UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO

INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Desarrollo de un videojuego para la configuración y análisis de redes de computadores

GNS3sharp

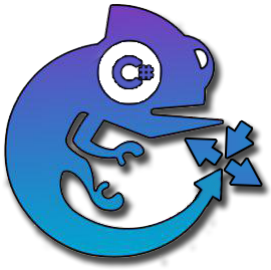
Autor
Ángel Oreste Rodríguez Romero

Directores
Juan José Ramos Muñoz
Jonathan Prados Garzón

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, agosto de 2018



Desarrollo de un videojuego para la configuración y análisis de redes de computadores

GNS3sharp

Autor
Ángel Oreste Rodríguez Romero

Directores
Juan José Ramos Muñoz
Jonathan Prados Garzón

Desarrollo de un videojuego para la configuración y análisis de redes de computadores: GNS3sharp

Ángel Oreste Rodríguez Romero

Palabras clave: palabra_clave1, palabra_clave2, palabra_clave3, ...

Resumen

El jugar ha sido desde siempre un gran amigo de la educación. Aprender jugando es un lema que cada vez se repite más. Los videojuegos, concretamente, toman en cierta forma el relevo de los juegos tal y como tradicionalmente estos se han entendido y amplían sus posibilidades.

La era digital afecta a casi todos los ámbitos de nuestro entorno. Las redes no iban a quedar excluidas de ese avance. Así, se pueden encontrar decenas de implementaciones virtuales de redes de telecomunicaciones, permitiéndonos visualizar su funcionamiento evitando el desembolso que equivale una real.

Digitalizados ambos ámbitos, ¿por qué no unirlos? ¿Y por qué no unirlos con un propósito educacional?

El presente documento pretende realizar un acercamiento a tal propósito. Se listará una serie de tecnologías que permiten llevar esto a cabo así como el desarrollo de mi aproximación.

Development of a videogame for the configuration and analysis of computer networks: GNS3sharp

Ángel Oreste, Rodríguez Romero

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Playing has always been a great friend of education. Learning by playing is a motto that is repeated more and more. Videogames, in particular, take over from games as they have traditionally been understood and expand their possibilities.

The digital age affects almost every area of our environment. Networks would not be excluded from this development. Thus, dozens of virtual implementations of telecommunication networks can be found, allowing us to visualize them working, avoiding the disbursement that is equivalent to a real one.

Digitized both areas, why not unite them, and why not unite them for an educational purpose?

This document is intended to bring this about. A number of technologies will be listed that allow this to be done as well as the development of my approach.

Yo, **Ángel Oreste Rodríguez Romero**, alumno de la titulación Ingeniería de Tecnologías de Telecomunicación de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 25351379C, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Fdo: Ángel Oreste Rodríguez Romero

Granada a 1 de septiembre de 2018.

D. **Juan José Ramos Muñoz**, Profesor del Área de Telemática del Departamento TSTC de la Universidad de Granada.

D. **Jonathan Prados Garzón**, Profesor del Área de Telemática del Departamento TSTC de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Desarrollo de un videojuego para la configuración y análisis de redes de computadores: GNS3sharp***, ha sido realizado bajo su supervisión por **Ángel Oreste Rodríguez Romero**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden y firman el presente informe en Granada a 1 de septiembre de 2018.

Los directores:

Juan José Ramos Muñoz **Jonathan Prados Garzón**

Agradecimientos

A Juanjo, por su increíble paciencia e inestimable ayuda, tanto en nuestras tutorías presenciales como aquellas improvisadas por Telegram. A todos aquellos profesores que confiaron en mí y en mis capacidades más que yo mismo en tantos momentos. A todos aquellos compañeros como Alfonso que no solo me facilitaron la vida académica con su conocimiento, si no también por su compañía. A Alberto, que aunque algo reticente de primeras, está dispuesto a echarme una mano de pedírselo. A mis compañeros de Francia, que me impulsaron a ampliar mis conocimientos. A Antonio por el logo tan genial que ha hecho. A mi grupo, porque sin él no habría tenido el ánimo suficiente durante este año para afrontar el proyecto. A todos aquellos amigos que me oyeron quejarme de mi proyecto con estoicismo. A la comunidad de StackOverflow que de tantos apuros me ha sacado.

