Bots vs humans: Spelunky

Borja Rodriguez

Resumen—Este proyecto es la creación y evolución de una inteligencia artificial para el juego llamado Spelunky. Este es un juego 2D de plataformas en el que se tienen que sortear varios obstáculos (trampas, desniveles de terreno, objetos) para llegar al final del nivel y conseguir una puntuación. Se empieza creando unas inteligencias preprogramadas donde se especifica que ha de hacer el jugador en cada situación del nivel para seguir adelante, después se crea una inteligencia que añade ruido aleatorio a las decisiones del jugador para hacerle salir de situaciones en las que la inteligencia se queda trabada o no sabe salir. Por último se crea una inteligencia que auto-aprende basada en redes neuronales evolutivas, se usaran redes neuronales aleatorias para hacer jugar al personaje y se les dará una puntuación según su acción en el juego, con estas redes y según sus puntuaciones crearemos otra generación de redes neuronales basadas en las primeras, dando favor a las de mayor puntuación y dando una probabilidad de que estas redes se modifiquen aleatoriamente, de esta forma a cada generación tendremos unas redes neuronales que actúan de una forma más propicia a conseguir una puntuación mayor.

Palabras clave—Inteligencia artificial, Spelunky, bot, botprize, redes neuronales, redes neuronales evolutivas, algoritmos genéticos, juego, IA, SNNS, ANN.

Abstract—This project is the creation and the evolution of an Artificial Intelligence of the game named Spelunky. This is a 2D platform game where you have overcome diferent obstacles (traps, uneven terrain, objects) to arrive to the end of the level and obtain a puntuation. It begins by creating a preprogrammed intelligences which specifies who has to make the player in each situation the level to move on after an intelligence that adds random noise to the player's decisions for him out of situations is created in which intelligence is locked or can not leave. Finally an intelligence that self-learning based on evolutionary neural network is created, random neural networks were used to play the character and will be given a score according to their action in the game, with these networks according to their scores and create another generation of neural networks based on the first, giving favor to the highest scoring and giving a chance that these networks are changed randomly, thus each generation we neural nets that act in a manner more conducive to get a higher score.

Index Terms— Artificial intelligence, Spelunky, bot, BotPrize, neural networks, evolutionary neural networks, genetic algorithms, game, IA, SNNS, ANN.

1 Introducción

ESTE proyecto fue propuesto para realizar un trabajo similar al que se ha realizado anteriormente en las competiciones llamadas botprize [5] que han tenido lugar años anteriores. Estas competiciones se basaban en el juego Unreal Tournament [6] los participantes tenían que crear una inteligencia artificial que se comportara lo más parecido posible a como lo haría un humano, después unos jueces intentarían identificar por medio de combates entre un humano jugando y un bot, quien era cada uno. El bot ganador seria el que consiguiese engañar mejor al juez. En este caso se intenta realizar una inteligencia artificial del mismo modo pero en este caso usando el juego Spelunky [4] donde la inteligencia artificial tendrá que intentar pasarse los niveles con una puntuación máxima usando los recursos que tiene al alcance. Esto se basa en el independent studi realizado por Daniel Scale [2] que creo unas herramientas para crear inteligencias artificiales para este juego.

Como objetivos del proyecto nos proponemos:

1) Familiarizarse con GameMaker, spelunky y su API para crear bots basados en AI.
   a) Situar los archivos básicos del código de SpelunkyBots.
   b) Comprender el funcionamiento de Spelunky.
   c) Comprender la interacción entre el bot y el juego.
   d) Reconocer las funciones básicas de reconocimiento del medio y de comportamiento del bot.

2) Crear bots que realicen tareas básicas
   a) Adaptar el código de los bots para una posible implementación de IA en el futuro.
   b) Crear nuevas funciones necesarias para crear bots más complejos.

- *E-mail de contacte: Borja.rgez@gmail.com*
- *Menció realitzada: Computació.*
- *Treball tutoritzat per: Felipe Lumbreras (CVC)*
- *Curs 2014/15*

3) Crear bots que simulen comportamiento inteligente (con objetivos concretos, tiempo, puntuación,. . .)
   a) Utilizar SNNS [9] para ejecutar redes evolutivas.
   b) Comprobar que parámetros dentro de la IA hacen que el bot sea más inteligente.
   c) Entrenar el bot para que auto-aprenda y evolucione en las forma de actuar en verso al medio que le rodea.

