Trabajo Fin de Grado
grado en ingeniería informática

Desarrollo de una arquitectura reactiva y deliberativa usando planificación en el entorno de juegos GVGAI

Anexo - Manual de usuario

Autor
Vladislav Nikolov Vasilev

Director
Juan Fernández Olivares

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Curso 2019-2020

Índice

1. Instalación

El proyecto se encuentra alojado en un repositorio público de `GitHub`. Este repositorio encontrarse en el siguiente enlace: `https://github.com/Vol0kin/gvgai-pddl`.

Las dependencias del sistema y el proceso de instalación se encuentran descritos en la página principal. En esta página también puede encontrarse otra información, como por ejemplo cómo generar plantillas de archivos de configuración, cómo ejecutar el sistema y cómo se puede ejecutar el planificador de `Planning.Domains` en local.

2. Estructura del proyecto

La estructura del proyecto puede verse en la figura 1. Ahí pueden verse los elementos más destacados, tanto directorios como archivos:

- El directorio `config` contiene algunos ejemplos de archivos de configuración en formato `YAML`. No es obligatorio que los archivos de configuración se generen en este directorio, pero sí que sería una buena idea tener un directorio específico donde agruparlos.
- El directorio `docs` contiene los `JavaDoc` que se han generado para el código fuente del sistema.
- El directorio `domains` contiene algunos ejemplos de dominios `PDDL` que se han creado para algunos de los juegos.
- El directorio `examples` es uno de los directorios originales de `GVGAI`. Aquí se incluyen los archivos en formato `VGDL` que describen los juegos y los niveles. Dentro de este directorio se encuentra el directorio `gridphysics`, el cual contiene todos los juegos que puede ejecutar el sistema.
- El directorio `sprites` es otro de los directorios originales de `GVGAI`. Aquí se encuentran los *sprites* de los elementos de los juegos.
- El directorio `src` contiene el código fuente del sistema, el código de los tests y los recuros que utilizan los tests (dominios, problemas y archivos de configuración). La estructura de este directorio sigue la estructura típica de un proyecto de Maven. El directorio `src/main/java` contiene el código fuente del sistema. Dentro de dicho directorio se encuentra el directorio `controller`, el cual contiene el código fuente implementado.

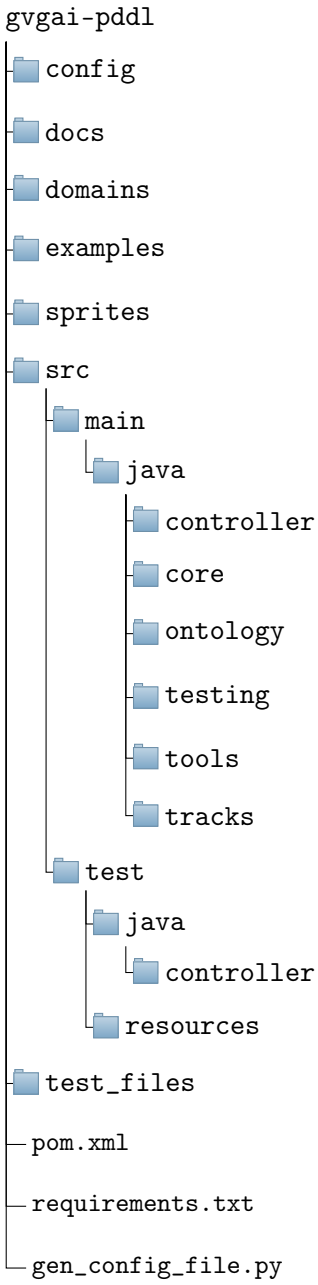

Figura 1: Estructura del proyecto.

- El archivo `pom.xml` contiene información sobre el proyecto de Maven, como por ejemplo las dependencias. También contiene la configuración del proyecto, como por ejemplo las tareas que se tienen realizar en algunos de los pasos de la construcción del proyecto. Se recomienda no modificar este archivo a

menos que se sepa muy bien qué es lo que se está modificando.

