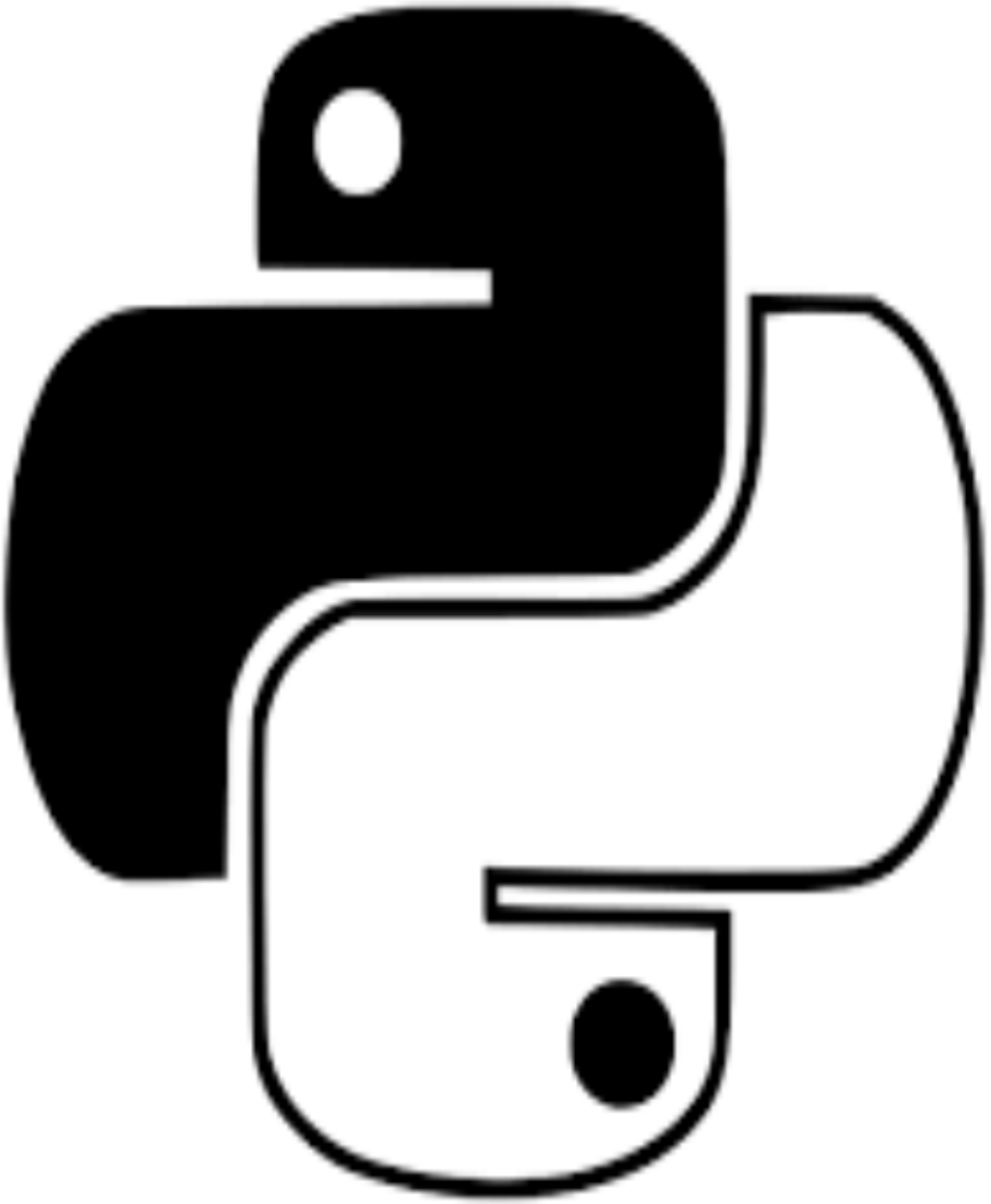


Alvaro De Vega Tolbaños
Mario Agra Taboada

Índice

INTRODUCCIÓN

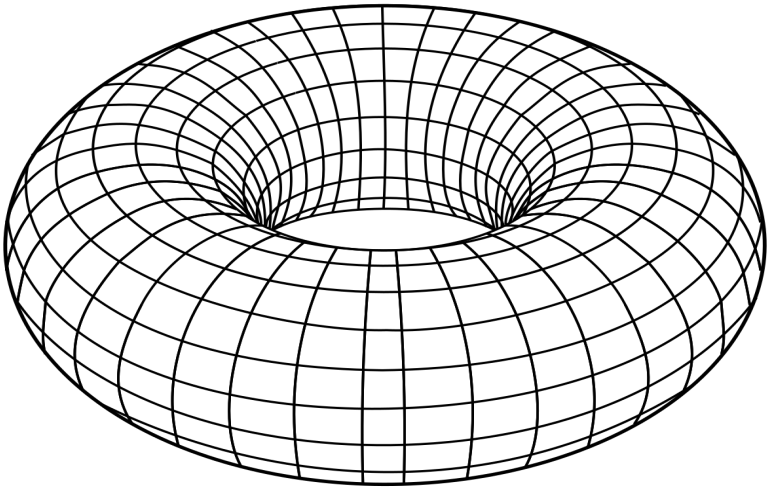
Como grandes fans de los videojuegos que somos Álvaro y yo, decidimos que podría ser una buena idea replicar el clásico juego de Snake de la década de los 70 en python. Como ya comentamos en el anteproyecto, nos fascinaba la idea de poder combinar un juego creado por nosotros con algoritmos de inteligencia artificial. Tras varias semanas investigando, dimos con un algoritmo de inteligencia artificial que se adapta perfectamente con el modelo de juego que nosotros estábamos creando.

