Investigación sobre Brazo Robótico creado en Unity

Florencia Peri, Luka Laplagne, Antonia Flores

***Abstract*—El problema a resolver en este proyecto fue la investigación y experimentación en base a un brazo robótico creado con Unity. El propósito principal fue familiarizarse con dicho programa, dominando sus funciones e investigando hasta su máximo potencial. Luego de esto, la intención fue agrupar estos conocimientos y seguir indagando específicamente en cómo se crea un brazo robótico en Unity. Cabe destacar que el grupo previamente contaba con experiencia en una plataforma bastante similar. Pese a no ser el mismo lenguaje a utilizar, claramente fue una ventaja contar con este conocimiento base. Para cumplir con el objetivo principal, la metodología que se implementó fue el estudio y análisis de diversos tutoriales para luego hacer pruebas de lo aprendido en Unity, de forma de complementar lo aprendido.**

***Keywords*—Unity, Brazo Robótico, MuJoCo, C#.**

I. INTRODUCCIÓN

Las plataformas de creación de contenido de simulación robótica son una herramienta clave para el conocimiento y comprensión de materia gráfica y de diseño digital para los estudiantes de ingeniería civil informática.

La creación de robots en la vida real es mucho más amena habiendo realizado una simulación digital en plataformas como estas. La posibilidad que entregan de no solo visualizar el diseño gráfico, sino también la simulación del movimiento y funcionalidad del objeto a crear, es una ventaja que no solo abarata costos, sino también optimiza el tiempo ahorrando fallas de prueba y error. El trabajo de creación de la simulación de un brazo robótico propone exactamente esto, el poder armar físicamente la máquina a partir de la simulación creada.

Plataformas como Webots o Unity permiten este trabajo. Esta última corresponde a un motor de videojuego multiplataforma que sirve para exactamente esto. Cuenta con las opciones de crear el objeto mismo, agregarle física para simular movimiento y probar el funcionamiento del objeto creado. Pero no es tan simple como suena, el objeto puede tener diferentes texturas, colores, y verse tan producido como un videojuego, finalmente ese es el objetivo de la plataforma. Además de esto, a los elementos debe agregárseles aspectos físicos como masa para hacer una simulación lo más cercana a la realidad. Ahora bien, el aspecto más profesional del trabajo en esta plataforma, o el que requiere mayor conocimiento, es el programar el robot. Así es, debe crearse un código en lenguaje de programación C# o Boo que permita el correcto funcionamiento del robot, desde el movimiento hasta la implementación de sensores o la detección de otros objetos, colores, etc.

Si bien en la Universidad Adolfo Ibáñez se utiliza mayormente la plataforma alternativa Webots, al ser tan similares, tanto en funcionamiento como en la gráfica, Unity es una herramienta ideal para crear simulaciones de robots de manera digital. La primera ha demostrado ser muy efectiva en términos académicos y de enseñanza, por lo tanto, transitivamente, se asume lo mismo para la segunda.

Pese a todo lo anteriormente mencionado, Unity también presenta aspectos negativos que en algunas ocasiones dificultó la investigación y la experimentación con dicha plataforma. Las dos principales que fueron detectadas para los propósitos de este trabajo fueron el rendimiento y la capacidad.

En los primeros usos de Unity, ya sea a modo de prueba o para efectos de la investigación, rápidamente se puede apreciar que, a la hora de ejecutar la plataforma, el computador disminuye su rendimiento y aumentando su temperatura. Ahora bien, fue fácil detectar que para el óptimo uso de Unity se precisa no tener gran cantidad de programas en ejecución al mismo tiempo, ya que así se evita pérdidas de rendimiento del computador en líneas generales. A modo de aterrizar esta idea, basta mencionar que con tener un tutorial de YouTube en reproducción junto con Unity, ya se podían sentir las diferencias de rendimiento, por lo que en muchas ocasiones trabajamos con más de un dispositivo en conjunto, para así utilizar únicamente Unity en una computadora.

En cuanto a la capacidad, se puede mencionar que, a medida que se desarrollaron nuevos proyectos de prueba con Unity, el uso de memoria en el computador se incrementó exponencialmente, dejando ver que esta plataforma también es demandante en este ámbito.

