Prácticas Movimiento

Práctica 1

Usando el proyecto de ejemplo implementad un seek steering behavior.

El programa debe leer de un fichero xml llamado params.xml localizado en la carpeta Content, toda la información necesaria:

- la posición del target
- la velocidad máxima del personaje
- la aceleración máxima

y dirigirse a ella usando el **steering behavior de Seek**.

El fichero tiene este formato:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <targetPosition x="" y=""> </targetPosition>
    </params>
</root>
```

Para la lectura de este fichero, dentro del esqueleto de prácticas suministrado existe una clase Params (en el fichero params.h) que lee los datos de un fichero. Está programado para el fichero de esta práctica, y deberéis modificarlo para los parámetros necesarios en el resto de prácticas.

Para la implementación de la práctica disponéis de una clase AAICharacter donde podéis añadir el código que necesitéis. Las funciones más importantes de la clase son:

- void BeginPlay(): Se llama al principio del juego una vez. Ahí podemos inicializar y leer todos los datos necesarios.
- void Tick(float DeltaTime): Se llama en cada iteración del bucle principal con el parámetro DeltaTime indicando el tiempo transcurrido desde la última llamada en segundos
- void DrawDebug(): Podéis usar esta función para pintar información de debug en la pantalla. En IA es fundamental usar bien la información de debug para saber lo que está pasando en el juego en todo momento y poder encontrar los posibles problemas de forma sencilla. Dibujar información es una parte fundamental de ese proceso.

Consejos:

Empezad paso a paso:

1. Poned una velocidad inicial en BeginPlay
2. Usad esa velocidad para actualizar la posición del personaje en Tick
3. Ver como el personaje se mueve por la pantalla con la velocidad configurada
4. Añadir la aceleración
    a. Definir una variable local dentro de Tick para definir una aceleración fija
    b. Usar esta aceleración para actualizar la velocidad en Tick
    c. Ver como el personaje va cambiando su velocidad con el tiempo y modificando su movimiento. Experimentad con diferentes velocidades iniciales para ver cómo se modifica la dirección del movimiento al aplicar la aceleración
5. Crear un interfaz Steering del que derivarán todos los steerings que hagamos
    a. Tened en cuenta que un steering tiene que devolver acceleracion lineal y aceleración angular
6. Añadir una clase SeekSteering para el cálculo de la aceleración
    a. Inputs:
        i. Personaje al que se va a aplicar (para obtener su estado: posición, velocidad)
        ii. Target al que queremos ir
    b. Outputs:
        i. Aceleración
7. SeekSteering tendrá una función GetSteering que realizará el cálculo de la nueva aceleración
8. Implementad esa función utilizando el método explicado en clase para Seek
    a. Calcular velocidad deseada: resta entre posición del target y posición del personaje
    b. Calcular aceleración necesaria para conseguir esa velocidad: resta entre velocidad deseada y velocidad actual
    c. Poned la longitud del vector aceleración a un número (máxima aceleración): Normalizar vector y escalar por la máxima aceleración.
9. SeekSteering tendrá una función DrawDebug donde se pintarán:
    a. Vector (Línea) de la velocidad deseada calculada en el frame anterior
    b. Vector (Línea) de la aceleración calculada en el frame anterior

10. Usad la aceleración calculada por SeekSteering para actualizar la velocidad en lugar de la variable local que estáis creando en el código.
11. Disfrutad con vuestra práctica terminada.

Para realizar el pintado de Debug tenemos algunas funciones de apoyo en debugdraw.h. Necesitan que los objetos que se van a pintar existan ya en el level o en el blueprint del personaje

- SetArrow: Configura un arrow component que tiene que estar en el actor
- SetCircle: Configura un Sprite círculo que tiene que estar en el level

El pintado de debug solo configura actores o componentes que existen y se pintarán por si solos, así que esas funciones pueden llamarse desde cualquier sitio en cualquier momento.

Práctica 2

Implementar ahora un **steering behavior de Arrive**.

Los parámetros también se especifican en un fichero params.xml igual que la práctica anterior.

Formato:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <arrive_radius value=""> </arrive_radius>
        <targetPosition x="" y=""> </targetPosition>
    </params>
</root>
```

