 3D Training For Pipeline Employees. *Gas and Pipeline Journal*.

ENFERMER@invisible. (2013). *Innovación en la prestación de servicios a domicilio en atención primaria, distribución de funciones y tecnologías.* Obtenido de https://agmasid.wordpress.com/2013/10/13/innovacion-en-la-prestacion-de-servicios-a-domicilio-en-atencion-primaria-distribucion-de-funciones-y-tecnologias/

EP Petroecuador. (2015). *Informe de Gestion.* Quito.

EP Petroecuador. (2016). *Informe Estadístico.* Quito.

EP PETROECUADOR. (2018). Plan estratégico Empresarial 2018-2021. Quito, Ecuador.

Flores J, A. E. (2014). Usos y aplicaciones de la Realidad Virtual en la Educación. *Convención Científica de Ingeniería y Arquitectura*.

Fragoso, E. (2007). Estudio numérico de la corrida de diablos para el mantenimiento de la producción en oleoductos. Mexico.

Games, P. D. (junio de 2013). Obtenido de https://rizkyyudhistira.wordpress.com/2013/06/11/pria-dan-games/

Girard. (s.f.). *Product Line*. Obtenido de Girard Industries: http://www.girardind.com/

Girard. (s.f.). *Product Line*. Obtenido de http://www.girardind.com/

González P, G. J. (1998). Realidad Virtual. En *Informática Gráfica* (págs. 235-237). España: Univerdiad de Castilla - La Mancha.

gregpalast. (2017). *Greg Palast Journalism and film*. Recuperado el 2018, de http://www.gregpalast.com/pig-burst-keystone-pipeline/

HTC VIVE. (2018). *VIVE*. Obtenido de https://www.vive.com/us/product/vive-virtual-reality-system/

Indraprastha A, et. al. (2009). The investigation on using Unity3D game engine in urban design study. *J. ICT Res. Appl.3*, 1-18.

Kalawsky, R. (1993). *The Science of Virtual Reality and Virtual Environments.* Boston, MA: Addison-Wesley Longman Publishing Co.

Kolb, D. (1984). *Aprendizaje basado en experiencias .*

LinLin W, et al. (2017). Research on the Surakarta chess game program based on unity 3D. *Chinese Control and Decision Conference - CCDC*, 7671-7674.

MakeUseOf. (8 de junio de 2016). *Virtual Reality Still Has 5 Big Problems to Overcome*. Obtenido de https://www.makeuseof.com/tag/virtual-reality-still-5-big-problems-overcome/

Manetta, C., & Blade, R. (1995). Glossary of virtual reality terminology. . *International Journal of Virtual Reality, vol. 1, núm. 2*, 35-39.

Marcano Y., T. R. (2006). Los Ambientes Virtuales Inteligentes como estrategia para el entrenamiento del capital humano en el área de Higiene y Seguridad Industrial Petrolera. *Multiciencias, Universidad del Zulia*, 135-140.

Mejía N. (septiembre de 2012). Trabajo Final de Máster. *Realidd Virtual: Estado del arte y análisis crítico*. Graanada.

Meza, G. (2004). El internet y la Realidad virtual en la implementación de un prototipo de interfaz gráfica para la visualización de modelos de yacimientos. *3era Convención Técnica de la ACGGP*.

Mons Midttun, et. al. (1998). Petroleum applications of virtual reality techology: introducing a new paradigm. *SEG Technical Expanded Abstracts*, 699-702.

Obrist V, et. al,. (2015). Aplicación de la Realidad Virtual en una experiencia de aprendizaje. *X Congreso de Tecnología en Educación & Educación en Tecnología*, 392-399.

Pallares R, C. (abril de 2016). *Petrotécnica.* Obtenido de Realidad virtual e impresión 3D hidráulica en la industria del petróole y del gas: http://www.petrotecnia.com.ar/abril16/Sin_Publicidad/Realidad.pdf

Pallares, R. (abril de 2016). *Petrotécnica.* Obtenido de Realidad virtual e impresión 3D hidráulica en la industria del petróole y del gas: http://www.petrotecnia.com.ar/abril16/Sin_Publicidad/Realidad.pdf

Parra, J.C., et al. (2001). Introducción pràctica a la realidd virtual. Universidad del Bío Bío.

Pérez, C. (2008). Realidad Virtual: Un aporte real para la Evaluación y el Tratamiento dee Personas con Discapacidad Intelectual. 253-262. Chile.

Perú.com. (agosto de 2016). *Pokémon GO: por qué mi smartphone no tiene realidad aumentada*. Obtenido de https://peru.com/epic/epic-mobile/pokemon-go-que-mi-smartphone-no-tiene-realidad-aumentada-fotos-noticia-468480

PointNatural. (2016). *Products TRACKIR*. Obtenido de https://www.naturalpoint.com/trackir/trackir5/

Ramírez, R., & Dutra, M. (2010). SIMULACIÓN DE FLUJO Y EVALUACIÓN DE LA FUERZA DE ARRASTRE ACTUANDO EN EL CUERPO DE UN PIG CON UN BYPASS. *Mecánica Computacional Vol XXIX*, 8741-8751.

REDES. (4 de diciembre de 2014). *Ventajas, Desventajas y uos de simuladdores.* Obtenido de http://redespctecnology.blogspot.com/2014/12/ventajas-desventajas-y-usos-de-los.html

Rizkyyudhis. (junio de 2013). *Pria Dan Games*. Obtenido de https://rizkyyudhistira.wordpress.com/2013/06/11/pria-dan-games/

Rolando F. (2012). *De la realidad virtual a la realidad aumentada.* Universidad de Palermo.

Santos L, et al. (2015). Guante de datos sensorizado para uso en cirugía laparascópica asistida por la mano (HALS). *Actas de las XXXVI Jornadas de Automática* , 779-785.

Science, L. (20 de abril de 2016). *Why Does Virtual Reality Make Some People Sick*. Obtenido de https://www.livescience.com/54478-why-vr-makes-you-sick.html

Smith, P. (2004). *Valve selection Handbook.*

Sutherland, I. (1965). The ultimate display. *IFIP Congress*, 506-508.

Themelsle. (2017). *ARcrowd la sencillez de un recurso ara hacer realidad aumentada*. Obtenido de http://blogs.upm.es/observatoriogate/2017/02/20/arcrowd-la-sencillez-de-un-recurso-para-hacer-realidad-aumentada-en-el-aula/

Virtual Realities, L. (2017). *VMG 30 PLUS Haptic Glove*. Obtenido de http://www.vrealities.com/products/data-gloves

Yiannakopoulou E, et al. (2015). Virtual reality simulators and training in laparoscopic surgery. *International Journal of Surgery*, 60-64.

YPF. (mayo de 2009). Especifiacion de Diseño. *Trampas de Scrappers*. Argentina.

Zhou Zewei, et. al. (2011). Virtual Reality Based Process Integrated Simulation Plataform in Refinery: Virtual Refinery and Its Application. *China Petroleum Pocessing and Petrochemical Technology*, 74-84.