Se trata de un algoritmo de inteligencia artificial llamado **A* o Astar**, que funciona evaluando una serie de **nodos del juego**, para ser capaz de discernir a qué dirección se tiene que mover la serpiente, en función de dónde esté el trozo de comida o recompensa que tiene que alcanzar. En nuestro caso, dentro del juego hemos implementado una modalidad para que cualquier jugador humano que se atreva, pueda medir sus habilidades contra las de nuestra IA y vea como probablemente sea terriblemente sobrepasado por esta.

Durante el desarrollo del algoritmo de nuestra IA, decidimos que la íbamos a dar un comportamiento mínimamente parecido a como interactuaría un ser humano con nuestro juego. De esta forma, aunque siga siendo mucho más eficiente capturando obstáculos que cualquier humano, en ciertos casos nos puede beneficiar para al menos poder acumular algo más de puntuación cuando juguemos contra ella.

Aparte de implementar este algoritmo, decidimos que también podíamos ir un paso más allá, incluyendo más funcionalidades y propiedades al juego para mejorar la experiencia. Para hacer el juego más dinámico, aparte de la comida, decidimos incluir **obstáculos**. Estos obstáculos se pueden generar en cualquier parte del mapa, y tienen un gran hándicap, en el momento que los tocas automáticamente pierdes la partida.

Por otra parte, cabe destacar que aparte de innovar ciertas funcionalidades con respecto al juego original, también decidimos incluir las básicas. En primer lugar tendríamos, el **comportamiento toroidal** de la ventana que gestiona el juego. Que principalmente debe su nombre a una figura geométrica sobre un eje en tres dimensiones que tendría el siguiente aspecto:



¿Qué conseguimos con esto?. Pues bien, esto aporta una **mayor jugabilidad** al juego dado que la serpiente se mueve a lo largo de un plano “infinito”. Es decir, cuando la serpiente en movimiento se encuentre con uno de los bordes del mapa, en vez de perder, saldrá automáticamente por la casilla correspondiente del otro lado de la ventana. La **segunda funcionalidad** incluida del juego de snake original sería: la gestión de colisiones propias de la serpiente.

¿Qué queremos decir con esto?. Cuando en algún momento de partida, la cabeza de la serpiente toque algún punto de su cuerpo, automáticamente se terminará la partida y habrás perdido. Si bien este **handicap** no puede parecer muy relevante al comienzo de la partida, conforme la serpiente va aumentando su tamaño es un serio problema a tener en cuenta si quieres prolongar la puntuación de tu partida.

Para cerrar la parte de funcionalidades incluidas en nuestra versión del juego, hablaremos de los **niveles de dificultad**. Tras investigar encarecidamente sobre la historia de este gran clásico, observamos que en la mayoría de adaptaciones no se ofrecía la posibilidad de poder variar la dificultad del mismo. Por esto, decidimos agregar cuatro niveles de dificultad **(Fácil, Medio, Difícil, Nivel Dios)** para mejorar la experiencia del usuario en el juego.

Conforme aumenta la dificultad del juego, la velocidad de la serpiente es más rápida,la cantidad de obstáculos es mayor y estos son más difíciles de esquivar. Incluso a partir del tercer y cuarto nivel hemos incluido obstáculos en movimiento para complicar más aún la jugabilidad del juego. Cuando has conseguido capturar toda la comida de un determinado nivel, automáticamente pasas al siguiente, excepto en el nivel Dios. Si consigues pasar el nivel Dios, seguirás en ese nivel hasta que pierdas la partida.

Para finalizar este tema, destacamos que también implementamos un sencillo sistema de puntuaciones, para que en todo momento puedas ver un ranking histórico de cuales han sido tus mejores partidas durante todo el tiempo que has invertido jugando.
Nuestro principal objetivo al embarcarnos en este proyecto, era el de ver hasta dónde éramos capaces de llegar con los conocimientos que habíamos adquirido en el curso hasta el momento. Y aparte, el de aprender a manejar y entender nuevas tecnologías, algoritmos y lenguajes de programación en el camino.