Pero ante todo, a mis padres, pilar fundamental y soporte absoluto de toda mi vida, universitaria o no. Por la educación que me han dado, por todos aquellos caprichos que me permitieron y por velar siempre por mi salud.

Índice general

1. Introducción **1**
1.1. Introducción 1
1.2. Los videojuegos docentes 1
1.3. Nuestro problema 1
1.4. "GNS3sharp" 1
1.5. Motivación personal 1

2. Introducción **3**
2.1. Descripción del problema 3
2.2. Estructura del trabajo 3

3. Estado del arte **5**
3.1. Motores de videojuegos 5
3.1.1. Motores más famosos 5
3.1.2. Nuestra elección 5
3.2. Simuladores de redes 5
3.2.1. Simuladores más famosos 5
3.2.2. Nuestra elección 5
3.3. Juegos docentes 5

4. Diseño de la solución **7**
4.1. Diseño de la API 7
4.1.1. Elección del lenguaje 8
4.1.2. Estructura de clases 9
4.1.3. GitHub y la comunidad 9
4.2. Diseño del videojuego 9
4.2.1. Interacción con el simulador 9
4.2.2. Propuestas de modelo de videojuego 9

Bibliografía **11**

Capítulo 1

Introducción

fgxdh

1.1. Introducción

huigk

1.2. Los videojuegos docentes

Mucha bibliografía

1.3. Nuestro problema

Aquí puedes poner tasas de abandono escolar

1.4. Nuestra solución

Explicar qué has hecho

1.5. Motivación personal

```
foreach(Node n in handler.Nodes){
        Console.Write("host: {0}, port: {1}, name: {2}, component: {3}
            ",
                n.ConsoleHost, n.Port, n.Name, n.GetType().ToString())
                    ;
```

```
        foreach(Link link in n.LinksAttached){
                Console.Write($", link: {link.ID}");
        }
        foreach(Dictionary<string,dynamic> port in n.Ports){
                Console.Write($",\n\tport, adapter number: {port["
                    adapterNumber"]}");
                Console.Write($",\n\tport, port number: {port["
                    portNumber"]}");
                Console.Write($",\n\tport, link: {port["link"]}");
        }
        Console.WriteLine();
}
```

Capítulo 2

Introducción

fgxdh

2.1. Descripción del problema

huigk

2.2. Estructura del trabajo

```
foreach(Node n in handler.Nodes){
        Console.Write("host: {0}, port: {1}, name: {2}, component: {3}
            ",
                n.ConsoleHost, n.Port, n.Name, n.GetType().ToString())
                    ;
        foreach(Link link in n.LinksAttached){
                Console.Write($", link: {link.ID}");
        }
        foreach(Dictionary<string,dynamic> port in n.Ports){
                Console.Write($",\n\tport, adapter number: {port["
                    adapterNumber"]}");
                Console.Write($",\n\tport, port number: {port["
                    portNumber"]}");
                Console.Write($",\n\tport, link: {port["link"]}");
        }
        Console.WriteLine();
}
```

Capítulo 3

Estado del arte

El estado del arte se define como el nivel de desarrollo de un ámbito concreto, generalmente relacionado con el mundo técnico-científico.

3.1. Motores de videojuegos

3.1.1. Motores más famosos

3.1.2. Nuestra elección

huigk

3.2. Simuladores de redes

3.2.1. Simuladores más famosos

3.2.2. Nuestra elección

3.3. Juegos docentes

Con todo lo anterior nuestro objetivo es tal. Ya hay ejemplos

Capítulo 4

Diseño de la solución

Expuestas ya las distintas tecnologías que serán implementadas en nuestro trabajo, queda definir de qué modo estas serán utilizadas y cuál será su papel en la creación del proyecto.

4.1. Diseño de la API

La programación se rige por capas de abstracción. Partiendo de conceptos concretos se desarrolla una capa de abstracción haciendo uso de ellos que permite elevar el trato de elementos concretos un nivel por encima[1]. De esta forma ganamos en eficiencia y agilidad de escritura sin perder flexibilidad de desarrollo.