ANEXOS

A. Código fuente en Unity 3D para el entorno virtual

A.1. Código visualización de menú principal y pantalla de avisos

```
using UnityEngine;

public class ControlGrip : MonoBehaviour
{
    public Transform cameraRigTransform;
    public Transform headTransform; // The camera rig's head
    private SteamVR_TrackedObject trackedObj;
    public GameObject panel; // The laser prefab

    private SteamVR_Controller.Device Controller
    {
        get { return SteamVR_Controller.Input((int)trackedObj.index); }
    }

    void Awake()
    {
        trackedObj = GetComponent<SteamVR_TrackedObject>();
    }
    void Start()
    {
    }

    void Update()
    {
        if (Controller.GetPress(SteamVR_Controller.ButtonMask.Grip))
        {
            panel.SetActive(true);
        }
        else
        {
            panel.SetActive(false);
        }

        if (Controller.GetPressUp(SteamVR_Controller.ButtonMask.Grip))
        {
            panel.SetActive(false);
        }
    }
}
```

A.2. Código de Interacción de objeto

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Runtime.InteropServices;
using UnityEngine.UI;
using System;

[RequireComponent(typeof(Controller))]
public class Hand : MonoBehaviour
{
    GameObject heldObject;
    Controller controller;

    public Valve.VR.EVRButtonId pickUpButton;
    public Valve.VR.EVRButtonId dropButton;
    public GameObject valvula;
    public GameObject valvula1;
    public GameObject valvula2;
    public GameObject valvula3;
    public GameObject valvula4;
    public GameObject valvula5;
    public GameObject valvula6;
    public GameObject valvula7;
    public GameObject valvula8;
    public GameObject valvula9;
    public GameObject cos;
    public GameObject coe;
    public GameObject pig;
    public Text texto;
    public Text texto1;
    public Text texto2;
    public Text texto3;
    public Text texto4;
    public Text texto5;
    public Text texto6;
    public Text texto7;
    public Text texto8;
    public Text texto9;
    public Material vas;
    public Material comp;
    public Material comp1;
    public Material vbs;
    public Material vcs;
    public Material vds;
    public Material ves;
    public Material vae;
    public Material vbe;
    public Material vce;
    public Material vde;
    public Material vee;
    public Color uno;
    public Color dos;
    public GameObject box;
    public GameObject box1;
    public GameObject box2;
    public GameObject box3;
    public GameObject box4;
    public GameObject box5;
```

```
    public GameObject box6;
    public GameObject box7;
    public GameObject box8;
    public GameObject box9;
    private float a;
    private float b;
    private float c;
    private float d;
    private float e;
    private float f;
    private float angcos;
    private float angcoe;
    private float angcs;
    private float angbs;
    private float angas;
    private float angds;
    private float anges;
    private float angce;
    private float angbe;
    private float angae;
    private float angde;
    private float angee;
    private float posi1;
    private float posi;
    private float posi2;
    private float posi3;
    private float posi4;
    private float posi5;
    private float posi6;
    private float posi7;
    private float posi8;
    private float posi9;


    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern int abrirMemoria(string nombre, int tipo);

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float readFloatValue(string nombre, int posicion);

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float writeFloatValue(string nombre, int posicion, float
valor);
    // Use this for initialization
    void Start()
    {
        controller = GetComponent<Controller>();
        abrirMemoria("Memoria0", 2);
        pig.gameObject.SetActive(false);
    }

    void Update()
    {

        if (heldObject)
        {
            if (heldObject.CompareTag("pig1"))
            {
                pig.gameObject.SetActive(true);
            }
            if (heldObject.CompareTag("cos"))
```

```
{
    angcos = heldObject.transform.rotation.y *100;
    if ((angcos < 0))
    {
        switch (controller.CurrentTouchPosition())
        {
            case TouchPosition.Up:
                cos.transform.Rotate(0, 1, 0);
                break;
            case TouchPosition.Down:
                cos.transform.Rotate(0, -1, 0);
                break;

            default:

                break;
        }
    }
    else
    {
        cos.transform.Rotate(0, 0, 0);
    }
    a = 0;b = 1;c = 0;d = 0;e = -1;f = 0;
    comp.SetColor("_Color", uno);
}

if (heldObject.CompareTag("coe"))
{
    angcoe = heldObject.transform.rotation.y * 100;
    if ((angcoe < 0))
    {
        switch (controller.CurrentTouchPosition())
        {
            case TouchPosition.Up:
                coe.transform.Rotate(0, 1, 0);
                break;
            case TouchPosition.Down:
                coe.transform.Rotate(0, -1, 0);
                break;
            default:
                break;
        }
    }
    else
    {
        coe.transform.Rotate(0, 0, 0);
    }
    a = 0; b = 1; c = 0; d = 0; e = -1; f = 0;
    comp1.SetColor("_Color", uno);
}

if (heldObject.CompareTag("vas"))
{
    angas = valvula2.transform.rotation.z * 150;
    box2.gameObject.SetActive(true);

    a = 0; b = 0; c = -1; d = 0; e = 0; f = 1;
    vas.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vbs"))
{
```

```
    angbs = valvula.transform.rotation.x * 150 * (-1);
    box1.gameObject.SetActive(true);
    a = 1; b = 0; c = 0; d = -1; e = 0; f = 0;
    vbs.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vcs"))
{
    angcs=valvula1.transform.rotation.x*150*(-1);
    a = -1; b = 0; c = 0; d = 1; e = 0; f = 0;
    box.gameObject.SetActive(true);
    vcs.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vds"))
{
    angds = valvula7.transform.rotation.z * 150 * (-1);
    box7.gameObject.SetActive(true);
    a = 0; b = 0; c = 1; d = 0; e = 0; f = -1;
    vds.SetColor("_Color", uno);
}
if (heldObject.CompareTag("ves"))
{
    anges = valvula6.transform.rotation.z * 150 * (-1);
    box6.gameObject.SetActive(true);
    a = 0; b = 0; c = 1; d = 0; e = 0; f = -1;
    ves.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vae"))
{
    angae = valvula4.transform.rotation.z * 150 * (-1);
    box4.gameObject.SetActive(true);
    a = 0; b = 0; c = 1; d = 0; e = 0; f = -1;
    vae.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vbe"))
{
    angbe = valvula5.transform.rotation.x * 150 ;
    box5.gameObject.SetActive(true);
    a = 1; b = 0; c = 0; d = -1; e = 0; f = 0;
    vbe.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vce"))
{
    angce = valvula3.transform.rotation.x * 150 * (-1);
    box3.gameObject.SetActive(true);
    a = 1; b = 0; c = 0; d = -1; e = 0; f = 0;
    vce.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vde"))
{
    angde = valvula8.transform.rotation.z * 150;
    print(angde);
    box9.gameObject.SetActive(true);
    a = 0; b = 0; c = 1; d = 0; e = 0; f = -1;
    vde.SetColor("_Color", uno);
}
if (heldObject.CompareTag("vee"))
{
    angee = valvula9.transform.rotation.z * 150;
    print(angee);
    box8.gameObject.SetActive(true);
    a = 0; b = 0; c = 1; d = 0; e = 0; f = -1;
```