Teniendo en cuenta que hemos sido capaces de desarrollar el juego íntegramente en un lenguaje totalmente desconocido para nosotros en ese momento, como era **Python**. Que hemos usado nuevos IDE 's y tecnologías para nosotros como son: **pycharm, visual studio code, git, gitHub.** Y, que hemos combinado todo esto con el aprendizaje y la inclusión de algoritmos de inteligencia artificial, considero que hemos cumplido con creces nuestro objetivo inicial.

Como he mencionado anteriormente, vamos a explicar algunas de las tecnologías que hemos utilizado para la realización de este proyecto. El lenguaje utilizado decidimos que fuese Python, debido a su sencillez y potencial combinado con su fácil inclusión para tecnologías relacionadas con la inteligencia artificial.

Por una parte, Mario ha utilizado **Visual Studio Code** para desarrollar su parte del código, mientras que Álvaro decidió utilizar un IDE llamado **Pycharm,** porque lo veía más intuitivo y sencillo a la hora de trabajar con código Python. Por otro lado, añadimos que para gestionar el proceso del proyecto, aprendimos a usar git y lo utilizamos para gestionar los commits y hitos del mimo. Y por último, combinamos el uso de git con un repositorio remoto de github, para poder tener todo nuestro código centralizado y que así fuese más sencillo poder acceder a ello y recuperarlo en caso de que ocurriera algún fallo.

PLANIFICACIÓN y SEGUIMIENTO

En la primera y segunda semana el equipo de desarrollo de software empezó a planificar por un lado la estructura, funcionamiento, diseño del front, gestión del juego y funcionamiento y por otro lado las herramientas, IDE's y demás sistemas de software que se usará en el futuro.

De la tercera a séptima semana el equipo de desarrollo de software se encargará de desarrollar el juego al completo, uno de los principales inconvenientes que se trataron es la gestión de la creación de objetivos obstáculos no superpuestos.

De la séptima a la novena semana, el equipo se enfrentó al mayor reto hasta la fecha, la concepción, desarrollo e implementación de una IA plenamente funcional.

De la cuarta a la última semana, el equipo había realizado innumerables comprobaciones y optimizaciones del código para una mayor eficiencia y mejor desarrollo.

En la última semana el equipo se dedicara a desarrollar la documentación que explicara con detalle el funcionamiento, desarrollo de la aplicación

A continuación mostraremos un diagrama de gantt el cual muestra la planificación tomada

CUERPO DEL DOCUMENTO

**Explicación de tecnologías y librerías usadas en el proyecto*

Como hemos mencionado anteriormente en una parte de la introducción del proyecto, para desarrollar nuestro juego hemos utilizado dos IDE's distintos que serían, **Visual Studio Code y Pycharm** respectivamente. Una vez dentro del código de proyecto, hemos utilizado una serie de módulos o librerías que han sido cruciales para el desarrollo final de nuestro juego. A continuación, explicaremos cuales son estas librerías/utilidades codificadas en el lenguaje de programación Python.

En primer lugar, cabe destacar mencionar la librería que sustenta el juego en sí, y todas las funcionalidades intrínsecas en él. Se trata de una librería llamada **Tkinter,** y por lo que hemos investigado, se trata de un módulo escrito íntegramente en python. Se trata de una librería utilizada a nivel mundial en toda la comunidad de python, para codificar las conocidas GUI's, o como se diría en español "interfaces gráficas de usuario".

Durante nuestra tiempo en el curso de DAM **(Desarrollo de Aplicaciones Multiplataforma)**, utilizamos el lenguaje de programación Java para la mayoría de tareas que se debían llevar a cabo con lenguajes de programación. Concretamente en la asignatura de **desarrollo de interfaces,** trabajamos durante todo el año con una librería codificada en Java llamada **JavaSwing**, que si bien decían que estaba anticuada, a nosotros nos valió perfectamente para poder llegar a cumplir los objetivos del curso.

Principalmente, esta librería se encargaba de dar la posibilidad de codificar **interfaces de usuario** en Java, que a nuestro parecer eran muy personalizables. Gracias al uso exhaustivo de esta librería durante todo el periodo del segundo año del ciclo, conseguimos implementar de manera **rápida y eficaz** la interfaz gráfica de usuarios que queríamos utilizar para gestionar nuestro juego.

En nuestro caso, debido a que se trataba de una **"remasterización"** de un juego popular en la década de los setenta, decidimos codificar nuestra interfaz más bien sencilla, intuitiva y ambientada en esa época en concreto. Para acabar con la información sobre este módulo, añadimos que la librería **tkinter** es mucho más amplia que las utilidades que hemos implementado en nuestra interfaz, y que no descartamos utilizar muchas más herramientas de la misma en próximas actualizaciones en el futuro, como por ejemplo: el uso de canvas, la implementación de otros widgets que ofrezcan otras funcionalidades etc.

La segunda librería más relevante durante el proceso de desarrollo de nuestro juego ha sido **Pygame**. Se trata de una librería enfocada principalmente al desarrollo de juegos sencillos en python. Es muy usada a nivel mundial por toda la comunidad de python, y es muy versátil sobre todo para juego en dos dimensiones.