- El archivo `requirements.txt` contiene las dependencias del *script* de `Python` para la generación de plantillas de archivos de configuración.
- Por último, el *script* `gen_config_file.py` es el que permite generar plantillas de archivos de configuración.

Los archivos que se encuentran en el directorio `controller` se pueden ver en la figura 2. En estos archivos se implementan las siguientes funcionalidades:

- En el archivo `Agenda.java` se implementa la clase `Agenda`, la cual permite la gestión de los objetivos de forma sencilla.
- El fichero `GameInformation.java` contiene la clase `GameInformation`, la cual es la responsable de guardar la información que se carga de los archivos de configuración.
- En el archivo `PDDLAction.java` se encuentra implementada la clase `PDDLAction`, la cual representa una acción `PDDL`. También se encuentra implementada la clase `PDDLEffect` como clase anidada a la anterior. Esta clase representa un efecto de una acción `PDDL`.

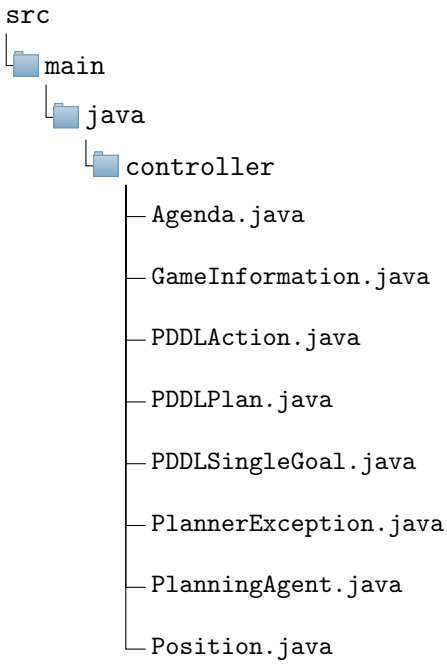

Figura 2: Contenido del directorio `controller`.

- En el archivo `PDDLPlan.java` se implementa la clase `PDDLPlan`, la cual representa un plan de `PDDL`.
- El archivo `PDDLSingleGoal.java` contiene la implementación de la clase `PDDLSingleGoal`, la cual representa un objetivo `PDDL`.
- En `PlannerException.java` se tiene la excepción que se lanza cuando se produce algún error al llamar al planificador en la nube.
- En el archivo `PlanningAgent.java` se encuentra la clase `PlanningAgent`, la cual contiene el agente implementado.
- El archivo `Position.java` contiene un enumerado que indica cuatro posibles posiciones: `UP`, `DOWN`, `LEFT`, `RIGHT`. Estas posiciones se utilizan en algunas de las clases anteriores.

Por último, en la figura 3 se observa la estructura del directorio que contiene los tests que se han creado. Estos tests se hacen sobre la funcionalidad pública que exponen las correspondientes clases. Como se mencionó anteriormente, se utilizan los recursos localizados en `src/test/resources` para llevarlos a cabo.

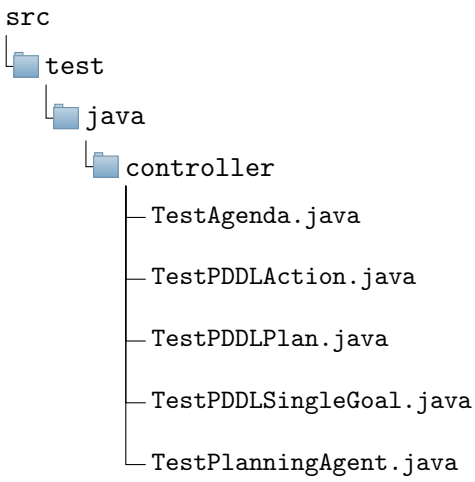

Figura 3: Estructura del directorio que contiene los tests para los archivos del paquete `controller`.

3. Documentación detallada del sistema

En la página principal del proyecto de `GitHub` se puede encontrar un enlace a la documentación detallada de todo el código fuente, incluyendo la del sistema.