```
                vee.SetColor("_Color", uno);
            }

            switch (controller.CurrentTouchPosition())
            {
                case TouchPosition.Up:
                    heldObject.transform.Rotate(a, b, c);
                    break;
                case TouchPosition.Down:
                    heldObject.transform.Rotate(d, e, f);
                    break;
                default:
                    break;
            }
            if ((controller.controller.GetPressUp(pickUpButton) &&
heldObject.GetComponent<HeldObject>().dropOnRelease) ||
(controller.controller.GetPressDown(dropButton) &&
!heldObject.GetComponent<HeldObject>().dropOnRelease))
            {
                heldObject.GetComponent<HeldObject>().Drop();
                vas.SetColor("_Color", dos);
                comp.SetColor("_Color", dos);
                comp1.SetColor("_Color", dos);
                vbs.SetColor("_Color", dos);
                vcs.SetColor("_Color", dos);
                vds.SetColor("_Color", dos);
                ves.SetColor("_Color", dos);
                vae.SetColor("_Color", dos);
                vbe.SetColor("_Color", dos);
                vce.SetColor("_Color", dos);
                vde.SetColor("_Color", dos);
                vee.SetColor("_Color", dos);

                box.gameObject.SetActive(false);
                box1.gameObject.SetActive(false);
                box2.gameObject.SetActive(false);
                box3.gameObject.SetActive(false);
                box4.gameObject.SetActive(false);
                box5.gameObject.SetActive(false);
                box6.gameObject.SetActive(false);
                box7.gameObject.SetActive(false);
                box8.gameObject.SetActive(false);
                box9.gameObject.SetActive(false);
                heldObject = null;
            }
        }
        else
        {
            if (controller.controller.GetPressDown(pickUpButton))
            {
                Collider[] cols = Physics.OverlapSphere(transform.position, 0.1f);
                foreach (Collider col in cols)
                {
                    if (heldObject == null && col.GetComponent<HeldObject>() &&
col.GetComponent<HeldObject>().parent == null)
                    {
                        heldObject = col.gameObject;
                        heldObject.GetComponent<HeldObject>().PickUp();
                    }
                }
            }
```

```
}
if ((angbs >= 0) && (angbs < 150))
{
    posi = (valvula.transform.rotation.x * 100) + 100;
}
else if (angbs < 0)
{
    posi = 100;
}
else
{
    posi = 0;
}
int salida = (int)posi;
texto.text = salida.ToString();
writeFloatValue("Memoria0", 0, posi);//bs
if ((angcs >= 0) && (angcs < 150))
{
   posi1 = valvula1.transform.rotation.x * 100;
}
else if (angcs<0){ posi1 = 0;
}
else
{
    posi1 = 100;
}
int salida1 = (int)posi1 * (-1);
texto1.text = salida1.ToString();
writeFloatValue("Memoria0", 1, posi1 * (-1));//cs

if ((angas <= 0) && (angas  > -150))
{
    posi2 = valvula2.transform.rotation.z * 100;
}
else if (angas > 0)
{
    posi2 = 0;
}
else
{
    posi2 = 100;
}
int salida2 = (int)posi2 *(-1);
texto2.text = salida2.ToString();
writeFloatValue("Memoria0", 2, posi2*(-1) );//as

if ((anges <= 0) && (anges > -150))
{
    posi6 = valvula6.transform.rotation.z * 100;
}
else if (anges > 0)
{
    posi6 = 0;
}
else
{
    posi6 = 100;
}
int salida6 = (int)posi6;
texto6.text = salida6.ToString();
writeFloatValue("Memoria0", 10, posi6);
```

```
if ((angds <= 0) && (angds > -150))
{
    posi7 = valvula7.transform.rotation.z * 100;
}
else if (angds > 0)
{
    posi7 = 0;
}
else
{
    posi7 = 100;
}
int salida7 = (int)posi7;
texto7.text = salida7.ToString();
writeFloatValue("Memoria0", 11, posi7);

if ((angce  <= 0) && (angce > -150))
{
    posi3 = valvula3.transform.rotation.x * 100;
}
else if (angce > 0)
{
    posi3 = 0;
}
else
{
    posi3 = 100;
}
int salida3 = (int)posi3;
texto3.text = salida3.ToString();
writeFloatValue("Memoria0", 3, posi3);//ce

if ((angae >= 0) && (angae < 150))
{
    posi4 = valvula4.transform.rotation.z * 100;

}
else if (angae < 0)
{
    posi4 = 0;
}
else
{
    posi4 = 100;
}
int salida4 = (int)posi4 * (-1);
texto4.text = salida4.ToString();
writeFloatValue("Memoria0", 4, posi4 * (-1));//ae

if ((angbe <= 0) && (angbe > -150))
{
    posi5 = (valvula5.transform.rotation.x * 100) + 100;
}
else if (angbe > 0)
{
    posi5 = 100;
}
else
{
    posi5 = 0;
```

```
        }
        int salida5 = (int)posi5;
        texto5.text = salida5.ToString();
        writeFloatValue("Memoria0", 5, posi5);//be

        if ((angde >= 0) && (angde <150))
        {
            posi8 = valvula8.transform.rotation.z * 100;

        }
        else if (angde < 0)
        {
            posi8 = 0;
        }
        else
        {
            posi8 = 100;
        }
        int salida8 = (int)posi8;
        texto8.text = salida8.ToString();
        writeFloatValue("Memoria0", 12, posi8);

        if ((angee >= 0) && (angee < 150))
        {
            posi9 = valvula9.transform.rotation.z * 100;

        }
        else if (angee < 0)
        {
            posi9 = 0;
        }
        else
        {
            posi9 = 100;
        }
        int salida9 = (int)posi9;
        texto9.text = salida9.ToString();
        writeFloatValue("Memoria0", 13, posi9);

    }
}
```

A.3. Código asociación de objetos para su utilización

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

[RequireComponent(typeof(Rigidbody))]
public class HeldObject : MonoBehaviour
{
    [HideInInspector]
    public Controller parent;
    public bool dropOnRelease;
    public UnityEvent pickUp;
    public UnityEvent drop;
```

```
    [HideInInspector]
    public Rigidbody simulator;

    void Start()
    {
        simulator = new GameObject().AddComponent<Rigidbody>();
        simulator.name = "simulator";
        simulator.transform.parent = transform.parent;
        simulator.useGravity = false;
    }

    private void Update()
    {
        if (parent != null)
        {
            simulator.velocity = (parent.transform.position - simulator.position) * 50f;
        }
    }

    public void PickUp ()
    {
        pickUp.Invoke();
        if (pickUp.GetPersistentEventCount() == 0)
        {
            DefaultPickUp();
        }
    }

    public void Drop ()
    {
        drop.Invoke();
        if (drop.GetPersistentEventCount() == 0)
        {
            DefaultDrop();
        }
        parent = null;
    }

    public void DefaultDrop ()
    {
        transform.parent = null;
        GetComponent<Rigidbody>().isKinematic = false;
        GetComponent<Rigidbody>().velocity = simulator.velocity;
        parent = null;
    }

    public void DefaultPickUp ()
    {
        transform.parent = parent.transform;
        transform.localPosition = Vector3.zero;
        transform.localRotation = Quaternion.identity;
        GetComponent<Rigidbody>().isKinematic = true;
    }
}
```