Gracias a su sintaxis sencilla e intuitiva se pueden desarrollar todo tipo de juegos en dos dimensiones sobre ella. Algunos ejemplos serían: el juego de la vida, el juego de space invaders, snake, adaptaciones del juego de Mario Bros etc. En nuestro caso era la primera vez que nos encontrábamos con un reto como este, porque nunca antes habíamos programado un juego en Java.

Las primeras semanas nos costó un poco adaptarnos a la sintaxis que utiliza la librería, principalmente porque nos faltaban conceptos básicos del funcionamiento de la estructura de datos de Python y de su sintaxis en general. Pero una vez que la empezamos a dominar decidimos que íbamos a implementar dos ventanas en nuestro proyecto que hicieran uso de la librería.

Concretamente se trata de las ventanas que gestionan los dos modos de juego que tenemos implementados en nuestro juego (esta ventana puede estar sujeta a cambios desde que hicimos esta captura de pantalla de la misma).

Como vemos en la captura de arriba, las ventanas codificadas serían lanzadas automáticamente cuando se pulsase uno de los dos botones. Gracias al escuchador

de eventos de los botones de la librería **tkinter,** cuando se produce el evento de click sobre cualquiera de los dos botones que gestionan los modos de juego, se llama a un método elaborado con la librería pygame que inicia el juego pertinente en una ventana propia de pygame.

En el tercer lugar de este apartado, vamos a mencionar algunas librerías utilizadas en nuestro juego que, aunque no tengan un papel tan relevante como las otras dos, han sido muy útiles a la hora de desarrollar el juego. Se trata de los módulos: **numpy, random y pickle.** A continuación, explicaremos de manera breve cuál ha sido la utilidad que hemos dado a estas librería en nuestro proyecto.

Comenzamos con la librería **numpy**. Se trata de una biblioteca para el lenguaje de programación Python que proporciona soporte a los usuarios de la misma para crear grandes vectores y matrices multidimensionales, junto con una gran colección de funciones matemáticas de un alto nivel para operar con ellas. Esta librería fue creada originalmente por Jim Hugunin, junto con contribuciones de varios otros desarrolladores.

Concretamente en el ámbito de nuestro proyecto, hemos utilizado esta librería para gestionar operaciones muy complejas, que están directamente relacionadas con la disposición del tablero (o array bidimensional) donde se mueve nuestra serpiente y con el estado del tablero del juego en cierto instantes.

Un pequeño ejemplo de esta librería sería la siguiente sentencia de código que se encarga de copiar toda la matriz o tablero con todo el estado del juego (guardándolo en una variable) en ese momento, para poder usarlo posteriormente en otra parte del código. Véase que se ha importado la librería de la siguiente manera: **import numpy as np**.

```
# Actualizamos el estado del juego
self.gameState = np.copy(newGameState)
```

Por otro lado tendríamos la librería **random**. Se trata de una librería estándar de Python que incluye un conjunto de funciones las cuales nos permiten obtener de distintos modos números aleatorios o, para ser rigurosos, pseudoaleatorios. Aunque es cierto que ha sido una de las librerías menos utilizadas, ha tenido un papel crucial en algunos aspectos de nuestro juego que comentaré a continuación.

Principalmente, ha sido utilizada en nuestro proyecto para generar aleatoriamente las posiciones tanto de las recompensas como de los obstáculos. De esta forma, aumenta la satisfacción del usuario debido a que mejora la jugabilidad general del juego. Un ejemplo del uso de esta librería en el código de nuestro proyecto sería el siguiente:

```
direccion = random.randrange(4)
self.direccion = lista_Direcciones[direccion]
```

Como vemos, se genera un índice aleatorio de la lista que contiene todas las direcciones posibles, para poder establecerlo como dirección en ese momento. En último lugar en lo que se refiere a este conjunto de tres librerías, tendríamos la librería **pickle**. Se trata de una librería escrita íntegramente en Python, utilizada principalmente para poder serializar todo tipo de objetos que consideremos pertinentes en nuestro código.