A simple vista, ambas falencias no parecen del todo graves, pero ciertamente enlentecen el trabajo a realizar y si estas se proyectan a proyectos más grandes, claramente la capacidad y el rendimiento serían problemáticas a tener en cuenta a la hora de trabajar con Unity y debe tenerse en cuenta que demandan una computadora con especificaciones técnicas más avanzadas.

El principal objetivo del presente trabajo es familiarizarse con el uso de Unity para entregar una base de conocimiento que debe tenerse para el armado de un brazo robótico que pueda ser utilizado para realizar un juego sencillo en un futuro. En este manuscrito se expone el trabajo de investigación que realizamos revisando bases de datos bibliográficas y repositorios de código abierto en línea, además de revisar el uso del motor de física MuJoCo para luego integrarlo en Unity.

Algunos de los métodos utilizados para llevar a cabo este estudio fue la recopilación de información de estudios similares tanto de internet como de estudiantes UTEM/UAI que ya estaban trabajando en él. Junto con esto, lo complementamos viendo y realizando tutoriales de la plataforma para aprender a

usarla y para comprender la lógica que sigue. De esta forma, una vez que se tuviera toda la información necesaria, sería más natural y amigable el desarrollo del armado del brazo robótico.

Los estudios mencionados anteriormente son validados en la sección 3 de este escrito, donde se expone la información encontrada que fue relevante para la mejor comprensión del estudio, así como los resultados obtenidos utilizando la plataforma de Unity para armar el robot.

El resto del manuscrito está estructurado de la siguiente manera: en la sección 1 se describe lo que se propuso hacer al inicio de la investigación, junto con detalles sobre la información extra y los problemas encontrados. En la sección 2 se expone lo que se logró hacer, los cambios hechos y la descripción de la resolución a los problemas. Finalmente, en la sección 3 se realiza la conclusión del trabajo desarrollado.

II. PROPUESTA

En esta sección describiremos más detalladamente lo que nos propusimos hacer para llevar a cabo la investigación. Adicionalmente mencionaremos los problemas que podrían encontrarse en su desarrollo.

A. Recopilar información

El principal objetivo de este trabajo es recopilar información acerca de todo lo relacionado con el uso de Unity, partiendo por lo más básico, como lo es el comprender el funcionamiento de cada panel y cada herramienta de la plataforma. Luego de habernos instruido en el manejo de Unity, es de vital importancia apoyarse de tutoriales, ya sean provenientes directamente de Unity o de algún otro usuario de internet. Para esto, se creará un repositorio de los distintos tutoriales vistos, junto con una descripción de los tópicos y el minuto exacto en el que se abordan en el video. De esta forma se hace más amigable toda la información reunida y más fácil de recordar más adelante.

Toda la información considerada útil es aquella que se puede utilizar para llevar a cabo el brazo robótico en Unity, ya sea para el correcto armado, funcionamiento o testeo del mismo. Este será el conocimiento que se adquirirá durante el trabajo y son estos los conceptos en los que entraremos en detalle más adelante en este escrito.

B. Trabajo en Unity

Primero, la propuesta fue buscar videos sobre proyectos similares que nos entregaran una visión general de lo que sería el proyecto terminado. De esta manera, podríamos generar un plan de trabajo acorde a las etapas del proceso. Siguiendo en la misma línea, luego de asimilar el tipo de trabajo, se decidió desarrollar tutoriales sobre el manejo de Unity. Esto para familiarizarnos con la plataforma, ya que nunca habíamos trabajado en ella. En base a estas ideas es que comenzamos a utilizar la aplicación. Seguimos videos tutoriales, donde la mayoría funcionaba enseñando un paso a paso que luego nosotros teníamos que desarrollar de forma paralela en nuestro computador. Así también, tutoriales originales de Unity, en los que se van superando niveles según el avance que se tenga.

Después de tener mayor conocimiento sobre el trabajo y la manipulación del editor de Unity, nos propusimos intentar crear el robot. Como guía usamos los videos tutoriales “Unity Inverse Kinematics” para crear un esqueleto de brazo robótico. En las imágenes a continuación se puede ver lo esperado, que es la rotación del brazo. Las bolas rojas y la bola azul corresponden a las articulaciones del brazo, destacando la azul como la de la punta, que pretendía ser la “mano” del brazo.

