Universidad de Murcia

Facultad de Informática

IA para el Desarrollo de Videojuegos

Real Time Wargame

Autores

Antonio López Martínez-Carrasco

antonio.lopez31@um.es

José María Sánchez Salas

josemaria.sanchez12@um.es

Profesores

Francisco Javier Martín-Blázquez Gómez

jgmarin@um.es

Luis Daniel Hernández Molinero

ldaniel@um.es

Índice

Part I
Introducción y estructura de la aplicación

1 Introducción

2 Estructura de la aplicación

Part II
Clase principal y mapa

3 Clase principal

4 Mapa

Part III
IA reactiva

5 Steerings

En el ámbito de los videojuegos, un sistema steering (o sistema de dirección, es español) es un mecanismo que *propone movimientos* a los agentes involucrados en el videojuego en base al entorno local que les rodea, es decir, usando la información del mundo que los agentes captan a través de sus sentidos. Los steerings concretos que se han estudiado y que, por tanto, han sido implementados es este proyecto son: **Steering Uniforme** y **Steering Uniformemente Acelerado**.

Un Steering Uniforme solamente maneja velocidad lineal y velocidad angular, puesto que se da por hecho que en el movimiento no va a intervenir ningún tipo de aceleración (en decir, la aceleración es 0). A partir de dicha información contenida en el steering, el agente modificará su posición y orientación. Por el contrario, un Steering Uniformemente Acelerado sí contiene una aceleración lineal y una aceleración angular, puesto que en este tipo de movimientos sí intervienen aceleraciones (aceleración constante). A partir de esta información contenida en el steering, el agente modificará su velocidad lineal y angular. Al modificar la velocidad lineal y angular, la posición y orientación del agente también será modificada en consecuencia.

5.1 Interfaz Steering

Durante el desarrollo del proyecto, nos ha parecido conveniente el poder manejar todos los steerings de manera uniforme independientemente de la clase de steering concreto. Esto ha sido posible gracias a ciertas características del lenguaje Java como son las clases abstractas, la herencia o la ligadura dinámica. Teniendo esto en cuenta, hemos implementado una clase abstracta vacía (clase `Steering`), tal que todos los steerings concretos heredarán de ella.

5.2 Steering Uniforme

Este tipo de steering ha sido implementado en la clase `Steering_NoAcceleratedUnifMov` y, tal y como se ha comentado, hereda de la clase abstracta `Steering`. En esta clase podemos encontrar dos atributos llamados `velocity` (de tipo `Vector3`) y `rotation` (de tipo float). El primero hace referencia a la velocidad lineal (el vector velocidad) y el segundo hace referencia a la velocidad angular (un escalar). En esta clase también están presentes los métodos *get* y *set* correspondientes a ambos atributos. Además de estos métodos, también hay otro método denominado `getSpeed`,

que devuelve el módulo del vector `velocity`.

Es muy importante remarcar que, es este ámbito, *velocity* y *speed* no significan lo mismo. El primero hace referencia al **vector** velocidad, mientras que el segundo hace referencia al **módulo** de dicho vector (es un escalar).

5.3 Steering Uniformemente Acelerado

Este tipo de steering ha sido implementado en la clase `Steering_AcceleratedUnifMov` y, tal y como se ha comentado, hereda de la clase abstracta `Steering`. Esta clase tiene dos atributos llamados `lineal` (de tipo `Vector3`) y `angular` (de tipo `float`). El primero corresponde con la aceleración lineal (un vector) y el segundo con la aceleración angular (un escalar). También se han implementado los métodos *get* y *set* correspondientes a ambos atributos.

6 Comportamientos

7 Árbitros

8 Modelo

Esta es una de las partes más importantes del videojuego, puesto que en ella podemos encontrar a los **agentes** que intervendrán e interactuarán en el transcurso del juego. Concretamente, en este paquete podemos encontrar la implementación de los objetos del mundo (concepto general que lo engloba todo), la implementación de los personajes, la implementación de los obstáculos y la implementación de las formaciones (que han sido tratadas como un tipo especial de personaje).

8.1 Clase WorldObject

La clase `WorldObject` es un concepto general que representa a una entidad del juego. Todas las entidades concretas (descritas en los siguientes subapartados) heredarán de esta **clase abstracta**. Una cosa muy importante a tener en cuenta es que esta clase abstracta hereda a su vez de la clase `Sprite` de la librería libgdx. Esto se ha hecho así para aprovechar todas las características y propiedades que ya posee la clase `Sprite`, como por ejemplo, la posición, la orientación, la anchura, la altura, las funciones de dibujo u otra funcionalidad ya preprogramada en la librería.

Solamente hay que tener en cuenta una cosa muy importante: a lo que nosotros llamamos orientación, la librería lo llama rotación. Esto a su vez puede llevar a confusiones con la velocidad angular (que para nosotros es la rotación). Para evitar todos estos problemas de nomenclatura y todas las confusiones que esto pueda provocar, se han tomado ciertas medidas:

- Se han creado el par de funciones `getOrientation` y `setOrientation` que llaman a las funciones correspondiente del padre (`super.getRotation` y `super.setRotation`).
- Se han sobreescrito las funciones `getRotation` y `setRotation` heredadas del padre para que si se llaman en el hijo (en la clase `WorldObject`), lancen una excepción. Estas funciones no se deben llamar nunca, puesto que si deseamos consultar o modificar la orientación de una entidad, deberemos hacer uso de las funciones `getOrientation` y `setOrientation` de la clase `WorldObject`.

Estas modificaciones solucionan el problema y evitan que se puedan producir confusiones con los nombres y errores de concepto a lo largo del resto del proyecto.

Part VII
Bibliografía

Bibliografía

[1] Old, L., *Confesiones de una oveja bizca.* 1ª ed. Madrid: Naturalistic, 2010.

[Prad] Montero, J., *Metodos matemáticos aplicados a la ganadería.* 3ª ed. Sevilla: Ediciones de la pradera, 2007.