Tecnológico de Costa Rica: Sede Cartago
Área académica de Ingeniería Mecatrónica

Desarrollo de una herramienta de simulación robótica para utilización y aprendizaje de conducción de vehículos pesados

Proyecto realizado en:

Universidad de São Paulo (USP)
Instituto de Ciencias Matemáticas y de Computación (ICMC)
Laboratorio de Robótica Móvil (LRM)

Autor:
Fletes Martínez, Ricardo
Carné: 201233476

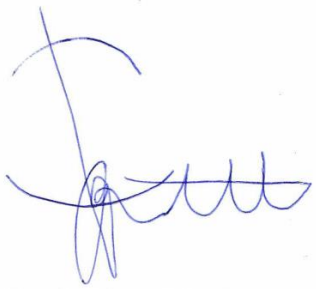
I Semestre, 2017

Declaratoria de Autenticidad

Declaro que el presente Proyecto de Graduación ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos propios.

En los casos en que he utilizado bibliografía, he procedido a indicar las fuentes mediante las respectivas citas bibliográficas.

En consecuencia, asumo la responsabilidad total por el trabajo de graduación realizado y por el contenido del correspondiente informe final.



Cartago, 20 de enero de 2017

Ricardo Fletes Martínez

Cédula: 4 - 0224 - 0039

INSTITUTO TECNOLÓGICO DE COSTA RICA

CARRERA DE INGENIERÍA MECATRÓNICA

PROYECTO DE GRADUACIÓN

ACTA DE APROBACIÓN

Proyecto de Graduación defendido ante el presente Tribunal Evaluador como requisito para optar por el título de Ingeniero en Mecatrónica con el grado académico de Licenciatura, del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal

M. Sc. Marta Eugenia Vílchez Monge

Profesor lector

M. Sc. Ing. Ignacio del Valle Granados

Profesor lector

Dr. Ing. Juan Luis Crespo Mariño

Profesor asesor

Los miembros de este Tribunal dan fe de que el presente trabajo de graduación ha sido aprobado y cumple con las normas establecidas por la Carrera de Ingeniería Mecatrónica

Cartago, 20 de Enero 2017

Dedicatoria

A mi familia en Costa Rica, quienes me han apoyado no solo en este proyecto sino que a lo largo de mi vida. Espero en **DIOS** este proyecto los llene de orgullo y que los pueda volver a ver pronto, los extraño.

Agradecimientos

Gracias **DIOS** mío, por dejarme vivir hasta este momento, por darme el potencial y la oportunidad para ir a otro país, conocer personas increíbles y desarrollar este proyecto del cual estoy tan orgulloso.

Gracias Sr. Osório, por darme la oportunidad de desarrollar mi proyecto en su universidad y por ser de tanta ayuda durante el traslado y el trabajo. Gracias también por sus atenciones cuando me fracturé el codo y por esos almuerzos que compartimos. Sinceramente le agradezco a **DIOS** por haberlo conocido, y siempre será bienvenido por mí y mi familia si decide venir a Costa Rica.

Quiero agradecer a algunos amigos, como Andreo y la pandilla, a la gente de la recepción, a Esteban, Lucas, Judas, João y la gente del LRM; ustedes hicieron de mi estadía una experiencia cálida y divertida. Gracias a Kyle, Rodolfo José y Gabriel, quienes se mantuvieron en constante contacto todos esos meses; realmente los aprecio amigos. Otros nombres importantes son Jennifer, Rodolfo, Sarai, Paula y Fabricio.

Finalmente, gracias a todas las personas que me motivaron a vivir esta experiencia y que siempre han creído en mí, como mi familia, Andrés y Aryeh, los profesores Crespo, Vílchez y Corrales; entre tantos otros.

Resumen

La introducción a una máquina implica el entrenamiento en el uso de sus mecanismos de actuación y control; mas esto conlleva desventajas como el consumo de combustible y el riesgo de dañar el equipo. Las industrias automotriz y aeronáutica han encontrado una solución en los simuladores, con una gama de opciones que equilibran su costo de compra y la recreación fidedigna de las condiciones de campo; pero otras áreas están restringidas a los extremos de soluciones profesionales de alta inversión o a módulos accesibles pero lúdicos. El presente proyecto discute el desarrollo de una solución de compromiso para el entrenamiento en excavadoras, con los objetivos de: instalar palancas de mando (joysticks) como controles, programar un detector de colisiones, diseñar los ambientes de entrenamiento y generar un algoritmo de puntajes. La plataforma de desarrollo fue el simulador robótico V-Rep, y todas las metas fueron alcanzadas.

Palabras clave: simuladores, excavación, V-Rep, simulador robótico.

Summary

The introduction to a new machine implies training with its actuation and control mechanisms, but that brings disadvantages like fuel-consumption and the risk of accidents. The racing and aeronautics industries have found a solution in computer simulations, with options that balance the acquisition price and the rigorousness of the field conditions recreation; but other areas are restricted to the extremes of expensive accuracy or affordable trivialness. The following project discusses the development of a middle-ground for excavator-training, with the objectives of: installing joysticks as the controls, programming a collision detector, designing the practice environments and generating a punctuation algorithm. The development platform was the robotics simulator V-Rep, and all the goals were achieved.

Keywords: excavator training, computer simulations, V-Rep, robotics simulator.

Tabla de Contenido

Tabla de Figuras

1. Introducción

1.1 Contexto

La empresa ARAMOV, dedicada al entrenamiento de personal en el uso de maquinaria pesada, solicitó informalmente al Laboratorio de Robótica Móvil (LRM) de la Universidad de Sao Paulo (USP) el desarrollo de una herramienta de entrenamiento que simulase la operación de una excavadora; con la idea de obtener una solución de compromiso entre un software de bajo costo y la capacidad de entrenamiento que ofrecería una opción profesional.

El LRM propuso el desarrollo de un prototipo mediante V-Rep: un software de simulación robótica gratuito con un motor de físicas, variedad de opciones de visualización, una interfaz de programación considerada por el laboratorio como accesible y versátil, y un modelo de un excavador de la compañía TEREX. De esta manera la inversión proyectada para ARAMOV o cualquier otra compañía interesada se reduciría a una computadora capaz de ejecutar el programa y los controles para operar la máquina. Se solicitó al estudiante desarrollar el proyecto en un plazo de 4 meses a partir del modelo de excavador mencionado.

La USP se encuentra en Sao Carlos, Sao Paulo, Brasil. El personal del LRM está conformado por personas en una gama de grados académicos, desde primer ingreso hasta post-doctorados, y se ha dedicado a la simulación y desarrollo de robots para varios fines. Entre sus proyectos destacan la generación de un control para la conducción automática de un camión y la implementación de un drone para la supervisión aérea de cultivos. Este personal se comprometió a la guía y apoyo del estudiante, facilitándole: el acceso a sus computadoras y accesorios, asistencia en el software utilizado, un presupuesto para expandir el hardware de la computadora, y otras comodidades referentes al acceso a las instalaciones de la universidad en general.

1.2 Problema

Las excavadoras son utilizadas en tareas que requieren la aplicación de fuerzas y el movimiento de pesos que resultarían muy difíciles o imposibles de realizar con la fuerza bruta de obreros, y por ende son equipos comunes en trabajos de construcción, excavación y demolición. No obstante, una operación indeseada por parte del operario incurre en gasto de combustible y tiempo, y puede resultar en accidentes letales y costosos: como el golpear a un obrero, el daño a una estructura aledaña o dañar a la excavadora en sí, entre otros.

Por ende, los nuevos operarios son previamente entrenados hasta dominar los controles y poder dimensionar espacialmente los movimientos del equipo. El problema es la novedad del esquema de controles: dos palancas de control (joysticks) para cambiar los ángulos en el brazo y dos pedales bidireccionales (pueden ser inclinados por la punta del pie o por el talón) para girar las orugas (Fig. 1).

Figura 1: Cabina de una excavadora. Extraído de Vantagetes (2009).

En un entrenamiento tradicional el operario entrena dentro de una excavadora común en tareas básicas y sin trabajadores alrededor, pero la inversión de recursos y la posibilidad de daños a la máquina siempre se mantienen. Una solución es usar simuladores profesionales, pero debido a su énfasis en recrear las condiciones de trabajo de manera realista la tecnología involucrada es costosa. Por otra parte existen simuladores más accesibles, empero presentan un carácter lúdico y están poco orientados en brindar entrenamiento formal.

El LRM propuso, bajo la petición de la compañía ARAMOV, la generación de un simulador de bajo costo pero orientado en entrenar apropiadamente al usuario en el uso de los controles, usando como plataforma de desarrollo el software de simulación robótica gratuito V-Rep. En un inicio, el proyecto consistió en adaptar un modelo de excavadora disponible en el programa para ser controlado por dos joysticks, y diseñar labores que entrenasen y evaluasen al operario.

El problema radicó en que el entrenamiento de un operario de excavadora se centra fuertemente en acostumbrarlo a los controles, pero el modelo del excavador disponible en la simulación no fue controlado mediante joysticks ni pedales sino por mouse (detalles en la Sección 2.). Además, el software no contó con tareas para entrenar ni mecanismos para evaluar ninguna habilidad específica del usuario. El LRM definió la siguiente línea de soluciones:

Delimitó el proyecto como un simulador para el entrenamiento en el control del brazo, el uso de los pedales quedó excluido. Para corregir los controles sugirió la búsqueda e implementación de palancas de control que se asemejasen a las utilizadas en la maquinaria real, empezando por pruebas con joysticks propiedad del laboratorio. Para las tareas propuso recrear ejercicios realizados en los entrenamientos tradicionales y evaluarlos mediante las métricas de su tiempo de ejecución (Tiempo) y el mantener una distancia mínima entre los objetos y la excavadora (Precisión). Esto lo separó en diseñar ambientes para cada tarea, generar un sistema de

detección de colisiones y/o distancias, y desarrollar un sistema que tradujese los parámetros medidos a puntajes; lo último a modo de facilitar al usuario el entendimiento de sus acciones y oportunidades de mejora.

Por todo lo anterior, el problema se delimitó como “el diseño e implementación de un sistema de control del brazo de la excavadora y evaluación del desempeño del usuario al utilizarlo, programado en V-Rep y basado en la línea de soluciones establecida por el LRM.”

1.3 Objetivos

1.3.1 Objetivo General

Desarrollar un simulador 3D para el entrenamiento de operarios en el uso del brazo de una excavadora, a partir del modelo de excavadora TEREX en V-Rep previamente manipulado por el LRM.

1.3.2 Objetivos Específicos

1. Implementar el sistema de control de posicionamiento de la herramienta basado en palancas de mando.

2. Desarrollar un algoritmo para detectar una distancia de separación específica entre los objetos simulados.

3. Desarrollar un algoritmo que asigne puntajes con relación al desarrollo de las tareas.

4. Diseñar ambientes y tareas de entrenamiento para la simulación.

1.4 Alcances y Limitaciones

El proyecto pretendió implementar los joysticks como controles del brazo, generar el detector de distancias/colisiones, desarrollar ambientes para evaluar el desempeño del usuario basados en los parámetros de Tiempo y Precisión, y desarrollar un sistema de puntajes para brindar retroalimentación de tal desempeño. No obstante, el proyecto no pretendió:

- Implementar el control de las orugas de la excavadora.
- Generar una interfaz específica entre los niveles o la presentación de los resultados.
- Estudiar o guardar en una base de datos los resultados del operario.
- Generar una cabina funcional basada en los entregables de los objetivos específicos o cotizar la posible manufactura de tal cabina.
- Proveer una simulación “realista” de los elementos que conforman los ambientes (solo funcional).
- Generar un ambiente para todas las prácticas sugeridas por alguna normativa o compañía (solo algunas y a criterio del diseñador).

2. Marco Teórico

El entrenamiento para el uso de una excavadora puede ser segmentado en el estudio de: los controles de la máquina, los trabajos de campo más frecuentes, las normas de seguridad y el mantenimiento del equipo. El proyecto se centró en los dos primeros campos, por lo que esta sección consiste en una descripción de los mismos. Además, se comentó las características del software utilizado en la solución con tal de entender varias de las decisiones de diseño tomadas más adelante.

2.1 Control de los movimientos

Los dos sistemas principales de una excavadora son su locomoción y el control de su brazo. El desplazamiento se consigue con el movimiento de dos orugas a los lados de la máquina, aunque en algunos casos pueden ser ruedas, y son controladas por un par de pedales que pueden ser inclinados hacia adelante usando la punta del pie o hacia atrás con la del talón. Cada oruga recibe la lectura de su pedal independientemente del otro y se mueve en el mismo sentido: los dos pedales hacia el frente mueven la máquina hacia adelante, los dos pedales hacia atrás activan la reversa e inclinaciones opuestas en los pedales generan giros (Edwards *et al,* 2003).

Por su parte, el control del brazo influye sobre los siguientes componentes (desde la punta y en orden): cubeta / herramienta, brazo, pluma y cabina (Fig. 2); aunque la cabina no suele ser considerada como “parte del brazo” (Lennie, 2014). La manipulación de los ángulos entre cada eslabón se media por pistones hidráulicos, mientras que la cabina rota usando un motor angular; todo esto es controlado a través de dos palancas de mando situadas una a cada costado del operario (Vantagetes, 2009). La lectura de los joysticks se basa en leer la posición de cada palanca de acuerdo a un eje horizontal y otro vertical, semejante a un eje cartesiano (Fig. 3), por lo cual se producen cuatro medidas (Chadli, 2013). Cada una es asignada a un pistón o motor diferente, de forma que se desplace en sentido positivo

o negativo con relación al sentido en que esta inclinada la palanca, y con una velocidad proporcional al grado de inclinación de la misma.

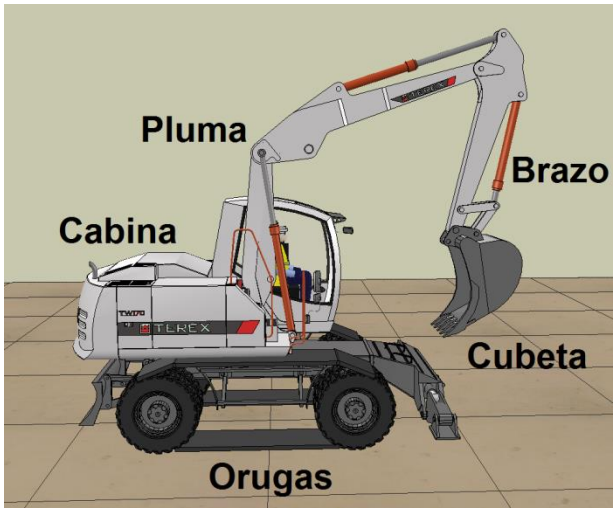


Figura 2: Partes de una excavadora. Desarrollado en V-Rep 2016.

Figura 3: Lecturas de un joystick. Editado en Paint 2016.

El mapeo de los registros a los motores tiene dos estandarizaciones posibles, la ISO y la SAE, aunque es decisión de cada compañía cuál usar o si agregar la opción de alternar entre las mismas. La normativa SAE asigna (siempre asumiendo una lectura positiva en cada eje): bajar la pluma al eje vertical derecho, abrir la cubeta al eje horizontal derecho, extender el brazo al eje vertical izquierdo y rotar la cabina hacia

la derecha al eje horizontal izquierdo (Fig. 4). El estándar ISO es igual excepto porque intercambia la función del eje vertical izquierdo con la del derecho (Singh *et al,* 1999).

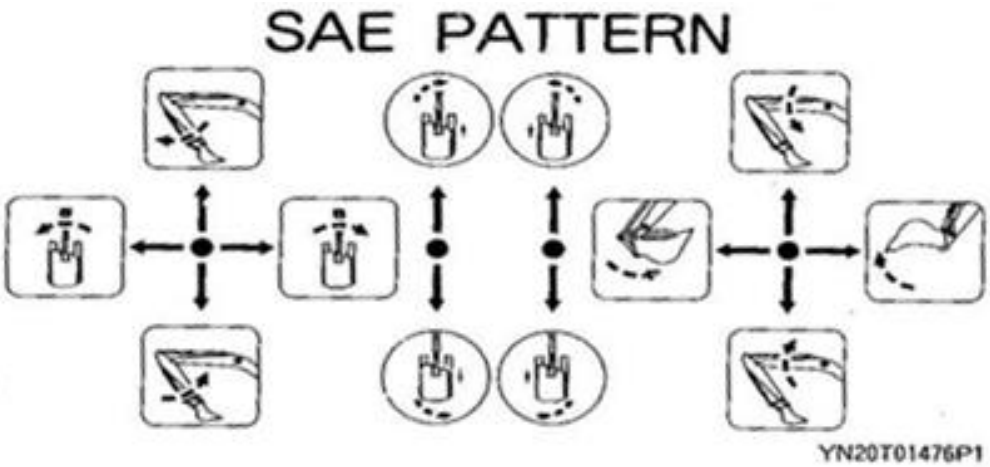


Figura 4: Estándar SAE. Provisto por Alves (2016).

2.2 Tareas De Entrenamiento

Todo entrenamiento comienza con la familiarización del usuario en el uso de los joysticks con el fin de adquirir la noción de dirección y sensibilidad de los movimientos (Alves, 2016), para luego explicar cómo estos están conectados a cada parte del brazo según la norma a usar. El objetivo es balancear la velocidad de los movimientos con la precisión para tocar suavemente o evitar superficies y cuerpos, además de memorizar los controles con tal de no realizar movimientos indeseados. Es posible crear una separación artificial si se usa un simulador, asignando una práctica previa que solo entrene el control de las palancas.

Una práctica más avanzada de estos conceptos es realizar cambios de herramienta, lo que conlleva posicionar el sistema lo suficientemente cerca para que un operario realice los ajustes manuales necesarios (Fig. 5). Otra labor afín es taladrar porque también consiste en posicionar la herramienta en una posición, ortogonal al piso, pero aumenta la complejidad al requerir reajustes dinámicos conforme se avanza hacia abajo.

Figura 5: Cambio de herramienta. Extraído de CATSimulators (2012).

Por su parte, los ejercicios de excavación enseñan al usuario los movimientos y ángulos a realizar con la pala dependiendo de la intención del corte, ya sea nivelar una superficie o remover material de un volumen. Esto limita las prácticas físicas a ser realizadas en campo, puesto que propiedades del material como la compresión, erosión, peso, fragmentación y dureza afectan su reacción ante el corte (Carter y Bentley, 2016), lo cual da las pautas para completar los movimientos. La complejidad aumenta si se toma en consideración el transporte del material desprendido a otra locación o vehículos (CATSimlators, 2012). Una práctica importante es “el corte de Talud”, un movimiento que permite nivelar la pendiente de un desnivel en el terreno (MrLumonca, 2011). Otra labor importante es la excavación de túneles rectos de poca anchura o trincheras para instalar tubos.