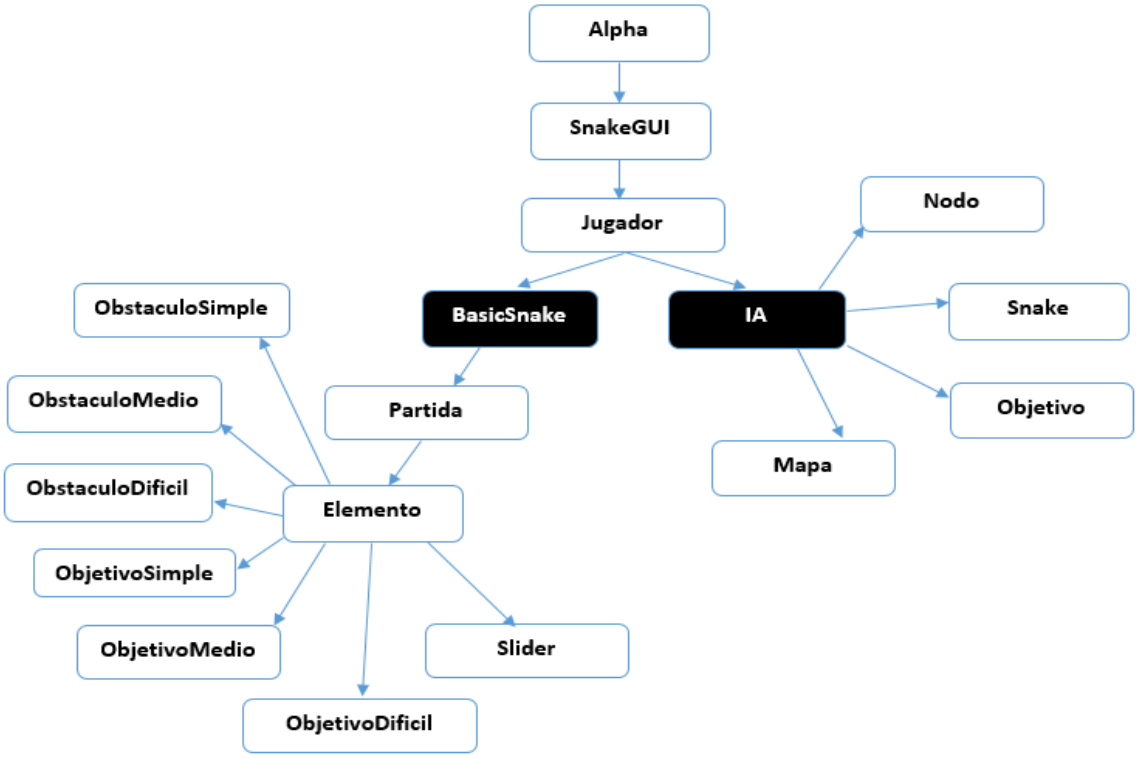
Ya que con anterioridad en la documentación mencionamos que hubo una parte en concreto de la asignatura de Desarrollo de interfaces que nos ayudó de cara al proyecto, esta vez no va a ser menos. El tema de la serialización y deserialización de objetos lo tratamos ampliamente en Java, con nuestra profesora de programación y acceso a datos Patricia Andrés.

Gracias a que hicimos bastantes ejemplos relacionados con el tema, y tras explicarnos todas las ventajas que podía tener serializar ciertos objetos en nuestro código, decidimos implementar este librería para gestionar nuestro sistema de puntuaciones. De esta manera lo hacemos más “seguro”, ya que sería bastante más difícil para alguien modificar el archivo concreto que tiene las puntuaciones. Principalmente porque toda la información relacionada con las puntuaciones y los nombres de los jugadores se encuentra en formato binario, en vez de en un simple archivo de texto fácilmente modificable por cualquiera.

**Diagrama de clases del proyecto*

A continuación, se muestra una captura de pantalla que contiene el diagrama (en forma de árbol) de la jerarquía de clases implementadas durante el desarrollo de nuestro proyecto. Cabe destacar que los nodos pintados en negro que corresponden a **BasicSnake y IA**, no son clases como tal, sino que estan ahí para ayudar a facilitar la compresión de la estructura del juego. Además, concretamente corresponden a las dos modalidades de juego que puedes disfrutar en el juego.

Para un mejor entendimiento del diagrama, decidimos añadir poca información en cada uno de los nodos que lo componen, y ya posteriormente en la siguiente sección explicamos detalladamente todas y cada una de las clases que conforman el juego con información sobre la función de los métodos contenidos en cada una de ellas.



**Estructura del código*

Para comenzar, hablaremos de la clase alpha, la cual es la encargada de poner todo en marcha. La clase alpha contiene un objeto de clase snakeGUI, la cual es la interfaz del juego, que se mantendrá constante durante toda la ejecución del juego.

La interfaz snakeGUI está dividida en dos secciones; la primera está situada en la parte izquierda, la cual es la encargada de recoger los datos del usuario y seleccionar el nivel al que se desea jugar, además hay dos botones para elegir el tipo de juego, el primero que es el juego por defecto, el cual tiene en cuenta lo seleccionado antes y el otro, en el que compites contra una IA. Y al otro lado de la pantalla se muestra el ranking de los mejores 5 jugadores que han jugado en el modo básico. Además, hay una serie de funciones que se ejecutarán en determinados momentos, una de ellas es la de cargar los datos del ranking, la cual se encargará de deserializar un archivo binario que alberga de manera encapsulada objetos de tipo jugador que contienen únicamente el nombre y la puntuación. Por último cabe decir que la interfaz está diseñada para emular una interfaz del siglo XX.

