SlimeLearn: una herramienta para enseñar inteligencia artificial

SlimeLearn: a tool for teaching artificial intelligence

Javier Burgos[1], Raúl Lara-Cabrera[1,2,*]

[1]*Escuela Técnica Superior de Ingeniería de Sistemas Informáticos, Universidad Politécnica de Madrid*
[2]*KNODIS Research Group*

Resumen

El rápido avance en el sector de la inteligencia artificial, sumado al aumento del interés por este campo, crea cada vez más demanda de recursos didácticos que ayuden a introducirse en estas tecnologías. Algo parecido ocurre con los videojuegos, que además de ser atractivos para generaciones jóvenes, pueden plantear infinidad de entornos y problemas, que utilizar para entrenar y poner a prueba agentes inteligentes. Este trabajo muestra el análisis, diseño e implementación de una herramienta, que pretende satisfacer esta demanda de recursos didácticos. Para ello hace uso de los videojuegos, con el objetivo de ser una alternativa versátil, atractiva y visual para diferentes perfiles de usuarios, incluidos nuevos programadores.

The fast growth and increasing interest in the artificial intelligence field, is creating a need for more and new didactic resources to help people learn this technology. Something similar occurs with videogames. Not only they are appealing to younger generations, games are a useful tools to make enviroments with different problems for artificial inteligence to solve and be tested on. Throughout this work, the analysis, design and implementation phases of a software application are shown. This app aims to satisfy this demand for didactic resources, making use of videogames with the intention of being a versatile, attractive and visual, for users in different skill levels.

Keywords

Inteligencia Artificial, framework, videojuego, investigación

1. Introducción

Los recientes avances y evolución del campo de la Inteligencia Artificial (IA) están fomentando un mayor uso de sus técnicas, tanto para la resolución de problemas ya existentes, como en casos de uso nuevos. Este aumento en popularidad y demanda está haciendo que cada vez más gente se interese por aprender sobre este campo.

Según *The Future Jobs Report 2020* [1] publicado por el World Economic Forum, el top tres de los empleos con mayor demanda en crecimiento, está ocupado por Data Science, AI/ML Specialist y Big Data. Todos ellos, trabajos fuertemente relacionados con el campo de la IA.

I Congreso Español de Videojuegos, December 1–2, 2022, Madrid, Spain
*Corresponding author.
j.b.valdes@hotmail.com (J. Burgos); raul.lara@upm.es (R. Lara-Cabrera)
0000-0002-7959-1936 (R. Lara-Cabrera)

Por lo tanto, no es de extrañar que los salarios para estas profesiones también se estén viendo favorecidos por la escasez de profesionales.

Dados los avanzados conocimientos estadísticos y matemáticos necesarios para comprender en profundidad el funcionamiento de muchas técnicas que engloba la IA, de ramas como el Machine Learning (ML) y el Deep Learning (DL), hay una creciente necesidad de recursos que permitan a nuevos desarrolladores aprender y practicar, de forma sencilla y visual, estas disciplinas.

Los videojuegos a menudo plantean de manera visual problemas y desafíos en base al marco de normas preestablecido por el mismo. Estas normas pueden variar drásticamente en dificultad y forma dando lugar a multitud de escenarios diferentes que nos pueden ayudar a desarrollar y poner a prueba agentes inteligentes. Dada la flexibilidad del medio, este caso de uso puede aportar valor a desarrolladores de todos los niveles, e incluso utilizarse como punto de entrada para niños y jóvenes en el mundo de la programación [2, 3, 4, 5, 6].

Desde otro punto de vista, la motivación de los estudiantes está estrechamente relacionada con el éxito académico [7]. Esto es especialmente importante en el campo de la informática, ya que tanto las titulaciones de ingeniería en general como las de informática en particular tienen importantes tasas de abandono [8]. Por lo tanto, es imperativo que la educación, y más especialmente la educación en ciencias de la computación, adopte enfoques de aprendizaje innovadores con alta eficacia instructiva y el potencial de impulsar la motivación de los estudiantes.

Los videojuegos educativos o serios son la solución ideal para esta cuestión. La utilidad de estos recursos como herramientas educativas y motivadoras ha llamado la atención de académicos y educadores durante los últimos 10 años. De hecho, revisiones de la literatura que examinan cientos de investigaciones empíricas llevadas a cabo en una serie de dominios de conocimiento y niveles educativos revelan que los videojuegos educativos son ventajosos para los estudiantes desde el punto de vista de potenciar la motivación y la adquisición de conocimientos [9, 10, 11, 12].