4) Participar en la competición.

Estos objetivos han sido modificados a lo largo del proyecto, anteriormente hubiéramos tenido que hacer pruebas para decidir qué tipo de inteligencia artificial hemos hubiéramos tenido que usar, pero gracias al profesor Juan Peralta [6] nos decantamos por usar redes neuronales genéticas usando las librerías SNNS, ya que participo en los antes mencionados botprize y por su experiencia entendió que eran los más adecuados y nos recomendó su uso.

Actualmente la industria de los videojuegos está muy desarrollada y se producen muchos juegos al año. Esta industria para triunfar se basa en la calidad del juego y la experiencia que proporciona al jugador. Los juegos han de ser un reto para los jugadores y para conseguir esto las productoras contratan a personas encargadas de jugar a las versiones iniciales de los juegos y reportar tanto los bugs cómo la calidad del juego dentro de estos. Pongámonos en la situación de que en esta misma industria se proporcionase las herramientas necesarias para que estos juegos sean evaluados por maquinas en vez de por personas. Si se crearan herramientas de inteligencia artificial para hacer esto posible las compañías de videojuegos ahorrarían tanto en dinero como en tiempo. Si en cada nuevo juego, los creadores de este se encargan de crear una ecuación de éxito que aportase un dato real de cómo de bien un jugador ha reaccionado dentro del juego, se podrían aplicar estas ecuaciones a máquinas de inteligencia artificial que jugarán al nuevo juego y obtener una relación entre al dificultad que ha tenido la máquina para jugar el juego con la puntuación que haya sacado y así crear una valoración objetiva, con datos experimentales de cómo de difícil será el juego para los jugadores reales.

Este tipo de idea podría tener más aplicaciones dentro de esta industria, actualmente la dificultad de los juegos es elegida por el jugador dentro de un menú, pero, ¿podríamos imaginar un mundo donde la experiencia de juego fuese más personalizada? Teniendo estas inteligencias artificiales basadas en aprendizaje, y las ecuaciones de evaluación podríamos crear un mundo virtual en que según la evaluación de éxito del jugador podríamos hacer que las inteligencias artificiales de los personajes del juego como los enemigos adaptasen su forma de actuar según la habilidad del jugador. Si el jugador evolucionase a lo largo de los niveles y su forma de jugar mejorase o cambiase de forma, los enemigos se adaptarían a este nuevo juego para hacer la experiencia del jugador se volviese un verdadero reto. Esta forma de crear juego es un sistema evolucionado de la inteligencia artificial llamada Némesis System [10] utilizada en el juego "Shadow of mordor" que salió a la venta este último año. Este sistema de inteligencia artificial, hace que los enemigos creador por la inteligencia artificial tengan su propia personalidad y recuerden batallas anteriores en las que se ha enfrentado al jugador. Este sistema ha seguido evolucionando y puede hacer que las inteligencias artificiales adapten su forma de combatir dependiendo de los enfrentamientos anteriores.

Cómo experiencia para los jugadores o los desarrolladores, se podrían dar facilidades para que en los juegos cada uno pudiera crear su propio bot que jugase en el juego dando un sistema de inteligencia artificial y publicando la ecuación de éxito en el juego. Gracias estas herramientas se podría evaluar la habilidad de los jugadores para crear sus propios bots que ayudarían al jugador o lo sustituirían para ponerse al mando del personaje y sacarle de situaciones complicadas o seguir jugando mientras el jugador no está delante de la pantalla, ya sea para subir de nivel, conseguir dinero, u otras cosas que podrían adaptarse según el juego.

Con este proyecto se pretende dar comienzo a un mundo de posibilidades donde todas estas cosas puedan ocurrir. Empezando por crear herramientas para poder ejecutar inteligencia artificial que pueda aprender en el juego y evolucionar y donde no se pueda diferenciar el estilo de juego de un bot al de un humano, y no se puedan identificar.

