TRABAJO FIN DE GRADO

INGENIERÍA EN INFORMÁTICA

Titulo del Proyecto

Subtitulo del Proyecto

Autor
Andrés Merlo Trujillo

Directores
Luis López Escudero

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

—

Granada, mes de 2023

Título del proyecto

Subtítulo del proyecto.

Autor
Andrés Merlo Trujillo

Directores
Luis López Escudero

Título del Proyecto: Subtítulo del proyecto

Andrés Merlo Trujillo

Palabras clave: Unreal, Simulación, IA

Resumen

Se pretende desarrollar una aplicación gráfica utilizando el motor gráfico *Unreal Engine* cuyo objetivo es de poder simular carreras de coches. Esta simulación dotará a los pilotos virtuales de la capacidad de tomar decisiones realistas y de cometer errores, basándose en las condiciones de cada piloto durante la carrera, que pueden variar dependiendo de las condiciones externas o ser modificadas por el usuario en tiempo real. Adicionalmente, se podrán configurar ajustes adicionales antes de comenzar la carrera, con el objetivo de hacer la simulación lo más personalizable posible.

Project Title: Project Subtitle

First name, Family name (student)

Keywords: Keyword1, Keyword2, Keyword3, ...

Abstract

Write here the abstract in English.

Yo, **Andrés Merlo Trujillo**, alumno de la titulación Ingeniería Informática de la **Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada**, con DNI 77147239H, 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: Andrés Merlo Trujillo

Granada a X de mes de 2023.

D. **Luis López Escudero**, Profesor del Área de XXXX del Departamento YYYYY de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado ***Título del proyecto, Subtítulo del proyecto***, ha sido realizado bajo su supervisión por **Andrés Merlo Trujillo**, 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 X de mes de 2023 .

Los directores:

Luis López Escudero

Agradecimientos

Poner aquí agradecimientos...

Índice general

1. Introducción **15**
1.1. Introducción y Motivación 15
1.2. Objetivos 16
1.3. Estado del arte 17
1.3.1. Motor gráfico 17
1.3.1.1. Unreal Engine 17
1.3.1.2. Unity 18
1.3.1.3. Godot 18
1.3.2. Algoritmo de navegación 19

Bibliografía **21**

Capítulo 1

Introducción

1.1. Introducción y Motivación

El mundo del motor es un deporte con una amplia base de seguidores y en el que se invierte una gran cantidad de dinero en tecnologías de todo tipo para poder tener una ventaja competitiva, incluyendo simuladores. Estos simuladores permiten a los equipos de carreras realizar una extensa variedad de actividades: entrenar a los pilotos antes y durante la temporada, modificar parámetros para lograr el máximo rendimiento del vehículo o simular carreras para analizar cuál será la estrategia más adecuada para alcanzar la victoria.

Este proyecto se centrará en la simulación de gestión de carreras, sector en el que existen pocas opciones para el público general y que es cada vez más demandado para aprender la manera en que las distintas estrategias pueden afectar al desenlace de una carrera y como diferentes situaciones influyen en la capacidad de los pilotos, lo cual a su vez puede afectar a la tasa de errores producidos durante la carrera.

Se añadirán ciertas características procedentes de videojuegos con elementos de simulación a la aplicación gráfica, como la posibilidad de elegir entre varios tipos de vehículos (*Gran Turismo B-Spec*), por mencionar un ejemplo.

La motivación detrás de realizar esta aplicación es profundizar en los conocimientos previamente adquiridos en otras asignaturas del manejo de *Unreal Engine* y en el aprendizaje de su API, enfocándolo al ámbito de las carreras y de los vehículos autónomos, área que me resulta muy interesante. Asimismo, se busca profundizar en el estudio de los distintos algoritmos de navegación para los vehículos del simulador, intentando escoger el que mejor se adapte a las necesidades del proyecto y que ofrezca buenos resultados.

1.2. Objetivos

En este proyecto se pretende desarrollar un simulador de gestión de carreras multidisciplinar y personalizable, en el que los pilotos tendrán un conjunto de capacidades que serán modificados por las condiciones de la carrera o por el usuario en tiempo real, haciendo que el piloto pueda cometer más errores o menos si su estado mejora. También existirá la opción de poder modificar la velocidad de simulación, con el objetivo de obtener los resultados de manera más rápida.