Nuestro objetivo es, por tanto, reducir la barrera entrada en el campo de la IA. Si creamos un entorno de aprendizaje visual e intuitivo, pero con la suficiente funcionalidad y flexibilidad, podemos adaptarnos a las necesidades cada usuario dándoles libertad para que usen la herramienta de la forma que les aporte mayor beneficio.

2. La herramienta

A continuación se detallan las características principales de SlimeLearn Framework así como la tecnología en la que se basa la herramienta.

2.1. JumpSlime!

SlimeLearn Framework ha sido concebido como una evolución de un producto anteriormente desarrollado por uno de los autores conocido como JumpSlime!

JumpSlime! es un videojuego de "plataformas", de estética y jugabilidad sencilla que consiste en hacer subir al personaje principal hasta el final del escenario mediante saltos coordinados y precisos. A pesar de contar con una premisa sencilla, la naturaleza del juego busca ser desafiante y difícil. Los errores a la hora de saltar a menudo suponen caer y perder una cantidad sustancial

de progreso, como es habitual en juegos de características similares como *JumpKing o Getting Over it with Bennet Foddy*.

JumpSlime! es un juego 2D, es decir, que el marco de acciones del personaje se restringe a un plano bidimensional. Cuando el personaje del jugador está listo para realizar un salto, una flecha se mueve a su alrededor señalando la dirección en la cual el personaje puede saltar. El jugador puede efectuar el salto con tan solo una única pulsación (de teclado o pantalla), en ese momento el personaje saltará en la dirección en la que señale la flecha en ese instante. Una vez comenzado el salto, el jugador no puede efectuar nuevos saltos hasta que este no termine de caer. Si durante el salto, este choca contra las paredes, o lateralmente contra otras plataformas, el personaje rebotará y el movimiento de caída cambiará de sentido. El salto siempre se realiza con la misma fuerza, por lo que el factor determinante para alcanzar la plataforma objetivo reside en el ángulo con el que se realiza.

El escenario se compone principalmente de plataformas, estas pueden ser de diferentes tipos y constan de diferentes comportamientos e interacciones con el jugador:

- **Plataforma Sencilla:** Permite al jugador atravesarse de abajo a arriba pero no de arriba a abajo. Esto hace que sea una de las plataformas más sencillas para el jugador, ya que admite mayor variedad de saltos para posicionarte encima, por norma general.
- **Plataforma Sólida:** Esta plataforma cuenta con detección de colisión en todas direcciones, por lo que solo será posible posicionarse encima si se lleva la trayectoria correcta como para caer encima. Si es golpeada desde abajo o por los laterales, el personaje se golpeará y caerá.
- **Plataforma Sólida Temporal:** Funciona de igual manera a la anterior, con la excepción de que una vez el personaje se posiciona encima, esta se destruirá a los tres segundos, dejando caer al personaje de seguir en la misma posición. Unos segundos después de desaparecer, vuelve a reconstruirse en la misma posición.
- **Rampas:** Este tipo de plataforma es de las más especiales, ya que el personaje no puede posicionarse encima suya. En caso de impactar contra una de ellas por la parte superior inclinada, este se deslizará por ella y caerá.
- **Plataforma Pegajosa:** Esta plataforma actúa como una plataforma sólida, salvo que si es impactada por el lado marcado, el jugador podrá agarrarse y quedar pegado a ella.

En el escenario también se pueden encontrar gemas. Estas piedras preciosas sirven de recompensa por tu progreso y, en ocasiones, por arriesgar en saltos difíciles. Además, estas pueden gastarse en grupos de cinco para deshacer el último salto.

Por último, el escenario cuenta con dos paredes a cada lado, que delimitan la zona de juego (figura 1).

JumpSlime! está hecho en *Godot Engine*. Este es un motor de videojuegos desarrollado siguiendo la filosofía de software libre y está financiado por su comunidad de usuarios a través de donaciones.

Su workflow está basado en árboles jerarquizados de nodos. Estos nodos son la entidad básica de creación. Pueden tener funcionalidades ya proporcionadas por el motor y/o tener comportamientos personalizados definidos mediante código.