En cuanto al manejo de la polea existen 2 prácticas especializadas. La primera consiste en transportar una carga por un laberinto (vortexsim, 2015) para que el usuario aprenda que movimientos cortados o bruscos generan oscilaciones en la carga, y que las inclinaciones de los joysticks deben ser coordinadas con tal de mantener la carga en un rango de alturas deseado (Fig. 6). La segunda práctica consiste en perturbar al objeto cargado y que el usuario use la rotación del cuerpo para contrarrestar tal oscilación.

Figura 6: Circuito de prueba. Extraído de vortexsim (2015).

Finalmente, una labor básica que amalgama todo esto es el uso de blindajes para túneles: armazones metálicos que actúan como paredes de contención al cavar a ciertas profundidades (Fig. 7). La actividad implica usar una polea para posicionar el blindaje en una zona específica, usar la cubeta para golpear las esquinas de la caja hasta hundirla lo suficiente, cavar dentro del espacio cerrado que se propició, volver a rellenar de material si era necesario y extraer la caja con la polea (ricoinode, 2009).

Figura 7: Blindaje para trincheras. Extraído de ricoinode (2009).

2.3 Software de Simulación V-Rep

V-Rep fue el software de simulación robótica 3D utilizado durante el proyecto (Fig. 8). La fortaleza principal de V-Rep es que permite ensamblar robots tanto a partir de geometrías simples y elementos genéricos como a partir de modelos completos o sensores y actuadores individuales de marcas específicas, sujetos a la modificación, acople y programación por parte del usuario (Coppelia Robotics[3], 2016). Esto permite simular el desempeño y posibles fallas del robot usando el código propio de los componentes, tomar en cuenta posibles errores lógicos relacionados a la naturaleza del procesamiento de cada unidad y exportar el código final a la contraparte física de cada componente sin tener que adaptarlo de nuevo.

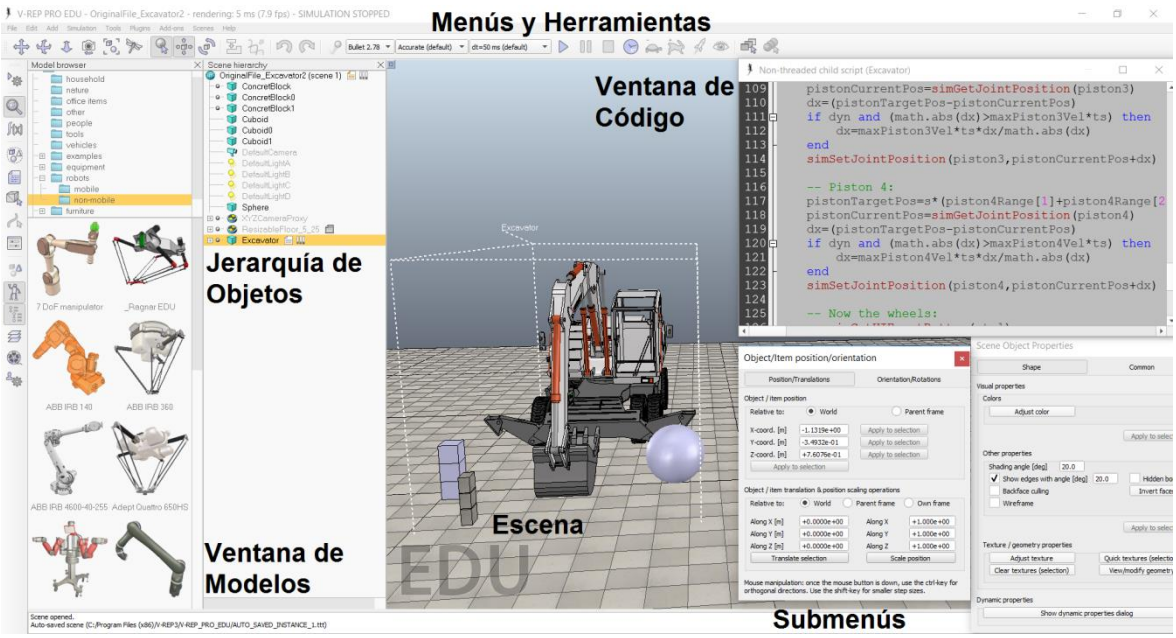


Figura 8: Ventana de V-Rep. Extraída de V-Rep 2016.

Con tal de extender estas comodidades, V-Rep se centra en facilitar la programación de los algoritmos al proveer Lua como su lenguaje base y complementarlo con una amplia gama de funciones especializadas para acelerar el proceso de compilación. La ventaja de Lua es su uso de “tablas y metatablas”, elementos que permiten

almacenar conjuntos de datos de cualquier tipo (enteros, flotantes, cadenas, buses, punteros, otras tablas) en una sola variable e interactuar con ellos mediante métodos especializados (Ierusalimschy, 2016). Esto facilita considerablemente el tratamiento de datos extraídos del ambiente y la forma de comunicarlos entre los componentes.

Por otra parte el simulador incluye varios motores de física con tal de añadir en consideración fuerzas como la gravedad, la fricción y los impactos, lo que permite verificar la validez de un diseño en cuanto a su locomoción e interacción con el ambiente. El programa también ofrece facilidades de visualización con un sistema de cámaras y ventanas, lo que permite centrar la ejecución en una perspectiva y cambiarla si es necesario o dividir la pantalla para visualizar algún ángulo adicional.

Otra ventaja es la capacidad de comunicarse con otros programas, como Python, C++, Matlab, SolidWorks o ROS, con tal de efectuar operaciones más complejas o de agregar objetos e interfaces más detallados. Esto permite extender la funcionalidad del simulador, como por ejemplo: al exportar modelos detallados de cuerpos, interpretar matricialmente la cinemática de un subsistema dado, estudiar el conjunto de datos recolectados por los sensores ya sea para el análisis de imágenes o la elaboración de mapas de puntos, o conectar controles externos como teclados, joysticks o mouse para aumentar la interactividad con la simulación (CoppeliaRobotics3, 2016).

No obstante, ya que las facilidades de programación son la prioridad las demás características poseen limitaciones importantes. Con respecto a lo comentado: la interfaz para manipular los objetos 3D es simple y rígida, los motores de física pueden presentar inestabilidades que imposibiliten el reposo de un cuerpo o dañen las conexiones entre los eslabones de un robot, cada adición de cámaras consume tiempo de procesamiento crítico, se experimenta de demoras de procesamiento al simular ambientes con una cantidad de objetos elevada o si estos cuerpos son muy complejos (CoppeliaRobotics1, 2014), y la conectividad con otros software es

altamente compleja si no se cuenta con una librería que lo facilite. V-Rep tampoco fue concebido para procurar el realismo más allá de las fuerzas comentadas, por lo que: deformar o cortar objetos no es posible aún (CoppeliaRobotics, 2015), no existen cables ni cuerdas (CoppeliaRobotics2, 2014), los ajustes de iluminación son muy limitados y carecen de un generador de sombras (CoppeliaRobotics, 2013), no hay librerías para añadir sonido, entre otros.

2.4 Antecedentes

El programa V-Rep se encontró instalado en las computadoras del LRM y fue instalado en el ordenador personal del estudiante, mas no existió ninguna cabina específica para la ejecución de las simulaciones. El laboratorio disposo dos joysticks Attack3 de la marca Logitech, predispuestos como los controles ideales del brazo del excavador (Fig. 9).

Figura 9: Joystick Attack 3 de Logitech. Foto tomada con la cámara de un Samsung S4 Mini.

Además, el laboratorio brindó una simulación con el modelo del excavador en escena y otros objetos de geometrías simples cercanos para realizar pruebas. Se reportó que la única modificación fue el cambio de color del vidrio de la cabina, de celeste translucido a transparente.

En cuanto a los detalles de la excavadora, esta fue extraída por el LRM de la sección de “modelos” de V-Rep; una colección de cuerpos complejos en cuanto a nivel de

detalle o funcionalidad, que incluyeron desde artículos decorativos hasta robots y vehículos digitales donados por sus compañías. El modelo fue un TW170 de TEREX: de cuatro ruedas independientes, cuerpo rotativo, un brazo de cuatro articulaciones, dos piernas de soporte extensibles y una cubeta como herramienta (Fig. 10). La cabina de trabajo careció de accesorios o conductor, y la pala trasera no fue funcional. Excepto por la cubeta y las ruedas el modelo traspasó los cuerpos en vez de interactuar con ellos.

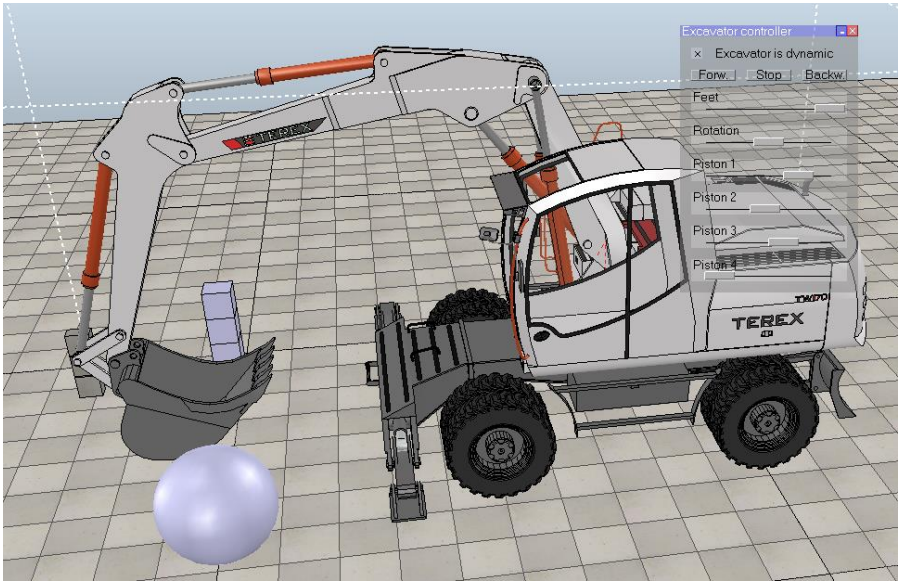


Figura 10: Modelo Original de la excavadora. Extraída de V-Rep 2016.

En el archivo original los ejes de cada llanta se encontraron simulados como un motor independiente con velocidad y torque ajustable y con retén de giro en caso de que su velocidad meta fuese cero. La rotación del cuerpo estuvo controlada por una articulación giratoria cuyo control fue respectivo a su posición angular, mientras que los eslabones del brazo eran manipulados al alterar la posición lineal de pistones. Este sistema se repitió en las piernas extensibles.

La excavadora estuvo programada para operar usando el mouse y un menú de botones y barras (Fig. 11). Los botones permitieron desplazar a la excavadora hacia

adelante o atrás mediante el reajuste de la velocidad de los motores de las ruedas a valores predeterminados. Por su parte, las barras deslizables determinaron la posición final de los demás pares cinemáticos: las posiciones del deslizador fueron mapeadas proporcionalmente a las posiciones de cada motor, y los límites de arrastre correspondieron a los límites de extensión y retracción de cada componente. Una vez alterada la posición de la barra una función se encargó de reajustar la posición del dispositivo respectivo hasta que llegase a la posición indicada. Este reajuste consistió en asignar una nueva posición para la extensión de la articulación al sumar a la actual un diferencial ya predeterminado, esto una vez por ciclo de la simulación. De esta manera el modelo simuló una “rapidez”, aunque los componentes no fuesen operables de este modo.

Figura 11: Interfase operada por mouse. Extraída de V-Rep 2016.

Cada pistón, articulación, motor, barra deslizable y botón estuvieron declarados de manera individual en la programación, y cada lectura y manipulación de estos tuvo su propia línea de código (Fig. 12).

```
-- Retrieve the joint handles:
frontLeftWheelMotor=simGetObjectHandle('frontLeftWheelJoint')
frontRightWheelMotor=simGetObjectHandle('frontRightWheelJoint')
rearLeftWheelMotor=simGetObjectHandle('rearLeftWheelJoint')
rearRightWheelMotor=simGetObjectHandle('rearRightWheelJoint')
leftFootPiston=simGetObjectHandle('footPistonLeft')
rightFootPiston=simGetObjectHandle('footPistonRight')
rotationMotor=simGetObjectHandle('BaseRotation')
piston1=simGetObjectHandle('Piston1')
piston2=simGetObjectHandle('Piston2')
piston3=simGetObjectHandle('Piston3')
piston4=simGetObjectHandle('Piston4')
```

Figura 12: Declaración original de los objetos. Extraída de V-Rep 2016.

Por último, la simulación estuvo mostrada desde una perspectiva en tercera persona, provista por una cámara desplazable a conveniencia.

2.5 Estado del Arte

Las simulaciones 3D se han consolidado como una alternativa de entrenamiento porque permiten realizar todas las sesiones y errores necesarios sin el subsecuente consumo de recursos ni daños al equipo. No obstante, siempre existe un punto donde es necesario pasar a entrenar en la máquina real, y la diferencia entre las soluciones es qué tan lejos está ese punto.

Los simuladores profesionales, como los de las compañías CAT (CATSimlators, 2012) y Vortex (vortexsim, 2015), pretenden enseñar al usuario todas las pericias necesarias en la zona de trabajo previo a siquiera introducir la máquina física; además de ser un recurso para evaluar rutinariamente a los operarios. El problema es que las labores más comunes, como excavar o transportar cargas, son fuertemente dependientes de las condiciones de trabajo; y la recreación de estas particularidades conlleva el uso de tecnologías que incrementan el precio del producto final.

Como ejemplos del hardware utilizado (Fig. 13): las cabinas de mando recrean tanto los controles como las dimensiones de una cabina real, hay motores que recrean las fuerzas y vibraciones que se sentirían durante la ejecución de cada labor, y hay pantallas que despliegan los ángulos de visualización de los vidrios y espejos de una cabina real. En cuanto al software hay varios módulos de física y accesorios que añaden: luces y sombras dependientes de la hora del día planteada en la simulación, trabajadores circundantes, sonidos de máquinas, tierra excarvable y cargas dinámicas, entre otros. También existen varios registros para recopilar e inclusive analizar datos como la frecuencia de errores o la cantidad de tierra excavada por movimiento.

Figura 13: Cabina de Simulación de la CAT. Extraída de CATSimulators (2012).

Por otra parte, los simuladores más económicos solo pretenden ser medios de entretenimiento, máximo mecanismos para aprender los conceptos más básicos de una operación. Por ejemplo, juegos como DIG IT!-A Digger Simulator (Fig. 14) y Power Shovel usan controles diferentes a los joysticks, se juegan en una perspectiva fuera de la cabina, implementan prácticas no estandarizadas y usan motores de física poco rigurosos. Aunque la perspectiva y los controles a veces pueden ser personalizados los últimos dos problemas imposibilitan el uso de estos juegos como herramientas formales de aprendizaje.

Figura 14: Simulación en DIG IT! – A Digger Simulator. Captura de pantalla de elaboración propia.

Finalmente el mercado carece de soluciones intermedias, a diferencia de otros campos como los simuladores de vehículos livianos. Una solución de este tipo procuraría el esquema de controles y prácticas de la variante profesional, con tal de enseñar las labores y los mecanismos involucrados durante su ejecución, pero mantendría su nivel de inversión bajo mediante el uso de hardware accesible y al ser desarrollado con una plataforma libre o no tan costosa; tal como pretende el presente proyecto.

3. Metodología

A continuación se describen las posibles soluciones contempladas para cada objetivo.

3.1 Implementación las palancas de mando

El primer objetivo hizo referencia a instalar dos joysticks como los controles de la excavadora, e idealmente usar los provistos por el laboratorio. Se decidió investigar sobre la posibilidad de realizar la lectura de las palancas mediante alguna función propia del software o una aportada por algún usuario anónimo, puesto que V-Rep es un software libre y su sitio oficial admite código generado por terceros. En caso de no hallar una función satisfactoria se contempló codificar el algoritmo de cero, probablemente con una comunicación a través de Python o C++, o investigar y comprar algún modelo de joysticks que si fuesen soportados por el software. Finalmente, en caso extremo se planteó el ensamblar los controles a partir de kits de Arduino, puesto que este microcontrolador si posee funciones para ser conectado a V-Rep.

Seguidamente, se planteó modificar el código del control de los movimientos de la excavadora con el fin de que usase las lecturas de los joysticks como entradas; esto con tal de ahorrar tiempo y conservar cualquier propiedad o parámetro especial que haya sido contemplado por los programadores de la excavadora original. En caso contrario se consideró codificar nuevas funciones para realizar los movimientos.

3.2 Desarrollo de un detector de distancias

Puesto que el segundo objetivo fue desarrollar un detector de distancias entre los objetos se planeó investigar la existencia de funciones que cumpliesen directamente con tal objetivo. En caso de no hallar, o de que la implementación de las mismas fuese perjudicial o inconveniente para los fines de la simulación, se consideró generar modificaciones a las funciones o programar nuevas de cero. En caso extremo, como que el uso de la función implicase fuertes demoras en el tiempo de procesamiento, se consideró limitar el uso del detector de distancias al mínimo necesario y diseñar los niveles para evitar su uso.

3.3 Desarrollo de un sistema de puntajes

El tercer objetivo, el desarrollo de un sistema de puntajes, fue concebido como un medio de retroalimentación para el usuario, de manera que tuviese presente su desempeño a lo largo de la tarea y todos los errores que hubiese cometido durante el proceso. Por ende, se consideró programar una interfaz que mostrase mensajes al usuario cada vez que cometiese un error, de modo que este tuviese claro cuándo se equivocó y pudiese corregirlo en el momento. En caso de no ser posible, se consideraron otras opciones como mostrar un compendio de todos los errores realizados solo al final de la simulación, o reducir el puntaje y valerse de elementos como colores y sonidos para indicar los accidentes.

Para demarcar los errores el LRM propuso con anterioridad el uso de los parámetros de Tiempo y Precisión, por lo que se decidió usar estos como base. Se planteó basar la penalidad por Tiempo en cronometrar la realización de movimientos simples o tareas pequeñas, o de no ser posible entonces cronometrar la compleción de los niveles. Para penalizar la Precisión se planeó usar el detector de distancias del objetivo anterior para corroborar la posición de los objetos y colisiones entre ellos. En caso de no poder evaluar con estas métricas se consideró conceptualizar nuevas.

Por último, para el sistema de puntaje se consideró las siguientes mecánicas: deducir puntos de acuerdo a las penalizaciones, sumar puntos al completar tareas pequeñas, o un contador de penalizaciones que al llegar a un máximo terminase el nivel instantáneamente. Para todas se planeó llevar registros numéricos dentro de la simulación.

3.4 Desarrollo de los ambientes y tareas

Para el desarrollo de los niveles se planteó investigar las tareas y entrenamientos más comunes al momento de usar una excavadora, y verificar la capacidad de recrearlos en V-Rep. También se consideró entrevistar a algún funcionario de ARAMOV sobre sugerencias nuevas o mejoras a los entrenamientos comentados. Se consideró que probablemente recrear los niveles involucraría generar nuevos objetos o funciones, para lo cual se usarían los módulos de objetos 3D y la programación en LUA propios del software.

