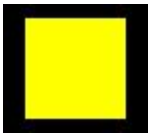
Bucle de Simulación
Y
Movimientos Básicos

Alumno: Lucas Merenciano Martínez
Asignatura: Inteligencia Artificial
Profesor: Antonio Barella
Fecha: 09 / 12 / 2019

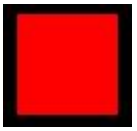
Índice

Sobre la práctica

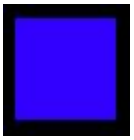
Junto a esta documentación, entrego el proyecto de un ejecutable que muestra cuatro agentes (representados por cuadrados fantasmales sin colisión), cada uno de un color, que se mueven por la ventana. Cada uno representa un tipo de movimiento.



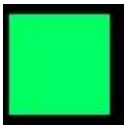
El cuadrado amarillo fantasma tiene asignado un movimiento determinista cuyo camino describe un cuadrado por los bordes de la ventana.



El cuadrado rojo fantasma tiene asignado el movimiento persecutorio y su objetivo es el cuadrado amarillo fantasma.



El cuadrado azul fantasma tiene asignado un movimiento por patrón, el cual le hace subir, bajar, ir a la derecha y por último ir a la izquierda. Este patrón le hace describir una L.



El cuadrado verte fantasma tiene asignado el movimiento aleatorio, por lo que irá desplazándose de un punto aleatorio dentro de la ventana a otro.

La práctica realizada tiene dos pilares fundamentales: el bucle de simulación y el movimiento de los agentes.

El bucle de simulación

Para que la velocidad a la que una máquina ejecuta el bucle de un videojuego no influya en la velocidad de movimiento de los agentes del mismo, medimos el tiempo que tarda el bucle en ejecutarse y multiplicamos la velocidad de los agentes por ese valor. De ese modo cuando el bucle ha tardado más en ejecutarse, el agente se moverá una distancia directamente proporcional a ese valor.

Esta solución es válida, pero nos crea un nuevo problema: un agente, tras un bucle especialmente lento, podría atravesar materiales sólidos ya que habría avanzado demasiado espacio sin realizar ningún cálculo de físicas.

Afortunadamente es aquí donde viene al rescate el maravilloso bucle de simulación, permitiendo que, aunque el tiempo a simular sea muy elevado tras un “frame” especialmente costoso, los cálculos de físicas se ejecuten cada cierto tiempo fijado previamente (en el caso del código de mi práctica, a ese tiempo lo llamo “time_step”).
De este modo controlamos mucho mejor la simulación y detectamos a tiempo cualquier evento producido en ella sin importar la cantidad de tiempo acumulado que nos quede por simular. Cabe destacar que para el correcto funcionamiento de esta técnica, debemos ser capaces de simular, en condiciones normales, cualquier cantidad de tiempo a más velocidad que el avance del propio tiempo acumulado.

Conocer con tal exactitud el tiempo acumulado y el tiempo que simulará cada bucle, nos permite realizar ciertos “trucos” o técnicas que ayudarán a un mejor funcionamiento del videojuego, pero en el estado actual de la práctica no se han implementado aún.

El movimiento de los agentes

Ahora que conocemos como funciona el bucle de simulación sabemos que debe multiplicarse el tiempo que se simula por la rapidez de el agente que movemos. Yo, en mi caso, empleo esta fórmula:

*Position = Position + (Velocity * Speed * DeltaTime)*

La expreso en inglés para diferenciar Velocity (el vector que dirige el movimiento) de Speed (la rapidez a la que se avanza).
Para calcular la posición del agente, calculamos el movimiento realizado durante la simulación y se lo añadimos a la posición que tenía previamente.
Dicho movimiento se calcula multiplicando la dirección del movimiento (Velocity) por el factor de rapidez (Speed, que es el resultado de multiplicar la rapidez base de los agentes por el factor de rapidez del agente que estamos moviendo), habiendo normalizado el vector de movimiento previamente, y por último multiplicamos por el tiempo que estamos simulando (DeltaTime, que será siempre fijo, de hecho es el valor de time_step que comentaba al explicar el bucle de simulación).

Ahora que Position, Speed y DeltaTime han quedado claros, me queda explicar cómo se obtiene Velocity.

Velocity es un vector normalizado que nos indica la dirección en la que un agente debe avanzar. Para calcular dicho vector, es necesario conocer el tiempo de movimiento que está empleando el agente.

Los tipos de movimiento

El movimiento determinista

Decimos que un movimiento es determinista cuando le indicamos al agente las posiciones exactas a las que debe desplazarse. Para este tipo de movimiento es aconsejable utilizar una estructura de datos que nos permita almacenar toda la información necesaria para recorrer un camino (en el caso de mi código es la clase Path).

El movimiento por patrón

Para este movimiento no se utilizan posiciones exactas, en vez de eso al agente se le indica una dirección y el tiempo o distancia que deberá recorrer antes de cambiar a la siguiente dirección. En mi caso de momento únicamente utilizo cuatro direcciones (arriba, abajo, a la derecha y a la izquierda).

El movimiento persecutorio

El clásico movimiento de persecución. A un agente se le indica una entidad a la que debe perseguir. Para dar una impresión de persecución más humana, en mi caso el vector de velocidad no es un vector directo entre el agente y la entidad, sino que el vector de velocidad lo calculo interpolando el vector directo entre ambos y el vector de velocidad que está siguiendo el agente actualmente. El punto exacto de dicha interpolación lo definimos nosotros (en caso de no hacerlo el valor por defecto es 0.5, lo que equivale a un vector a medio camino entre ambos).
Con este mismo tipo de movimiento también podemos realizar huidas, simplemente debemos invertir el vector de velocidad resultante.

El movimiento aleatorio

Cualquier agente que emplee este tipo de movimiento, avanzará hacia un punto aleatorio en el mapa, y cuando llegue a él, calculará el siguiente de forma aleatoria también y así sucesivamente. Cabe destacar que los puntos no serán aleatorios en el sentido más estricto de la palabra, ya que previamente se acota un área para que no salgan de ella.