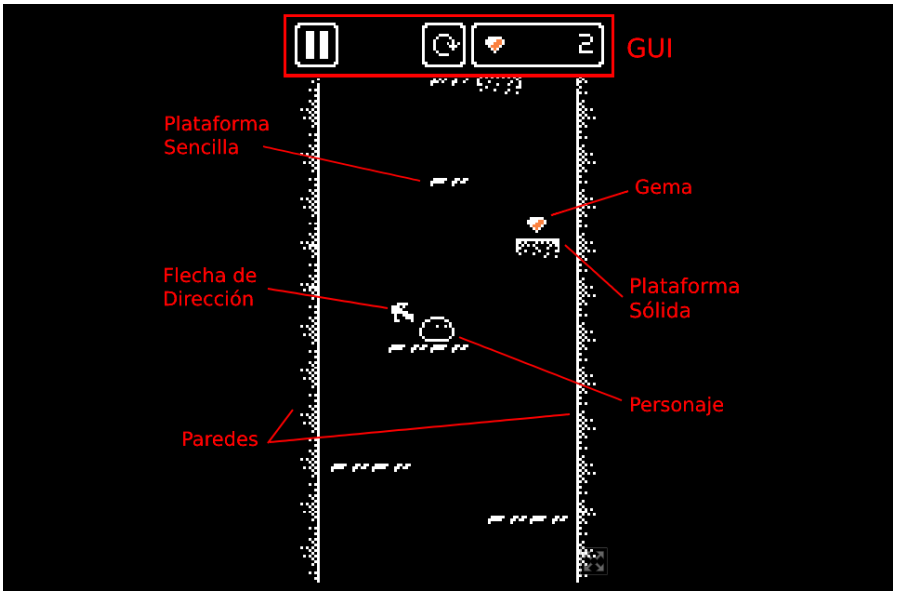

Figura 1: Elementos por pantalla de *JumpSlime!*

2.2. SlimeLearn App

Como ya se ha comentado, *SlimeLearn App* es la herramienta basada en *JumpSlime!* para la enseñanza y aprendizaje de aprendizaje por refuerzo y otras técnicas de inteligencia artificial de un modo ameno y desafiante. Se puede acceder a la herramienta mediante un repositorio de Github[1].

En la figura 2 podemos ver una representación sencilla de la topología que sigue el sistema completo. En el lado izquierdo se encuentra representada la aplicación de *SlimeLearn*, creada en *Godot*. En el lado derecho, comunicada mediante *websocket*, está representado el cliente, con los iconos de varios lenguajes en los que estos pueden ser desarrollados.

Para poder comunicar el entorno con el código del usuario se ha implementado un servidor dentro de la misma aplicación del juego, utilizando la *API* de multijugador proporcionada por *Godot*. Esta *API* dispone de varias alternativas, de más alto y bajo nivel, para el envío de información por red. Dado que *Godot* es un motor de videojuego y no está del todo enfocado a escenarios de uso como el que planteamos en este proyecto, la integración de estas utilidades con aplicaciones externas no es trivial. De todas las opciones que disponemos, se ha escogido la opción que utiliza tecnología *websocket*, por las siguientes razones:

- **Fácil implementación**: Ya que está basado en un protocolo web bastante extendido, es compatible con cualquier lenguaje de programación capaz de utilizarlo. Muchos, como es el caso de *Python*, disponen de librerías que agilizan todavía más la integración.
- **Full Duplex**: La tecnología *websocket* permite una conexión full-duplex *TCP*. Esto quiere decir que permite la comunicación bidireccional paralela. Traducido a un nivel más alto de

[1] https://github.com/javierburgosv/slimelearn

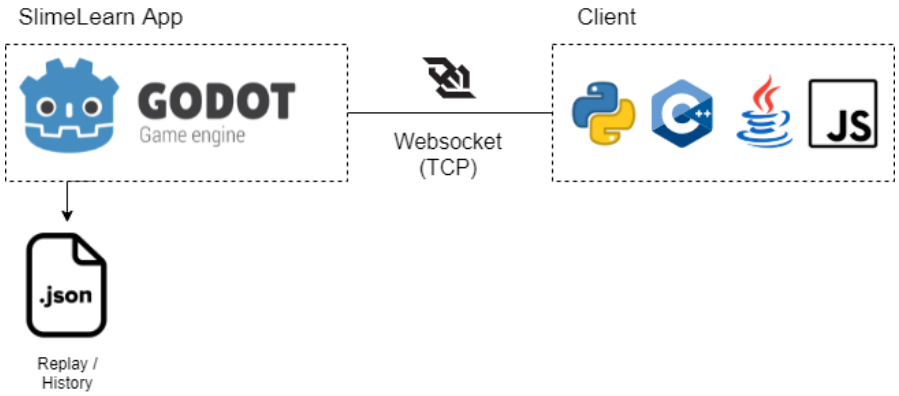

Figura 2: Diagrama de topología

abstracción, nos permite que el servidor pueda mandar mensajes a los clientes conectados, sin necesidad de petición previa. Permitiendo así la implementación de patrones de diseño que agilicen las comunicaciones y aporten comodidad al usuario.