En este documento se explicara cual ha sido la evolución del proyecto, cuál es su funcionamiento y cuál es el estado actual en que se encuentra. También daremos a conocer la metodología usada y los resultados que hemos obtenido con esta metodología. Por último se expondrán las conclusiones según estos resultados y cuál es el trabajo que se podría hacer en un futuro para mejorar este proyecto o seguir avanzando.

2 Estado del arte

Desde que se crearon los primeros bots sus creadores han querido comprobar lo parecidos que eran los comportamientos o reacciones de un bot, al de un humano.

Para hacer esto se han creado concursos en que las máquinas compiten para hacer creer a otros humanos que ellas son humanas, uno de los casos de ejemplo es "Turing Test 2014 Prize" [7] que se trata de que los bots participantes tienen que convencer a personas de que ellos son humanos en una conversación de chat. Como este ejemplo, se ha realizado varios años seguidos un concurso llamado "BotPrize" [5] que se trataba de intentar engañar a humanos intentando hacer que unos bots jugaran inteligentemente a un juego llamado "Unreal Tournament"[6].

De este último concurso proviene la idea de realizar un bot que se pase un juego, ya que actualmente se está inten-

tando crear un nuevo concurso de este tipo, esta vez basado en el juego Spelunky [4]. Spelunky es videojuego indie de acción y aventura creado por Derek Yu [1] e initheir own player bots" [2]. Este estudio a la vez está basado en querer crear unas herramientas básicas para que usuarios puedan crear bots para el juego de Spelunky. El juego

Figura 1: Uno de los nivel del juego Spelunky.

cialmente lanzado como freeware para Microsoft Windows. Luego fue relanzado para Xbox 360, PlayStation 3, PlayStation Vita y nuevamente para Microsoft Windows. En Spelunky, el jugador controla a un espeleólogo que explora una serie de cavernas en 2D coleccionando tesoros, salvando damiselas y evadiendo trampas. La primera versión del juego fue lanzada el 21 de diciembre de 2008. El código fuente de la versión de 2008 de Windows fue hecho público el 25 de diciembre de 2009. La versión mejorada para Xbox Live Arcade fue lanzada el 4 de julio de 2012. Esta versión también fue lanzada en PC el 8 de agosto de 2013 y para PlayStation 3 el 27 y 28 de agosto de 2013.

Actualmente existen varios bots creados con estas herramientas que realizan tareas básicas cómo matar un tipo de enemigo especifico, coger barras de oro o utilizar algún objeto en especial. Basándome en estos bots y añadiendo IA se ha conseguido hacer que el bot aprenda por su propia experiencia en los niveles y consiga tener un comportamiento más parecido al de los humanos.

3 Marco

Bot vs humano es un proyecto basado en un estudio independiente de Daniel Robert Scale llamado "Create a set of AI Tools for Spelunky that will allow users to program

está hecho en GameMaker, basado en C++. Por lo tanto, este proyecto se realizará usando el programa GameMaker, usando como código base el proyecto de Daniel Robert Scale aprovechando las herramientas que aporta su trabajo para crear bots para Spelunky. El leguaje usado para crear los algoritmos que serán los bots, es C++ para poder usar las funciones y el código de Spelunky que facilitará la interacción entre el bot y el entorno.

Para generar las redes neuronales y utilizarlas, utilizaré la librería SNNS [9]. Este está formado de un conjunto de ejecutables que con los ficheros adecuados generarán las redes en forma de ficheros que posteriormente se usarán con otro ejecutable para conseguir las salidas a través de los inputs. El uso de estas librerías fue recomendado por el profesor Juan Peralta Donate que ya conocía el uso de estas por el proyecto de un alumno llamado Joan Marc Llargues Asensio [8].

El hecho de trabajar con ficheros, para poder trabajar con el lenguaje C++ se tendrán que crear diferentes parsers para leer estos ficheros y escribirlos, lo que ralentizará la ejecución del código. Estos archivos son archivos de texto sin nigun tipo de codificación expecifica.