Se consideró 8 niveles, en específico para: entrenar el uso de los joysticks sin usar la excavadora, entrenar el uso de los joysticks al mover la excavadora, practicar los ángulos del brazo al enganchar herramientas, realizar pequeños ajustes en el ángulo de la herramienta, practicar el transporte de objetos en espacios cerrados, practicar enganchar y liberar cargas, hundir blindajes en el suelo, y realizar cortes sobre la tierra.

4. Desarrollo de la Solución

A continuación se discute el desarrollo de cada objetivo, con hincapié en la etapa conceptual y en los resultados parciales más influyentes.

4.1 Implementación las palancas de mando

Primero se intentó de reemplazar el control mediante la interfaz y el mouse usando los controles provistos por el laboratorio, por lo que se investigó sobre la existencia de librerías que permitiesen la comunicación entre V-Rep y los dos joysticks. Se encontró que la función simExtJoyGetData( ) sirvió para este cometido y que estuvo instalada desde la instalación del programa a partir de un pluggin llamado “v_repExtJoystick.dll”. Empero, la función solo extrajo datos de un joystick por lo que se usó la modificación propuesta por el usuario jlouis2k4 (2013) para extenderlo habilitar la lectura del segundo.

En conjunto, con tal de facilitar la evaluación y edición de los ambientes fuera de las instalaciones del LRM y para agregar funcionalidades cuyo acceso no se consideró conveniente desde los joysticks, se programó la lectura del teclado mediante la función simGetSimulatorMessage( ). Se almacenó cada lectura en una tabla llamada “control”, en el siguiente orden: eje vertical derecho, eje horizontal derecho, botón derecho, eje vertical izquierdo, eje horizontal izquierdo, botón izquierdo y lectura del teclado.

Se programó un modo de lectura adicional para el caso de que no se detectasen los joysticks, donde la lectura del teclado fue mapeada con la función mapping( ) para que equivaliese a las 6 otras lecturas de los controles. Debido a que la función simGetSimulatorMessage( ) solo detectó la última tecla presionada, en vez de detectar todas las teclas en uso en un momento dado, esta opción se caracterizó por no permitir movimientos simultáneos ni fluidos, y por eso se reservó como una

herramienta de edición. Además, se dispuso la posibilidad de intercambiar entre el control por joysticks o por el teclado en cualquier momento al presionar una tecla.

Se desarrolló la función pairMovement( ) para traducir las lecturas de los controles como las acciones sobre los motores de la excavadora. La función conservó la premisa del modelo original, realizar pequeños cambios de posición durante cada ciclo de la simulación basado en un diferencial pre-programado, pero se modificó para que los movimientos fuesen realizados solo si se detectase una lectura desde el control. También se implementó código con tal de variar el tamaño de los cambios de posición con respecto al grado de inclinación del control. Primero, se discretizó la lectura del control en valores entre el -1 y el 1 con pasos de 0,1. Después, se agregó una región muerta: una línea que cambió los valores entre el -0.3 y 0.3 a 0; esto con tal de que ligeros desajustes del control o movimientos demasiado leves por parte del usuario no fuesen detectados. El resultado fue usado como un escalar sobre el diferencial de reajuste, de forma que a mayor inclinación el paso se acercó más a su magnitud máxima. Además, sirvió para asignar sentido al paso debido a la correlación entre el signo del escalar y la dirección en la que apuntó la palanca.

Finalmente se programó la función pairCheck( ) para mapear cada lectura a su respectiva articulación y aplicar la función pairMovement( ) con cada pareja (Tabla 1). Se agregó la opción de cambiar entre el modo ISO y SAE a través de una orden desde el teclado, lo que se basó en intercambiar las articulaciones controladas por el eje vertical derecho e izquierdo.

Tabla 1. Mapeo de las lecturas al manejar el excavador en configuración SAE. Desarrollado en Word 2013.

Acción	Lectura del Joystick	Lectura del teclado
Extensión del brazo	Eje Vertical Izquierdo hacia Arriba	W
Retracción del brazo	Eje Vertical Izquierdo hacia Abajo	S
Bajar la pluma	Eje Vertical Derecho hacia Arriba	Q
Alzar la pluma	Eje Vertical Derecho hacia Abajo	A
Abrir la cubeta	Eje Horizontal Derecho hacia la Derecha	E
Cerrar la cubeta	Eje Horizontal Derecho hacia la Izquierda	D
Rotar hacia la derecha	Eje Horizontal Izquierdo hacia la Derecha	X
Rotar hacia la izquierda	Eje Horizontal Izquierdo hacia la Izquierda	Z
Botón de acción	Botón Central Derecho	P
Claxon	Botón Central Izquierdo	B
Ajuste de Velocidad	Regulado con la inclinación	C / V
Elevar la cámara	Eje Vertical Derecho hacia Arriba + Gatillo Derecho	I
Bajar la cámara	Eje Vertical Derecho hacia Abajo + Gatillo Derecho	K
Rotar la cámara hacia la izquierda	Eje Horizontal Derecho hacia la Izquierda + Gatillo Derecho	J
Rotar la cámara hacia la derecha	Eje Horizontal Derecho hacia la Derecha + Gatillo Derecho	L
Habilitar/Deshabilitar la lectura del joystick	Tab	Tab
Habilitar/Deshabilitar la configuración SAE	9	9

4.2 Desarrollo de un detector de distancias

Para el detector de distancias/colisiones se investigó si V-Rep tuvo funciones pre-programadas con el mismo objetivo o similar. El detector de distancias fue la función simReadDistance( ), que calculó vectorialmente la menor distancia entre dos superficies (CoppeliaRobotics[2], 2016). No obstante, esto involucró el procesamiento de varias operaciones sobre una colección de puntos, y por ende fue más costoso en recursos y tiempo. Por otra parte el detector de colisiones fue la función simCheckCollision( ), que verificó la superposición entre dos superficies (CoppeliaRobotics[1], 2016). Esta fue la alternativa escogida (Fig. 15).

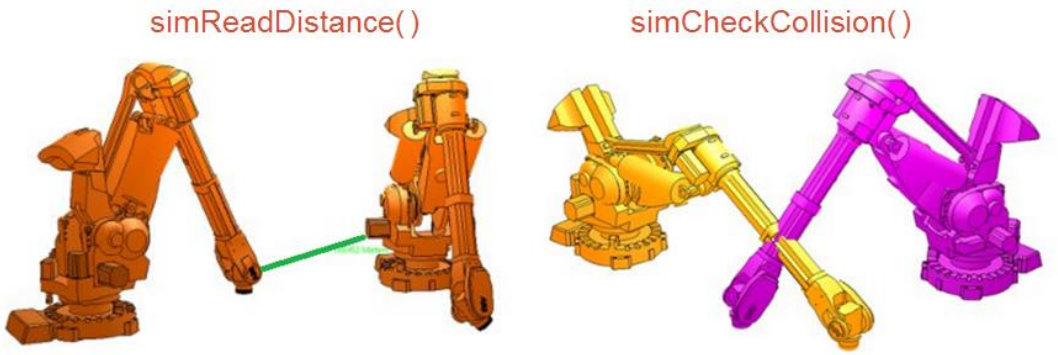


Figura 15: Comparación entre el detector de distancias y el de colisiones. Extraído de CoppeliaRobotics[1] (2016) y CoppeliaRobotics[2] (2016).

Empero, la superposición de dos cuerpos sólidos fue imposible, ya que en su lugar colisionaron como lo hubieran hecho dos cuerpos en la realidad. Como solución se encerró a los objetos con geometrías simples, como cubos y cilindros, y traspasables por cualquier objeto, en una técnica conocida como “hitbox” (Fig. 16). La idea fue que si otro cuerpo sólido se acercase lo suficiente al objeto encerrado tendría que atravesar el hitbox antes de realizar contacto, y esa superposición si sería detectable. El método fue versátil, ya que en caso de encerrar a los cuerpos superficialmente el detector fue de colisiones, mientras que al encerrarlos con un mayor margen se detectó la violación de una distancia mínima de separación. Otras ventajas fueron: la

posibilidad de escoger cuál de los cuerpos de interés encerrar con base en su simplicidad, la capacidad de detectar colisiones entre dos hitbox y la oportunidad de asignar hitbox a regiones del espacio en vez de a objetos, creando “detectores de presencia.

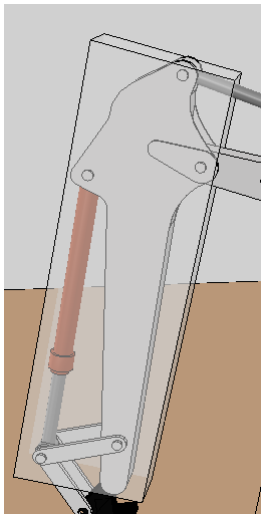


Figura 16: Prisma usado como hitbox. Desarrollado en V-Rep (2016).

Se agregó al modelo de la excavadora su propio hitbox superficial (Fig. 17), y cada tarea elaborada en la sub-sección 4 fue provista con hitbox para los cuerpos, obstáculos y zonas necesarias.

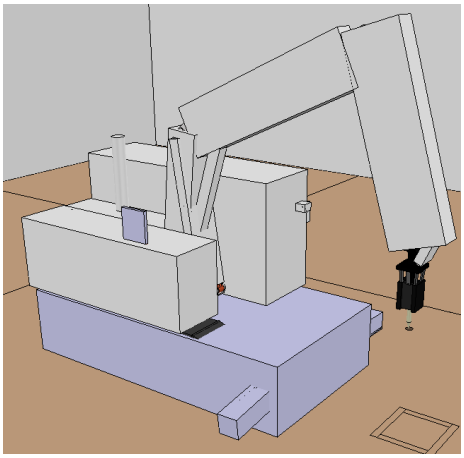


Figura 17: Sistema de hitbox de la excavadora. Desarrollado en V-Rep (2016).

4.3 Desarrollo de un sistema de puntajes

El sistema de puntaje se basó en la asignación inicial de 100 puntos para la tarea y dos algoritmos de penalización que fueron ejecutados durante cada ciclo, de modo tal que durante la práctica solo se pudo perder puntos en vez de ganarlos. El sistema se diseñó de esta manera al tomar en consideración que el operario es pagado por completar todas las labores asignadas en un tiempo límite, y en caso de fallar con el tiempo o cometer accidentes sufre de deducciones en su salario.

Como paso preliminar se desarrolló una interfaz básica para desplegar mensajes al usuario, llamada con la función printingTime( ). Cada vez que se usó: desactivó el control del usuario durante un tiempo especificado por el programador y desplegó un mensaje pre-programado. Esto fue programado para informar al usuario detalles importantes de cada nivel o la pérdida de puntos por cometer errores (Fig. 18).

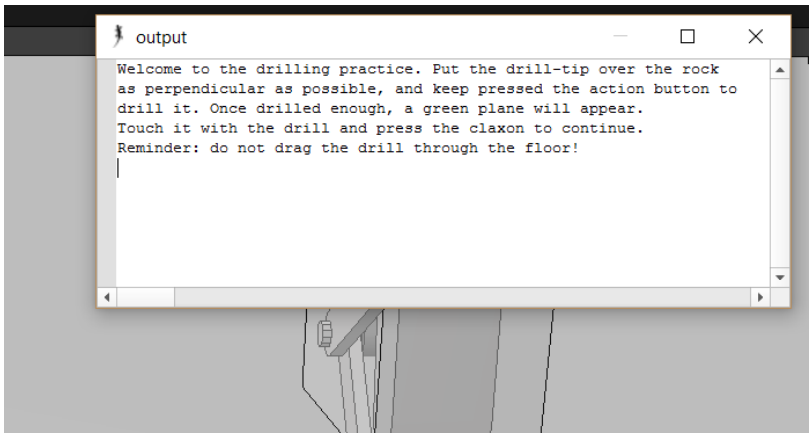


Figura 18: Impresión de un mensaje. Desarrollado en V-Rep (2016).

La primera penalización fue por cometer errores durante el proceso, como por ejemplo arrastrar una carga por el piso o golpearla contra un obstáculo (Fig. 19). En tal caso ocurrió una deducción de puntos, se le notificó al usuario y se le concedió 5 segundos con tal de corregir la situación sin ejecutar el algoritmo. Este tiempo de gracia fue necesario porque en caso contrario la resta de puntos y el despliegue del

mensaje hubiesen ocurrido nuevamente durante cada ciclo, interrumpiendo los intentos del usuario por solucionar su error.

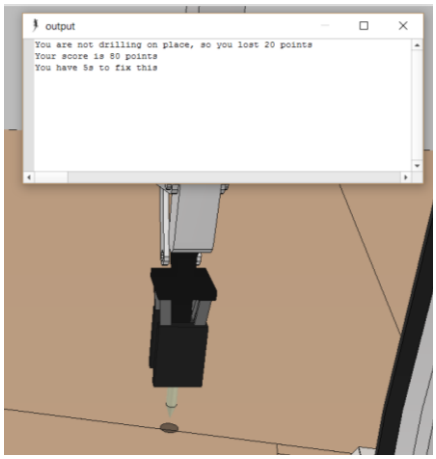


Figura 19: Penalización de Precisión. Desarrollado en V-Rep (2016).

Por otra parte cada tarea fue dividida en subprocesos con un tiempo de realización dado, y si el usuario completó la acción fuera de esa ventana de tiempo recibió una penalización (Fig. 20). Este algoritmo no se usó en algunos segmentos con tal de brindar rutinas “de práctica” para que el usuario se acostumbrase a los controles. La decisión de cronometrar subprocesos en las prácticas en vez de la práctica entera fue con tal de alertar al usuario de cualquier ineficiencia tempranamente e incentivarlo a corregirla antes de finalizar la sesión.

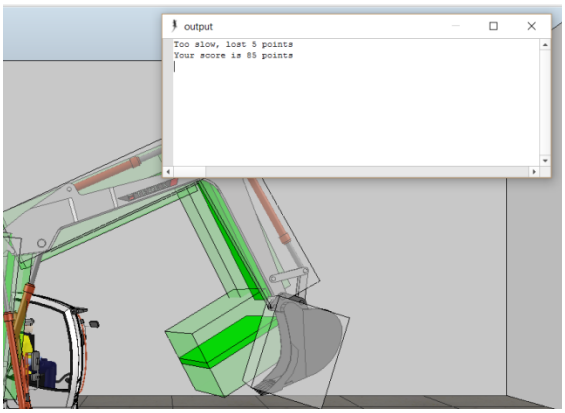


Figura 20: Penalización de Tiempo. Desarrollado en V-Rep (2016).

Todos los muestreos de tiempo se realizaron con la función simGetSystemTime( ). La asignación de los tiempos y las penalizaciones iniciales fueron arbitrarias y se corrigió reiteradamente hasta obtener tareas cuya completitud fuese alcanzable por el usuario y cuyos valores de penalización fuesen aprobados por el asesor local.

4.4 Desarrollo de los ambientes y tareas

Previo a la elaboración de los ambientes, se desarrollaron varios módulos y mecánicas adicionales, con tal de optimizar código y de proveer el material necesario para cada nivel. La primera sub-sección se centra en esto, y luego cada ambiente posee su propia explicación.

4.4.1 Funciones Preliminares

Como primera simplificación se cambió la manera de declarar los componentes de la excavadora. Por ejemplo, en el archivo original el motor de cada rueda tuvo un nombre que especificó su posición entre delantera o trasera y derecha o izquierda, con declaraciones individuales como “frontLeftWheelJoint”. En el nuevo sistema el nombre fue “WheelX”, donde “X” fue un número representativo del uno al cuatro representativo de cada rueda.

Esto se aplicó también a los pistones y articulaciones de la excavadora, ahora con el nombre de “PairX”, y a obstáculos y objetos importantes para cada actividad. La ventaja es que permitió desarrollar una función llamada modelCall( ), que generó una tabla con los punteros de cada uno de los objetos de una categoría en vez de almacenar cada uno individualmente, y otras funciones especializadas en recorrer cada miembro de la tabla para realizar una función. Esto comprimió los algoritmos considerablemente, a cambio de mantener un registro de la relación número-objeto (Fig. 21).

```
164    excavator = simGetObjectAssociatedWithScript( sim_handle_self ) -- The excavator itself
165    pair = modelCall( 'Pair' , 9 ) -- The kinematic pairs: 1) Piston 1 , 2) Piston 2 , 3) Piston 3 , 4) Piston 4 , 5) Rotate Body
166    shovel = modelCall( 'Shovel' , 4 )
167    soil = modelCall( 'Soil' , 5 )
```

Figura 21: Declaración de variables usando modelCall( ). Desarrollado en V-Rep (2016).

Con tal de facilitar la percepción del espacio tridimensional se desarrolló el equivalente a una sombra simple. Se agregó a la simulación un plano rectangular o circular translúcido de color negro, posicionado levemente arriba del piso y programado para seguir a un objeto sin abandonar su altura o ángulo respecto al suelo, usando las funciones simSetObjectPosition( ) y simSetObjectOrientation( ). Configurada para estar por debajo de una herramienta o carga, la sombra brindó una ayuda visual para calcular la posición y trayectoria del objeto (Fig. 22). Como variante más avanzada se programó una sombra cuyo diámetro cambió dinámicamente con respecto a la altura de su objeto guía, con la función simScaleObject( ).

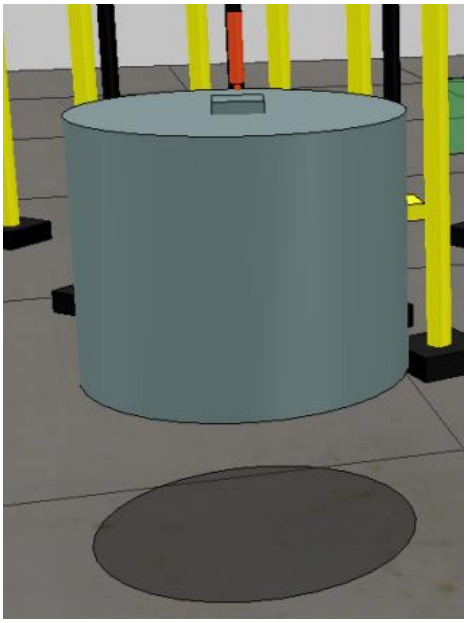


Figura 22: Sombra de una carga. Desarrollado en V-Rep (2016).

Se configuró un sistema de visión en primera persona desde la cabina de la excavadora. Con tal de asegurar una posición semejante a la de una persona real se usó un modelo de trabajador sentado que V-Rep ofreció bajo el nombre de “Sitting Bill”. Una vez centrado en la cabina se instaló una cámara en la posición de sus ojos. La cámara fue conectada a la excavadora mediante dos articulaciones, una para el

ángulo de elevación y otra para el ángulo de giro, controlables a través de los joysticks (Fig. 23).