A.4. Código de generación de posiciones en el TouchPad

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Controller : MonoBehaviour
{
    public SteamVR_Controller.Device controller
    {
        get
        {
            return
SteamVR_Controller.Input((int)GetComponent<SteamVR_TrackedObject>().index);
        }
    }

    public TouchPosition CurrentTouchPosition ()
    {
        Vector2 pos =
controller.GetAxis(Valve.VR.EVRButtonId.k_EButton_SteamVR_Touchpad);
        bool isTop = pos.y >= 0;
        bool isRight = pos.x >= 0;

        if (isTop)
        {
            if (isRight)
            {
                if (pos.y > pos.x)
                {
                    return TouchPosition.Up;
                }
                else if (pos.y < pos.x)
                {
                    return TouchPosition.Right;
                }
            }
            else
            {
                if (pos.y > -pos.x)
                {
                    return TouchPosition.Up;
                }
                else if (pos.y < -pos.x)
                {
                    return TouchPosition.Left;
                }
            }
        }
        else
        {
            if (isRight)
            {
                if (-pos.y > pos.x)
                {
                    return TouchPosition.Down;
                }
                else if (-pos.y < pos.x)
                {
                    return TouchPosition.Right;
                }
            }
            else
            {
```

```
                if (-pos.y > -pos.x)
                {
                    return TouchPosition.Down;
                }
                else if (-pos.y < -pos.x)
                {
                    return TouchPosition.Left;
                }
            }
        }

        return TouchPosition.Off;
    }
}

public enum TouchPosition
{
    Off, Up, Down, Left, Right

}
```

A.5. Código verificación

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System.Runtime.InteropServices;
public class entrada : MonoBehaviour {
    public Text box;
    public GameObject aviso;
    public GameObject pig;
    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern int abrirMemoria(string nombre, int tipo);

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float readFloatValue(string nombre, int posicion);

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float writeFloatValue(string nombre, int posicion, float
valor);

    void Start () {
        abrirMemoria("Memoria2", 2);
    }

       void Update () {

       }

       void OnTriggerEnter(Collider otro)
       {
              if (otro.CompareTag ("Player")) {
            box.text = "ON";
            writeFloatValue("Memoria2", 9, 1.0F);
            aviso.gameObject.SetActive(true);
            Destroy(pig);
        }
       }
```

```
}
```

A.6. Código para interactuar con el menú y botones

```
using UnityEngine;
namespace Wacki {
    abstract public class IUILaserPointer : MonoBehaviour {

        public float laserThickness = 0.002f;
        public float laserHitScale = 0.02f;
        public bool laserAlwaysOn = false;
        public Color color;

        private GameObject hitPoint;
        private GameObject pointer;


        private float _distanceLimit;

        // Use this for initialization
        void Start()
        {
            pointer = GameObject.CreatePrimitive(PrimitiveType.Cube);
            pointer.transform.SetParent(transform, false);
            pointer.transform.localScale = new Vector3(laserThickness, laserThickness,
100.0f);
            pointer.transform.localPosition = new Vector3(0.0f, 0.0f, 50.0f);

            hitPoint = GameObject.CreatePrimitive(PrimitiveType.Sphere);
            hitPoint.transform.SetParent(transform, false);
            hitPoint.transform.localScale = new Vector3(laserHitScale, laserHitScale,
laserHitScale);
            hitPoint.transform.localPosition = new Vector3(0.0f, 0.0f, 100.0f);

            hitPoint.SetActive(false);

            Object.DestroyImmediate(hitPoint.GetComponent<SphereCollider>());
            Object.DestroyImmediate(pointer.GetComponent<BoxCollider>());

            Material newMaterial = new Material(Shader.Find("Wacki/LaserPointer"));

            newMaterial.SetColor("_Color", color);
            pointer.GetComponent<MeshRenderer>().material = newMaterial;
            hitPoint.GetComponent<MeshRenderer>().material = newMaterial;
            Initialize();
            if(LaserPointerInputModule.instance == null) {
                new GameObject().AddComponent<LaserPointerInputModule>();
            }


            LaserPointerInputModule.instance.AddController(this);
        }

        void OnDestroy()
        {
            if(LaserPointerInputModule.instance != null)
                LaserPointerInputModule.instance.RemoveController(this);
        }
```

```
        protected virtual void Initialize() { }
        public virtual void OnEnterControl(GameObject control) { }
        public virtual void OnExitControl(GameObject control) { }
        abstract public bool ButtonDown();
        abstract public bool ButtonUp();

        protected virtual void Update()
        {
            Ray ray = new Ray(transform.position, transform.forward);
            RaycastHit hitInfo;
            bool bHit = Physics.Raycast(ray, out hitInfo);

            float distance = 100.0f;

            if(bHit) {
                distance = hitInfo.distance;
            }

            if(_distanceLimit > 0.0f) {
                distance = Mathf.Min(distance, _distanceLimit);
                bHit = true;
            }

            pointer.transform.localScale = new Vector3(laserThickness, laserThickness,
distance);
            pointer.transform.localPosition = new Vector3(0.0f, 0.0f, distance * 0.5f);

            if(bHit) {
                hitPoint.SetActive(true);
                hitPoint.transform.localPosition = new Vector3(0.0f, 0.0f, distance);
            }
            else {
                hitPoint.SetActive(false);
            }

            _distanceLimit = -1.0f;
        }

        public virtual void LimitLaserDistance(float distance)
        {
            if(distance < 0.0f)
                return;

            if(_distanceLimit < 0.0f)
                _distanceLimit = distance;
            else
                _distanceLimit = Mathf.Min(_distanceLimit, distance);
        }
    }

}
```

