# Laboratorio #2
# Sonic NEAT.


Universidad de Panamá, Facultad de Informática, Electrónica y Comunicación.

| | |
|---|---|
| Roderick Aparicio | 8-916-593 |
| Isaac Bethancour | 8-917-2263 |
| Alan Castro | 8-912-1890 |
| Jesús De Gracia | 8-1086-1646 |
| Efrain Escobar | 8-772-1697 |
| Guillermo Espino | 8-925-2235 |
| Allan Marín | EC-20-12127 |
| Elvin Marín | EC-20-12126 |
| David Morán | 2-741-87 |
| Lourdes Moreno | 8-920-640 |

| | |
|---|---|
| Daniel González | 8-907-1404 |
| Kayser Obaldia | 8-898-703 |
| Yeny Ortega | 8-923-1263 |
| Fermin Povaz | 8-932-1661 |
| Kevin Ruedas | 8-910-1800 |
| Krisbel Sanjur | 2-738-644 |
| Alexander Santana | 8-933-1167 |
| Stephanie Tejeira | 8-924-2239 |
| Victor Valenzuela | 8-931-2246 |
| Ana Villarreal | 8-917-2049 |

*Abstract* - The following article presents the project: Sonic NEAT, where we will use the NEAT genetic algorithm to train an AI in a video game developed in Unity. We will also use a tool that Unity provides us, which is the Navigation section, for the use of artificial intelligence, specifically intelligent agents.

*Resumen* - En el siguiente artículo se presenta el proyecto: Sonic NEAT, donde utilizaremos el algoritmo genético NEAT para el entrenamiento de una IA en un videojuego desarrollado en Unity. También utilizaremos una herramienta que nos proporciona Unity que es la sección de Navigation, para el uso de inteligencia artificial, específicamente el agentes inteligentes.

*Palabras claves* - Algoritmo genético, NEAT, Python, C#, Unity, Agente, NavMesh.

## *I - Introducción*

El concepto de aprendizaje a través de la evolución también se puede aplicar a la Inteligencia Artificial. Podemos entrenar a las IA para realizar determinadas tareas utilizando NEAT, Neuroevolution of Augmented Topologies. En pocas palabras, NEAT es un algoritmo que toma un lote de IA (genomas) que intentan realizar una tarea determinada. Las IA de mejor rendimiento "se reproducen" para crear la próxima generación. Este proceso continúa hasta que tenemos una generación que es capaz de completar lo que necesita.

En el caso de los agentes inteligentes son como una entidad software que, basándose en su propio conocimiento, realiza un conjunto de operaciones destinadas a satisfacer las necesidades de un usuario o de otro programa, bien por iniciativa propia o porque alguno de éstos se lo requiere. Todos los agentes inteligentes son programas, pero no todos los programas que realizan búsquedas son agentes inteligentes.

## *II - Objetivos*

El objetivo de este proyecto consiste en la creación de un agente inteligente que le permita al personaje "Sonic" moverse libremente y de manera autónoma en el mapa para poder, de esa manera completar el nivel o stage 1.

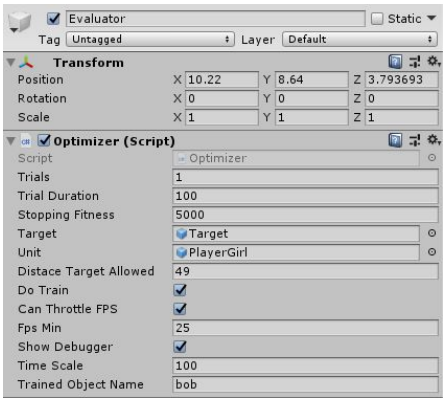
## *III - Metodología*

### Metodo SharpNEAT

Utilizamos el motor de videojuegos Unity para la carga del mundo y escena que pertenece a un videojuego con el nombre Sonic, el cual utilizaremos para el proyecto.

Aplicamos el algoritmo NEAT para la evolución de redes neuronales de una inteligencia artificial, la implementación que utilizamos fue SharpNEAT, el cual provee de mecanismos de mutaciones, metas y objetivos a cumplir[1].

En el caso de la plataforma Unity existe un port de SharpNEAT, conocido como UnityNEAT[2].
Para empezar, añadimos el objeto Evaluator el cual utilizará el script Optimizer para controlar las generaciones de redes neuronales y su comportamiento en la búsqueda del mejor camino.