- **Aplicaciones Futuras**: A pesar de que la aplicación de *SlimeLearn* está orientada, en su versión actual, a utilizarse como aplicación de escritorio, el uso de esta tecnología hace posible su integración con sistemas web y/o su utilización de forma remota.

El servidor ha sido implementado en un script ejecutado de forma automática con la ejecución de la aplicación utilizando la función de auto-load. Esta funcionalidad del motor funciona siguiendo el patrón singleton, es decir, que se asegura de instanciar una única vez el servidor.

Este componente es capaz de manejar conexiones mediante eventos (conocidas como señales en *Godot*) predefinidos por la *API* del motor, de manera que al producirse alguno de ellos (paquete entrante, conexión nueva, desconexión...) se llame al método indicado para actuar en consecuencia.

De cara a implementar las conexiones con el servidor en forma de subscripción, se han establecido varios modos de funcionamiento para el servidor, que determinan el tipo de evento que se va usar como activador del envío del estado del juego. Los modos permitidos son:

- **Jump**: Cuando el servidor se encuentra configurado en este modo, trasmitirá el estado del juego cada vez que el personaje termine un periodo de salto, es decir, cada vez que el personaje vuelva a estar listo para saltar.
- **Frame**: Con esta configuración la transmisión del estado del juego se realiza a cada fotograma del juego.
- **Seconds**: Este mecanismo hace que el servidor transmita la información de salida en intervalos regulares específicos de segundos que el usuario puede configurar.

El modo *Frame* mencionado, es la aproximación más habitual que podemos encontrar para aplicar inteligencia artificial a videojuegos. Dependiendo de la naturaleza del juego y del problema a resolver, puede resultar más o menos útil. En algunos casos, este método nos puede

dificultar la tarea si queremos ser selectivos y controlar en qué momento el agente debe aprender o actuar. Por ello, para aportar otro recurso más que facilite el uso de la herramienta, existen los otros dos modos, *Seconds* y *Jump*.

En lo concerniente a las acciones que puede tomar el agente para jugar al juego, en principio este solo debería requerir de una, la orden de saltar (en la dirección en la que se encuentre la flecha de dirección en el momento de la orden), como cualquier otro jugador. No obstante, hacer que un agente calcule el momento preciso de saltar, puede suponer gran esfuerzo. Es por esto, que con la finalidad, de nuevo, de rebajar la curva de dificultad, se ha implementado también la opción de indicar la orden con el ángulo de la dirección de salto que queremos. Desligando al agente de la flecha de dirección.

Además, ya que los agentes inteligentes son dados a cometer muchos errores, se ha incluido la orden para reiniciar el entorno de juego. Esta opción permite llevar a cabo series de entrenamientos de forma mucho más ágiles y rápidas.

Con la implementación del servidor, la aplicación ya es capaz de gestionar conexiones, mandar y recibir paquetes. El siguiente paso en el desarrollo ha sido la inclusión de un componente capaz de entender e interpretar los paquetes recibidos y actuar en consecuencia. Este componente ha sido implementado como un singleton que agrupa la lógica detrás de las comunicaciones, y actúa como intermediario entre el servidor y el entorno. Su separación como componente independiente del servidor tiene como objetivo desacoplar ambas implementaciones, haciendo así más fácil el desarrollo y evitando fallos a largo plazo ante actualizaciones.

En esencia, este intérprete, funciona de manera semejante a una *API* muy sencilla (ver tabla 1). Los paquetes recibidos por el servidor llevan un formato específico que este intérprete es capaz de leer. Si el intérprete reconoce las palabras clave que componen el paquete, es capaz de tomar las acciones oportunas. En caso contrario, este notifica al servidor de que el paquete recibido es erróneo para que notifique a su remitente.

Este componente también se encarga de la composición de paquetes a enviar por parte del servidor al cliente. Como se ha mencionado antes, en caso de recibir algún paquete que este no entiende, el intérprete sera el encargado de construir un paquete con el tipo y mensaje de error concreto, e indicárselo al servidor para su envío. Por otro lado, durante un funcionamiento nominal de la aplicación, se ocupa de dar formato y preparar el paquete a enviar al cliente, con la información que va recibiendo del resto de componentes sobre el entorno del videojuego.

El objetivo fundamental de la aplicación, y la razón por la que se han integrado los dos componentes anteriores en ella, es la recolección de datos para su envío a los agentes inteligentes que desean jugar. Esta información se encuentra dentro del juego de diferentes maneras. En algunos casos es información directamente extraíble de las propiedades de algunas entidades (ej: la posición cartesiana del personaje). En otros casos, se requiere de lógica y procesamiento previo.