4 METODOLOGÍA

Como anteriormente se había dicho se utiliza la librería SNNS para ejecutar las redes neuronales creadas y con un input generar el output esperado. Las IAs usadas serán unas redes neuronales que se generarán aleatoriamente, se ejecutarán en el juego y se les asignarán una puntuación de éxito. Una vez hecho esto se creará una siguiente generación de redes basadas en las de la anterior generación, y así cada generación tendrá unas IAs con mayor probabilidad de obtener mayor puntuación de éxito.

4.1 Redes neuronales

Las redes neuronales se tratan de un sistema de interconexión de neuronas que colaboran entre sí para producir un estímulo de salida. Estas neuronas son llamadas perceptrones, estos perceptrones como se ve en la figura 1, tienen uno inputs, y se introducen en una función de activación si el resultado de esta función llega a la neurona se activa. Si esto lo extendemos una red, las neuronas se interconectan entre ellas, dando cómo entrada de las neuronas del siguiente nivel las salidas del nivel anterior, estas conexiones tienen unos pesos asociados que deciden cuanto valorará la siguiente neurona la entrada por esa conexión.

De la librería SNNS [9] usaremos dos ejecutables que cumplen dos funciones diferentes, el ffbignet, que crea la red inicializada con todos los pesos y bias a 0, y la función batchman que con un fichero llamado por esta función tiene los pasos a seguir por este ejecutable que en nuestro caso será, cargar la red que queramos usar, ejecutarla con los inputs que tenemos y generar los outputs.
Para inicializar las redes de forma aleatoria para la primera generación leeremos el archivo que contiene la forma de la red y generaremos los bias y pesos de las neuronas de forma aleatoria y las volveremos a guardar en el mismo fichero para poderlas usar posteriormente con el batchman. El archivo que representa la red, se llamara con el nombre de la red y con la extensión .net.

```
SNNS result file V1.4-3D
generated at Sat Feb 07 12:05:03 2015

No. of patterns    : 1
No. of input units : 27
No. of output units : 10
startpattern       : 1
endpattern         : 1
input patterns included
#1.1
0 0 0 0 0 0 0 0 0 0.18182
0 0 0.18182 0 0 0.18182 0 0 0 0
0 0 0 0 0 0 0
0.28757 0.67521 0.52071 0.65446 0.91462 0.08729
0.52558 0.90152 0.21487 0.57341
```

Figura 3: Formato de los archivos file.in y file.out. Contiene el número del patrón, el número de entradas y el número de salidas. Las líneas finales indican los valores de las entradas en los primeros 27 números y las salidas en los últimos 10 del documento.

El ejecutable batchman será ejecutado con un archivo llamado ANN.script que contiene el nombre del archivo de nuestra red, y las instrucciones para cargar esta red. Este archivo contiene todas las instrucciones sobre cómo actuar en con las redes, en este archivo primero se especificarán los nombres de los archivos que serán utilizados para la ejecución, después se inicializarán las variables necesarias para las redes, como el tipo de red que en este caso será una de backpropagation. Con los archivos especificados y las variables inicializadas se cargara la red desde su documento y se cargara el archivo de inputs file.in, que contiene los inputs, que en este caso serán 27 doubles con la información de las casillas que envuelven al jugador, y se ejecutará la función testNet que utilizando la red cargada generará un archivo llamado file.out donde nos devolverá los inputs que habíamos usado seguidos por los outputs generados que en este caso son 10 doubles. Tanto el archivo file.in como file.out tienen el mismo formato y están basados en el mismo tipo de plantilla como el que se muestra en la figura 3. La posición en el vector del máximo de esto doubles es el que nos indicara el movimiento que realizará el jugador.

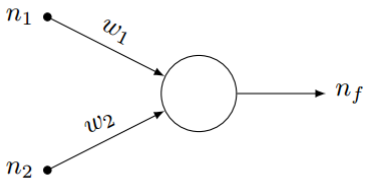

Figura 2: Neurona artificial i funcionamiento con entradas n_1, n_2, pesos w_1, w_2, y salida n_f.

4.2 Evolución genética

La evolución se basa en los algoritmos genéticos que son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

La forma en cómo evolucionarán las redes por cada generación será usando: mutación, reproducción y elitismo. La forma en que se usarán las redes y estos métodos para generar la siguiente generación es la siguiente.