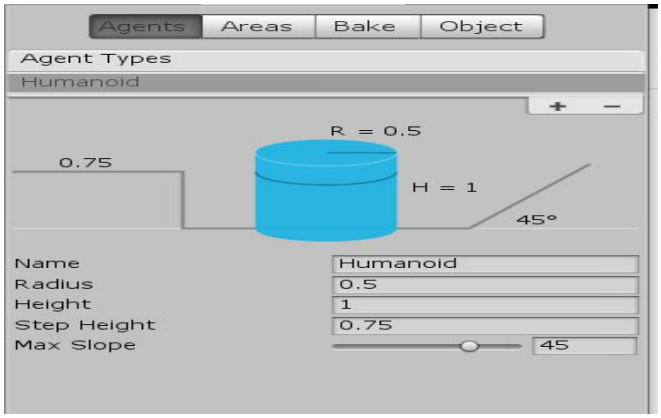



Luego correremos el juego en la plataforma y empezamos el entrenamiento por medio del botón Start EA.


Se generarán copias del modelo de Sonic para moverlo hacia su objetivo y así empezar el entrenamiento.

### Metodo NavMesh

Se utilizó la metodología NavMesh que es una colección de polígonos convexos bidimensionales que define qué áreas de un entorno pueden atravesar los agentes. En otras palabras, un personaje en un juego podría caminar libremente dentro de estas áreas sin obstáculos por árboles, lava u otras barreras que son parte del entorno. Los polígonos adyacentes están conectados entre sí en un gráfico.



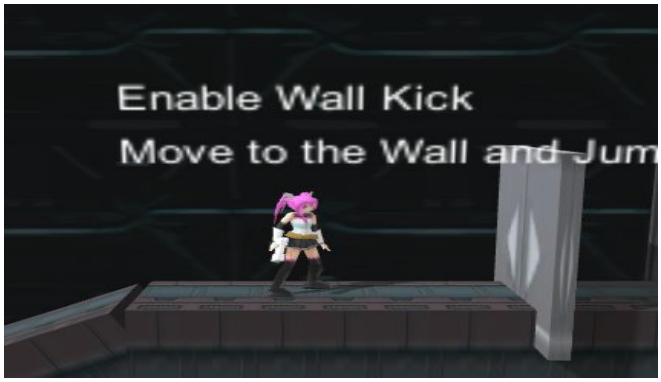
Este método nos permitió personalizar nuestro agente, como deseamos, posee las opciones de radio, altura y muchas más opciones. Pero lo más sorprendente es que se utilizando un script que con la función GetComponent<NavMeshAgent>(); y creando una variable para la posición e igualarla al objetivo al cual se quiere llegar, el personaje se mueve según las configuraciones que uno simplemente, hacía el objetivo, que en nuestro caso fue una puerta, siempre y cuando en el objetivo esté colocado "Navigation Static" y se haga un bake para que se marquen los objetos y el área donde debe pasar el personaje, llegará a su destino.


## *IV - Análisis de Resultados*


Copias de Sonic generadas por el algoritmo SharpNEAT

El algoritmo NEAT buscó correctamente enemigos y objetivos con el tag "Enemy" e incluso seguía sus movimientos correctamente, en algunas ocasiones, unas copias de Sonic se iban a buscar otros enemigos en vez de quedarse con el enemigo más cercano.
Colocar el tag "Target" en algún objeto colocado en el mapa no causaba ningún movimiento a pesar de los intentos de cambios de objetivo o búsqueda de entidades con algún tag.

Con el NavMesh el personaje buscó el objetivo asignado con completo éxito, las pruebas que realizamos fueron, colocar una puerta y asignarla como objetivo al personaje y hacerle un bake a la puerta para que el personaje lo reconociera y no se fuera de largo, la prueba se realizó en diferentes escenas para que se viera el funcionamiento del agente y de cómo llegaba hasta la puerta, sin importar la distancia que tuviera que recorrer. Lo trabajamos son obstáculos para que se pudiera apreciar el funcionamiento.




Personaje detectando la puerta con el Navmesh

## *V - Conclusión*

NEAT a menudo llega a redes efectivas más rápido que otras técnicas neuroevolutivas contemporáneas y métodos de aprendizaje por refuerzo. No conocíamos la metodología de NavMesh pero logramos hacer que el personaje lograra llegar al objetivo que colocamos. Si hubiéramos tenido más experiencia en Unity hubiéramos alcanzado mejores resultados.

## *VI - Referencias*

[1] SharpNEAT Neuroevolution Framework.

(s. f.). SharpNEAT. https://sharpneat.sourceforge.io/

[2] lordjesus. (s. f.). lordjesus/UnityNEAT. GitHub. https://github.com/lordjesus/UnityNEAT

[3] Unity3D. (Abril de 2019). *Unity Documentation*. Obtenido de https://docs.unity3d.com/es/2019.4/Manual/nav-BuildingNavMesh.html
m