Para obtener la información relativa al personaje, se a ampliado su funcionalidad para que sea este quien se encargue de mantener actualizado al intérprete para que el siguiente paquete de salida con el estado del juego se encuentre actualizado.

Por otro lado, para el reconocimiento del resto del entorno, se ha ampliado la lógica de la que disponía la cámara del juego. Este componente, en el juego original, se encargaba únicamente de seguir el personaje, usando una lógica personalizada. Para *SlimeLearn*, se le han añadido dos sistemas de reconocimiento del entorno. El primero se encarga de detectar todos los objetos

Tabla 1
Tabla de peticiones de la API. Un asterisco indica que el atributo es opcional. Dos asteriscos indican que el atributo depende de otro

Valor	Descripción	Atributos	Cod. Resp.
reset	Reestablece el estado original del entorno (manteniendo la configuración).	N/A	200
jump	Envía una orden de salto al personaje.	angle* (float): Indica el ángulo de salto.	200, 400
config	Envía los parámetros de configuración al servidor.	mode (enum): Selecciona el tipo de evento que usará el servidor como señal para transmitir el estado del juego. Puede tener los siguientes valores: ▪ jump: Evento lanzado cada vez que el personaje toca el suelo y está listo para saltar. ▪ sec: Evento lanzado cada periodo indicado de segundos. Rquiere del atributo "delay". ▪ frame: Evento lanzado cada fotograma. speed* (float): Multiplicador de velocidad del entorno. Usado para agilizar entrenamientos. (Valores superiores a 4 pueden alterar el motor de físicas). delay** (float): Intervalo en segundos entre eventos para el modo "sec". replay_path* (string): Ruta donde se guarda el fichero de historial de la sesión. Si no se indica no se guarda repetición alguna.	200, 400

mostrados en pantalla, los filtra según los tipos especificados, y encapsula su información (posición, tipo, tamaño...) en una estructura de datos, representativa del estado de la pantalla, que se mandará al cliente. Por otro lado, este componente, usando la *API de Godot*, es capaz de hacer capturas de pantalla, que transformará en matrices binarias (0 para píxel vacío, 1 para píxel blanco) como alternativa al método anterior.

Si examinamos los datos que podemos obtener a través del estado del juego que genera el framework, veremos que encontramos la misma información sobre determinados aspectos del juego en diferentes formas. Esta redundancia, tiene la intención de aportar versatilidad a la hora de usar la herramienta. Permite a cada desarrollador, usar el tipo de datos que mejor se adapte

al enfoque para su agente inteligente.

Otra de las funcionalidades con las que cuenta *SlimeLearn* es la capacidad de generar un historial de movimientos que son ejecutados por el personaje. Esta característica tiene como intención facilitar el estudio del comportamiento de los agentes que juegan en la aplicación, permitiendo una mejor comparativa. Si esta funcionalidad está activada en la configuración, el componente encargado (nombrado ReplayManager anteriormente) crea y actualiza un fichero *json* con las acciones tomadas por el personaje (saltar, resetear entorno...) en orden secuencial y acompañadas de información temporal.

2.3. Python SDK

Si bien es verdad que la aplicación de *SlimeLearn* descrita en el apartado anterior (sección 2.2) es capaz de comunicarse con cualquier código escrito en lenguajes de programación compatibles con la tecnología *websocket*, este proceso puede no ser trivial. Por ello se ha implementado un *Software Development Kit (SDK)* que facilita la programación a aquellos que quieran usar *SlimeLearn* en combinación con *Python*. Dando de esta manera, mayor facilidad a desarrolladores menos experimentados y reducir drásticamente la curva de dificultad de entrada al uso de la herramienta.

Entre los diferentes lenguajes a elegir para desarrollar esta *SDK*, se ha escogido *Python*, debido a su extendido uso en materia en el campo de la IA, y por tanto, recursos de este ámbito compatibles.

