Memoria: Juego con Predicción



Gabriel Laguna de la Puente

Manuel Cabañas Álvaro

Videojuegos 4-3

Inteligencia Artificial

Ángel Luis Marinas

01/02/2018

**Descripción del proyecto**

La entrega consiste en un juego en el que se ha de elegir entre cuatro posibles opciones (ataque alto, ataque bajo, defensa alta y defensa baja). Una vez el jugador hace su elección el jugador rival, controlado por una inteligencia artificial, realiza la suya y se resuelve cual es el ganador. Está basado en el sistema de juego de piedra papel tijera.

Si ambos jugadores eligen defenderse no ocurre nada, si por el contrario ambos atacan y el ataque es dirigido a diferentes partes, arriba y abajo, entonces se produce un doble KO. Para ganar la partida es necesario que uno de los contendientes se defienda y el otro ataque por aquel lugar donde no se ha defendido.

Una vez finalizada una partida es posible reiniciar el juego y jugar de nuevo.

**Estructura de clases**

**class Predictor:**

Guarda las secuencias de movimiento hechas por el jugador y genera el movimiento más probable para contraatacar al del jugador rival.

**class IAController:**

Contiene al predictor y utiliza el mismo para decidir qué movimiento le enviara al GameController.

**class GameController:**

Clase encargada de controlar el flujo de juego y decidir quien gana y quién pierde. llama a los métodos de AnimatorManager para lanzar las animaciones.

**class DataRecord:**

Clase encargada de almacenar los datos de cada movimiento del usuario y el número de veces que se realizan los mismos.

**class AnimatorManager:**

Clase contenedora de los métodos necesarios para lanzar las animaciones y controlar los estados de animación de los personajes. Contiene también referencias a los animadores de cada uno de los citados personajes.

**Instrucciones**

La elección de movimiento se lleva a cabo con las flechas de dirección siendo las verticales las indicativas de la altura del movimiento y las horizontales las que marcan que de que movimiento se trata. La tecla “R” sirve para reiniciar la partida.

