Universidad de Córdoba
Escuela Politécnica Superior

Grado en Ingeniería Informática
Especialidad en Computación

Trabajo de Fin de Grado

Entrenamiento de una IA mediante aprendizaje por refuerzo para un juego hecho en Unreal Engine

- Manual de Código -

Autor:

Francisco David Castejón Soto

Directores:

Dr. Manuel Jesús Marín Jiménez

Dr. Javier Sánchez Monedero

Córdoba, 10 de junio de 2024

Índice general

Índice de figuras

Capítulo 1

Acceso al Código Fuente

El código fuente completo del proyecto se puede encontrar en el siguiente enlace alojado en Gitlab de forma pública:

https://gitlab.com/Silver812/tfg

Cada función está auto-explicada mediante comentarios dentro del propio código en C++ y Python, lo que facilita su comprensión y mantenimiento. Es importante señalar que, debido a que este proyecto ha sido desarrollado utilizando Unreal Engine, cierta parte del código está escrita en el lenguaje visual de Blueprints. Para acceder a esta parte del código, es necesario abrir el proyecto desde Unreal Engine 5.3.X. En concreto, el código en Blueprints está dentro de la carpeta denominada *FiringRange*.

El resto del código desarrollado por el estudiante se encuentra en la siguiente ruta del proyecto:

`Plugins/GameFeatures/FiringRange/Source`

En esta ubicación, se puede encontrar tanto las clases de C++ de Unreal Engine como los *scripts* de aprendizaje por refuerzo en Python.

Capítulo 2

Archivos

Este capítulo describe los archivos y componentes más importantes que forman este proyecto, tanto del videojuego como de la IA.

2.1. Archivos de Unreal Engine

2.1.1. Blueprints

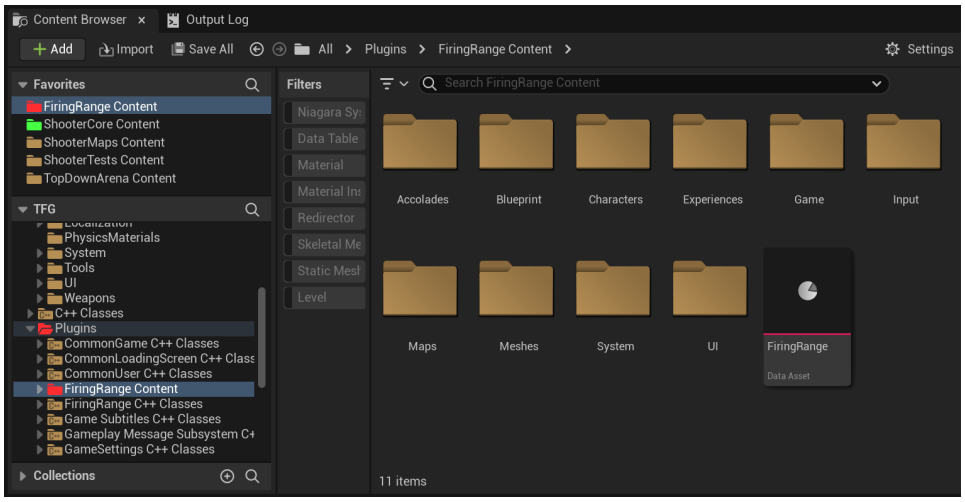

Figura 2.1: Blueprints

En la Figura 2.1 se puede ver la estructura de carpetas del proyecto den-

tro del editor de Unreal Engine. Cada una contiene diferentes Blueprints que controlan el comportamiento del minijuego. Muchos de ellos son modificaciones de otras partes del código de Lyra, como:

- *BP_FiringRangeScoring*: controla el sistema de puntuación del minijuego, así como la lógica de inicio y finalización.
- *W_ScoreWidget_FiringRange*: en esta modificación de la interfaz estándar de Lyra, se muestra la puntuación de un solo jugador, en lugar de la de dos equipos de jugadores.

Pero otros son completamente nuevos, como *B_FiringRangeDummy*, que controla la rotación de los maniquíes en el minijuego. Además, cada vez que se dispara a un maniquí, este notifica a *B_FiringRangeScoring* para que actualice la puntuación.