1) Primero generaremos la primera generación de forma aleatoria.
2) Usaremos estas redes en el juego y les daremos una puntuación a cada una según su actuación

dentro de juego, se basara en el tiempo de vida, el número de vidas que tiene, el oro que consiga y si acaba el nivel o no.

3) Las dos redes con mayor puntuación pasaran a la siguiente generación (Elitismo).
4) Entre todas las redes de la primera generación las reproduciremos entre ellas, escogeremos dos redes aleatorias y las partiremos por un punto aleatorio y mezclaremos la primera parte de una red con la segunda de la otra y a la inversa para crear dos hijos de estas redes. (Reproducción, método sobre-cruzamiento)
5) Después para cada red la siguiente generación, cogeremos por cada bias y cada peso de estas redes y le daremos un 10 por ciento de probabilidades de que mute por otro número aleatorio. (Mutación)
6) Con esto tendremos la siguiente generación y repetiremos todos los pasos para cada nueva generación desde el segundo.

4.3 Calculo de la puntuación de éxito

Para obtener la puntuación de éxito (*Pe*) que nos dirá que IA tiene más éxito que otras, se utilizarán varias variables, como son, el tiempo en que el jugador se mantenga vivo, la puntuación interna del juego (basada en el oro recogido), la cantidad de vida que le queden y si llega al final de la partida o muere en el proceso. Con lo que la fórmula que quedaría sería el sumatorio de los segundos de vida el número de vidas (se empieza con 4) y la cantidad de oro recogida, aparte este valor lo multiplicaremos por 10 si llega al final del nivel.

$$Pe = (gold + lifes + time) * 10 \; if \; finish$$
$$\text{O}$$
$$Pe = (gold + time) \; if \; not \; finish$$

Ecuación 1: Ecuación de éxito utilizada para evaluar las inteligencias artificiales.

4.4 Flujo de ejecución

El flujo de ejecución con todos unido seria el siguiente:

1) Primero generaremos 20 redes neuronales aleatorias, esto solo se ejecutara una vez.

2) Ejecutaremos el juego 20 veces con cada una de las redes neuronales y en el juego se les asociará una puntuación de éxito. Por cada ejecución, el juego leerá la red neuronal, usará batchman para obtener los outputs por cada paso que haga, según los inputs que recoja.

3) Una vez tengamos la puntuación de todas las redes, se usaran el elitismo, la reproducción y la mutación para crear la siguiente generación de redes neuronales.

4) Se ejecutarían las redes de esta nueva generación en el juego, y se volvería a repetir el paso 3 con las nuevas puntuaciones obtenidas.

Por cada generación obtendremos unas redes más inteligentes, y al final de esta obtendremos 20 nuevas redes neuronales con toda la información de sus pesos y bias en el fichero .net. Aunque se pare la ejecución nunca se volvería a empezar el entrenamiento de cero, si no que empezaremos desde la última generación conseguida. Se empezarían las ejecuciones de las generaciones más jóvenes con mapas sencillos, solo con puertas y desniveles, a cada nueva generación se le irán poniendo mayores dificultades como enemigos, trampas u objetos.

En este caso el juego no deja automatizar este proceso del todo con lo que cada ejecución de red requiere interacción humana. Cuando una red neuronal muere o acaba el nivel el juego se queda en un menú de pausa en el cual se ha de presionar la tecla escape para volver al menú de niveles. Cuando seleccionamos el nivel de test (epecificamente creado para las pruebas), el juego carga la siguiente red siguiendo el orden y en caso de haber ejecutado la última red generará la nueva generación.

Acciones básicas
global.goLeft
global.goRight
global.jump
global.duck
global.lookUp
global.attack
global.running
global.payp (pay in a shop)
global.ropep (release a rope)
global.bombp (release a bomb)

Tabla 1: Tabla de variables que definen las acciones básicas que puede realizar el jugador.