A.7. Código para ejecutar la función de Teletransportación

```
using UnityEngine;
public class LaserPointer1 : MonoBehaviour
{
    public Transform cameraRigTransform;
```

```
    public Transform headTransform; // The camera rig's head
    public Vector3 teleportReticleOffset; // Offset from the floor for the reticle to
avoid z-fighting
    public LayerMask teleportMask; // Mask to filter out areas where teleports are
allowed

    private SteamVR_TrackedObject trackedObj;

    public GameObject laserPrefab; // The laser prefab
    private GameObject laser; // A reference to the spawned laser
    private Transform laserTransform; // The transform component of the laser for ease
of use

    public GameObject teleportReticlePrefab; // Stores a reference to the teleport
reticle prefab.
    private GameObject reticle; // A reference to an instance of the reticle
    private Transform teleportReticleTransform; // Stores a reference to the teleport
reticle transform for ease of use

    private Vector3 hitPoint; // Point where the raycast hits
    private bool shouldTeleport; // True if there's a valid teleport target

    private SteamVR_Controller.Device Controller
    {
        get { return SteamVR_Controller.Input((int)trackedObj.index); }
    }

    void Awake()
    {
        trackedObj = GetComponent<SteamVR_TrackedObject>();
    }

    //new
    void Start()
    {
        laser = Instantiate(laserPrefab);
        laserTransform = laser.transform;
        reticle = Instantiate(teleportReticlePrefab);
        teleportReticleTransform = reticle.transform;
    }

    void Update()
    {
        // Is the touchpad held down?
        if (Controller.GetPress(SteamVR_Controller.ButtonMask.Touchpad))
        {
            RaycastHit hit;

            // Send out a raycast from the controller
            if (Physics.Raycast(trackedObj.transform.position, transform.forward, out
hit, 100, teleportMask))
            {
                hitPoint = hit.point;

                ShowLaser(hit);

                //Show teleport reticle
                reticle.SetActive(true);
                teleportReticleTransform.position = hitPoint + teleportReticleOffset;
                shouldTeleport = true;
```

```
        }
        else // Touchpad not held down, hide laser & teleport reticle
        {
            laser.SetActive(false);
            reticle.SetActive(false);
        }

        // Touchpad released this frame & valid teleport position found
        if (Controller.GetPressUp(SteamVR_Controller.ButtonMask.Touchpad) &&
shouldTeleport)
        {
            Teleport();
        }
    }

    private void ShowLaser(RaycastHit hit)
    {
        laser.SetActive(true); //Show the laser
        laserTransform.position = Vector3.Lerp(trackedObj.transform.position, hitPoint,
.5f); // Move laser to the middle between the controller and the position the raycast
hit
        laserTransform.LookAt(hitPoint); // Rotate laser facing the hit point
        laserTransform.localScale = new Vector3(laserTransform.localScale.x,
laserTransform.localScale.y,
            hit.distance); // Scale laser so it fits exactly between the controller &
the hit point
    }

    private void Teleport()
    {
        shouldTeleport = false;
        reticle.SetActive(false); // Hide reticle
        Vector3 difference = cameraRigTransform.position - headTransform.position;
        difference.y = 0

        cameraRigTransform.position = hitPoint + difference;    }
}
```

A.8. Código para rutas de patrullaje de operadores extra

```
using UnityEngine;
using System.Collections;
public class patrullaje : MonoBehaviour
{
    public Transform[] points;
    private int destPoint = 0;
    private UnityEngine.AI.NavMeshAgent agent;
    void Start()
    {
        agent = GetComponent<UnityEngine.AI.NavMeshAgent>();
        agent.autoBraking = false;
        GotoNextPoint();
    }


    void GotoNextPoint()
    {
        // Returns if no points have been set up
        if (points.Length == 0)
```

```
            return;
        agent.destination = points[destPoint].position;
        destPoint = (destPoint + 1) % points.Length;
    }

    void Update()
    {
        if (agent.remainingDistance < 0.5f)

        GotoNextPoint();
    }

}
```

A.9. Código de obtención y generación de datos

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Runtime.InteropServices;
using UnityEngine.UI;
using System;
public class simulador : MonoBehaviour
{

    public Text dato;//presion1800
    public Text dato1;//caudal255
    public Text dato2;//presion150
    public Text dato3;//API(24-30)
    public Text dato4;//caudal254
    public Text p2s;
    public Text p3s;
    public Text p2e;
    public Text p3e;
    public Text ras;
    public Text rae;
    public GameObject pig;
    public int v1;
    public int v2;
    private float p1s=1800;
    private float p1e=150;
    private float q = 255;
    private float api = 24;
    private float q2 = 254;
    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern int abrirMemoria(string nombre, int tipo);
    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float readFloatValue(string nombre, int posicion);

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float writeFloatValue(string nombre, int posicion, float
valor);

    void Start()
    {

        abrirMemoria("Memoria2", 2);
        abrirMemoria("Memoria0", 2);
        dato.text = p1s.ToString();
```

```
        writeFloatValue("Memoria2", 0, p1s);
        dato2.text = p1e.ToString();
        writeFloatValue("Memoria2", 1, p1e);
        dato1.text = q.ToString();
        writeFloatValue("Memoria2", 2, q);
        dato3.text = api.ToString();
        writeFloatValue("Memoria2", 3, api);
        dato4.text = q2.ToString();
        writeFloatValue("Memoria2", 8, q2);
        ras.text = "OFF";
        writeFloatValue("Memoria2", 9, 0.0f);
        rae.text = "OFF";
        writeFloatValue("Memoria2", 10, 0.0f);
        writeFloatValue("Memoria0", 0, 100.0f);
        writeFloatValue("Memoria0", 5, 100.0f);
        writeFloatValue("Memoria0", 5, 100.0f);
        writeFloatValue("Memoria0", 1, 0.0f);
        writeFloatValue("Memoria0", 2, 0.0f);
        writeFloatValue("Memoria0", 3, 0.0f);
        writeFloatValue("Memoria0", 4, 0.0f);
        writeFloatValue("Memoria0", 6, 0.0f);
        writeFloatValue("Memoria0", 7, 0.0f);
        writeFloatValue("Memoria0", 8, 0.0f);
        writeFloatValue("Memoria0", 9, 0.0f);
        writeFloatValue("Memoria0", 10, 0.0f);
        writeFloatValue("Memoria0", 11, 0.0f);
        writeFloatValue("Memoria0", 12, 0.0f);
        writeFloatValue("Memoria0", 13, 0.0f);

    }
    void Update()
    {
        dato1.text = readFloatValue("Memoria2", 2).ToString();
        dato3.text= readFloatValue("Memoria2", 3).ToString();
        p3s.text = readFloatValue("Memoria2", 4).ToString();
        p2s.text = readFloatValue("Memoria2", 5).ToString();
        p3e.text = readFloatValue("Memoria2", 6).ToString();
        p2e.text = readFloatValue("Memoria2", 7).ToString();
        dato4.text = readFloatValue("Memoria2", 8).ToString();
        dato.text = readFloatValue("Memoria2", 0).ToString();
        dato2.text = readFloatValue("Memoria2", 1).ToString();
        float pig1 = readFloatValue("Memoria2", 9);
        float pig2 = readFloatValue("Memoria2", 10);

        if (pig1 < 1)
        {
            ras.text = "OFF";
        }
        if (pig2 > 0)
        {
            rae.text = "ON";
        }
        if (v1 > v2)
        {
           pig.gameObject.transform.Translate(0,0.1f, 0);
        }
    }
}
```

A.10. Código intercambio de posiciones entre estaciones

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class posicionesreal : MonoBehaviour
{
    public GameObject personaje;
    void Start()
    {
    }

    void Update()
    {
    }
    public void entrada()
    {
        personaje.transform.position = new Vector3(356.2f, 1.03f, 294.8f);
    }
    public void salida()
    {
        personaje.transform.position = new Vector3(119.5f, 1.03f, 382.72f);
    }
}
```

