Proyecto individual Videojuegos 1

El objetivo de este proyecto era realizar un juego parecido al Raven, juego descrito en el capitulo 7 del libro “Programming Game AI by Example”. Debido a innumerables problemas de compatibilidad, inexperiencia y falta de tiempo, no se pudo realizar el proyecto adecuadamente.

El primer paso, obviamente, era leer el capitulo pertinente para ver las especificaciones del juego. Tras haberlo leído, procedí a empezar a trabajar, aunque también seguí leyendo el libro para aprender más sobre IA.

El profesor nos explicó que íbamos a trabajar en Manjaro, una versión de Linux. Al principio, intenté crear una partición en mi disco duro para poder tener Windows y Manjaro en un mismo equipo. Sin embargo, el proceso era muy complicado. Incluso con diversos tutoriales, fui incapaz de hacerlo funcionar. Al final, fui a la segunda opción: máquina virtual de Virtualbox.

Aquí surgió un problema inesperado: Ya había trabajado con Virtualbox en el pasado, pero resulta que había actualizado mi equipo hace poco, y accidentalmente creé una máquina virtual con ajustes incorrectos, lo cual llevó una pérdida de tiempo considerable.

Tras configurar la máquina virtual correctamente, tuve que acostumbrarme a los comandos de Manjaro, que son ligeramente distintos a los de Ubuntu, que es con lo que estaba más familiarizado en lo que se refiere a programar. Con un poco de ayuda de Internet, conseguí instalar Netbeans y SFML, para ayudarme con el desarrollo del juego.

Al fin empecé a trabajar. Primero configuré diversas funciones simples que me harían falta en el futuro, como detectar cuando se pulsan ciertas teclas, o la posición del ratón en la ventana de SFML.

El siguiente paso era crear la clase enemigo. Empecé con algo simple, así que cree un círculo con SFML e hice que se moviera en base a un int y un switch. Una máquina de estados supersimple.

Tras pelearme un poco con el vector de enemigos del main para que todo se dibujara correctamente, pase a pensar en el movimiento. Decidí crear una clase nodo, que tendría coordenadas que el enemigo usaría para moverse por la pantalla de forma más inteligente. Le di al enemigo un vector con punteros a nodos, que usaría como lista de destinos. El resultado fue un éxito. Obviamente aun era simple (los nodos se creaban a mano en main e iba en línea recta), pero pensé que me valdría como concepto y que lo mejoraría más tarde. La falta de tiempo me impidió desarrollar el sistema de nodos mas allá de lo explicado.

El siguiente paso eran los disparos. Obviamente son la clave del juego, y por eso era muy importante que se realizaran bien. El primer paso era darle al enemigo un arma (un rectángulo), que básicamente sirve como los ojos del enemigo (o sea, que siempre mira donde está disparando). Tras modificar los ajustes del origen del arma para que se dibujara correctamente, me encontré con un problema grave: no sabía “calcular” los giros. Investigando por Internet, encontré algunas formulas para calcular un giro en base a la posición actual y a la posición del cursor. Aunque eso serviría para cuando el jugador controle a alguien, no vale para la IA porque no tiene cursor.

Tras pensarlo detenidamente, me di cuenta de que la formula de Internet usaba dos vectores. Eso quiere decir que solo tengo que simular un cursor dándole a la IA un vector de dirección. Tras realizar diversos cálculos matemáticos para comprobar cómo obtener un vector genérico para cualquier dirección, conseguí implementar la fórmula. Tras crear una clase bala y unirla al enemigo, además de corregir diversos errores de cálculo, conseguí que la bala fuera en la dirección indicada por la rotación de la pistola cuando se crea.

Pensaba que ya lo tenía todo bien, pero resulta que la fórmula actual, por algún motivo, no sirve para girar durante la ejecución del programa. Los resultados obtenidos no concuerdan, y eso hace que la pistola se quede con rotaciones extrañas.

Al final el tiempo se me echó encima y no conseguí descubrir la fórmula correcta, así que dejé el comportamiento de la bala que funcionaba bien, y metí la formula incorrecta en un comentario. Llegados a este punto, estaba claro que el juego no iba a salir bien, pero aun así intenté hacer una cosa más: una máquina de estados bien desarrollada.

La idea era buena, según mis conocimientos de máquinas de estados: El enemigo tendría una máquina de estados (ME), que se crea junto al enemigo. La ME tendría un puntero al enemigo al que pertenece. La ME tendría un vector con punteros a diversos estados, indicando los posibles estados del enemigo. Luego, habría una clase Estado, que tendría un int de id, y diversos métodos virtuales. Luego, cada estado sería su propia clase, heredando de Estado y con sus propios métodos. Eso quiere decir que el enemigo solo tendría que hacer ME.update(), y la ME se ocuparía de todo. La ME haría update del estado activo en ese momento, el estado haría todos los cambios y modificaciones pertinentes, y si hace falta, informa a la ME de que cambie el estado.

Claramente el programa no estaba de acuerdo con mi idea. Para cuando lo tenía todo puesto, no me dejó compilar. Por algún motivo, no aceptaba la herencia de la clase Estado en los estados hijos. No tengo claro si el problema era la herencia, o me decía eso por una dependencia cíclica debido a los includes necesarios para que mi idea salga bien. De todas maneras, no pude resolverlo y tuve que volver a la máquina de estados simple, la del int y el switch. Aun así, dejé los archivos de esta máquina de estados en el código, dentro de comentarios, para que se vea lo que he trabajado.

Como última oportunidad, investigué diversos métodos de pathfinding, pero ya no me quedaba tiempo para implementarlos.

Resumiendo, esto es lo que tiene el “juego”:

-Enemigos con pistolas. Realizan un único disparo en la dirección en la que miran (dirección de rotación de la pistola). Uno se mueve de forma simple. El otro implementa una lista de nodos con coordenadas simples, y luego realiza un movimiento simple.

-Se detecta si se pulsan ciertas teclas (las indicadas en las especificaciones de Raven)

-Se detectan diversos detalles sobre el cursor (cuando entra o sale de la ventana, su posición, cuándo hace clic y dónde)