Una API, a grandes rasgos, no es más que una capa de abstracción sobre un lenguaje de programación o incluso sobre un framework del mismo. Comprende una serie de funciones que facilitan en mayor o menor medida trabajar sobre un cierto motivo. Como ejemplo de API famosa tenemos la de Google Maps, que contiene un compendio de métodos para JavaScript que permiten interactuar directamente con la plataforma de Google y crear nuestros programas jugando con ella[2]. Podemos verlo como una biblioteca de funciones.

Como se ha citado previamente, GNS3 hace uso de una API REST (*REpresentational State Transfer*). Esto es, mediante una serie de métodos (los conocidos GET, POST...) asociados a una URI concreta podemos interactuar vía web con una aplicación. La diferencia fundamental con otra clase de servicios web es que REST está orientado a recursos y no a métodos. Esto permite a la web utilizar comunicaciones sin estado, facilitando de este modo su escalabilidad[3].

Aunque de increíble utilidad y ya veremos qué papel concreto juega en

nuestro trabajo, necesitamos algo más para poder propiciar la interacción entre el simulador de redes y el videojuego. Necesitamos crear nuestra propia API que haga uso de la API REST de GNS3 y que defina métodos que permita a Unity interactuar con el SR de forma automática.

4.1.1. Elección del lenguaje

En pleno 2018 la cantidad de lenguajes de programación existentes roza el absurdo. Desde el tradicional C, pasando por el multifuncional Java, el sencillo Python o incluso los llamados lenguajes esotéricos como LOLCODE[4]. De entre todos ellos nosotros elegiremos uno sobre el que trabajar. Esta decisión está condicionada, como es natural, por el motor de videojuegos a utilizar.

Ya que nuestra intención es que el MV sea capaz de establecer interacción con el SR, es necesario que la API a desarrollar esté escrita en un lenguaje con el que el propio motor sea capaz de trabajar. C# parecía la opción más sensata. ¿Por qué? Las razones se exponen a continuación:

- **Porque es el lenguaje más usado en Unity**. Unity admite varios lenguajes de programación con los que desarrollar los scripts asociados a los juegos. C++, usado en otros muchísimos otros motores de videojuegos como Unreal Engine, es uno de ellos. Aunque se trate de un lenguaje increíblemente potente y eficiente, su complejidad de uso es mucho mayor, ya que su nivel de abstracción es altamente inferior. JavaScript es otro de ellos, pero no es tan recomendable como C#, pues entre otras razones, a diferencia de JS es fuertemente tipado[5]. En general, Unity es el lenguaje usado por defecto en Unity y el recomendado por todos.

- **Porque otros motores comienzan a usarlo**. Unity está en el podio de entre los motores de videojuegos más usados en el mundo. Como tal se convierte en un referente. El resto de motores miran hacia él y si quieren atraer a nuevos programadores, tendrán que hacer de su incursión en el nuevo motor algo sencillo. Este es el caso de Godot Engine, que hace unos meses decidió incluir tal lenguaje entre los soportados[6]. Esto quiere decir que la API no solo podrá ser usada en juegos creados en Unity, si no que su terreno de juego se verá ampliado. CryEngine es otro de los motores que permiten el uso de C#.

- **Porque, ante todo, es un gran lenguaje**.

4.1.2. Estructura de clases

Clase principal

Nodes

Links

Otras clases

4.1.3. GitHub y la comunidad

4.2. Diseño del videojuego

4.2.1. Interacción con el simulador

4.2.2. Propuestas de modelo de videojuego

Bibliografía

[1] Pavol Návrat. Hierarchies of programming concepts: Abstraction, generality, and beyond. *Sigse Bulletink*, 26(3):17–28, 1994. Available at `https://dl.acm.org/citation.cfm?id=187397`.

[2] Google maps platform. Available at `https://cloud.google.com/maps-platform/?hl=es`.

[3] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, 2000. Available at `https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm`.

[4] Esolang, the esoteric programming languages wiki. Available at `https://esolangs.org/wiki/Hello_world_program_in_esoteric_languages`.

[5] Joseph Hocking. *Unity in action*. Manning, 2000.

[6] Ignacio Roldán Etcheverry. Introducing c# in godot. Available at `https://godotengine.org/article/introducing-csharp-godot`.