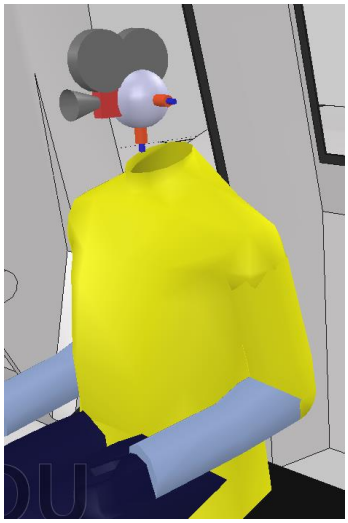


Figura 23: Cámara para la vista dentro de la cabina. Desarrollado en V-Rep (2016).

Se estableció un sistema de cámaras adicionales para reemplazar la ayuda que suministraría un compañero en tierra para guiar al operario. El sistema consistió de un hitbox para la región de interés y una cámara (Fig. 24). Cuando el hitbox detectó un objeto específico se usó la función simAdjustView( ) para abrir un recuadro adicional y mostrar la toma de la cámara. Al terminar la acción o dejar de realizar contacto con el hitbox se cerró el recuadro con simFloatingViewRemove( ).

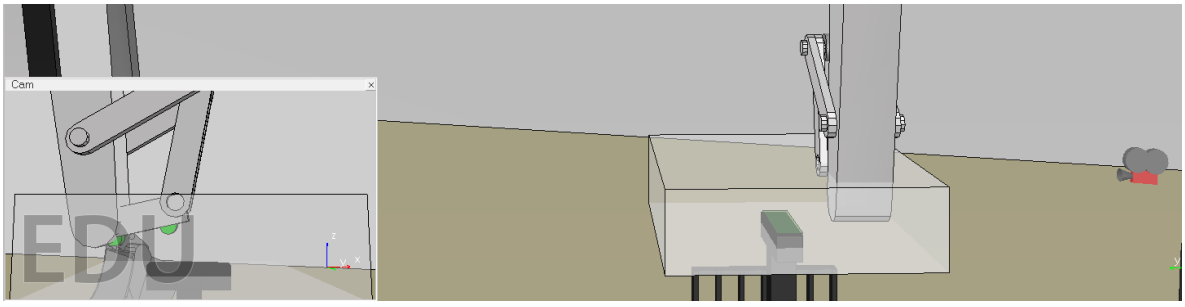


Figura 24: Sistema hitbox-cámara. Desarrollado en V-Rep (2016).

Se desarrollaron los modelos para tres herramientas nuevas: una polea, un taladro y un tenedor (Fig. 25). El cable de polea consistió en una serie de cilindros conectados mediante articulaciones bajo la recomendación del fórum de CoppeliaRobotics2 (2014). El taladro fue elaborado con base en imágenes del Cat Hammer H55E (CAT[2], 2016) y el tenedor con base en el CAT Fork 1220mm (CAT[1], 2016). Se programó la función toolChange( ) para poder intercambiar de herramienta, basada en la función simSetObjectParent( ) con tal de conectar y desconectar cada herramienta del brazo de la excavadora.

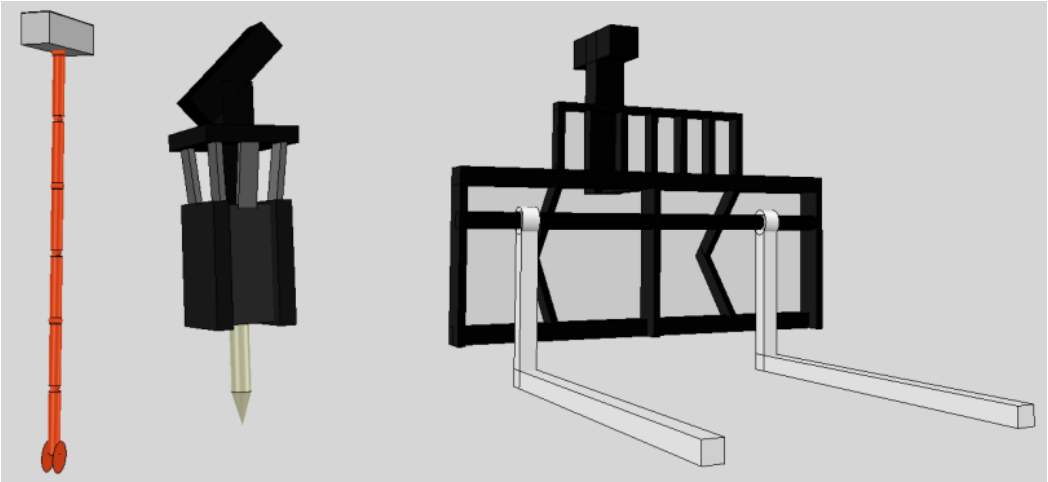


Figura 25: Polea (izquierda), taladro (centro) y tenedor (derecha). Desarrollado en V-Rep (2016).

Otras decisiones estéticas fueron agregar geometrías para que las ruedas se viesen como orugas, agregar al modelo del conductor los colores de un traje de seguridad y agregar el logo de la empresa ARAMOV (Fig. 26).

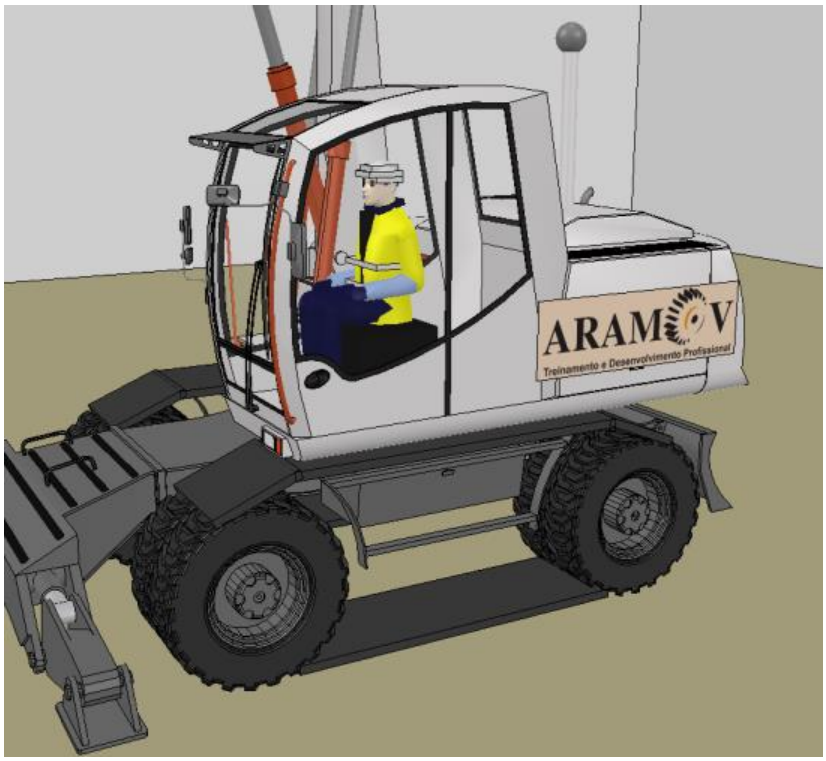


Figura 26: Cambios estéticos. Desarrollado en V-Rep (2016).

En cuanto a los niveles, estos fueron conceptualizados como una única tarea corta desarrollada en un ambiente específico. Esta decisión se tomó con tal de garantizar la atención del usuario en una sola labor y debido a las limitaciones en la capacidad de procesamiento del software al agregar objetos y código. Los niveles fueron escogidos con ayuda del tutor y un operario de ARAMOV (Alves, 2016).

4.4.2 Primera Práctica: JoystickPractice

Se programó primeramente un ejercicio para aprender a usar los joysticks, con tal de controlar la sensibilidad y dirección de los movimientos. La práctica consistió en mover un círculo a través de un recorrido lleno de obstáculos, con la intención de alcanzar una estación meta pero con penitencias en caso de tocar los obstáculos o paredes (Fig. 27). Se programó la cámara para observar esto desde una vista superior y seguir cada movimiento del círculo.

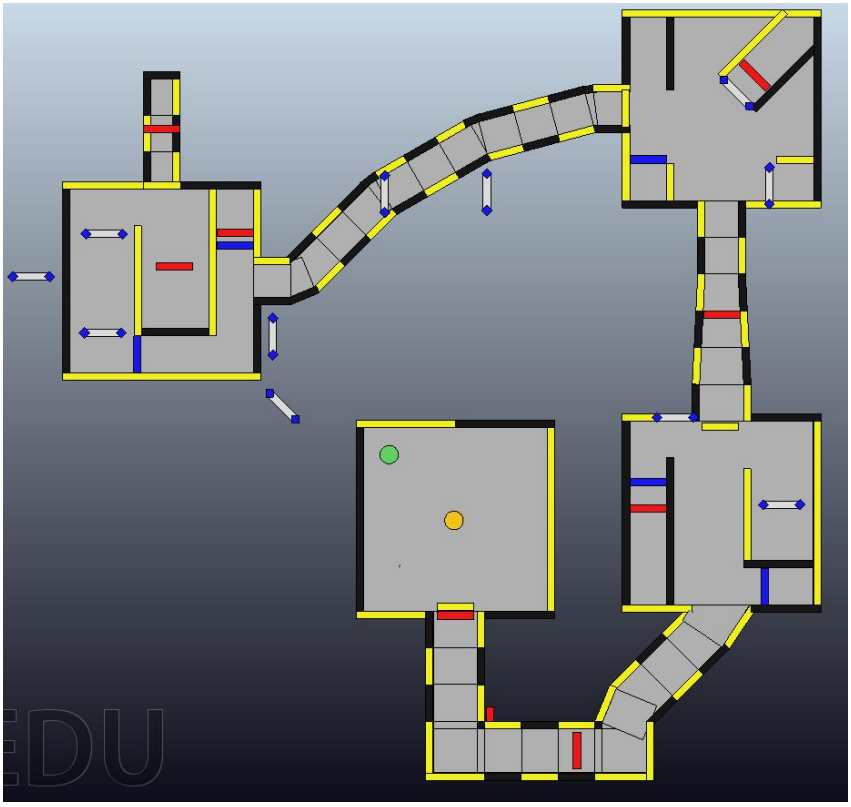


Figura 27: Vista superior del nivel JoystickPractice. Desarrollado en V-Rep (2016).

El recorrido comenzó en una arena cuadrada, con un mensaje para el usuario que indicó su libertad para experimentar movimientos con la palanca de control sin ningún límite de tiempo mientras no tocase un círculo verde aledaño. La intención de esto fue introducir al usuario a los movimientos a su propio ritmo hasta que se

sintiese cómodo. Estos movimientos fueron controlados con la función movement( ), una modificación ligera de la función pairMovement( ) para que en vez de alterar la extensión de motores en la excavadora alterase la posición del círculo controlado por el usuario.

El círculo verde consistió en un hitbox ligado a una posición específica en el recorrido. Al detectar al usuario el hitbox fue trasladado a la siguiente coordenada, mientras se ligó la coordenada vieja para el usuario. La coordenada del usuario se usó como punto de reinicio en caso de realizar contacto con una pared u obstáculo, a modo de castigo por el error además de la reducción de puntos. De esta manera el avance a través del recorrido fue progresivo, recompensando la superación de cada sección con un nuevo punto de reinicio más cercano a la meta (Fig. 28).

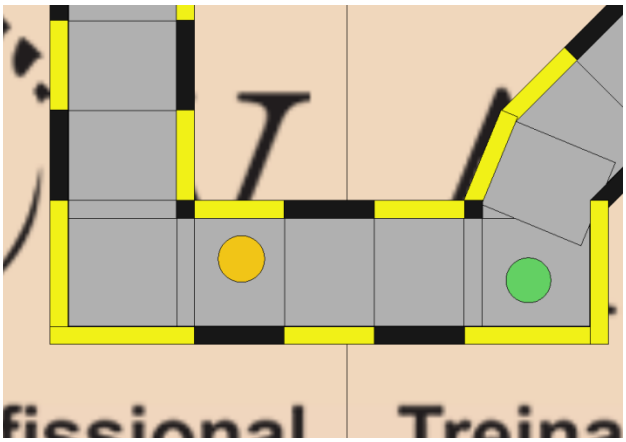


Figura 28: Seguimiento paulatino del hitbox verde. Desarrollado en V-Rep (2016).

Una vez impactada la primera estación verde se le dio al usuario la instrucción de recorrer un puente a la siguiente arena. Los puentes tuvieron el propósito de entrenar movimientos en una dirección definida y sin vibraciones, por ser caminos angostos. Las arenas fueron más anchas pero con más obstáculos y estaciones repartidas en varios puntos, para incentivar la concentración en resolver un problema a la vez. Las 3 arenas y 4 puentes fueron provistos con un tiempo para ser recorridos, registrado

entre una estación a cada inicio y una a cada final. En caso de sobrepasar el tiempo el usuario fue castigado con una pérdida de puntos.

Los obstáculos fueron: paredes rojas cuya elevación fue controlada con el botón central del joystick derecho, paredes azules cuya elevación fue controlada con el botón central del joystick izquierdo, y paredes grises cuya posición fue controlada con movimientos del joystick izquierdo. Esto se configuró para acostumbrar al usuario a operar con ambas manos, y se manejó con las funciones levelCheck( ) y arrowCheck( ).

Una vez alcanzada la meta al final del cuarto puente se desplegó la puntuación final. Toda acción no contenida por las funciones mencionadas anteriormente fue parte de una función llamada colliding( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 29).

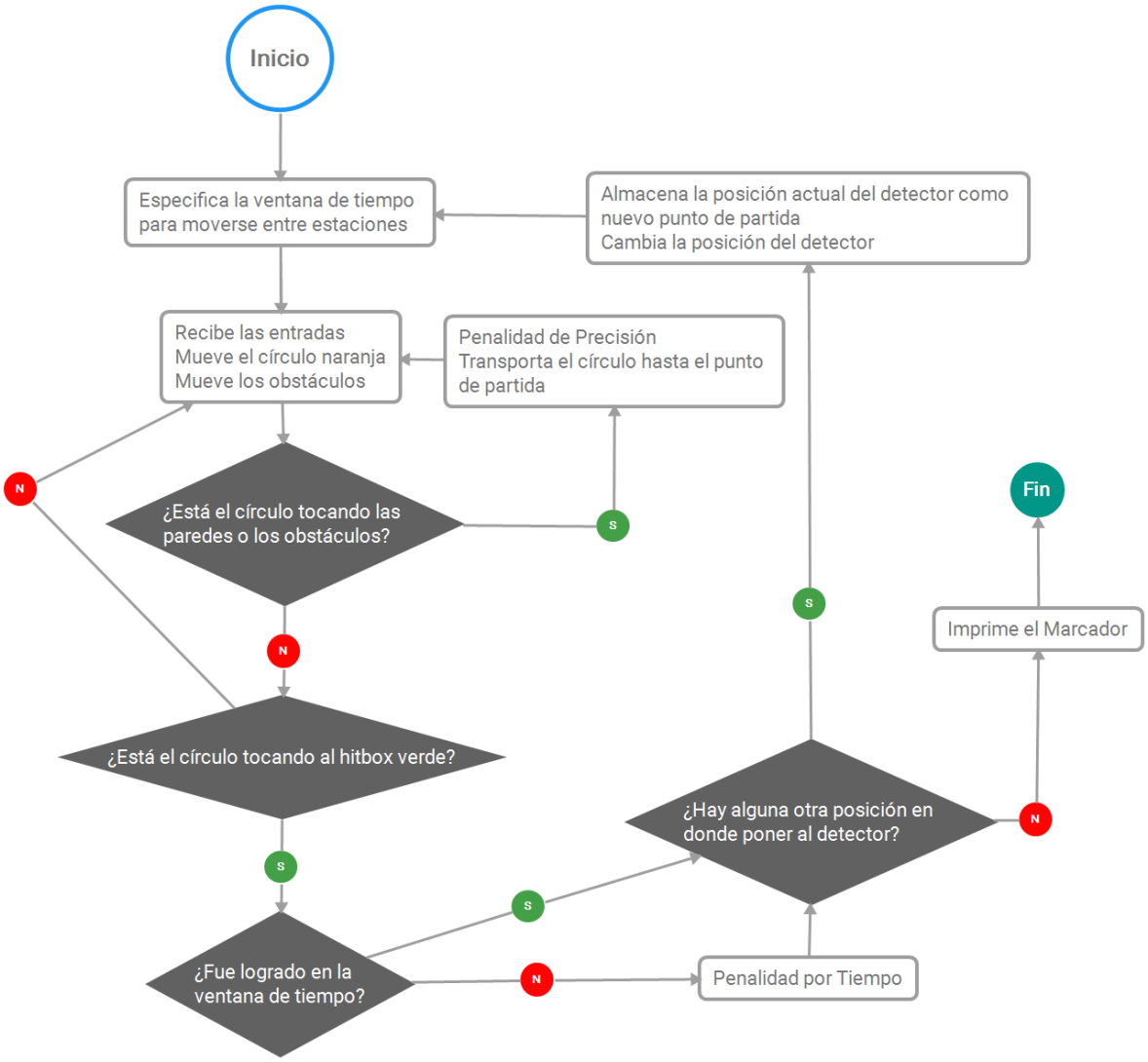


Figura 29: Diagrama de flujo conceptual del nivel JoystickPractice. Desarrollado en X Mind8 (2016).

4.4.3 Segunda Práctica: ArmPractice

La segunda práctica fue un ejercicio para el control de los movimientos de la excavadora. La cámara fue puesta estática, fuera de la cabina, en vista lateral con tal de facilitar la visualización por parte del usuario. La herramienta fue la cubeta.

Los hitbox del brazo de la excavadora fueron hechos visibles y de un color gris traslucido. Se recreó esta cadena de hitbox y sus respectivas articulaciones, con lo que se obtuvo un segundo brazo color verde traslúcido. Los hitbox de este brazo adicional fueron agrandados 10cm en cada dirección y se añadió a cada uno una copia adicional pero alterada para que fuese de solo 10cm de altura, centrada en su prisma de origen y de color verde opaco pero configurados como invisibles (Fig. 30). También se generó código para alterar los ángulos de cada una de las cuatro articulaciones del brazo secundario, basado en listas de cuatro números almacenadas en una lista general.

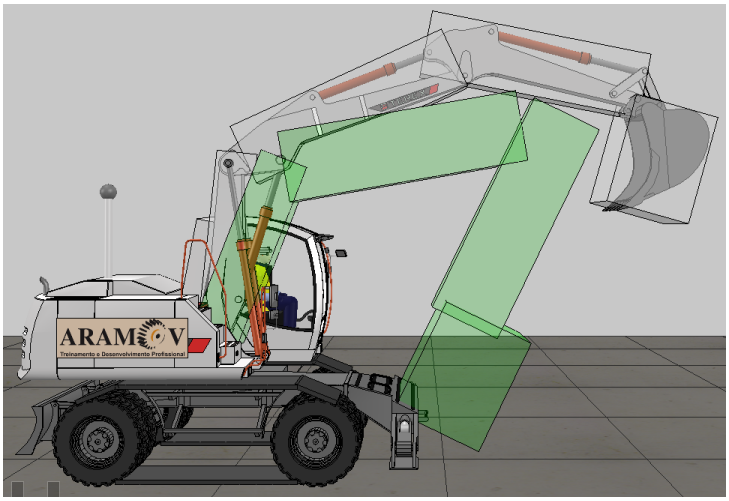


Figura 30: Sistema de brazos gris y verde. Desarrollado en V-Rep (2016).