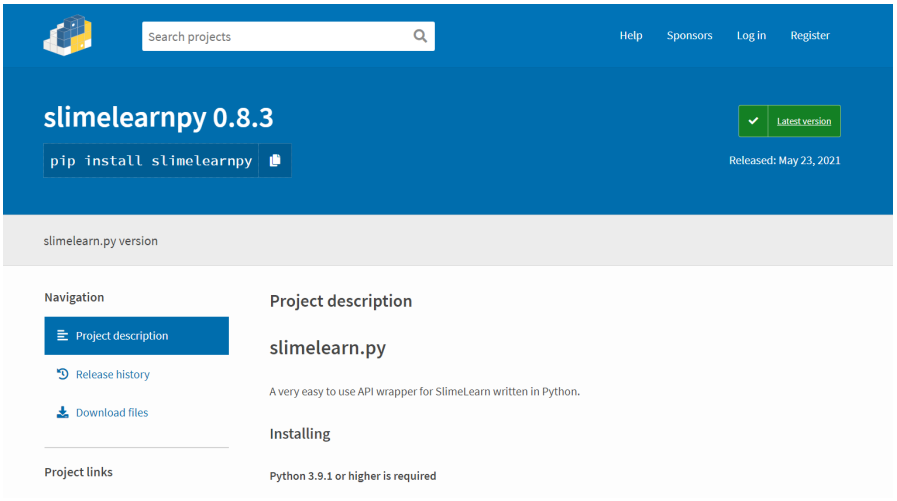

Figura 3: Página de slimelearn.py en PyPI.org

El funcionamiento de este componente es bastante simple. Se encarga de manejar las conexiones, en la parte del cliente, facilitando una serie de métodos que permiten la conexión, configuración y comunicación con el servidor de *SlimeLearn*. Es decir, actúa como un *API wrapper*, de forma que el usuario no tenga que codificar las peticiones de forma manual.

Uno de los objetivos principales de la *SDK* consiste en abstraer al usuario del manejo de la asincronicidad de las comunicaciones con el servidor. De no hacerlo, la comunicación puede

resultar muy laboriosa, ya que se necesita saber en cada momento que mensaje debemos enviar al servidor, con el formato adecuado y que respuesta esperar. Por ello la *SDK* facilita unos métodos síncronos al usuario, que llaman y esperan a otros métodos asíncronos, de uso interno, encargados del envío y recepción de paquetes.

Con el objetivo de facilitar el uso de nuestras herramientas lo máximo posible, se ha empaquetado la *SDK* y subido a *PyPI* para contar con este servicio y no depender de instalaciones manuales (ver figura 3).

Además de la librería para facilitar el desarrollo de nuevos agentes inteligentes, se proporcionan algunos agentes de ejemplo dentro del repositorio de GitHub de la librería[2]:

- **Basics.py:** En este ejemplo podemos encontrar un cliente que consta de un agente muy sencillo capaz de escribir sus coordenadas por consola y realizar saltos cada, si es posible, cada 3 segundos. Este fichero cuenta con multitud de comentarios y explicaciones que detallan el funcionamiento y propósito de cada sección de código.
- **BruteForce.py:** Esta es una demostración de como se puede plantear un algoritmo que escale por fuerza bruta el escenario.
  Para ello, este ha sido configurado con el modo jump y velocidad x2. El agente será llamado cada vez que el personaje esté listo para hacer un salto. En ese momento se leerá el estado del juego y estudiará la posición de las plataformas respecto a la posición del personaje para decidir cual es la siguiente plataforma a la que debe saltar. Una vez fijada se calcula el ángulo que forma, la línea que une el personaje con la plataforma objetivo, con el suelo. Saltar en linea recta a la plataforma objetivo sin tener en cuenta la distancia no garantiza llegar exitosamente, por ello el agente probará a añadir una pequeña desviación al ángulo de salto para intentar acertar su parábola. En caso de acertar, y caer bien en la siguiente plataforma, el ángulo de salto quedará guardado en una lista, que contendrá en orden secuencial, los pasos a seguir para subir. Si el salto falla entonces hará un reset y volverá al punto de partida, repitiendo los saltos que ya sabe hacer primero, hasta volver al salto que trata de aprender para probar con una nuevo ángulo.
  El principal inconveniente de este agente es que el tiempo que tarda en aprender a subir aumenta exponencialmente con el número de plataformas (y por tanto de saltos) que hay en el mapa. Además este agente, una vez termine su entrenamiento y sepa llegar a la cima, solo sabrá escalar este mapa en específico.
  Aunque obtiene buenos resultados, y siempre que los saltos que hay en el mapa sean posibles suele llegar a la cima, podemos encontrarnos con algunos problemas en saltos a la misma altura. El agente no es muy efectivo en situaciones en las que tiene que saltar a un mismo nivel para reposicionarse para poder llegar a su siguiente salto. Para intentar solventar esto, el agente está configurado para que al terminar un salto a la misma altura que en la que comenzó, este no sea siempre considerado como fallo. Aunque esta solución funciona para algunos casos, este cambio también puede introducir algunos saltos innecesarios dados como válidos en situaciones en las que el personaje salta y cae en el sitio.
  El agente consta de unos valores de configuración que juegan con las tolerancias y márgenes de error para retocar su funcionamiento. Además puede ser fácilmente optimizado y

[2] https://github.com/javierburgosv/slimelearn.py

modificado para alterar el filtro de éxito que evalúa los saltos o los métodos de detección de la siguiente plataforma objetivo.
Aunque no es la solución más eficiente para el problema, este agente sirve como prueba conceptual, que demuestra como sin mucho procesamiento respecto a la información de salida se pueden obtener buenos resultados. Además muestra como se puede hacer que el agente guarde información necesaria entre llamadas y la utilice para evaluar decisiones pasadas, de manera básica.