2.1.2. C++

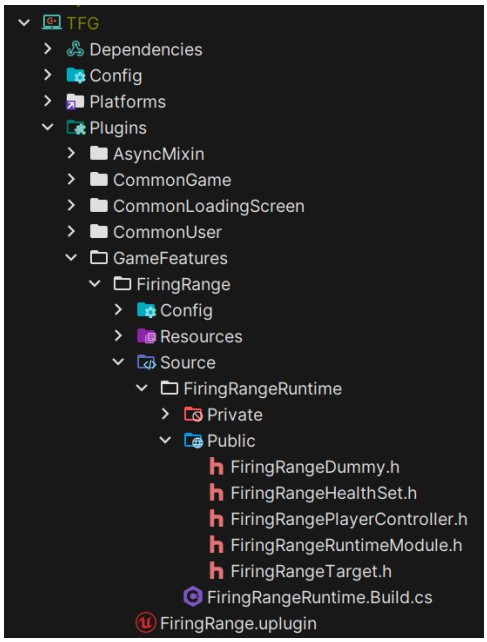

Figura 2.2: Clases de C++

En la Figura 2.2 se puede ver la estructura de carpetas del proyecto en Rider. En esta ubicación, se encuentran el código C++ que se ha desarrollado para el proyecto. Además de otras modificaciones al código fuente de Lyra, se han creado cuatro nuevas clases:

- *FiringRangeTarget*: clase base que define un objetivo en el minijuego.
- *FiringRangeHealthSet*: componente de los objetivos que controla su salud y la capacidad de recibir daño.
- *FiringRangeDummy*: clase hija de *FiringRangeTarget* que define un maniquí en el minijuego. Esta, a su vez, es la clase de padre del Blueprint *B_FiringRangeDummy*.
- *FiringRangePlayerController*: controlador del jugador en el minijuego. Se encarga de controlar el movimiento de la cámara y el disparo. Además, también gestiona todo lo relativo a la conexión con el servidor local en Python.

2.2. Archivos de Python

En la Figura 2.3 se puede ver la estructura de carpetas del proyecto en Visual Studio Code. Veamos los archivos y directorios uno a uno:

- *envs*: directorio con todo lo relativo a los entornos de Gymnasium.
- *logs*: directorio donde se guardan los registros.
- *a2c, ppo, ppo_exp*: directorios con los registros de cada uno de los ajustes de los hiperparámetros y entrenamientos de los modelos. Estos datos son los que se han usado para generar las gráficas presentes en el manual técnico.
- *__init__.py*: archivo relativo a la generación de paquetes en Python.

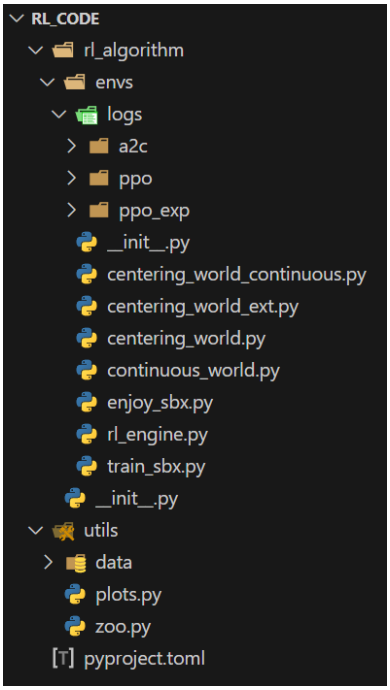

Figura 2.3: *Scripts* de Python

- *centering_world_continuous.py*: entorno de Gymnasium del prototipo que se desarrolló en la primera parte del proyecto.
- *centering_world_ext.py*: entorno de Gymnasium con el espacio de acción extendido.
- *centering_world.py*: entorno de Gymnasium con el espacio de acción original.
- *continuous_world.py*: otro entorno de Gymnasium que se desarrolló en la primera parte del proyecto a modo de prototipo.
- *enjoy_sbx.py*: pequeño *script* que permite evaluar un modelo usando la versión en JAX de Stable Baselines 3, la cual es más rápida que la de Python.
- *rl_engine.py*: archivo principal del proyecto que usa la clase de *cen-*

tering_world.py y es el encargado de conectarse con el controlador de Unreal Engine.

- *train_sbx.py*: *script* que entrena un modelo usando la versión en JAX de Stable Baselines 3.
- *_init_.py*: otro archivo relativo a la generación de paquetes en Python.
- *utils*: directorio con *scripts* auxiliares.
- *plots.py*: pequeño *script* que genera gráficas de los registros dentro de data.
- *zoo.py*: pequeño *script* que simplemente comprueba que los entornos están debidamente instalados como paquetes de Python.
- *pyproject.toml*: archivo de configuración que permite crear paquetes de Python.