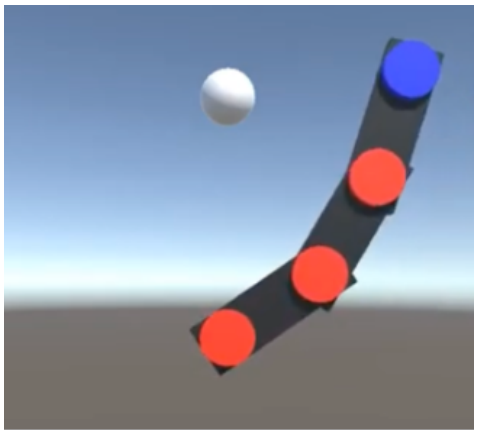

Fig. 1. Prototipo objetivo a crear un Unity

III. RESULTADOS

En esta sección describiremos lo que logramos de lo propuesto en la sección previa. Mencionaremos detalladamente como afrontamos los problemas que se presentaron.

A. Información recopilada

Dentro de todos los tutoriales acerca de juegos y simulaciones en Unity, de los que estudiamos a lo largo del semestre, hay gran variedad de ellos de los cuales no se pudo obtener información pertinente para el desarrollo del brazo robótico. Ya sea por ser demasiado específico o simplemente redundar en conceptos ya aprendidos en tutoriales anteriores. Debido a esto, se incluirán los tutoriales más relevantes con los que se trabajó y sus respectivas enseñanzas.

Antes de comenzar a desarrollar los tutoriales, decidimos estudiar los comandos básicos de la plataforma. Para esto se priorizaron los instructivos que están dentro de la aplicación de Unity, ya que varios de los encontrados en internet correspondían a versiones anteriores que no contaban con todas las funcionalidades actuales. El más básico y, así mismo, el más útil, Unity Editor [1] enseñaba cómo maniobrar en la ventana de edición. En este presentaba las herramientas para transformar, mover, rotar, cambiar de tamaño y posición y escalar. En la figura 2 se observa el tablero para seleccionar estas funcionalidades junto con la tecla correspondiente que permite seleccionarla.

Siguiendo con esto, el visor del editor muestra los ejes sobre los que se está trabajando respecto al objeto y, también,

Fig. 2. Tablero comandos editor de Unity

permite hacer los movimientos directamente desde ellos. Esta función permite manipular la figura de manera más fácil y tener mayor claridad sobre que planos se está trabajando. En la figura a continuación se expone lo recién mencionado.

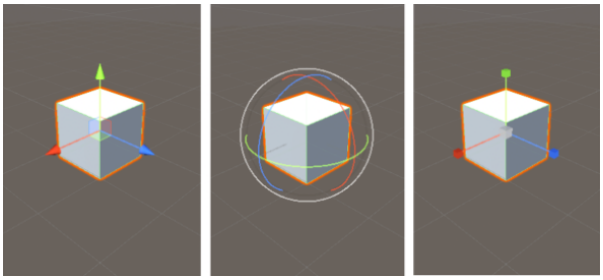

Fig. 3. Exposición de los ejes trabajados en el objeto

Con este tutorial donde se enseñaban los básicos de Unity Editor, se logró comprender de mejor manera los comandos de la aplicación. Así, se pudo dar comienzo al trabajo de los tutoriales y próximamente al armado del brazo.

Uno de los primeros tutoriales que utilizamos para hacernos una idea de cómo llevar a cabo el brazo robótico fue el tutorial "Simple Tank Game" [2]. Este tutorial a simple vista parece poco útil para los efectos del trabajo, al tratarse de un juego acerca de tanques disparando entre sí, pero en realidad aborda varios conceptos que fácilmente se podrían replicar en la obtención del brazo robótico.