2.4. Uso Básico

Para la puesta en marcha de cualquier cliente, debemos de seguir los siguientes pasos:

1. Realizar conexión con el servidor. Por defecto la aplicación de *SlimeLearn* se levanta en local en el puerto 8080, por lo que podemos conectarnos a ella a través de la dirección localhost:8080.
2. Indicar configuración deseada. Para ello, una vez conectados debemos mandar un paquete en formato *json* con la configuración deseada.
3. Comenzar la escucha de eventos. Una vez conectado y configurado el servidor comenzará a mandar el estado del juego por la conexión establecida. Debemos asegurarnos de tener algún mecanismo para escuchar la llegada de estos paquetes y poder así usarlos.

En la figura 4, se puede ver de manera gráfica el comportamiento de las comunicaciones mediante un diagrama de secuencia *UML 2.0*. Este describe un escenario nominal de uso, sin incidencias ni interrupciones previas.

3. Conclusiones

En este trabajo, se ha abordado la gran barrera de entrada presente en el campo de la Inteligencia Artificial. Para ello, se procedió entonces con el análisis, diseño e implementación de una aplicación de escritorio con un entorno virtual, creado mediante adaptación de un videojuego desarrollado previamente, y una interfaz de comunicación. Esta aplicación, permite al usuario comunicar agentes inteligentes codificados en diferentes lenguajes de programación y ponerlos a jugar al juego, recibiendo la información de su estado de forma sencilla y tomando acciones acordemente. Finalmente se complementó la aplicación, con la implementación de una *SDK* que facilita todavía más la labor de comunicación desde *Python*, uno de los lenguajes más extendidos para IA.

Uno de los puntos a destacar de este trabajo es la demostración de que los videojuegos pueden ser utilizados como herramienta complementaria para otros sectores tecnológicos, proporcionando entornos hechos a medida con multitud de propósitos. *SlimeLearn* es un prototipo que ejemplifica uno de los muchos usos que le podemos dar al videojuego, y de como puede ser adaptado al ámbito didáctico.

Encontramos margen de mejora a corto plazo, principalmente, en las utilidades del framework. Este puede ser dotado de muchas más facilidades para procurar cubrir completamente todo el espectro de desarrolladores posible, con utilidades que procesen la información en mayor