A.11. Código de ejecución de emergencias

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System.Runtime.InteropServices;
using UnityEngine.UI;
using System;
public class pruebaerror : MonoBehaviour
{
    public GameObject exp;
    public GameObject llam;
    public GameObject sonexp;
    public Text contador;
    public Text text1;
    public Text text2;
    public Text text3;
    private float tiempo = 20f;
    private float tiempo1 = 3f;
    public float R1;
    private float R2;
    private float R3;
    private float R4;
    private Boolean e;
    private float RotY2;
    public Color HighLightColor;
    public Color HighLightColor2;
    public Color HighLightColor3;
    public Color DefaultColor;
    private Renderer colorCubo;
    public Material pieza;
    public int v;

    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern int abrirMemoria(string nombre, int tipo);
    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
```

```
    private static extern float readFloatValue(string nombre, int posicion);
    [DllImport(@"C:\Users\Daniel\Desktop\Memoria Compartida\smClient.dll")]
    private static extern float writeFloatValue(string nombre, int posicion, float
valor);
    void Start()
    {
        abrirMemoria("Memoria0", 2);
        abrirMemoria("Memoria1", 2);
        abrirMemoria("Memoria2", 2);
        pieza.SetColor("_Color", DefaultColor);
    }
    void Update()
    {
        R1 = readFloatValue("Memoria2", 2);
        R2 = readFloatValue("Memoria0", 0);
        R3 = readFloatValue("Memoria0", 1);
        R4 = readFloatValue("Memoria0", 2);
    }

    void FixedUpdate()
    {
        float RotY1 = R1;

        if ((RotY1 <= 1)&&(R2<=1)&&(R3<=1))
        {
            contador.gameObject.SetActive(true);
            text1.gameObject.SetActive(true);
            pieza.SetColor("_Color", HighLightColor3);
            tiempo -= Time.deltaTime;
            contador.text = " " + tiempo.ToString("f0");
            if ((tiempo <= 10)&&(tiempo > 0))
            {
                if((R2 > 0) && (R3 > 0))
                {
                        e = true;
                }
                contador.gameObject.SetActive(true);
                text2.gameObject.SetActive(true);
                text1.gameObject.SetActive(false);
                pieza.SetColor("_Color", HighLightColor2);
            }
            if (tiempo <= 1)
            {
                if ((R2 > 0) && (R3 > 0))
                {
                    e = true;
                }
                contador.gameObject.SetActive(true);
                text3.gameObject.SetActive(true);
                text2.gameObject.SetActive(false);
                text1.gameObject.SetActive(false);
                pieza.SetColor("_Color", HighLightColor);
                sonexp.gameObject.SetActive(true);
            }
            if (tiempo < 0)
            {
                llam.gameObject.SetActive(true);
                exp.gameObject.SetActive(true);
                tiempo = 0f;
            }
        }
```

```
        else
        {
            text1.gameObject.SetActive(false);
            text2.gameObject.SetActive(false);
            text3.gameObject.SetActive(false);
            contador.gameObject.SetActive(false);
            pieza.SetColor("_Color", DefaultColor);
            tiempo = 20f;
            e = false;
            exp.gameObject.SetActive(false);
            sonexp.gameObject.SetActive(false);
            llam.gameObject.SetActive(false);
        }
        if (e == true)
        {
            pieza.SetColor("_Color", DefaultColor);
        }
    }
}
```

B. Código fuente de Matlab

```
clear all;
close all;
clc;
loadlibrary('./smClient.dll','./main.h')
calllib('smClient','abrirMemoria','Memoria0',2);
calllib('smClient','abrirMemoria','Memoria1',2);
calllib('smClient','abrirMemoria','Memoria2',2);
calllib('smClient','writeFloatValue','Memoria2',4,15.0);%p3
i=0;
j=0;
a=2;
g=9.8;
ro=910;%kg/m^3
q=0.676;%m^3/s
f=0.03;
k=0.2;
dc=0.203;%m
da=0.508;
db=0.508;
la=7;
lb=4;
lc=11.50;
l2=18.5;
m3=0;
m3e=0;
m4=0;
api=0;
k1bs=1;
k1as=1;
k1cs=1;
k1be=1;
k1ce=1;
k1ae=1;
z3=15;
n=1;
pig2=0;
pig1=0;

while(a>0)

p1s=calllib('smClient','readFloatValue','Memoria2',0);
p1e=calllib('smClient','readFloatValue','Memoria2',1);
cs=calllib('smClient','readFloatValue','Memoria0',1)/100;
bs=calllib('smClient','readFloatValue','Memoria0',0)/100;
as=calllib('smClient','readFloatValue','Memoria0',2)/100;
ce=calllib('smClient','readFloatValue','Memoria0',3)/100;
be=calllib('smClient','readFloatValue','Memoria0',5)/100;
ae=calllib('smClient','readFloatValue','Memoria0',4)/100;
es=calllib('smClient','readFloatValue','Memoria0',10)/100;
ds=calllib('smClient','readFloatValue','Memoria0',11)/100;
de=calllib('smClient','readFloatValue','Memoria0',12)/100;
ee=calllib('smClient','readFloatValue','Memoria0',13)/100;
pigl=calllib('smClient','readFloatValue','Memoria2',9);

if (bs>0.95)
    k1bs=1;%k1b
elseif((bs>0.84)&&(bs<=0.95))
    k1bs=5;
elseif((bs>0.68)&&(bs<=0.84))
```

```
    k1bs=20;
elseif((bs>0.57)&&(bs<=0.68))
    k1bs=100;
elseif((bs>0.48)&&(bs<=0.57))
    k1bs=200;
elseif((bs>0.36)&&(bs<=0.48))
    k1bs=500;
elseif((bs>0.28)&&(bs<=0.36))
    k1bs=1000;
elseif((bs>0.22)&&(bs<=0.28))
    k1bs=2000;
elseif((bs>0.14)&&(bs<=0.22))
    k1bs=5000;
elseif((bs>0.1)&&(bs<=0.14))
    k1bs=10000;
end

if (cs>0.95)
    k1cs=1;
elseif((cs>0.84)&&(cs<=0.95))
    k1cs=5;
elseif((cs>0.68)&&(cs<=0.84))
    k1cs=20;
elseif((cs>0.57)&&(cs<=0.68))
    k1cs=100;
elseif((cs>0.48)&&(cs<=0.57))
    k1cs=200;
elseif((cs>0.36)&&(cs<=0.48))
    k1cs=500;
elseif((cs>0.28)&&(cs<=0.36))
    k1cs=1000;
elseif((cs>0.22)&&(cs<=0.28))
    k1cs=2000;
elseif((cs>0.14)&&(cs<=0.22))
    k1cs=5000;
elseif((cs>0.1)&&(cs<=0.14))
    k1cs=10000;
end