El primer y más básico punto que aborda el tutorial es la creación de un objeto, en este caso un tanque, y cómo moverlo utilizando las teclas del computador. Esto se logra mediante un "Input.GetKey()", donde se ingresa la tecla a configurar para luego darle las coordenadas y la velocidad asignadas a este movimiento. Esto es pertinente para la creación del brazo debido a que se puede evidenciar que la lógica del movimiento del objeto es mediante las coordenadas (X,Y,Z), por lo que efectivamente es de utilidad para los movimientos básicos del brazo. Además de asignarle una función a las teclas, también se puede asignar una función al mouse/ratón, lo cual inmediatamente se puede relacionar con el abrir y cerrar la mano del brazo. Lo anteriormente mencionado se logra mediante otro input "Input.GetMouseButtonDown()", si se ingresa 0 se hace alusión al click izquierdo, si se ingresa 1 se está configurando el click derecho y, finalmente, si se ingresa el 2 configuraremos el click del "scroll" del ratón. Pensado en el brazo robótico, estos comandos podrían configurarse para que con el click izquierdo se abra el brazo y con el click derecho se cierre.

```
void Update()
{
    currentPos = transform.position;

    if (Input.GetKey("a"))
    {
        GetComponent<Transform>().eulerAngles = new Vector3(0, -90, 0);
        GetComponent<Rigidbody>().velocity = transform.forward * 2.5f;
    }
    else
    if (Input.GetKey("d"))
    {
        GetComponent<Transform>().eulerAngles = new Vector3(0, 90, 0);
        GetComponent<Rigidbody>().velocity = transform.forward * 2.5f;
    }
    else
    if (Input.GetKey("w"))
    {
        GetComponent<Transform>().eulerAngles = new Vector3(0, 0, 0);
        GetComponent<Rigidbody>().velocity = transform.forward * 2.5f;
    }
    else
    if (Input.GetKey("s"))
    {
        GetComponent<Transform>().eulerAngles = new Vector3(0, 180, 0);
        GetComponent<Rigidbody>().velocity = transform.forward * 2.5f;
    }
    else
    {
```

Fig. 4. Código para el movimiento de un objeto en el plano

Ahora bien, suponiendo que después de finalizado el brazo hay que ponerlo a prueba, el tutorial también brinda una manera de hacer esto posible. Dentro del juego se puede apreciar que van apareciendo tanques enemigos de manera aleatoria dentro del plano, por lo que esto se podría llevar a la práctica y que aparezcan objetos de manera aleatoria para atraparlos con el brazo robótico. Esto principalmente se logra con un "spawnTimer()" que determina el tiempo en el que estos objetos aparecen. Además de esto, otros conocimientos necesarios son los de "Box Collider" y "RigidBody", los cuales le asignan "forma" y "masa", respectivamente, a los objetos con el fin de que estos no sean traspasados y puedan interactuar correctamente entre sí, como es necesario en la hipotética situación entre el brazo y objetos a cargar.

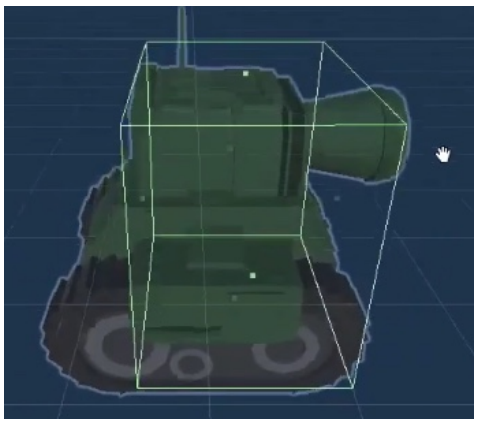

Fig. 5. Ejemplo de utilización de box collider

Continuando con la investigación, encontramos varias páginas que hablaban sobre este proyecto. Entre ellas, un repositorio de github [3] donde se encuentra el trabajo de la simulación del funcionamiento de brazo robótico creado en unity pero que utiliza el motor MuJoCo. Además de la diferencia de la física de ese proyecto (el motor), este está realizado en lenguaje Python, por lo que ahí se encuentra otra diferencia con nuestro

trabajo. De igual manera fue muy útil tener este material para entender mejor esta investigación.

B. Desarrollo Brazo en Unity

Como se mencionó en el apartado de Propuesta en II-B, Trabajo en Unity, se revisaron y desarrollaron varios tutoriales. Si bien en un comienzo la instrucción solo era revisar los tutoriales que parecieran útiles, optamos por desarrollar un plan que siguiera una lógica que permitiera ser más prácticos y lograr tener mayor claridad con el entorno de trabajo y los pasos a seguir para el armado de brazo tentativo.

Ahora, considerando todo lo aprendido en la investigación, además del seguimiento de la serie de tutoriales mencionada [4] [5], se pudo crear la estructura buscada, con las articulaciones rojas y la "mano" azul.