En cuanto al juego básico, tras pulsar el botón nombrado antes y con la etiqueta “basicSnake”, lanzara un objeto tipo partida, el cual de manera automática en el constructor creará por un lado un objeto de tipo elemento que se encargará de crear todos y cada uno de los elementos de la partida(se explica con más detenimiento más adelante), pero antes de todo se definirá el nivel, nombre del jugador, el timing del juego y demás variables que serán necesarias para la ejecución de la partida. Por otro lado una vez que se haya creado el objeto elemento por completo, se lanzará la función partida, la cual se encargará de llevar a cabo todo el proceso e iteraciones que sean necesarios para llevar acabo la partida, en primer lugar hay una función llamada dificultad que irá modificando el timing de la partida en función del nivel en el cual nos encontremos, en segundo lugar, la función inicializarMapa se encarga de dar valores de la matriz o mapa segun la posicion de la serpiente, objetivos y obstáculos, en tercer lugar la función actualizarMapa se encarga de tanto modificar el mapa según el movimiento de objetivos, obstáculos y la serpiente, en cuarto lugar la función cambiaNivel, la cual es llamada únicamente en el momento en el que todos los objetivos son asimilados por la serpiente, por último la función restante es la ya nombrada partida, que como bien se ha comentado antes es la encargada de todo el proceso de captura de movimientos por teclado, gestión de reglas, almacenar la puntuación al finalizar la partida, etc.

La clase elemento contiene una serie de funciones que se encargan de la creación de objetivos, obstáculos, además de calcular en base a las dimensiones del mapa una cantidad más que razonable de elementos. En primer lugar crea un objeto de tipo slider que estará situado en el centro del mapa, después se lanzarán un par de métodos, los cuales funcionan de forma muy similar, en primer lugar dentro de las funciones crearemos una variable, la cual asuvez llamará a un método que nos devolverá una cantidad n de objetivos o de obstáculos depende del método en el que estemos, luego en función del nivel que sea crearan mas o menos elementos de diferentes niveles, el sistema por el cual crea los elementos, es un compendio de una multitud de iteraciones sumadas con diferentes verificaciones para conseguir que en el 100% de los casos ningún elemento se superponga a otro durante su creación, y una vez creados se guardan los elementos y obstáculos en sus listas correspondientes.

En cuanto a la clase ObjetivoSimple u ObstaculoSimple, ambas son prácticamente idénticas, empezando por el constructor, el cual contiene únicamente su posición, dirección, estado, etc. En cuanto a su función la cual se encarga de modificar su posición en función de la dirección que tenga, sin embargo y el equipo de desarrollo de software, ha diseñado estos atributos para que si se hace una modificación únicamente en el constructor sea capaz de manera sencilla cambiar su comportamiento

En cuanto a la clase ObjetivoMedio u ObstaculoMedio, ambas son prácticamente idénticas, empezando por el constructor, el cual contiene únicamente su posición, dirección, estado, pero ha diferencia de los elementosSimples, estos contienen un añadido, el cual de manera aleatoria y tan solo en su inicialización obtendrán una dirección que se mantendrá hasta el fin de la partida. En cuanto a su función la cual se encarga de modificar su posición en función de la dirección que tenga, sin embargo y el equipo de desarrollo de software, ha diseñado estos atributos para que si se hace una modificación únicamente en el constructor sea capaz de manera sencilla cambiar su comportamiento.

En cuanto a la clase ObjetivoDificil u ObstaculoDificil, ambas son prácticamente idénticas, empezando por el constructor, el cual contiene únicamente su posición, dirección, estado, pero ha diferencia de los elementos medios, éstos contienen un añadido, el cual se observa en el método explicado más adelante, haciendo que manera aleatoria y cada vez que se llame al método obtendrán una dirección aleatoria, la cual se mantendrá hasta el fin de la partida. En cuanto a su función la cual se encarga de modificar su posición en función de la dirección que tenga, sin embargo y el equipo de desarrollo de software, ha diseñado estos atributos para que si se hace una modificación únicamente en el constructor sea capaz de manera sencilla cambiar su comportamiento.

En cuanto a la clase slider es la que gestiona todo lo relacionado con la serpiente, en el constructor podemos observar una serie de atributos los cuales unos nos servirán para el posicionamiento y movimiento, otro para la gestión con otros elementos, etc. En cuanto al método que modifica la dirección y por ende la posición, se encarga de modificar la posición de la cabeza, a su vez se encarga de gestionar si se choca con algo, ya sea un objetivo, obstáculo, o sí mismo, dando lugar a diversos fines. El método objetivoAlcanzado() se encarga de gestionar el caso cuando un objetivo es comido por la serpiente, asimilando el objetivo a su cuerpo.

Por otro lado si pulsamos el botón con la etiquete SnakeVsIA, nos lanzará la clase ia, la cual ejecutará el método main, el cual es el encargado de poner la partida en marcha, la cual nada más empezar, se inicializan una serie de características para diseñar las características de la partida, ahora creará un objeto de tipo mapa, el cual es una matriz bidimensional, un objeto tipo snake que será la IA y otro de tipo snake que será el jugador, una vez creado estos elementos, se empezará con el flujo del juego, al principio del bucle que controla se llamará a la función astar, la cual sigue un algoritmo que detecta cuál es el camino más corto desde su posición actual al objetivo crear un par de nodos de tipo nodos, para su posterior gestión y cálculo.