Además, la aplicación permitirá modificar diversos parámetros antes de la carrera, con el propósito de poder personalizar la simulación lo máximo posible. Cabe destacar que estos parámetros no podrán ser modificados de nuevo, debido a que no tendría demasiado sentido y podrían hacer que la simulación no sea del todo precisa.

Siendo más específicos, los objetivos mínimos relativos a cambios antes de ejecutar la simulación son:

- Cambio de número de pilotos.
- Cambio de número de vueltas.
- Cambios de las aptitudes de los pilotos; es decir, el nivel máximo de cada condición que puede alcanzar.
- Cambio de la velocidad máxima de los vehículos.
- Permitir (opcionalmente) que haya variación de prestaciones entre vehículos.
- Cambio del tipo de vehículo.

En cuanto a cambios durante la simulación, debe cumplir estos objetivos:

- Seleccionar piloto de la lista de pilotos.
- Obtener las condiciones actuales del piloto seleccionado.
- Modificar las condiciones actuales del piloto seleccionado.
- Cambiar la velocidad de simulación.

Cabe destacar que todos estos objetivos y algunos opcionales serán explicados en detalle en apartados siguientes.

1.3. Estado del arte

———ARREGLAR LAS MULETILLAS DE ADEMAS———

Como ya se ha descrito anteriormente, este proyecto hará uso de *Unreal Engine* como motor gráfico y como *framework* para la realización del simulador especificado. En cuanto a la forma en la que los pilotos se moverán por el circuito, he decidido usar el algoritmo de navegación **PONER AQUÍ EL ALGORITMO DE NAVEGACIÓN**.

A continuación, voy a explicar las distintas opciones que hay disponibles en motores gráficos y en algoritmos de navegación y explicar el motivo de la elección realizada. Separaré esto en dos subsecciones.

1.3.1. Motor gráfico

En el mercado hay gran variedad de motores gráficos de videojuegos, los más conocidos son los siguientes:

1.3.1.1. Unreal Engine

Unreal Engine [6] es una plataforma de desarrollo que permite, entre otras cosas: el desarrollo de videojuegos, creación de contenido para programas y películas, simulación, etc. Posee un soporte para un gran número de plataformas, tanto consolas como móviles.

Unreal además ofrece *Blueprint Visual Scripting*, el cual es un sistema de *scripting* visual que hace uso de nodos para programar las distintas partes del sistema. Permite programar de manera visual sin la necesidad de conocer la *API* de C++.

Además, a partir de la versión 5.0, Unreal incluye diversas tecnologías como:

- TSR *(Temporal Super Resolution)*, mejorando el rendimiento haciendo que el juego se renderice internamente a una resolución menor y luego sea reescalado al tamaño deseado.
- Nanite, que permite tener objetos con una gran cantidad de polígonos en la pantalla,
- Lumen, que genera una iluminación más realista.

Todo esto permite tener un resultado a nivel gráfico muy convincente sin necesitar dedicar demasiado trabajo en este apartado y sin requerir un hardware demasiado potente.

Asimismo, Unreal incluye una tienda con una gran cantidad de recursos, permitiendo simplificar mucho el desarrollo de videojuegos.

Sin embargo, no tiene una licencia de código abierto, pero su código fuente es accesible a través de un repositorio de GitHub.

1.3.1.2. Unity

Unity [5] es una plataforma de desarrollo que permite: desarrollar videojuegos, visualizar construcciones en el ámbito de la arquitectura, para uso en la industria automotriz y para la creación de películas y series.

Entre sus características se encuentra:

- Uso de C# como lenguaje de programación.
- Motor de físicas 2D separado del de físicas en 3D.
- Posee un *Scriptable Rendering Pipeline* altamente personalizable, permitiendo mediante el uso de scripts escritos en C# personalizar sombras, iluminación, oclusión ambiental, entre otras.
- Soporte para un gran número de plataformas, incluyendo consolas y móviles, aparte de Windows, macOS y Linux.
- Incluye un lenguaje de programación visual, similar a *Unreal* que permite programar sin necesitar conocer la *API*.

Además, Unity tiene varias versiones del programa, incluyendo uno gratuito y los demás de pago, con prioridad en la asistencia y un mayor abanico de herramientas.

1.3.1.3. Godot

Godot [3] es un motor gráfico gratuito y multiplataforma de código abierto que permite crear videojuegos en 2D y 3D.
Entre las características más notables se encuentran las siguientes:

- Motor gráfico 2D dedicado.
- Programación usando GDScript, C# o C++ de manera oficial. No obstante, se puede programar usando otro lenguaje como Rust, Python o JavaScript.
- Soporte para un gran número de plataformas, incluyendo móviles y consolas.