El concepto se basó en indicar al usuario que posicionase el brazo de la excavadora, y por ende sus hitbox grises, dentro de los hitbox verdes del brazo secundario. En caso de lograrlo con un eslabón su respectivo hitbox verde opaco se volvió visible

(Fig. 31), y en caso de lograrlo con todos se habilitó la opción de presionar el botón central del joystick derecho para registrar la acción y mover el brazo secundario a su siguiente posición. La lógica se basó en preguntar durante cada ciclo de la simulación si el hitbox gris de un eslabón tocó al verde traslucido y al verde opaco del brazo secundario, si las respuestas fueron no y si respectivamente significó que el eslabón se encontró dentro de la posición deseada.

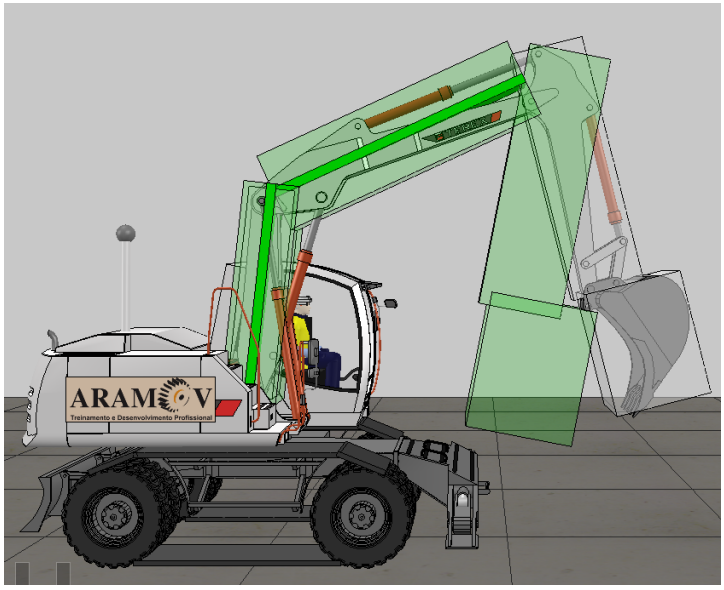


Figura 31: Prisma usado como hitbox. Desarrollado en V-Rep (2016).

Las primeras posiciones requeridas involucraron ajustes con un solo pistón y no fueron cronometrados, pero paulatinamente se agregó posiciones más complicadas y límites de tiempo con penalizaciones por incumplimiento. También se agregó un plano a ras del piso a modo de hitbox, con tal de penalizar el contacto de la cubeta con el suelo. Todo esto fue realizado con la función angleFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 32).

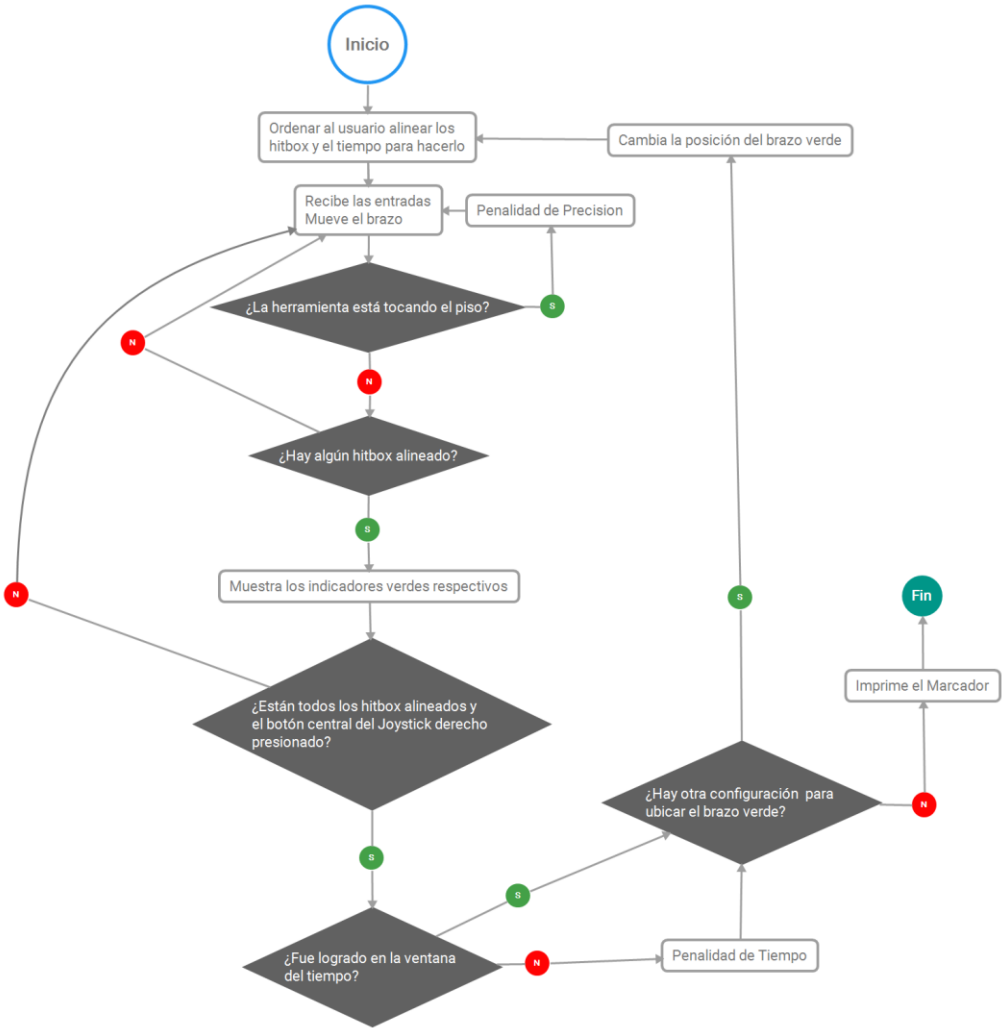


Figura 32: Diagrama de flujo conceptual del nivel ArmPractice. Desarrollado en X Mind8 (2016).

4.4.4 Tercera Práctica: ToolPractice

La práctica siguiente fue el posicionamiento del brazo para acoplar una herramienta, ya que permitió volver al concepto de controlar el brazo pero esta vez con la cámara en primera persona.

Se tomó la cubeta y las tres herramientas generadas para las prácticas y se posicionaron alrededor de la excavadora, sobre bases diseñadas para simular una apariencia de madera, en ángulos y distancias configuradas para que el final del brazo fuese capaz de alcanzar el mecanismo de acople de cada herramienta (Fig. 33). El tenedor y la polea sirvieron para introducir un ángulo plano de acople pero a alturas diferentes, mientras que la cubeta y el taladro forzaron ajustes angulares más complicados.

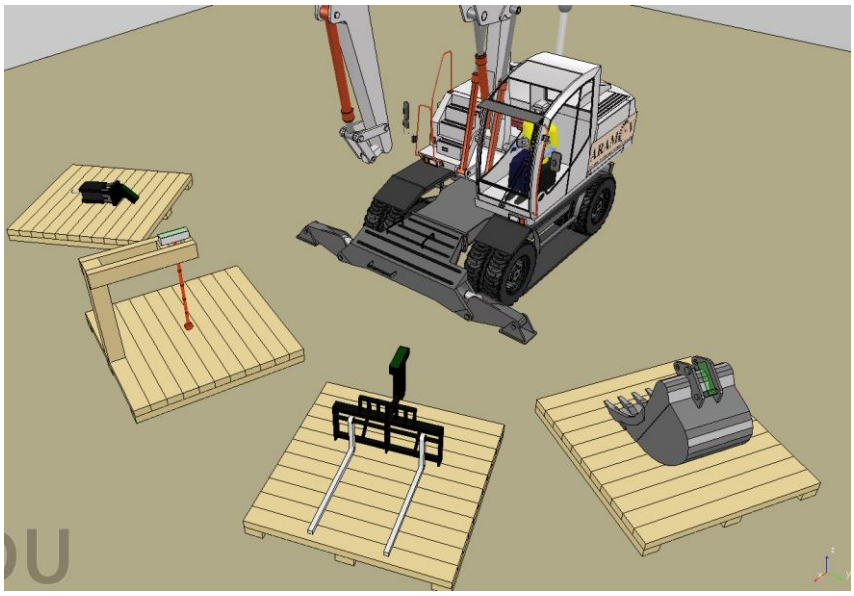


Figura 33: Vista superior del nivel ToolPractice. Desarrollado en V-Rep (2016).

La programación consistió en agregar un plano rectangular sobre el acoplador de cada herramienta como primer hitbox, y dos cilindros pequeños separados unos 10cm y salientes del final del brazo como segundo y tercer hitbox. Para determinar la

alineación del brazo con cada herramienta se revisó si el hitbox rectangular detectó a los dos cilindros al mismo tiempo. La separación entre los hitbox cilíndricos fue para asegurar tanto que el extensor final estuviese centrado como paralelo a la superficie (Fig. 34). Como recursos auxiliares se configuró una luz que se encendió en caso de alcanzar la posición de acople y un hitbox alrededor de cada herramienta para activar una vista auxiliar del acople al detectar al brazo (Fig. 35).

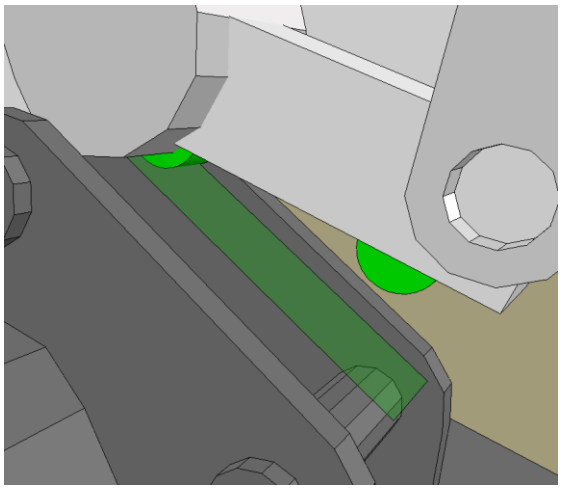


Figura 34: Sistema de alineamiento. Desarrollado en V-Rep (2016).

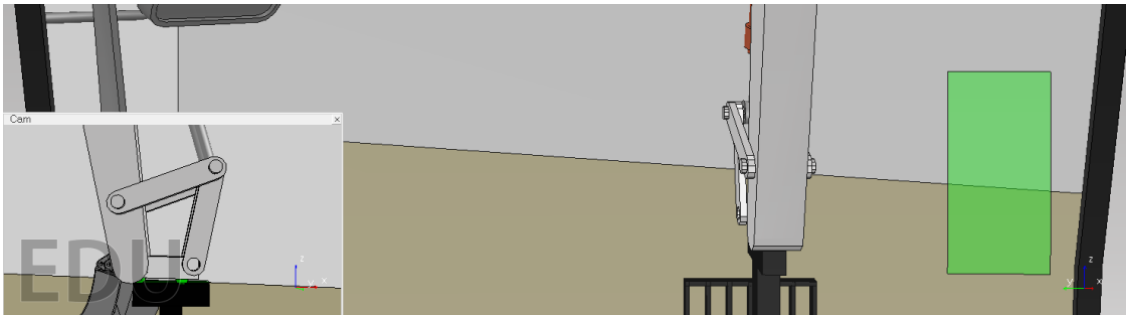


Figura 35: Sistemas auxiliares de corroboración. Desarrollado en V-Rep (2016).

La práctica se basó en especificar al usuario que herramienta tomar, que este posicionase el brazo hasta el acople de la herramienta, presionase el botón central del joystick derecho para conectar, moviese la herramienta hasta una zona marcada y presionase el botón central del joystick izquierdo para terminar la acción. Una vez

alcanzado este punto, un hitbox amarillo para detectar la presencia de la herramienta (Fig. 36), se retornó de manera automática la herramienta a su lugar de reposo. Este proceso fue organizado en una primera ronda sin tiempo para conectar con cada objeto, a modo de entrenar el control del brazo, y una segunda ronda con tiempo y un orden diferente de las herramientas, para entrenar la eficiencia y la precisión. En caso de no satisfacer el límite de tiempo se descontó puntos al usuario y se lo forzó a realizar el acople de nuevo hasta superar la prueba. Todo esto se manejó con la función toolFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 37).

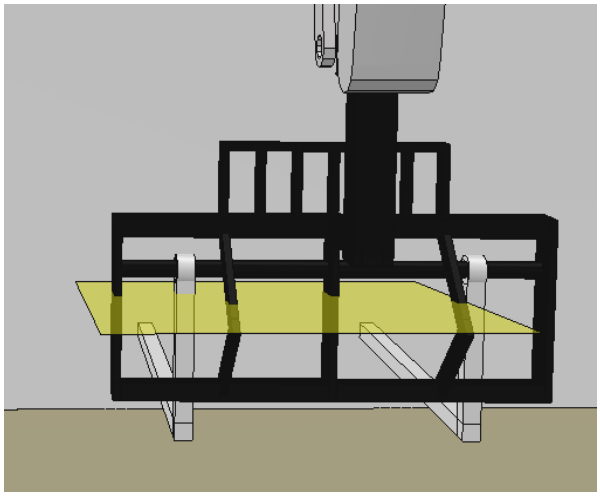


Figura 36: Detector de la herramienta. Desarrollado en V-Rep (2016).

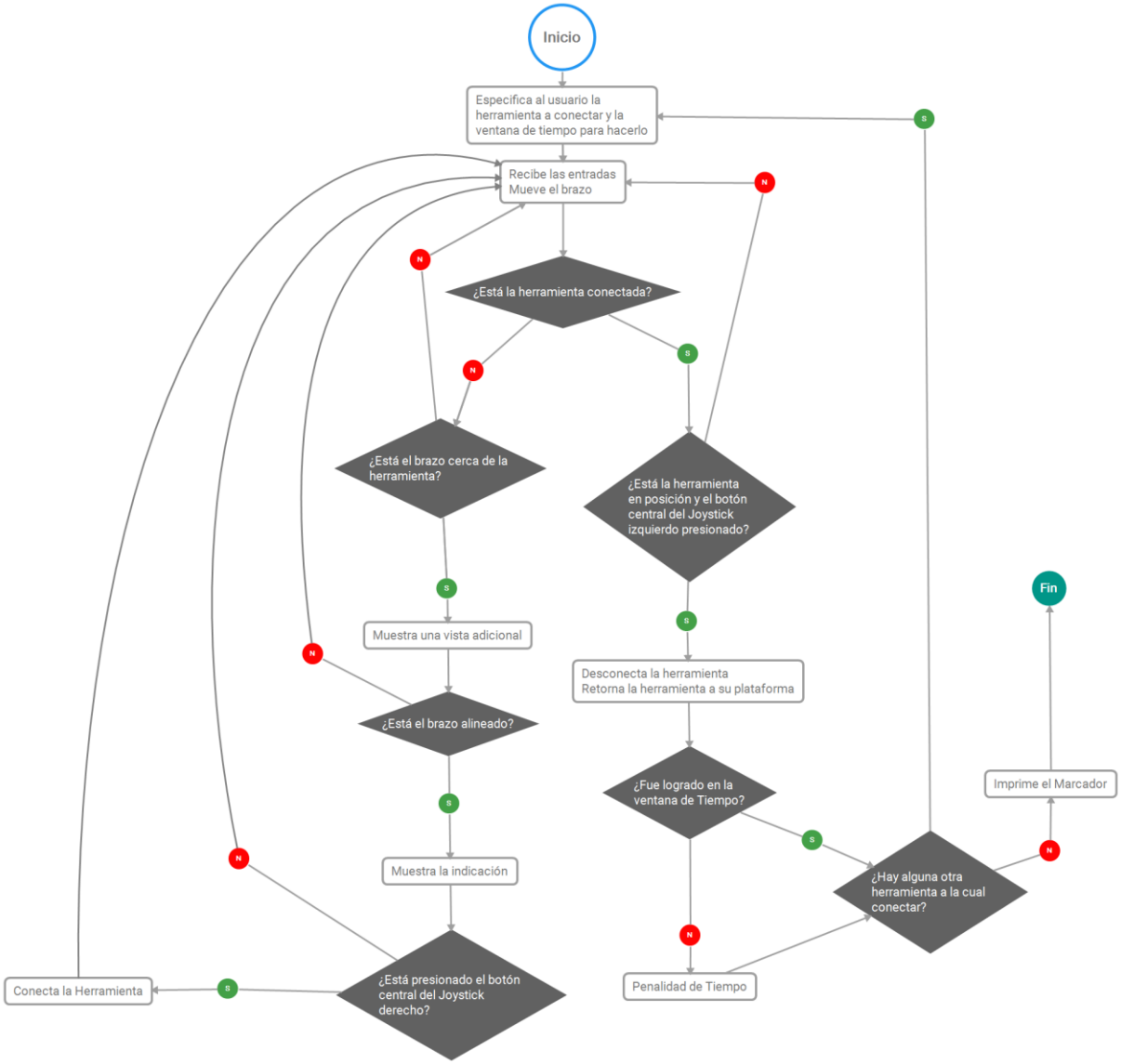


Figura 37: Diagrama de flujo conceptual del nivel ToolPractice. Desarrollado en X Mind8 (2016).

4.4.5 Cuarta Práctica: DrillPractice

La cuarta práctica fue centrada en el uso del taladro, con tal de entrenar más el uso de los botones y el reajuste constante de la herramienta.

La práctica se basó en posicionar la punta del taladro ortogonal a la superficie de una roca, mantener presionado el botón central del joystick derecho y taladrar hasta medio metro por debajo del nivel del piso, extraer el taladro hasta un área marcada y presionar el botón central del joystick izquierdo (Fig. 38). Una vez realizado esto la piedra fue reposicionada en un punto cercano para comenzar nuevamente.