Práctica 3

Implementar ahora un **steering behavior de Align**, para que independientemente de cual sea la orientación inicial, termine mirando a un ángulo definido en params.xml

Formato:

```
<root>
    <params>
        <max_angular_velocity value=""> </max_angular_velocity>
        <max_angular_acceleration value=""> </max_angular_acceleration>
        <angular_arrive_radius value=""> </angular_arrive_radius>
        <targetRotation value=""> </targetRotation>
    </params>
</root>
```

Los ángulos se especifican en grados

Práctica 4

Utilizad un **steering behavior de Arrive** para el movimiento y un **steering behavior AlignToMovement que use Align** como **delegado** para la rotación de forma que el personaje mire hacia donde se mueve.

Los parámetros también se especifican en un fichero params.xml igual que la práctica anterior.

Formato:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <arrive_radius value=""> </arrive_radius>
        <targetPosition x="" y=""> </targetPosition>

        <max_angular_velocity value=""> </max_angular_velocity>
        <max_angular_acceleration value=""> </max_angular_acceleration>
        <angular_arrive_radius value=""> </angular_arrive_radius>
        <targetRotation value=""> </targetRotation>
    </params>
</root>
```

Opcionales

Práctica 5

En esta práctica hay que implementar dos personajes: un enemigo y un personaje principal

Hay que implementar el comportamiento para estos dos personajes

Enemigo

Éste puede ser gestionado directamente por el Actor que estamos implementando para el personaje 2 y su pintado puede hacerse con un circle que se mueve por la pantalla con las funciones de debugdraw.h

Opción 1 (más fácil)

- Implementad un personaje que se mueva a lo largo del eje X de un lado a otro de la pantalla, con la coordenada Y fija a -200.

Opción 2 (más interesante)

- Pensad como implementaríais un personaje que se dedique a moverse de forma aleatoria por la pantalla, como si estuviera explorando o paseando.

Personaje principal

El personaje 2 utiliza el **steering behavior de Pursue** para perseguir al personaje 1. Cuando llega a la posición de personaje 1 se resetea su posición y vuelve al 0,0.

Opción 1:

params.xml:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <arrive_radius value=""> </arrive_radius>
        <targetPosition x="" y=""> </targetPosition>

        <max_angular_velocity value=""> </max_angular_velocity>
        <max_angular_acceleration value=""> </max_angular_acceleration>
        <angular_arrive_radius value=""> </angular_arrive_radius>
        <targetRotation value=""> </targetRotation>

        <enemy_speed value=""></enemy_speed>
        <enemy_minPosition x="" y=""></enemy_minPosition>
        <enemy_maxPosition x="" y=""></enemy_maxPosition>
    </params>
</root>
```

Opción 2:

params.xml:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <arrive_radius value=""> </arrive_radius>
        <targetPosition x="" y=""> </targetPosition>

        <max_angular_velocity value=""> </max_angular_velocity>
        <max_angular_acceleration value=""> </max_angular_acceleration>
        <angular_arrive_radius value=""> </angular_arrive_radius>
        <targetRotation value=""> </targetRotation>
    </params>
</root>
```

enemy_params.xml:

```
<root>
    <params>
        <max_velocity value=""> </max_velocity>
        <max_acceleration value=""> </max_acceleration>
        <arrive_radius value=""> </arrive_radius>
        <targetPosition x="" y=""> </targetPosition>

        <max_angular_velocity value=""> </max_angular_velocity>
        <max_angular_acceleration value=""> </max_angular_acceleration>
        <angular_arrive_radius value=""> </angular_arrive_radius>
        <targetRotation value=""> </targetRotation>
    </params>
</root>
```