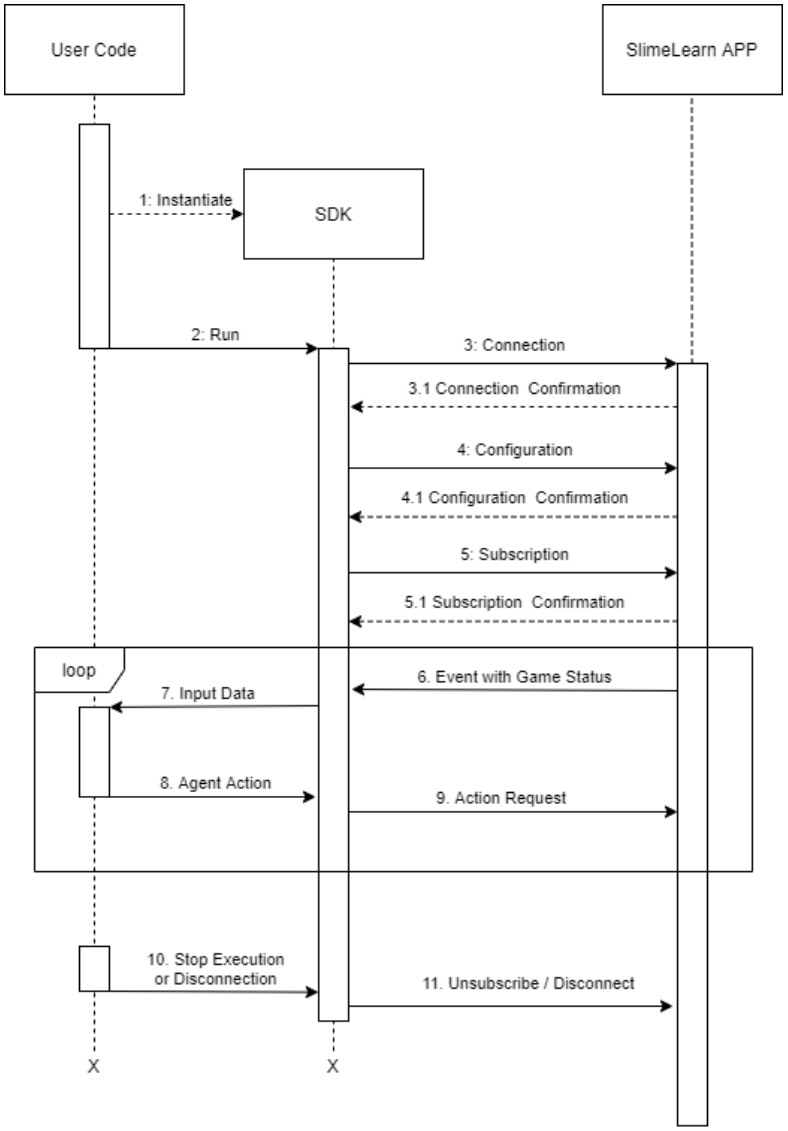

Figura 4: Diagrama de secuencia de comunicaciones nominal

profundidad y ampliando la cantidad de parámetros configurables por el usuario. También sería interesante añadir la funcionalidad para cambiar los obstáculos del escenario mediante algún tipo de editor dedicado, para que todo usuario pueda utilizarlo sin modificar el proyecto fuente. Además, utilizar el sistema de historial de movimientos actual para poder reproducir entrenamientos a modo de repetición, podría ser una manera muy visual y satisfactoria de guardar y compartir tus logros.

A medio plazo, el framework podría ser transformado en una aplicación web. De esta manera sería posible la utilización del mismo desde un mayor número de equipos y podría facilitar enormemente su utilización y puesta en marcha (ideal para centros educativos). El prototipo

actual, fue creado con tecnologías preparadas para funcionar en navegadores y a través de la red con este punto de expansión en mente.

Referencias

[1] World Economic Forum, The future jobs report, 2020.

[2] K. Hartness, Robocode: using games to teach artificial intelligence, Journal of Computing Sciences in Colleges 19 (2004) 287–291. doi:10.5555/1050231.1050275.

[3] J. W. Ho, M. Scadding, S. Kong, D. Andone, G. Biswas, H. Hoppe, T. Hsu, Classroom activities for teaching artificial intelligence to primary school students, in: Proceedings of International Conference on Computational Thinking Education, 2019, pp. 157–159.

[4] J. DeNero, D. Klein, Teaching introductory artificial intelligence with pac-man, in: First AAAI Symposium on Educational Advances in Artificial Intelligence, 2010.

[5] H. Burgsteiner, M. Kandlhofer, G. Steinbauer, Irobot: Teaching the basics of artificial intelligence in high schools, in: Proceedings of the AAAI Conference on Artificial Intelligence, volume 30, 2016.

[6] J. M. Alonso, Teaching Explainable Artificial Intelligence to High School Students, International Journal of Computational Intelligence Systems 13 (2020) 974–987. doi:10.2991/ijcis.d.200715.003.

[7] R. S. Alsawaier, The effect of gamification on motivation and engagement, International Journal of Information and Learning Technology 35 (2018) 56–79. doi:10.1108/IJILT-02-2017-0009.

[8] I. O. Pappas, M. N. Giannakos, L. Jaccheri, Investigating factors influencing students' intention to dropout computer science studies, in: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, Association for Computing Machinery, New York, NY, USA, 2016, p. 198–203.

[9] E. A. Boyle, T. Hainey, T. M. Connolly, G. Gray, J. Earp, M. Ott, T. Lim, M. Ninaus, C. Ribeiro, J. Pereira, An update to the systematic literature review of empirical evidence of the impacts and outcomes of computer games and serious games, Computers & Education 94 (2016) 178–192. doi:10.1016/j.compedu.2015.11.003.

[10] T. M. Connolly, E. A. Boyle, E. MacArthur, T. Hainey, J. M. Boyle, A systematic literature review of empirical evidence on computer games and serious games, Computers & Education 59 (2012) 661–686. doi:10.1016/j.compedu.2012.03.004.

[11] B. Karakoç, K. Eryılmaz, E. Turan Özpolat, I. Yildirim, The Effect of Game-Based Learning on Student Achievement: A Meta-Analysis Study, Tech. Know. Learn. 27 (2022) 207–222. doi:10.1007/s10758-020-09471-5.

[12] C. A. Bodnar, D. Anastasio, J. A. Enszer, D. D. Burkey, Engineers at Play: Games as Teaching Tools for Undergraduate Engineering Students, Journal of Engineering Education 105 (2016) 147–200. doi:10.1002/jee.20106.