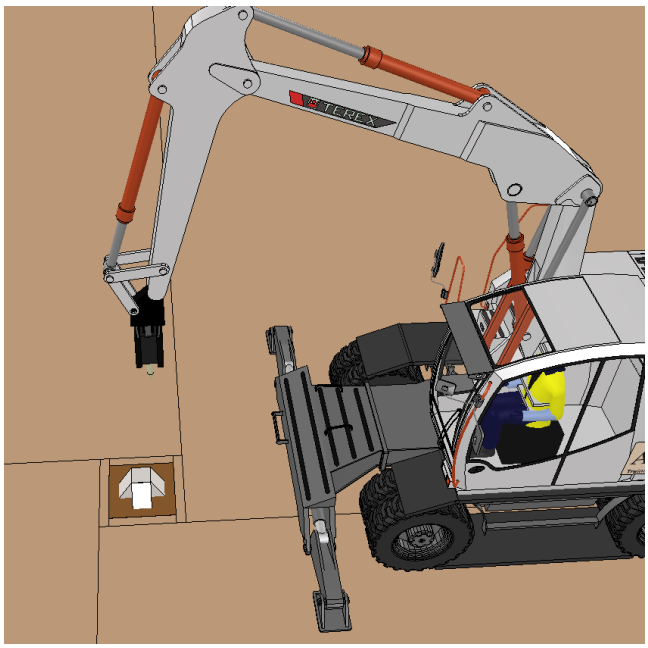


Figura 38: Vista general del nivel DrillPractice. Desarrollado en V-Rep (2016).

Para detectar la ortogonalidad del taladro se posicionaron dos planos cuadrados a modo de hitbox, paralelos a la superficie de la piedra, uno elevado por sobre el otro; y se encerró la punta del taladro con un hitbox cilíndrico de altura ligeramente mayor a la separación de susodichos planos. Solo si la punta del taladro fue puesta perpendicular a la roca su hitbox atravesó ambos hitbox (Fig. 39).

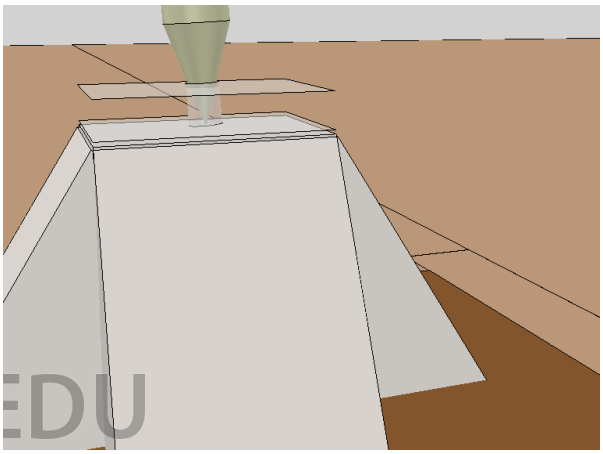


Figura 39: Corroboración de la ortogonalidad del taladro. Desarrollado en V-Rep (2016).

Se creó una penalización por usar el botón central del joystick derecho fuera del área de excavación, ya que en la práctica eso hubiera implicado desperdiciar gasolina en taladrar el aire. Para detectar la situación se agregó un hitbox a la roca, y se preguntó constantemente si la activación del botón fue simultánea a la detección del taladro con el hitbox. También se agregó un plano a ras del piso para detectar acciones fuera de posición y penalizarlas. Con tal de ayudar al operario a evitar eso, se agregó una sombra para la punta del taladro (Fig. 40).

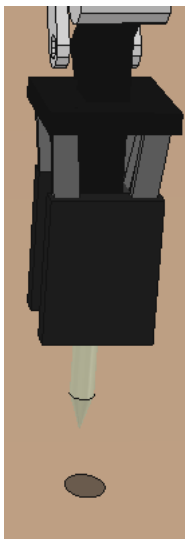


Figura 40: Sombra de la punta del taladro. Desarrollado en V-Rep (2016).

Para la mecánica de taladrar se programó que la roca cambiase su posición en el eje vertical si se detectó el taladro en posición y el botón presionado, pero en pasos pequeños para exigir la constancia del usuario (Fig. 41). El cambio de altura y el adentramiento en el agujero forzaron al usuario a realizar reajustes con los pistones para garantizar la posición de la herramienta. Después, al posicionar la herramienta fuera del agujero y presionar el botón izquierdo se elevó la roca a la altura del piso y se desplazó a otra coordenada, previamente almacenada en el nivel. Este proceso se repitió varias veces, al principio sin límites de tiempo para adicionarlos paulatinamente. Todo esto fue organizado con la función drillFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 42).

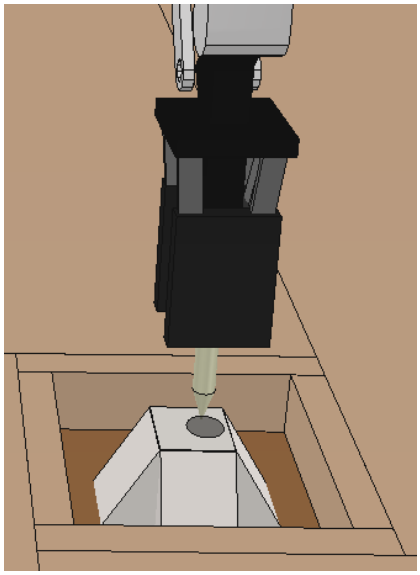


Figura 41: Hundimiento paulatino de la roca. Desarrollado en V-Rep (2016).

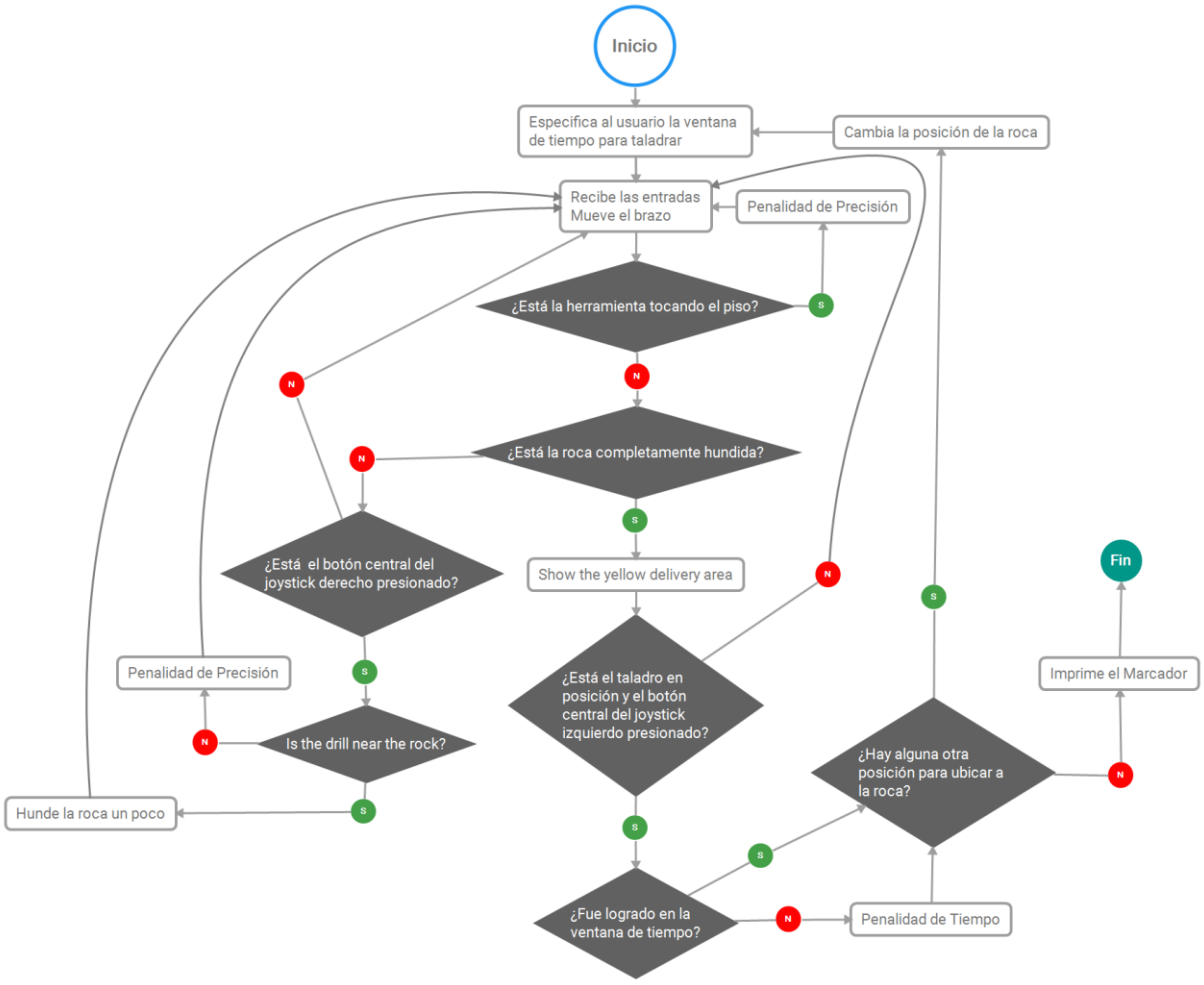


Figura 42: Diagrama de flujo conceptual del nivel DrillPractice. Desarrollado en X Mind8 (2016).

4.4.6 Quinta Práctica: MazePractice

La quinta práctica fue el uso de la polea para transportar una carga a través de un laberinto (Fig. 43), previamente comentada en el marco teórico.

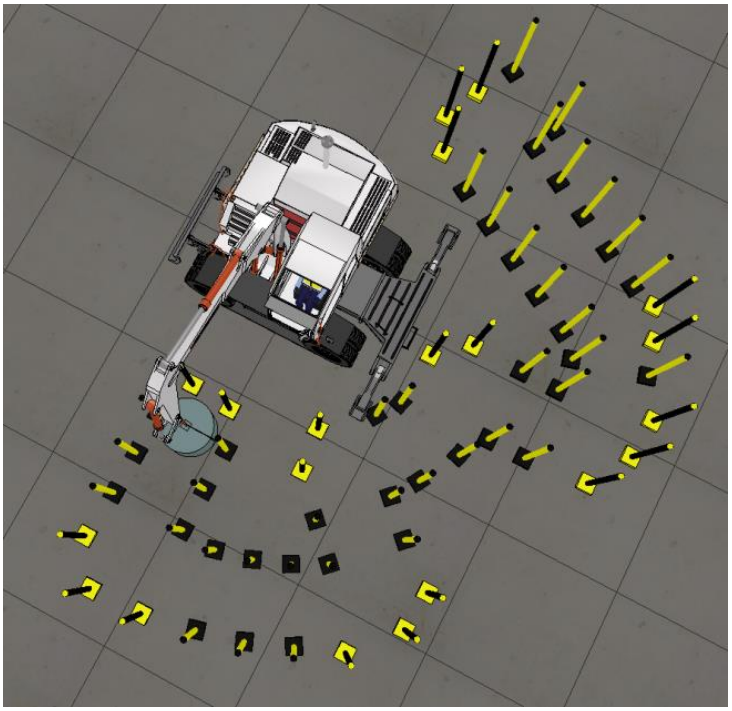


Figura 43: Vista superior del laberinto. Desarrollado en V-Rep (2016).

Como masa se agregó un cilindro color gris al final de la polea, de masa suficiente para oscilar de ser movida bruscamente, con una sombra con cambios de tamaño para facilitar la ubicación 3D del sistema y con un hitbox cilíndrico superficial para detectar posibles choques. También se configuró postes de color amarillo y negro, colores estándar de seguridad, para demarcar el laberinto. Finalmente, se programó una línea de meta color verde traslúcido que fue básicamente un plano rectangular con función de hitbox.

El ejercicio consistió en trasportar la masa de forma tal que traspasase la línea de meta (Fig. 44), y al detectar el choque trasportar esa línea a su siguiente

coordenada. Esto brindó al usuario una pauta simple de seguir y a la programación una forma de verificar el final de la sesión: detectar una colisión con la línea de meta en su última coordenada. Se programó un hitbox a ras del suelo con tal de penalizar el arrastre de la carga, y uno a dos metros de altura para penalizar el elevar la carga demasiado. También se configuró una penalización al existir contacto entre los postes del laberinto y el hitbox de la masa.

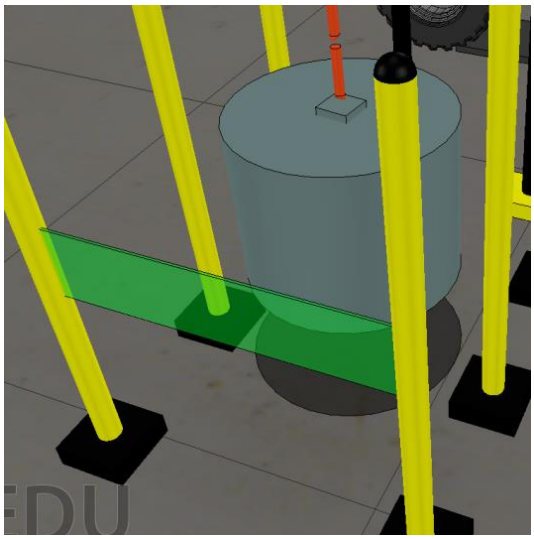


Figura 44: Transporte de la masa a la línea de meta. Desarrollado en V-Rep (2016).

El laberinto se diseñó con secciones lineales de acercamiento y alejamiento de la carga, lo que forzó a mantener en consideración como la retracción del brazo afectó a la altura de la masa, secciones circulares para el desplazamiento transversal, lo que fomentó el uso del joystick izquierdo con sensibilidad suficiente para no generar oscilaciones, y una sección diagonal que combinó ambas exigencias. Las primeras secciones se programaron para no tener límite de tiempo, a diferencia de las últimas. Todo lo comentado se desarrolló en la función loadFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 45).

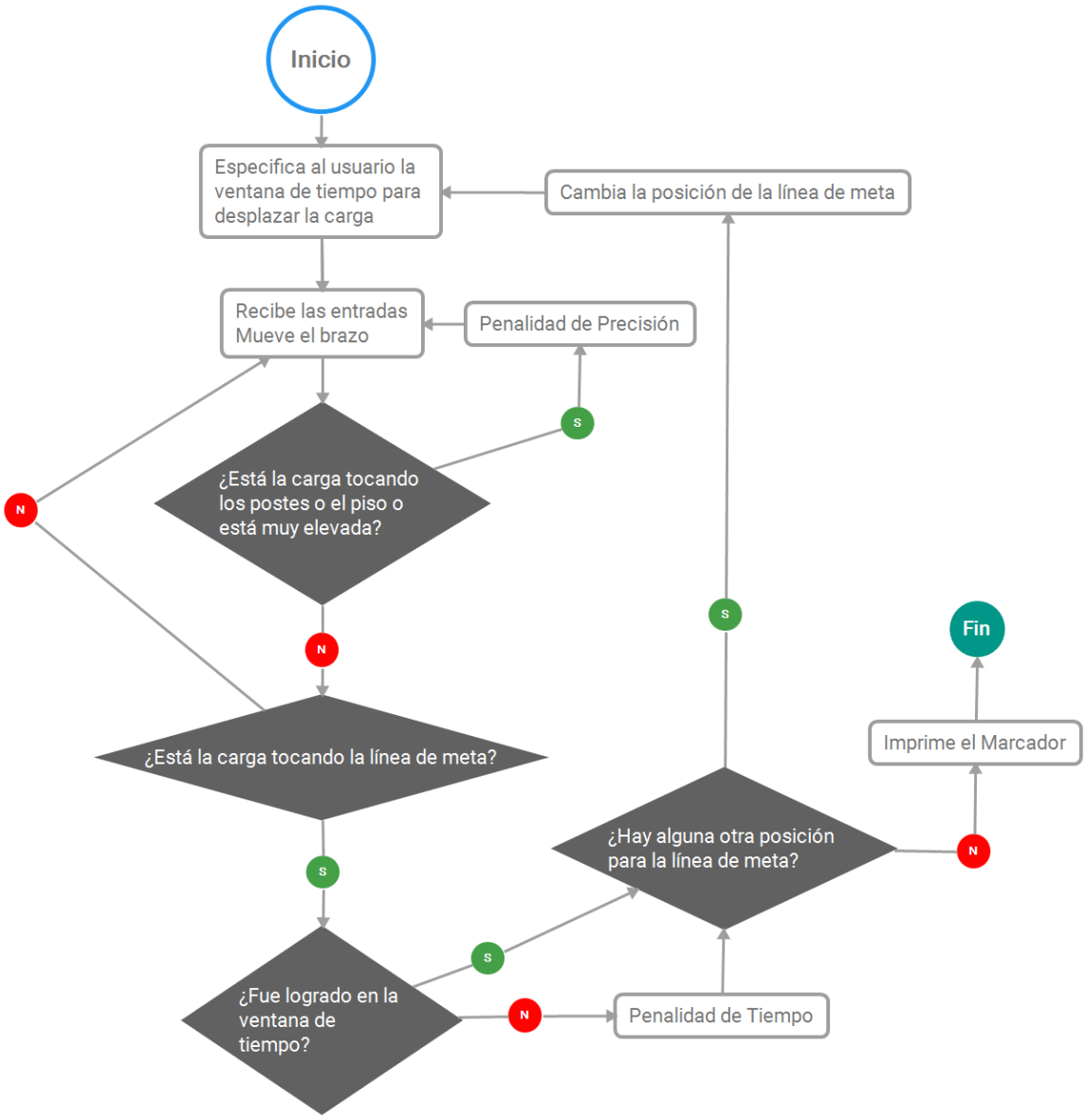


Figura 45: Diagrama de flujo conceptual del nivel MazePractice. Desarrollado en X Mind8 (2016).

4.4.7 Sexta Práctica: BlindagePullPractice

El próximo ejercicio consistió en conectar la polea a un blindaje para trasportarlo a otra posición.

Se creó el blindaje combinando prismas de diferentes tamaños y colores para imitar la apariencia de un modelo real (Fig. 46), y se agregó en su sección superior una conexión para la polea consistente de dos prismas rectangulares separados por 10cm. La mecánica de conexión consistió en ubicar el final de la polea de forma tal que tocase ambos hitbox del conector y presionar el botón central del joystick derecho para conectar. Para ayudar al operario se configuró una vista auxiliar en acercamiento a la sección de acople, activada con la detección de la polea por parte de un hitbox dentro de la zona del blindaje.

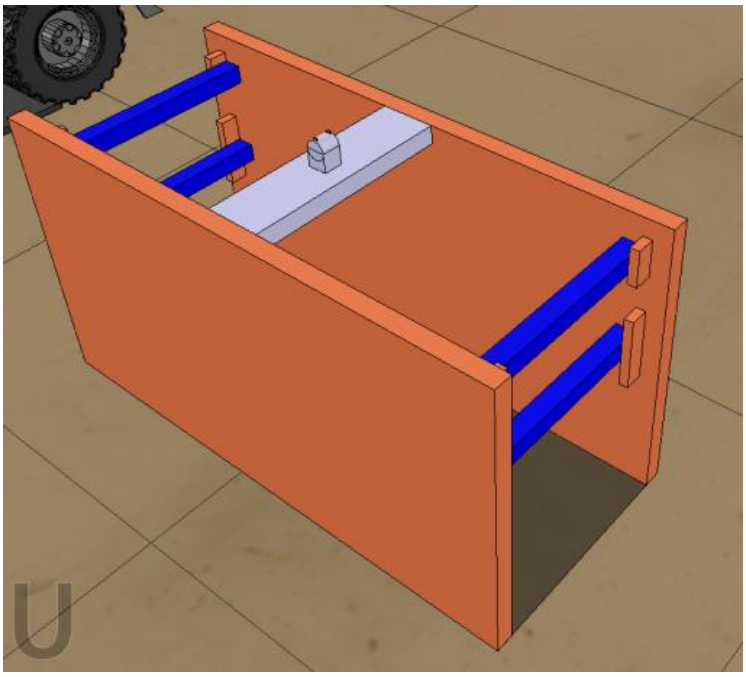


Figura 46: Blindaje con conector. Desarrollado en V-Rep (2016).