5 Resultados

La ejecución del código necesita de interacción humana para ir cargando los niveles, y el hecho de tener que hacer ejecuciones en la terminal de Windows desde el juego hace que la ejecución sea lenta. Aparte de esto el juego solo se ejecuta en primer plano, con lo que si las maquinas son usadas para otro fin y no está el juego ejecutándose en primer plano este no continua con la partida y deja de ejecutarse el código. Al día tan solo se pueden ejecutar una media de tres redes, con lo que con una generación de 20 redes neuronales como se había establecido para el experimento tan solo hemos llegado a generar dos generaciones. Con tan pocas redes y tan pocas generaciones no se pueden sacar conclusiones sobre la forma de generar las redes y las generaciones.

Para realizar los experimentos se ha creado un nuevo nivel sobre el que evaluar a todas las redes que se observar en la figura 4. Este nivel tiene diferentes pruebas sobre las que la inteligencia artificial tiene que reaccionar, se les ha puesto dos enemigos contra los que enfrentarse tres monstones de oro y una trampa de pinchos que saltar, aparte se ha añadido un desnivel en frente de la salida para obstaculizar su avance. Se ha evaluado a todas las redes sobre el mismo nivel para que la puntuación se basase en un mismo entorno y tener las mismas condiciones y probabilidades de supervivencia. De este modo muchas inteligencias obtienen la misma puntuación puesto que el resultado final de sus acciones deriva en el mismo tiempo de vida.

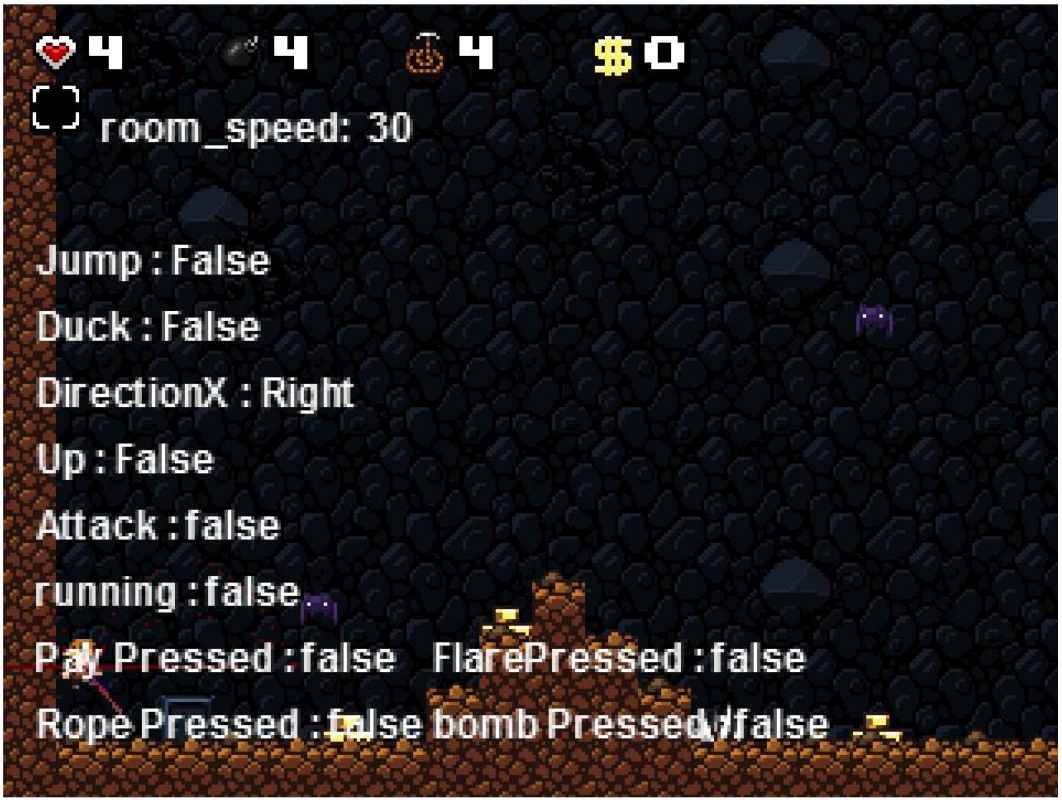

Figura 4: Nivel de prueba utilizado para evaluar a las redes neuronales con el bot ejecutándose.