if (as>0.95)
    k1as=1;%k1b
elseif((as>0.84)&&(as<=0.95))
    k1as=5;
elseif((as>0.68)&&(as<=0.84))
    k1as=20;
elseif((as>0.57)&&(as<=0.68))
    k1as=100;
elseif((as>0.48)&&(as<=0.57))
    k1as=200;
elseif((as>0.36)&&(as<=0.48))
    k1as=500;
elseif((as>0.28)&&(as<=0.36))
    k1as=1000;
elseif((as>0.22)&&(as<=0.28))
    k1as=2000;
elseif((as>0.14)&&(as<=0.22))
    k1as=5000;
elseif((as>0.1)&&(as<=0.14))
    k1as=10000;
end

if (be>0.95)
    k1be=1;%k1b
```

```
elseif((be>0.84)&&(be<=0.95))
    k1be=5;
elseif((be>0.68)&&(be<=0.84))
    k1be=20;
elseif((be>0.57)&&(be<=0.68))
    k1be=100;
elseif((be>0.48)&&(be<=0.57))
    k1be=200;
elseif((be>0.36)&&(be<=0.48))
    k1be=500;
elseif((be>0.28)&&(be<=0.36))
    k1be=1000;
elseif((be>0.22)&&(be<=0.28))
    k1be=2000;
elseif((be>0.14)&&(be<=0.22))
    k1be=5000;
elseif((be>0.1)&&(be<=0.14))
    k1be=10000;
end

if (ce>0.95)
    k1ce=1;
elseif((ce>0.84)&&(ce<=0.95))
    k1ce=5;
elseif((ce>0.68)&&(ce<=0.84))
    k1ce=20;
elseif((ce>0.57)&&(ce<=0.68))
    k1ce=100;
elseif((ce>0.48)&&(ce<=0.57))
    k1ce=200;
elseif((ce>0.36)&&(ce<=0.48))
    k1ce=500;
elseif((ce>0.28)&&(ce<=0.36))
    k1ce=1000;
elseif((ce>0.22)&&(ce<=0.28))
    k1ce=2000;
elseif((ce>0.14)&&(ce<=0.22))
    k1ce=5000;
elseif((ce>0.1)&&(ce<=0.14))
    k1ce=10000;
end

if (ae>0.95)
    k1ae=1;%k1b
elseif((ae>0.84)&&(ae<=0.95))
    k1ae=5;
elseif((ae>0.68)&&(ae<=0.84))
    k1ae=20;
elseif((ae>0.57)&&(ae<=0.68))
    k1ae=100;
elseif((ae>0.48)&&(ae<=0.57))
    k1ae=200;
elseif((ae>0.36)&&(ae<=0.48))
    k1ae=500;
elseif((ae>0.28)&&(ae<=0.36))
    k1ae=1000;
elseif((ae>0.22)&&(ae<=0.28))
    k1ae=2000;
elseif((ae>0.14)&&(ae<=0.22))
    k1ae=5000;
elseif((ae>0.1)&&(ae<=0.14))
    k1ae=10000;
end
```

```
kb=k*k1bs;
kc=k*k1cs;
ka=k*k1as;

kbe=k*k1be;
kce=k*k1ce;
kae=k*k1ae;

d2=nthroot((l2/((lc/(dc^5))+(la/(da^5)))),5);
k2=(d2^4)*((kc/(dc^4))+(ka/(da^4)));
k1t=((f*(lb/db))+kb);
k2t=((f*(l2/d2))+k2);
h=((8*(q^2))/((pi^2)*g*(((db^2)/sqrt(k1t))+((d2^2)/sqrt(k2t)))^2));
q2=(((2*g*h)/k2t)^(1/2))*pi*((d2^2)/4);
k2e=(d2^4)*((kce/(dc^4))+(kae/(da^4)));
k1te=((f*(lb/db))+kbe);
k2te=((f*(l2/d2))+k2e);

ec5=(p1s-
(ro*g*((8*(q^2))/((pi^2)*g*(((db^2)/sqrt(k1t))+((d2^2)/sqrt(k2t)))^2)))/6894.757)
;
ec6=(p1s-((ro*g*((8*(q^2)*k1t)/((pi^2)*g*(db^4))))/6894.757));
ec7=(p1s-((ro*g*((8*(q^2)*k2t)/((pi^2)*g*(d2^4))))/6894.757));
ec8=(p1s-((ro*g*((f*(lc/dc))+kc)*((8*(q2^2))/((pi^2)*g*(dc^4))))/6894.757));

ec5e=(p1e-
(ro*g*((8*(q^2))/((pi^2)*g*(((db^2)/sqrt(k1te))+((d2^2)/sqrt(k2te)))^2)))/6894.75
7);
ec6e=(p1e-((ro*g*((8*(q^2)*k1te)/((pi^2)*g*(db^4))))/6894.757));
ec7e=(p1e-((ro*g*((8*(q^2)*k2te)/((pi^2)*g*(d2^4))))/6894.757));
ec8e=(p1e-
((ro*g*(((f*(lc/dc))+kce)*((8*(q^2)*k1te)/((pi^2)*g*(dc^4)))))/6894.757));

h1=((8*(q^2)*k1te)/((pi^2)*g*(db^4)));
h2=(((f*(lc/dc))+kce)*((8*(q^2)*k1te)/((pi^2)*g*(dc^4))));
ht=h1+h2;
ec9=p1e-((ro*g*ht)/6894.757);


%---------presiones salida-----------------%
if ((as<0.1)&&(bs<0.1)&&(cs<0.1)&&(ds<0.1))
     flujo=0.0;
     calllib('smClient','writeFloatValue','Memoria2',0,1890.0);
     calllib('smClient','writeFloatValue','Memoria2',5,429.42);%p2

end
if ((as<0.1)&&(bs<0.1)&&(cs<0.1)&&(ds>=0.1))
    flujo=0.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1890.0);
     calllib('smClient','writeFloatValue','Memoria2',5,429.42);%p2
     calllib('smClient','writeFloatValue','Memoria2',4,0.0);
     m4=1;
end
if ((as<0.1)&&(bs<0.1)&&(cs>=0.1)&&(ds<0.1))
    flujo=0.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1890.0);
     calllib('smClient','writeFloatValue','Memoria2',5,429.42);%p2
     calllib('smClient','writeFloatValue','Memoria2',4,ec8);
     m3=1;
end
if ((as<0.1)&&(bs<0.1)&&(cs>=0.1)&&(ds>=0.1))
```

```
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',5,429.42);%p2
    calllib('smClient','writeFloatValue','Memoria2',4,0.0);
    m3=1;
    m4=1;
end
if ((as<0.1)&&(bs>=0.1)&&(cs<0.1)&&(ds<0.1))