La parte del trasporte se basó en reposicionar el blindaje en una zona marcada, de forma tal que contactase dos hitbox de color verde traslucido a ras del piso. La posición de los hitbox fue ajustada para forzar al usuario a posicionar al blindaje completamente sobre el piso, en vez de permitir solo el contacto de un costado o vértice de la estructura (Fig. 47). Se agregó un hitbox a las secciones del piso que el blindaje no debió tocar con tal de penalizar tal error, pero se añadió una sombra rectangular con tal de facilitar la ubicación del blindaje.

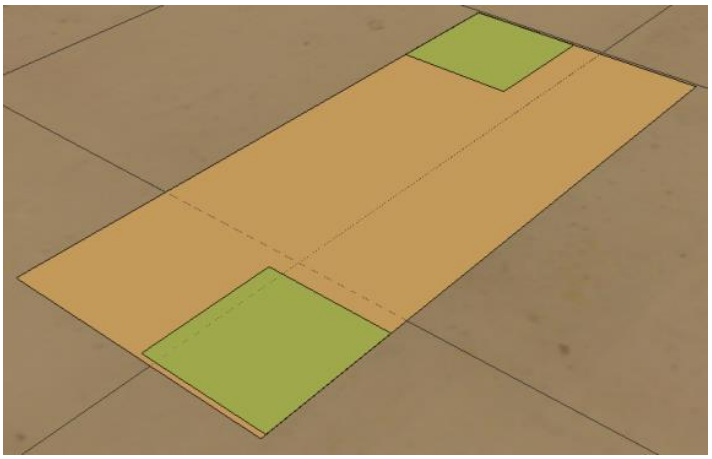


Figura 47: Zona de descarga del blindaje. Desarrollado en V-Rep (2016).

Finalizado el trasporte se reubicó el blindaje a una nueva posición, para reiniciar la acción varias veces. Los primeros ciclos de este trabajo no evaluaron el tiempo de ejecución, pero los últimos si lo hicieron. Todo esto fue controlado por la función pullFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 48).

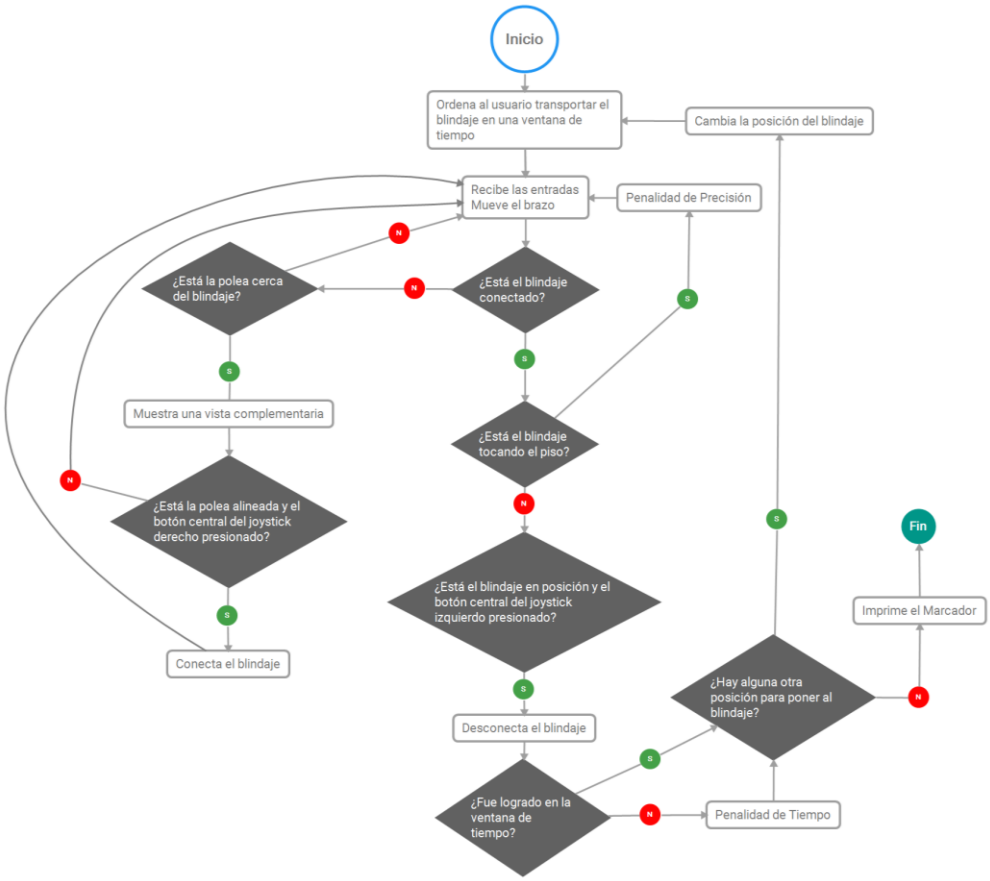


Figura 48: Diagrama de flujo conceptual del nivel BlindagePullPractice. Desarrollado en X Mind8 (2016).

4.4.8 Séptima Práctica: BlindagePushPractice

El penúltimo ejercicio fue una continuación de la manipulación del blindaje, esta vez la labor de hundirlo con golpes propinados con la cubeta.

Se colocó al blindaje sobre cuatro planos rectangulares, uno en cada esquina del borde que tocó al piso, y estos cubrieron un agujero de forma rectangular por debajo del blindaje. En el costado superior del blindaje se colocaron cuatro hitbox rectangulares de color verde traslúcido, uno a cada esquina y todos invisibles excepto por uno (Fig. 49). Se colocó más detectores en secciones donde la cubeta nunca tuvo que realizar contacto, y uno en el piso.

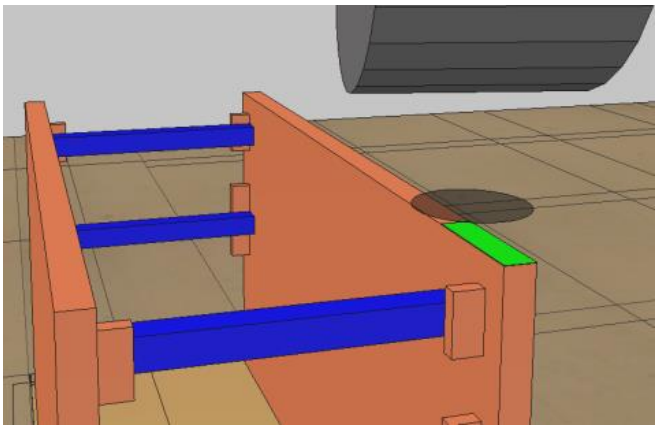


Figura 49: Hitbox visible sobre el blindaje. Desarrollado en V-Rep (2016).

La mecánica consistió en: impactar el fondo de la cubeta contra el hitbox visible en la esquina superior del blindaje, al detectar tal impacto bajar 5cm la plataforma de apoyo respectiva a esa esquina (Fig. 50), cambiar a invisible el hitbox impactado y cambiar a visible el hitbox de la siguiente esquina. Debido al cambio consecutivo de la posición del apoyo de cada esquina, se simuló el hundimiento e inclinación paulatinos de la caja propios de esta práctica. Los demás hitbox fueron para detectar y penalizar cualquier mal praxis al intentar de hundir al blindaje, incluyendo la posibilidad de hacerlo caer al piso.

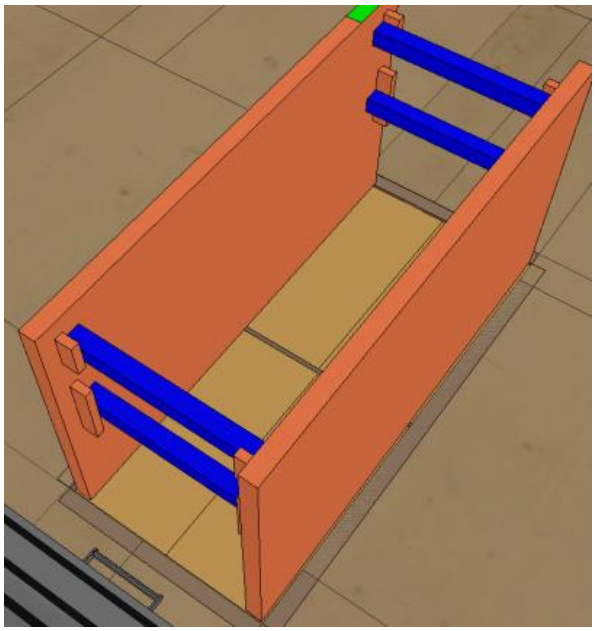


Figura 50: Desnivel causado por el hundimiento de las plataformas. Desarrollado en V-Rep (2016).

Los primeros impactos fueron sin tiempo para entrenar los movimientos, pero en ciclos subsecuentes se agregó un límite y su penalización. Todo esto se organizó con la función pushFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 51).

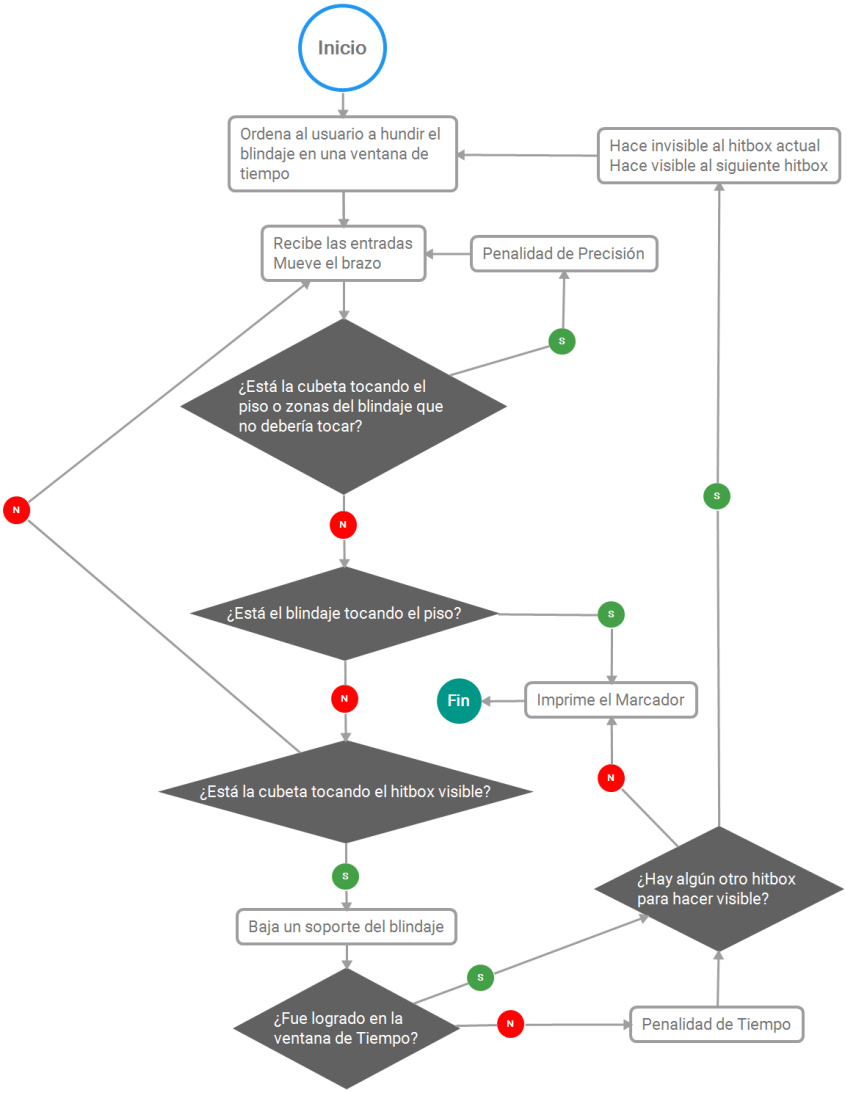


Figura 51: Diagrama de flujo conceptual del nivel BlindagePushPractice. Desarrollado en X Mind8 (2016).

4.4.9 Octava Práctica: TaludPractice

El ejercicio final fue un entrenamiento para los movimientos del corte de talud en tres variantes.

Como terreno excarvable se generó un prisma color café traslucido, cubierto por un plano con textura de tierra y delimitado a su largo por dos hitbox café que representaron el inicio y final de la excavación (Fig. 52). Se agregó a la cubeta tres hitbox prismáticos: uno en cada esquina del filo de la cubeta y uno en el centro del fondo de la cubeta (Fig. 53).

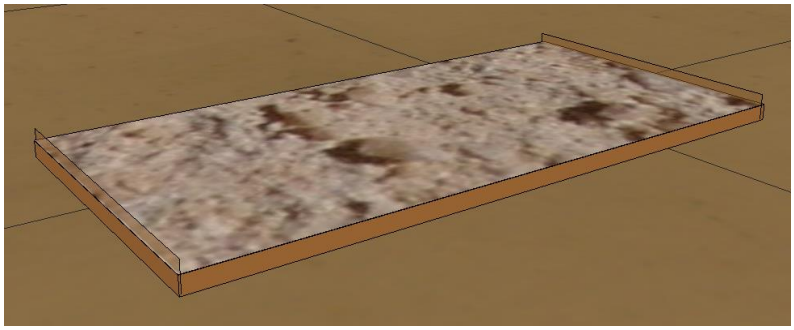


Figura 52: Terreno excarvable. Desarrollado en V-Rep (2016).

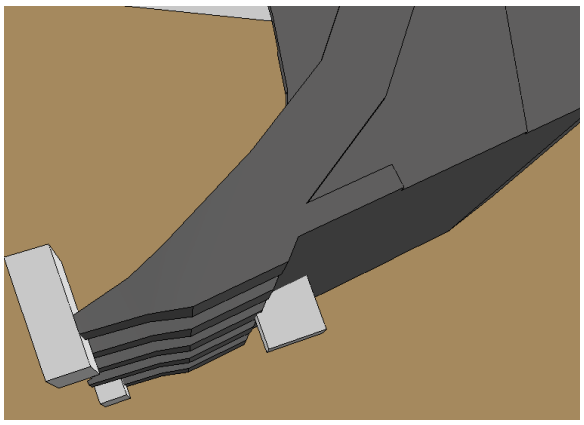


Figura 53: Hitbox en la cubeta. Desarrollado en V-Rep (2016).

El primer corte fue el raspado, el que se detectó bajo la siguiente secuencia de contactos: los hitbox del filo contra el hitbox del inicio del terreno, los hitbox del filo contra el prisma del terreno y los hitbox del filo contra el hitbox del final del terreno. El segundo corte fue el talud horizontal, y se detectó como: los hitbox del filo contra el hitbox del inicio del terreno, los hitbox del filo contra el prisma del terreno, los tres hitbox de la cubeta contra el prisma, y los hitbox del filo contra el hitbox del final del terreno. El tercer corte fue el mismo que el segundo, pero se trasladó el terreno a una superficie inclinada. La diferencia radicó en que el raspado consistió en rastrillar el terreno con la punta de la cubeta (Fig. 54) mientras que el talud fue aplanar el terreno al arrastrar la cubeta paralela a este (Fig. 55); por ende en el talud se exigió el contacto del fondo de la pala con el suelo, mientras que en el raspado se prohibió eso.

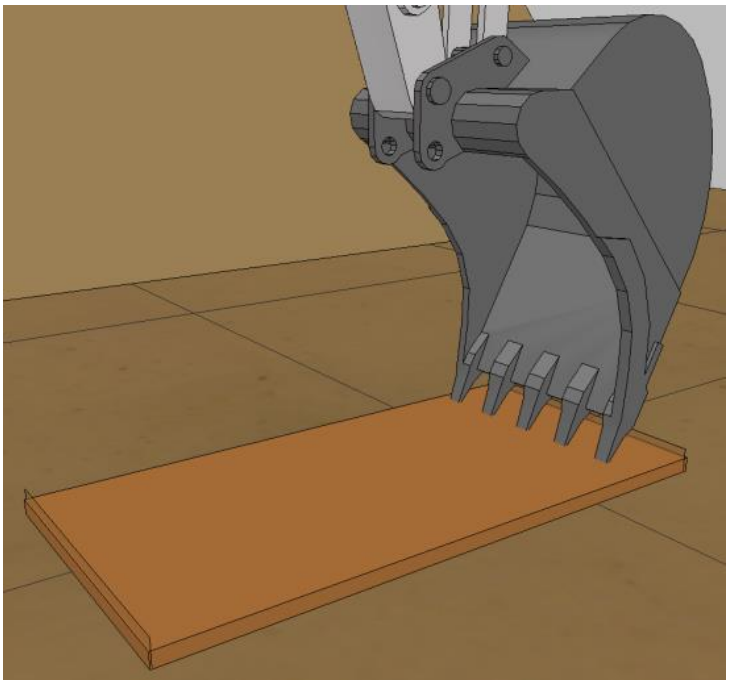


Figura 54: Raspado. Desarrollado en V-Rep (2016).

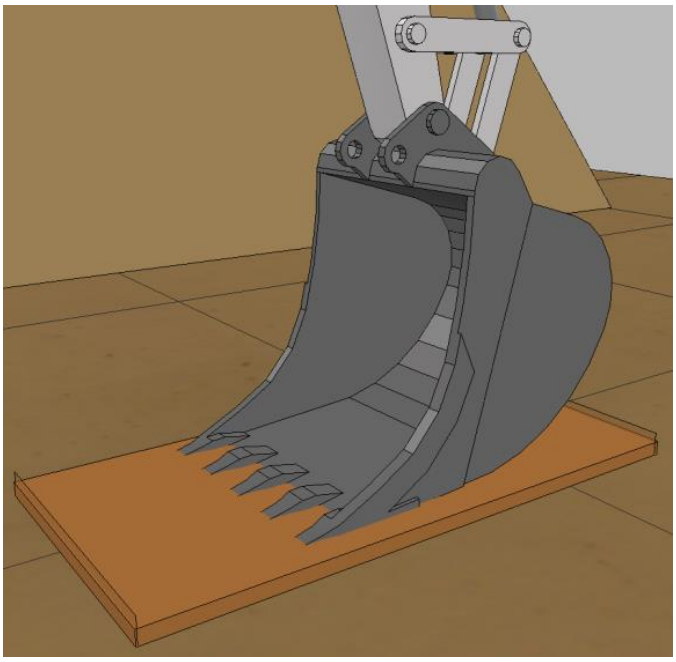


Figura 55: Corte de Talud. Desarrollado en V-Rep (2016).