**Enlaces de interés**

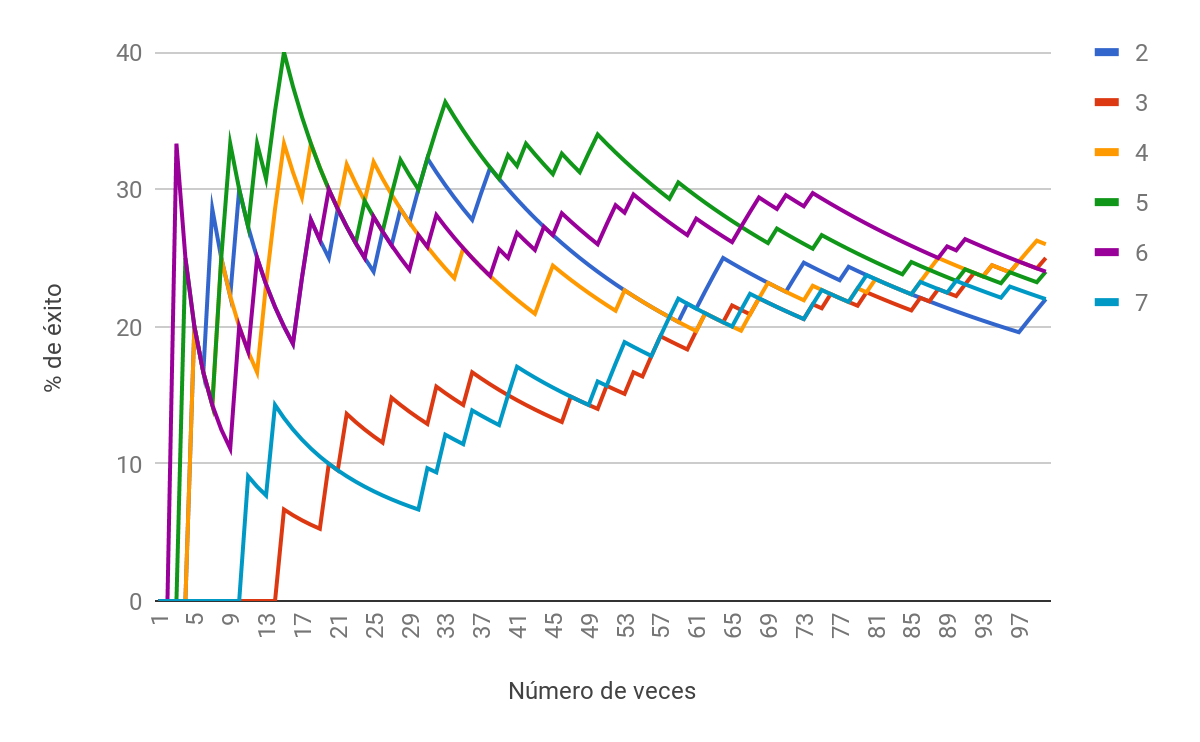
<https://www.spriters-resource.com/>

<http://evo.shoryuken.com/>

<https://en.wikipedia.org/wiki/Rock%E2%80%93paper%E2%80%93scissors>

<http://mundogeek.net/archivos/2009/02/20/piedra-papel-tijeras-lagarto-o-spock/>

**Análisis de datos**



Para realizar el análisis de datos se ha creado un script que genera las acciones del jugador aleatoriamente un número de veces dado, en este caso 100, para conocer el porcentaje de aciertos a medida que avanza el tiempo de juego, con diferentes tamaños de ventana.

Dicha información reflejada en el gráfico anterior nos muestra que al inicio el porcentaje de acierto es dispar entre los diferentes tamaños de ventana, y que a medida que pasa el tiempo la disparidad tiende a estabilizarse.

Los valores a su vez tienden a ser bajos debido al uso de una generación de acciones pseudoaleatoria y no a la de un jugador humano, hecho que al contrario de lo ocurrido crea acciones más predecibles que las que se han presentado durante la prueba como hemos comprobado al realizar las mismas acciones repetidamente.

**Futuras mejoras**

Para la mejora de este proyecto se podría tener en cuenta las animaciones de cada uno de los jugadores, de tal manera que tenga la misma velocidad para el mismo tipo de movimiento, algo que no se hay podido solucionar por las diferencias del número de frames entre los dos personajes en sus movimientos, consecuencia de haber cogidos los sprites de fuentes distintas. También se podría usar un sistema de control más adecuado, ya que hay veces que al pulsar dos botones al mismo tiempo no se produce ninguna reacción. Esto es por la precisión que parece que es necesaria para que el juego detecte estos inputs al mismo tiempo. También se ha pensado en nuevos tipos de movimiento para contrarrestar los ya existentes para hacer el juego mucho más equilibrado, como un stun al recibir varios ataques consecutivos defendiendo (para evitar el uso excesivo del bloqueo), y una barra de vida en vez de únicamente el sistema de conteo de victoria, haciendo mucho más cercano a un juego de lucha.

**Cambios realizados tras la revisión**

Tras la primera revisión del proyecto se ha revisado el principal problema que se ha encontrado: que la IA no aprendía los movimientos del usuario, haciendo los movimientos totalmente aleatorios. Se investigó todo el proceso buscando el problema, y se encontró algo que no habíamos caído cuando se entregó la primera vez: que no se distinguían las llaves de los diccionarios, de tal manera que nunca se daba cuenta de que ya existía esa llave, generando siempre llaves con los mismos movimientos. Esto se debe al uso de listas como llaves, que el diccionario no es capaz de comparar. Al principio se preparó un IEqualityComparer personalizado que se pasase al constructor de la lista para que comparase correctamente, pero no funcionó como se esperaba. Al final se optó por una opción más sencilla: pasar las opciones a simples char (1,2,3,4) y el conjunto de los movimientos a strings. De esta manera se asemeja más a lo que el profesor enseñó con los proyectos realizados en clase, e hizo que todo funcionase correctamente.