***Gestión de persistencia de las puntuaciones.**

En este caso, durante todo el transcurso de nuestro proyecto no hemos visto la necesidad de crear una base de datos donde almacenar cierta información relacionada con el juego. En un momento dado, decidimos que para hacer el juego más completo, íbamos a proporcionar la posibilidad de poder registrar un ranking de las mejores puntuaciones alcanzadas.

Debido a que durante el segundo año del ciclo hicimos mucha insistencia en la persistencia, se nos ocurrió una idea para poder un sistema de puntuaciones de la forma más sencilla posible. Este sistema se podría explicar de la siguiente manera: Utilizamos concretamente dos métodos y una lista de puntuaciones para implementar el sistema. La lista de puntuaciones tendría un handicap al rellenarse, por el cual mediante una condición, filtra las puntuaciones obtenidas de las anteriores partidas y únicamente se queda con las cinco más altas.

Decidimos que fuera así, porque no nos parecía muy lógico guardar la información de todas las puntuaciones de las partidas que se habían jugado desde un principio. Ya que esto aumentaría el volumen de datos que se deben guardar drásticamente en el tiempo, cuando en nuestra opinión si no está entre las cinco mejores partidas, no es una puntuación relevante.

Una vez actualizada la lista de las mejores puntuaciones, las guardamos de forma serializada con la librería pickle que ya he explicado previamente en la documentación del proyecto. Utilizamos el método **guardar_puntuacion()** que en código se vería de la siguiente forma:

```
def guardar_puntuacion(self, nombre_archivo):
    """ Guarda la lista de puntajes en el archivo.
    Pre: nombre_archivo corresponde a un archivo válido,
        puntajes corresponde a los valores a guardar
    Post: se guardaron los valores en el archivo en formato pickle.
    """

    archivo = open(nombre_archivo, "wb")
    lista_Jugadores = []
    for i in range(len(self.lista_partidas)):
        #Añadimos objetos de jugador a la lista según la info de la lista de partidas
        lista_Jugadores.append(Jugador(self.lista_partidas[i].nombreJugador, self.lista_partidas[i].score))

    #Serializamos
    pickle.dump(lista_Jugadores, archivo)
    archivo.close()
```

Una vez que ya tenemos guardadas y serializadas las cinco puntuaciones más altas obtenidas desde el principio del juego, el objetivo es recuperar esas puntuaciones del archivo serializado, para mostrarlas en el ranking que aparece al cargar la ventana que gestiona el juego.

Para poder recuperar la información del archivo serializado, de nuevo, utilizamos otro método llamado **recuperar_puntuaciones()** que usa la librería **pickle** para poder deserializarlo y extraer la información que nos interesa. El código del método que gestiona este proceso sería el siguiente:

```
def recuperar_puntuacion(self, nombre_archivo):
    """ Recupera los puntajes a partir del archivo provisto.
        Devuelve una lista con los valores de los puntajes.
    Pre: el archivo contiene los puntajes en formato pickle
    Post: la lista devuelta contiene los puntajes en el
        mismo formato que se los almacenó.
    """

    archivo = open(nombre_archivo, "rb")
    lista_jugadores = pickle.load(archivo)
    archivo.close()
    return lista_jugadores
```

Como podemos ver en la captura de arriba, este método va a devolver una lista con la información de las cinco puntuaciones más altas de los jugadores. El único paso que quedaría es cargar el resultado de este método antes de iniciar la interfaz del juego y modificar las posiciones en la ventana, con los valores obtenidos del método cargar puntuaciones.

CONCLUSIONES

Como conclusión podemos discernir que tras haber completado con éxito nuestra misión, la cual es aprender y mejorar, nos hemos dado cuenta que aun habiendo sido costoso llevarla a cabo nos llena de satisfacción el ver culminada la obra que con ansia hemos esperado por dos largos años, verla por fin completa la cual nos ha llevado a romper barreras y sobrepasar nuestros propios límites.

Sobre el trabajo, nuestra principal y más voraz preocupación se ve calmada por el logro de la implementación funcional de un sistema de aprendizaje, la cual mediante un algoritmo es capaz de discernir el camino a tomar por la IA, otro gran problema es el sistema que se encarga de controlar la partida, la cual tiene en cuenta diversos factores de manera simultánea, como la captura de las teclas del teclado para modificar la dirección, el sistema que actualiza el mapa, el sistema que se encarga de aplicar las reglas del juego, otro pero no el último de los grandes problemas ya resueltos es el sistema de creación de los elementos que componen la partida, los cuales se crean teniendo en cuenta los demás elementos, de tal forma que en ningún caso no se superpone a otro

La tabla que tenemos seguidamente a este párrafo nos recoge una muestra de el número de comprobaciones [objetivos,obstáculos] que tiene que procesar el juego según las dimensiones que se hayan establecido y la dificultad que se haya escogido.