    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
end
if ((as<0.1)&&(bs>=0.1)&&(cs<0.1)&&(ds>=0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
    calllib('smClient','writeFloatValue','Memoria2',4,0.0);
    m4=1;
end
if ((as<0.1)&&(bs>=0.1)&&(cs>=0.1)&&(ds<0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
    z3=ec8;
    m3=1;
    %m4=1;
end
if ((as<0.1)&&(bs>=0.1)&&(cs>=0.1)&&(ds>=0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1785.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
    calllib('smClient','writeFloatValue','Memoria2',4,0.0);
    m3=1;
    m4=1;
end
if ((as>=0.1)&&(bs<0.1)&&(cs<0.1)&&(ds<0.1))
    flujo=0.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1890.0);
    calllib('smClient','writeFloatValue','Memoria2',5,429.42);%p2
    calllib('smClient','writeFloatValue','Memoria2',4,429);

end
if ((as>=0.1)&&(bs<0.1)&&(cs<0.1)&&(ds>=0.1))
    flujo=0.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1890.0);
    calllib('smClient','writeFloatValue','Memoria2',5,150.0);
    calllib('smClient','writeFloatValue','Memoria2',4,150.5);
    m4=1;
end
if ((as>=0.1)&&(bs<0.1)&&(cs>=0.1)&&(ds<0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec7);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
    m3=1;
end
if ((as>=0.1)&&(bs<0.1)&&(cs>=0.1)&&(ds>=0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1785.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec7);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
    m3=1;
```

```
    m4=1;
end
if ((as>=0.1)&&(bs>=0.1)&&(cs<0.1)&&(ds<0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
end
if ((as>=0.1)&&(bs>=0.1)&&(cs<0.1)&&(ds>=0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1785.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec6);
    calllib('smClient','writeFloatValue','Memoria2',4,0.0);
    m4=1;
end
if ((as>=0.1)&&(bs>=0.1)&&(cs>=0.1)&&(ds<0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1800.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec5);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
    m3=1;
end
if ((as>=0.1)&&(bs>=0.1)&&(cs>=0.1)&&(ds>=0.1))
    flujo=255.0;
    calllib('smClient','writeFloatValue','Memoria2',0,1785.0);
    calllib('smClient','writeFloatValue','Memoria2',5,ec5);
    calllib('smClient','writeFloatValue','Memoria2',4,ec8);
    m3=1;
    m4=1;
end



%---------presiones entrada-----------------%
if ((ae<0.1)&&(be<0.1)&&(ce<0.1)&&(de<0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,172.0);
     calllib('smClient','writeFloatValue','Memoria2',7,5.8);%p2
     if (m3e==0)
        calllib('smClient','writeFloatValue','Memoria2',6,0.0);%p3
     else
         calllib('smClient','writeFloatValue','Memoria2',6,5.8);
     end
end
if ((ae<0.1)&&(be<0.1)&&(ce<0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,172.0);
     calllib('smClient','writeFloatValue','Memoria2',7,5.8);%p2
     calllib('smClient','writeFloatValue','Memoria2',6,0.0);
end
if ((ae<0.1)&&(be<0.1)&&(ce>=0.1)&&(de<0.1))
    flujo2=0.0;
    calllib('smClient','writeFloatValue','Memoria2',1,172.0);
     calllib('smClient','writeFloatValue','Memoria2',7,5.8);%p2
     calllib('smClient','writeFloatValue','Memoria2',6,5.8);
end
if ((ae<0.1)&&(be<0.1)&&(ce>=0.1)&&(de>=0.1))
    flujo2=0.0;
    calllib('smClient','writeFloatValue','Memoria2',7,0.2);%p2
    calllib('smClient','writeFloatValue','Memoria2',6,0.12);
end
if ((ae<0.1)&&(be>=0.1)&&(ce<0.1)&&(de<0.1))
    flujo2=254.0;
```

```
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    if (m3e==0)
        calllib('smClient','writeFloatValue','Memoria2',6,0.0);%p3
     else
         calllib('smClient','writeFloatValue','Memoria2',6,ec9);
     end
end
if ((ae<0.1)&&(be>=0.1)&&(ce<0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    calllib('smClient','writeFloatValue','Memoria2',6,0.0);
end
if ((ae<0.1)&&(be>=0.1)&&(ce>=0.1)&&(de<0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec9);
    m3e=1;
end
if ((ae<0.1)&&(be>=0.1)&&(ce>=0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,139.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    calllib('smClient','writeFloatValue','Memoria2',6,0.0);
end
if ((ae>=0.1)&&(be<0.1)&&(ce<0.1)&&(de<0.1))
    flujo2=0.0;
    calllib('smClient','writeFloatValue','Memoria2',1,172.0);
    calllib('smClient','writeFloatValue','Memoria2',7,5.8);%p2
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
end
if ((ae>=0.1)&&(be<0.1)&&(ce<0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,5.8);
    calllib('smClient','writeFloatValue','Memoria2',6,0.0);%%revisar valor
end
if ((ae>=0.1)&&(be<0.1)&&(ce>=0.1)&&(de<0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec7e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
end
if ((ae>=0.1)&&(be<0.1)&&(ce>=0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,139.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec7e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
end
if ((ae>=0.1)&&(be>=0.1)&&(ce<0.1)&&(de<0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
    m3e=1;
end
if ((ae>=0.1)&&(be>=0.1)&&(ce<0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,139.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec6e);
    calllib('smClient','writeFloatValue','Memoria2',6,0.0);
```

```
end
if ((ae>=0.1)&&(be>=0.1)&&(ce>=0.1)&&(de<0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,150.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec5e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
end
if ((ae>=0.1)&&(be>=0.1)&&(ce>=0.1)&&(de>=0.1))
    flujo2=254.0;
    calllib('smClient','writeFloatValue','Memoria2',1,139.0);
    calllib('smClient','writeFloatValue','Memoria2',7,ec5e);
    calllib('smClient','writeFloatValue','Memoria2',6,ec8e);
end

%--------------------------------------------------------------------%


calllib('smClient','writeFloatValue','Memoria2',2,flujo);
calllib('smClient','writeFloatValue','Memoria2',3,api);
calllib('smClient','writeFloatValue','Memoria2',8,flujo2);
calllib('smClient','writeFloatValue','Memoria2',10,pig2);


if (calllib('smClient','readFloatValue','Memoria2',9)>0)
    j=j+1;
    if(j==900000)
       pig2=1;
       calllib('smClient','writeFloatValue','Memoria2',9,0.0);
       j=0;
    end
end
i=i+1;
if(i==90000)
api=6*rand+24
i=0;
end

% figure
y = linspace(0,1000,0.5);
f1 = calllib('smClient','readFloatValue','Memoria2',0);
f2 = calllib('smClient','readFloatValue','Memoria2',5);
f3 = calllib('smClient','readFloatValue','Memoria2',4);
ff1(n)=f1;
ff2(n)=f2;
ff3(n)=f3;
n=n+1;
pause(0.2);
plot(ff1,'b');
plot(ff2,'g');
plot(ff3,'r');
legend('P1','P2','P3');

hold on;
end
```