En caso de iniciar la secuencia se hizo invisible la textura a modo de confirmación, y al terminar el corte se hizo visible de nuevo. La secuencia de cada corte fue terminada automáticamente en caso de detectar un paso no especificado o fuera de orden, como por ejemplo detectar el fondo de la cubeta durante un raspado.

La práctica consistió en: dos raspados sin límite de tiempo y dos cronometrados, dos cortes de talud horizontales sin límite de tiempo y dos cronometrados, y dos cortes de talud inclinados sin límites de tiempo y dos cronometrados. Todo este sistema fue regulado con la función cutFunction( ). El diagrama de flujo conceptual del nivel se encuentra a continuación (Fig. 56).

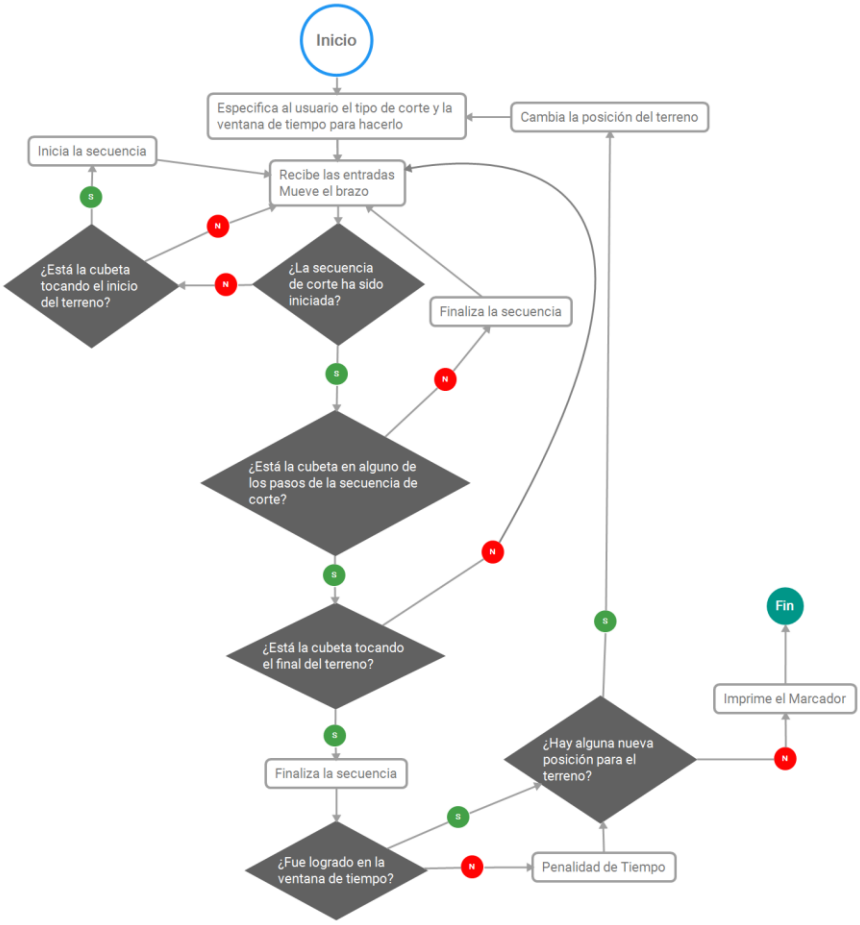


Figura 56: Diagrama de flujo conceptual del nivel TaludPractice. Desarrollado en X Mind8 (2016).

4.5 Validaciones y Aprobaciones

La primera validación fue para el sistema de hitbox, usando un nuevo nivel con el excavador en su centro y tres planos paralelos al frente de este. El plano central fue un hitbox verde opaco, mientras que los otros dos fueron planos traspasables color rojo translúcido pero configurados para ser invisibles, uno a cada lado del hitbox y separado 1cm. La idea fue acercar la punta del taladro hasta atravesar el hitbox, y que la detección de esto hiciese visibles a los planos rojos. Después, corroborar con la cámara la posición de la punta del taladro: si este se encontró entre los dos hitbox rojos significó que el contacto fue detectado en el margen de ±1cm. Los resultados fueron exitosos, sugiriendo inclusive un margen en la escala de los milímetros (Fig. 57). El margen de ±1cm fue escogido por el estudiante y el profesor asesor al ser considerado suficientemente pequeño para permitir detectar colisiones entre objetos.

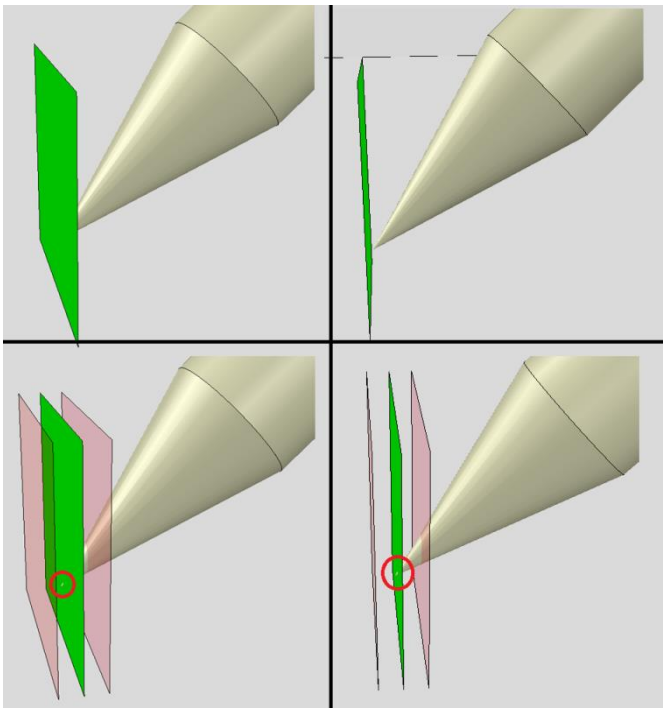


Figura 57: Corroboración del sistema de hitbox. Desarrollado en V-Rep (2016).

Con el hitbox como recurso validado, se generó la validación del control del brazo. Se diseñó otro nivel basado en el sistema hitbox verde-placa roja, pero esta vez con los hitbox de 1cm x 1cm orientados paralelos al piso y con solo una placa roja, distanciada a 5mm por encima y usada solo para indicar el contacto. El sistema fueron varios hitbox distanciados por 1cm en los ejes cartesianos y otros separados a 1° desde el centro de rotación de la cabina (Fig. 58 y 59). La prueba consistió en intentar de realizar contacto con cada hitbox de manera individual, lo que verificó la capacidad de controlar el brazo con márgenes de ±1cm y ±1°. Los resultados fueron exitosos (Fig. 60). Los márgenes fueron escogidos arbitrariamente por el estudiante y el asesor a modo de corroborar la sensibilidad de los movimientos y de verificar el mapeo de las lecturas en vez de verificar alguna distancia dada, esto porque los reajustes entre movimientos fueron realizados con las medidas de la excavadora virtual oficial de TEREX y por ende se asumieron como correctos.

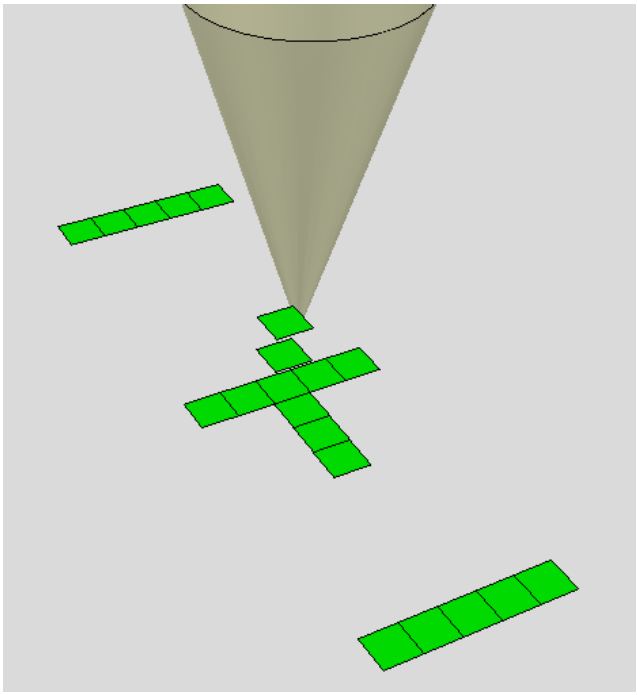


Figura 58: Sistema de corroboración de los controles. Desarrollado en V-Rep (2016).

Figura 59: Escala y posición del sistema con respecto a la excavadora. Desarrollado en V-Rep (2016).

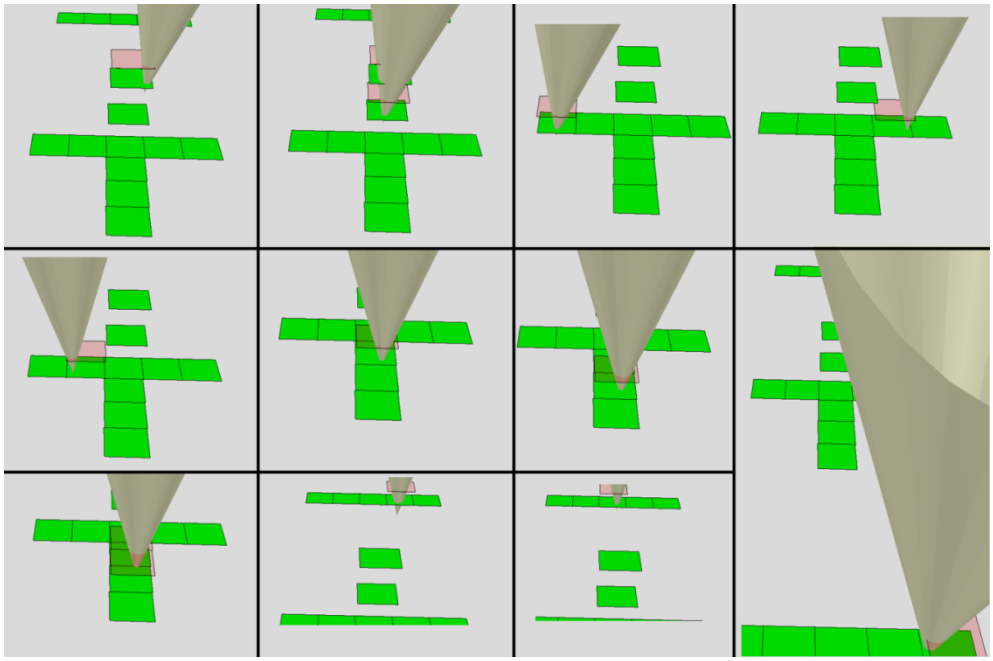


Figura 60: Acceso a cada placa usando los controles. Desarrollado en V-Rep (2016).

Finalmente, los ambientes, tareas y puntuaciones fueron aprobados por el profesor.

5. Conclusiones y Recomendaciones

5.1 Conclusiones

Se cumplieron los objetivos asignados:

- Se implementó el sistema de control de posicionamiento de la herramienta basado en palancas de mando.
    - Se usó la función simExtJoyGetData( ) para extraer información de los joysticks.
    - Se modificó el pluggin llamado “v_repExtJoystick.dll” con tal de poder leer los dos joysticks a la vez.
    - Se usó la función simGetSimulatorMessage( ) para extraer información del teclado.
    - Se creó la función pairMovement( ) con tal de escalar la “rapidez” de los pistones por medio de las lecturas de los joysticks.
    - Se creó la función pairCheck( ) para permitir el control del brazo mediante el modo SAE y el modo ISO.
    - Se creó la función mapping( ) para permitir el control del brazo mediante los joysticks y mediante el teclado.
    - Se validó el control mediante un nivel personalizado.

- Se desarrolló un algoritmo para detectar una distancia de separación específica entre los objetos simulados, basado en la técnica de hitbox.
    - Se encerró a los cuerpos y regiones de interés con geometrías simples no sólidas, también llamados hitbox.
    - Se usó simCheckCollision( ) sobre los hitbox para corroborar la distancia mínima entre el ente que encerraron y cualquier otro cuerpo de interés.
    - Se usó la técnica del hitbox para encerrar a la excavadora y a los obstáculos y detectores usados en los niveles.

- Se validó los hitbox mediante un nivel personalizado.

- Se desarrolló un algoritmo que asignó puntajes y realizó penalizaciones con base en las métricas de Precisión y Tiempo.
  - Se diseñó un algoritmo de puntaje basado en una puntuación inicial máxima y deducciones por accidentes y demoras durante la práctica.
  - Se programó la deducción por accidentes, o Precisión, para activarse al detectar colisiones o violaciones de distancias mínimas.
  - Se programó la deducción por demoras, o Tiempo, para activarse por culminar una acción fuera de una ventana de tiempo asignada.
  - Se creó la función printingTime( ) para desplegar mensajes en pantalla, en especial indicaciones de la pérdida de puntos.
  - El profesor tutor en Brasil aprobó el sistema.

- Se diseñaron 8 niveles de entrenamiento y 2 mecanismos de validación.
  - Se programó sombras para los objetos.
  - Se implementó una vista desde la cabina de la excavadora.
  - Se programó un sistema para brindar ángulos de visión auxiliares durante una tarea.
  - Se programó una polea, un taladro y un tenedor.
  - Se programó varios cambios estéticos en la excavadora.
  - Se diseñó un nivel para entrenar el uso de los joysticks.
  - Se diseñó un nivel para entrenar los movimientos generales del brazo.
  - Se diseñó un nivel para entrenar la alineación del brazo con una herramienta.
  - Se diseñó un nivel para entrenar reajustes angulares del brazo.
  - Se diseñó un nivel para entrenar el transporte de cargas.
  - Se diseñó un nivel para entrenar la conexión de la polea a los Blindajes para Trincheras.

- Se diseñó un nivel para entrenar el hundimiento de los Blindajes para Trincheras.
- Se diseñó un nivel para entrenar los cortes de tierra.
- Se diseñó un nivel para corroborar la validez de los hitbox.
- Se diseñó un nivel para corroborar la validez del control del brazo.
- El profesor tutor en Brasil aprobó los niveles.

5.2 Recomendaciones

El simulador posee el potencial de ser extendido con más ejercicios y módulos; por ejemplo: el control de las orugas, más ejercicios con el tenedor, el desarrollo de herramientas como una garra mecánica, una mejor interfaz de comunicación, submenús para desplazarse entre niveles, o inclusive experimentos con sistemas de cámaras auxiliares (tecnología en la que el LRM se encuentra interesado actualmente). No obstante, la plataforma de desarrollo puede ser inestable cuando se usan varias articulaciones (como en el brazo), no soporta escenarios muy complejos y carece de las herramientas para simular el excavar la tierra (lo que es imprescindible para formar al usuario). Por ende, si se busca un entrenamiento preliminar en los controles básicos esta línea de desarrollo es suficiente, pero si se busca entrenar otras pericias o generar una solución comercial debería optarse por otras plataformas.

6. Lista de Referencias

Alves, S. (4 de Octubre, 2016). Comunicación personal. Oficinas de ARAMOV, Araraquara.

Carter, M; Bentley, S. P. (2016). Soil Properties and their correlations, pp. 1-33. Inglaterra: WILEY.

CAT[1]. (2016). New Forks 1220mm (48IN). Rescatado el 10 de Agosto del 2016 de: http://www.cat.com/en_US/products/new/attachments/forks/block-forks/18379582.html

CAT[2]. (2016). New Hammers H55E. Rescatado el 10 de Agosto del 2016 de: http://www.cat.com/en_US/products/new/attachments/hammers/hammers/18495741.html

CAT Simulators. (7 de Febrero, 2012). Introduction to Cat® Simulators Hydraulic Excavator Training Exercise: Setting Trench Box & Pipe. Rescatado el 1 de Agosto del 2016 de: https://www.youtube.com/watch?v=SeMrUONlayU

Chadli, M. et al. (2013). Command-control for Real-time Systems, p. 273. Estados Unidos: WILEY.

CoppeliaRobotics. (13 de Septiembre, 2013). Shadows. Rescatado el 16 de Septiembre del 2016 de: http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=859

CoppeliaRobotics[1]. (29 de Octubre, 2014). V-Rep getting slower. Rescatado el 4 de Septiembre del 2016 de: http://www.forum.coppeliarobotics.com/viewtopic.php?f=5&t=2725

CoppeliaRobotics[2]. (18 de Diciembre, 2014). How to make a Flexible Cable? Rescatado el 13 de Agosto del 2016 de: http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=2665

CoppeliaRobotics. (8 de Mayo, 2015). Soft-Body Deformations. Rescatado el 13 de Agosto del 2016 de: http://www.forum.coppeliarobotics.com/viewtopic.php?f=7&t=3517

CoppeliaRobotics[1]. (2016). Collision detection. Rescatado el 6 de Agosto del 2016 de: http://www.coppeliarobotics.com/helpFiles/en/collisionDetection.htm

CoppeliaRobotics[2]. (2016). Distance calculation. Rescatado el 6 de Agosto del 2016 de: http://www.coppeliarobotics.com/helpFiles/en/distanceCalculation.htm

CoppeliaRobotics[3]. (2016). Virtual Robot Experimentation USER MANUAL. Rescatado el 6 de Agosto del 2016 de: http://www.coppeliarobotics.com/helpFiles/

Edwards, D.J. et al. (2003). Management of off-highway plant and equipment, p. 27. CRC Press.

Ierusalimschy, R. (2016). Programming in Lua (First Edition). Rescatado el 5 de Septiembre del 2016 de: https://www.lua.org/pil/contents.html

jlouis2k4. (29 de Septiembre, 2013). Multiple Joystick Control. Rescatado el 4 de Agosto del 2016 de: http://www.forum.coppeliarobotics.com/viewtopic.php?f=9&t=1805

Lennie, C. (2014). Excavators, pp. 8-13. Estados Unidos: ABDO.

MrLumonca. (8 de Diciembre, 2011). Corte de Talud con Excavadora. Rescatado el 4 de Octubre del 2016 de: https://www.youtube.com/watch?v=x9uU29MydEs

ricoinode. (6 de Agosto, 2009). The third generation of allaround steel trench lining systems. Rescatado el 20 de Agosto del 2016 de: https://www.youtube.com/watch?v=SD_sdg4X1bg

Singh, A. et al. (1999). Implementation of Safety and Health on Construction Sites, pp. 500-502. CRC Press.

Vantagetes. (25 de Enero, 2009). How the controls in an excavator work [CC English/Spanish]. Rescatado el 1 de Agosto del 2016 de: https://www.youtube.com/watch?v=VKWaFoglX70

vortexsim. (29 de Julio, 2015). Vortex Excavator Training Module. Rescatado el 1 de Agosto del 2016 de: https://www.youtube.com/watch?v=fzJf24VGrng