	Fácil	Medio	Difícil	Imposible
10*10	46,20	29,128 29,160	16,174	7,184
20*20	106,330	79,728	56,902	37,954
30*30	191,1340	154,2178 154,2286	121,2464	92,2674
40*40	301,3425	254,4853	211,5334	172,5719
50*50	436,6960	379,9128	326,9854	277,10464
100*100	1486,59510	1279,67628	1276,70329	1177,72814

La información de esta tabla es una de las razones por las que, debido a los pocos recursos con los que cuentan nuestros equipos, de vez en cuando el juego no llega a arrancar. Es una de los puntos más urgentes en los que estamos trabajando para próximas versiones.

Por otro lado, en cuanto a los problemas de la interfaz hemos tenido que diseñar desde cero, de tal forma que hemos necesitado tener en cuenta las dimensiones de la pantalla y de esta manera poder colocar los diversos elementos que la componen.

En cuanto a las observaciones, nos hemos dado cuenta que el proyecto aunque ambicioso ha sido factible gracias a ciertos sacrificios que han propiciado el éxito del proyecto. Sin embargo, con nuestros escasos conocimientos sobre el lenguaje nos hemos visto obligados a dirigir y a tomar ciertas decisiones como por ejemplo el sistema de ventanas que íbamos a utilizar, el algoritmo a implementar etc.
Para finalizar la conclusión cabe decir que el proyecto ha cumplido con las expectativas del equipo de programadores, dando lugar a la apertura de una puerta que nos conducirá a aprender más y mejor sobre el Machine Learning, deep learning, Neural Networks, Natural Language Processing, Natural Language Generation, Predictive Analysis, Python, Visual Studio Code y Pycharm entre muchos otros.

“Macbeth:But screw your courage to the sticking-place, And we'll not fail”

TRABAJO FUTURO

Tras haber concluido el proyecto, ambos miembros del mismo estamos de acuerdo en que sin ninguna duda se puede ampliar. Sentimos que en tan poco tiempo hay muchas cosas y funcionalidades que se nos quedan en el tintero, por lo que a continuación explicaremos las más importantes.

Comenzando en primer lugar por la interfaz que gestiona el juego. Como ya mencionamos anteriormente, decidimos darle un estilo más “clásico” para que pareciera que está ambientada en la década de los setenta cuando el juego se empezó a popularizar. Pero sin duda, estamos de acuerdo en que se podría mejorar el diseño visual de la interfaz añadiendo un vídeo de fondo en el que se pudieran ver algunas tomas de cómo se juega.

Hemos contrastado también que podríamos agregar algunos widgets a la interfaz gráfica que permitieran gestionar otro tipo de funcionalidades, como por ejemplo: la posibilidad de añadir más datos acerca del jugador entre otras.

Por otro lado, tras debatir bastante sobre este tema, decidimos que podría ser buena idea aumentar la posibilidades de personalización de los elementos del juego para el usuario. Por ejemplo, dejar elegir al usuario el aspecto de la serpiente con la que va a jugar, el aspecto de la recompensa o objetivo a alcanzar por la serpiente, el diseño visual de los obstáculos que ha de esquivar la serpiente etc.

Otra mejora interesante a tratar en un futuro, sería la posibilidad de poder jugar en batallas online de distintas magnitudes: uno contra uno, dos contra dos, tres contra tres etc. De esta forma, nos aseguraríamos de que va a haber más gente que opte por descargar el juego debido a este característica.

Por último, otra mejora muy significativa sería la inclusión de obstáculos en el modo de juego de la IA, y la posibilidad de poder regular el nivel al que quieres que se comporte la IA contra ti. De esta forma mejoraríamos la jugabilidad añadiendo más posibilidades al jugador, y sin duda sería otra de las características que atraería más usuarios a nuestro juego.

BIBLIOGRAFÍA

Canal de Youtube de Dotcsv, enfocado a resolver problemas de inteligencia artificial:
https://www.youtube.com/results?search_query=dotcsv

Vídeo de Dotcsv, explicando cómo elaborar el juego del que hemos cogido inspiración para nuestro proyecto:
https://www.youtube.com/watch?v=qPtKv9fSHZY&t=305s

Documentación oficial de la librería pygame utilizada en nuestro proyecto:
http://www.pygame.org/docs/ref/display.html

Massachusetts Institute of Technology(Curso introductorio del MIT sobre métodos de aprendizaje profundo con aplicaciones para visión por computadora, procesamiento del lenguaje natural, biología y más):
http://introtodeeplearning.com/

Documentación oficial de la librería tkinter utilizada en nuestro proyecto:
https://docs.python.org/es/3/library/tk.html

Documentación oficial utilizada para aprender a usar git:
https://git-scm.com/doc

Curso de python desde cero de píldoras informáticas:
https://www.youtube.com/watch?v=G2FCfQj-9ig&list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS

Tutorial para principiantes de la librería numpy:
https://www.youtube.com/watch?v=QUT1VHiLmmI

Tutorial enfocado a aprender a usar pycharm:
https://www.youtube.com/watch?v=hc50ALh_x5g