El problema fue que, lamentablemente, y como finalmente no era el objetivo del proyecto, se dejó de lado el trabajo de crear el brazo en Unity para enfocarnos en la investigación. Así entonces solo alcanzamos a crear la estructura del brazo pero no logramos hacerla funcionar. En las figuras 6 y 7 se puede ver la estructura creada pero la falla en el movimiento al correr la simulación respectivamente. Mientras trabajamos en esto, usamos además la plataforma para escribir código de programación de manera colaborativa, "prod liveshare visualstudio" [6].

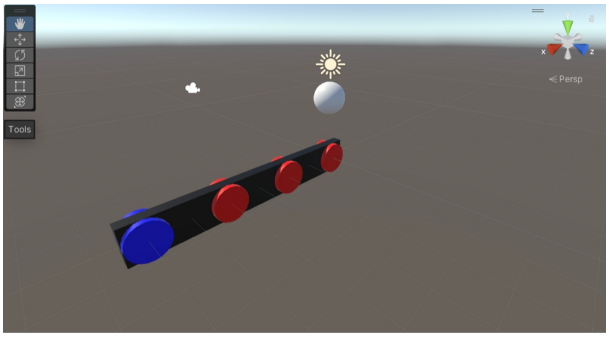

Fig. 6. Estructura satisfactoriamente creada en Unity

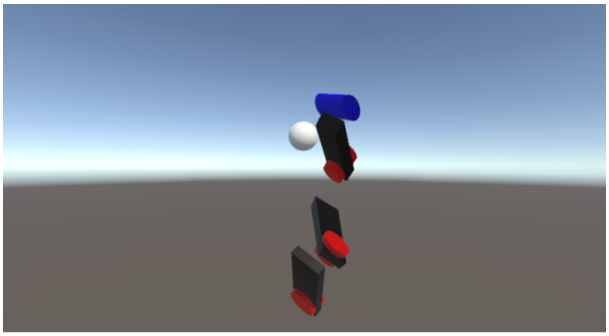

Fig. 7. Falla en el funcionamiento de la figura creada

IV. CONCLUSIONES

Durante el período de trabajo se recopiló información acerca del programa Unity y de la creación de una simulación de brazo robótico creado en esta plataforma, se revisaron video-tutoriales, páginas web, e incluso un repositorio de github. Considerando esto como el objetivo principal del trabajo, aún así quisimos aplicar un poco de lo aprendido e iniciamos la creación de nuestro propio brazo robótico. Como ya se mencionó, esta misión fue dejada de lado al darnos cuenta que no era ahí donde debíamos estar gastando nuestro tiempo, sino en la investigación en sí misma. Aún así este intento sirvió mucho de aprendizaje y, por consecuencia, fue un aporte a la finalidad del proyecto, familiarizarnos con unity y una simulación de un brazo robótico.

Todo lo aprendido y obtenido en este trabajo se encuentra disponible en un repositorio de github [7] donde se tiene acceso de manera más directa a cada enlace de donde fue obtenida información, además de lo logrado en el intento de creación de nuestra propia simulación de un brazo robótico.

REFERENCIAS

[1] U. Learn, "Explore the unity editor," 2022. [Online]. Available: https://learn.unity.com/tutorial/explore-the-unity-editor-1

[2] U. A. Store, "Tanks! tutorial — tutorial projects," 2015. [Online]. Available: https://assetstore.unity.com/packages/essentials/tutorial-projects/tanks-tutorial-46209

[3] J96W, "A robotic arm (ur5) simulation built with the physics of mujoco and the rendering of the unity." 2018. [Online]. Available: https://github.com/j96w/MuJoCo_Unity_UR5

[4] Guidev, "Unity inverse kinematics - arm animation - part 1/2," 2020. [Online]. Available: https://www.youtube.com/watch?v=VdJGouwViPs

[5] ——, "Unity inverse kinematics - arm animation - part 2/2," 2020. [Online]. Available: https://www.youtube.com/watch?v=0YiePoTe2Xo

[6] "Visual studio code for the web." [Online]. Available: https://visualstudio.microsoft.com/es/services/live-share/

[7] "Tid brazo robótico," 2022. [Online]. Available: https://github.com/antonflores/TID-Brazo-Robotico