Los jugadores con el tipo de redes que se utilizan generadas de manera aleatoria, solo comprenden las acciones básicas nombradas en la tabla 1. Solo ejecutarán una acción y si esta acción no modifica la situación del jugador no se modificarán los inputs y seguirá generando los mismos outputs, y el jugador seguirá en la misma situación algo de su entorno cambie. En el caso de que el entorno se modifique, los inputs cambiarán pero no se modificarán los outputs necesariamente. Si el cambio producido es en un nodo de la red neuronal que no tiene unos pesos altos, este no modificará la salida. Para evitar que la inteligencia artificial se quede trabada en la misma posición ejecutando la misma acción durante un tiempo infinito, cada cierto tiempo se le restará una vida. Así nunca habrá una inteligencia que se quede infinito tiempo trabado en el mismo tiempo.

Las redes aún están en entrenamiento con lo que se puede deducir que con más tiempo de entrenamiento. También si se utilizarán métodos más complejos de aprendizaje estos harán evolucionar las redes de forma más dirigida al objetivo final.

Como resultado final podríamos decir que a falta de tiempo seguir experimentando con las redes el proyecto, se ha conseguido crear unas herramientas para introducir redes neuronales evolutivas dentro del juego de Spelunky haciendo que así otras personas puedan usar este proyecto para crear nuevas pruebas y experimentos para adaptar estas redes a un uso más afinado para el juego, modificando en sus experimentos, las probabilidades de reproducción, de elitismo o de mutación y probando diferentes estrategias de juego.

Estas herramientas no solo sirven para generar otros proyectos si no que se abre nuevas posibilidades para que personas puedan crear sus propios bots que auto aprendan y poder usarlos en concursos u otras finalidades.

6 Conclusiones

Finalmente se entiende que para tener mejores resultados sería necesario más tiempo de entrenamiento, aunque el tiempo necesario para mejorar cambia de manera exponencial, es decir se necesitan cada vez más generaciones para notar cambios en el comportamiento del jugador.

También se puede ver que la forma en que se evalúa a la red neuronal se podría mejorar encontrando un punto intermedio en que cuente la cantidad de tiempo que vives en relación a si acabas el juego, es decir, no es bueno que una red acabe el juego muy rápido sin conseguir dinero pero tampoco es bueno que tarde mucho tiempo en acabar el nivel si no tiene nada más que recoger o hacer en el mismo nivel.

Cómo se comentaba en la introducción si por parte de los creadores del juego se hubiera liberado una ecuación de puntuación de éxito según ellos crean se tiene que evaluar a un jugador, las inteligencias artificiales evolucionarían a un estilo de juego dirigido a cada tipo de juego.

Si el juego se pudiera ejecutar en segundo plano y no necesitase de interacción humana para ejecutarse, el tiempo de ejecución se hubiera visto reducido de tal forma que se hubiera podido hacer un experimento más extenso sobre la evolución de las redes a través de las generaciones y el cambio en sus ejecuciones.

Haber creado estas herramientas para crear bots con inteligencia artificial que auto-aprendan, pone inicio a lo comentado en el inicio del documento. Estas herramientas pueden ser usadas no tan solo en la inteligencia de artificial del bot que maneja al jugador sino que también pueden ser usadas para los enemigos y que estos se adapten a cada jugador dependiendo de la experiencia de este en el juego.

6.1 Estado del proyecto

El proyecto ha sido finalizado en los términos de tiempo establecidos y ha cumplido con las fechas previstas hasta el momento. Aunque haya sido finalizado es un proyecto que a medida que es ejecutado funciona mejor con lo que se podría decir que se puede seguir usando para llegar a un estado óptimo donde el jugador sepa pasarse los niveles de forma eficiente con puntuaciones máximas.

También este proyecto puede seguir evolucionando adoptando otro tipo de estrategias, por lo que, podríamos decir que este proyecto podría servir como base para otros proyecto basados en esta técnica. Usando el mismo algoritmo y las mismas funciones se podría llegar a crear bots con diferentes comportamientos, cambiando la forma de inicializar las redes y modificando la fórmula de puntuación para estas mismas, también se puede cambiar las probabilidades de mutación, tanto como la forma de reproducirse y la proporción del elitismo.