- Incluye un motor de físicas (solo para el motor gráfico 3D), aunque es algo más simple que el de sus competidores.

A partir de la versión 4, Godot ha dejado de incluir el lenguaje visual, debido a su falta de uso, por lo que solo es posible utilizar lenguajes de programación convencionales [2].

Además, Godot tiene una biblioteca con una gran cantidad de recursos para utilizar durante el desarrollo de un videojuego.

Al final he decidido usar Unreal Engine debido a que la versión gratuita incluye toda la funcionalidad, ofrece unos buenos resultados a nivel visual sin requerir demasiado esfuerzo, es el más potente en cuanto a programación visual y es con el que más familiarizado estoy de todas las opciones citadas anteriormente.

1.3.2. Algoritmo de navegación

En cuanto a algoritmos de navegación para los vehículos, los más destacados son:

- Aprendizaje por refuerzo (Q-Learning): Es una técnica de aprendizaje automático en la que el agente aprende a tomar decisiones en un entorno mediante la interacción a través de acciones (en este caso es: acelerador, freno y volante) y recibiendo una recompensa en caso de acercarse al objetivo deseado o de hacerlo bien.

  Es importante obtener un equilibrio entre exploración y explotación, para que el agente sea capaz de actuar ante situaciones desconocidas de manera correcta. En caso de estar desequilibrado, puede darse la situación en el que se produzca un máximo local y no sea capaz de descubrir la mejor estrategia.

  Para evitar esto, se suelen utilizar métodos de equilibrio. Uno de ellos es el denominado *epsilon-greedy*, que permite controlar la cantidad de exploración mediante un parámetro *epsilon*, que determina la aleatoriedad en la selección de acciones [4], permitiendo elegir en función de dicho parámetro una acción aleatoria o la mejor hasta el momento.

  Una desventaja que tiene es la necesidad de entrenar el modelo, cosa que puede llevar demasiado tiempo, pero con el suficiente tiempo y con un conjunto de recompensas y castigos bien realizado, es uno de los mejores algoritmos para resolver este tipo de problema.

  No obstante, al trabajar en un espacio continuo, no se puede hacer una tabla de *Q-Values* porque, aparte de haber infinitos valores, la tabla no cabría físicamente en memoria y sería computacionalmente

muy costoso. En este caso, sería necesario usar alguna alternativa como puede ser una red neuronal, permitiendo así trabajar con este tipo de valores [1].

- Mediante reglas: Otra forma de resolver este problema es mediante reglas, que como su nombre indican, son un conjunto de reglas que se accionarán cuando se dé la condición deseada y normalmente están priorizadas. Suelen tener forma de árbol, al hacer preguntas en cadena hasta llegar con la solución.

  Esto tiene la ventaja de ser más sencillo de implementar que el de aprendizaje por refuerzo (sin usar ninguna biblioteca ya implementada), pero suele tener peores resultados, al poder darse el caso de no haber priorizado bien las reglas o no haber tenido en cuenta algún caso especial.

- Máquina de estados: Este algoritmo implementa una máquina de estados finitos mediante una estructura de datos, donde cada nodo representa un estado en el que se encuentra el piloto y cada arco representa la transición a otro estado.

  Una de las ventajas que tiene es que suele ser más simple describir el comportamiento en agentes más complejos y además, suelen ser más fáciles de modificar que los basados en reglas.

Bibliografía

[1] Rémi Coulom. *Reinforcement Learning Using Neural Networks, with Applications to Motor Control*. Theses, Institut National Polytechnique de Grenoble - INPG, June 2002. URL: `https://theses.hal.science/tel-00003985`.

[2] Enlace a la noticia sobre la eliminación del lenguaje visual en godot. URL: `https://godotengine.org/article/godot-4-will-discontinue-visual-scripting/`.

[3] Página oficial de Godot. URL: `https://godotengine.org/`.

[4] Michel Tokic and Günther Palm. Value-difference based exploration: Adaptive control between epsilon-greedy and softmax. In Joscha Bach and Stefan Edelkamp, editors, *KI 2011: Advances in Artificial Intelligence*, pages 335–346, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[5] Página oficial de Unity. URL: `https://unity.com/`.

[6] Página oficial de Unreal Engine. URL: `https://www.unrealengine.com/en-US/unreal-engine-5`.