7 Futuro trabajo

Como futuro trabajo se podría investigar otras maneras de relacionar el juego con las redes y la forma de evaluarlas, es decir cambiar la estrategia utilizada para avanzar en el juego. De forma que si ahora usamos solo las casillas más cercanas, con unas redes más complejas se podría usar el conocimiento que tenemos sobre todo el mapa y evaluar las redes según la distancia que tenemos a unos objetivos fijados.

También se puede avanzar mirando de evaluar la cantidad de neuronas ocultas que tenemos y buscando el número que mejor se adapta al juego, de forma que tengamos unos mejores resultados. También se podría hacer evolucionar un poco a las redes neuronales de forma individual antes de poderlas a evolucionar generacionalmente, con lo que tendríamos una primera generación con más conocimiento y las siguientes generaciones darían mayor resultado.

Para acelerar las ejecuciones, se podría usar algún tipo de librería de redes neuronales o implementar unas funciones propias para poder cargar en memoria las redes y ejecutarlas sin hacer llamadas al terminal del sistema. De esta forma las herramientas creadas para crear bots que usan inteligencia artificial de auto-aprendizaje podrían llevarse a cualquier otro juego que haya sido creado con C++ o que se base en este lenguaje. Con esto se abrirían muchas posibilidades y tan solo se necesitaría crear una ecuación de éxito para cada juego y un decidir las probabilidades de reproducción, elitismo o mutación. De esta forma para cada juego se podrían crear diferente experimento, por ejemplo, según la dificultad del juego cuantas generaciones necesitaríamos para crear un bot que se pasase los niveles, probar con las mismas redes y las mismas generaciones probar diferente s estrategias en un juego para ver cuál de ellas es la que más beneficios nos aporta o probar con diferentes tamaños de generación y diferentes probabilidades para averiguar de qué forma evolucionan más rápido.

Agradecimientos

Agradezco la ayuda del profesor Juan Peralta Donate por los consejos y las explicaciones que me ha dado a lo largo del proyecto sobre las redes neuronales basadas en su experiencia personal en los proyectos que ha realizado a lo largo de su carrera. Doy gracias también al profesor Felipe Lumbreras por el seguimiento en el proyecto y las horas pasadas en su despacho para solucionar problemas acaecidos en el código y la atención prestada a lo largo de todo este tiempo, gracias a él también pude ponerme en contacto con el profesor Juan Peralta. Gracias al trabajo de Joan Marc Llargues Asensio por su trabajo y atención que atendió las peticiones para poder acceder a su trabajo y poder aprender sobre su proyecto. Gracias también a mi pareja Anna Castaño por su ayuda en la vida cotidiana mientras yo realizaba este proyecto.

BIBLIOGRAFÍA

[1] Mossmouth, LLC: Página oficial del Juego Spelunky, 2009-2012.

[2] Daniel Robert Scale: Create a set of AI Tools for Spelunky that will allow users to program their own player bots, University of Derby : School of Computing and Mathematics,2013-2014

[3] Tommy Thompson: Getting Started With SpelunkBots, Retrieved 2014.

[4] Tommy Thompson: Página de Spelunkybots API, Retrieved 2014.

[5] 2KAustralia: Página de The 2K BotPrize, Retrieved 2014.

[6] G.Bedia, Manuel; Arrabales, Raúl; Peralta Donate, Joan; Hingston, Philip: Página de 2014 BotPrize competition, Retrieved 2014.

[7] University of reading: Página de Spelunkybots API, Release Date 08 June 2014.

[8] Joan Marc Llargues Asensio, Juan Peralta, Raul Arrabales, Manuel González Bedia, Paulo Cortez, Antonio López Peña: "Artificial Intelligence approaches for the generation and assessment of believable human-like behaviour in virtual characters". Expert Systems with Applications 41 (2014) 728187290

[9] University of Stuttgart: Stuttgart Neural Network Simulator, User Manual, Version 4.2

[10] Fernandez, Ricardo: "Así es el Sistema Némesis de Tierra Media: Sombras de Mordor", June of 2014.