Segundo Proyecto Programado

Estudiante: Jose Alfredo Gamboa Román

Carné: 2019196401

Profesor: Diego Mora

Curso: Taller de programación

Primer Semestre 2019

Fecha de entrega: 27/5/2019

Tabla de contenidos:

Introducción

En este proyecto programado fue realizado un videojuego de cuatro en línea con interfaz gráfica en Python mediante la utilización de matrices. El juego difiere en algunos aspectos de la versión original dado que la matriz es infinita, para la interfaz gráfica se utilizó pygame y diferentes archivos multimedia. El programa consta de 3 secciones singleplayer, multiplayer y rankings. En el primero se utiliza un algoritmo de IA contra el que se juega, el segundo es un modo multijugador en el que 2 oponentes se enfrentan y por último la sección de rankings muestra los primeros 10 puestos de las partidas mutiplayer.

Descripción del problema

Para el segundo proyecto programado debe implementar en Python un juego de 4 en línea, sin embargo, difiere en algunas características del juego original. A continuación, se explican las características del proyecto:

1. El objetivo del juego, la cantidad de jugadores (2), la forma de los turnos entre jugadores, se mantienen.

2. La matriz no será de 6X7 únicamente, sino que es infinita. La matriz inicia con 6x7, pero los jugadores podrán colocar la ficha entre las 7 columnas o bien, a la derecha o izquierda 7 posiciones de cualquier ficha y la matriz lo permitirá, pues es infinita hacia los lados, pero con esa restricción, solo se puede colocar fichas en columnas que estén a una distancia máxima de 7 columnas de una ficha.

Hacia arriba la matriz también es infinita.

1. Representación gráfica: en pantalla siempre debe verse 6X7 fichas y tendrá que implementar un “scroll“ para ver la matriz si ha crecido, sin embargo, para más facilidad, solo se verán 6x7 en todo momento.

Importante es que las columnas y filas deben estar siempre numeradas, para saber en todo momento en cuál posición de la matriz se está.

El programa debe ser gráfico, mostrar colores de fichas según el estado de la matriz.

También debe mostrar cuál es el jugador que le corresponde el turno.

2. Jugadores: solamente pueden participar dos jugadores, donde cada uno tendrá un color de ficha. Los turnos son intercalados. En la pantalla debe mostrarse turno.

Antes de iniciar el juego, debe solicitarse el nombre de ambos jugadores.

3. Juego contra la computadora: puede jugar un solo jugador contra la computadora. Este modo de juego tendrá que implementar un algoritmo que haga el segundo turno. Debe ser un algoritmo que se base en colocar la ficha donde no se pierda, es decir, donde el otro jugador no pueda formar cuatro en fila. Cuidado porque habrá ocasiones en que no hay alternativa y se le podría “enciclar“ el algoritmo.

4. Puntajes: debe tener un archivo donde guarde los puntajes (cantidad de juegos ganados) de todos los jugadores del juego. Esto adicional a una pantalla donde se puedan consultar los puntajes, ordenados del mayor a menor.

5. Guardar partida: debe permitir guardar una partida no finalizada y luego poder jugarla nuevamente en el estado en el cual quedó. Para esto utilice archivos para almacenar en disco las partidas.

Análisis de resultados:

El juego avanza correctamente por turnos intercalados entre los 2 jugadores ya sea el sigleplayer o multiplayer.

La matriz es infinita y puede crecer correctamente hacia la derecha, izquierda o hacia arriba, además de esto solo se puede jugar a máximo 7 columnas de distancia de una ficha.

Siempre se muestra 6x7 en la interfaz grafica y se puede realizar correctamente el scroll en caso de que la matriz termine esta se expandirá hacia el lado que se intenta hacer scroll a excepción de hacia abajo que al llegar al fondo no bajara más todas las columnas y filas están enumeradas con el fin de saber en que parte de la matriz se está jugando (los índices inician en 0).

El programa cuenta con una gran cantidad de gráficos incluidas las piezas que son de diferente diseño y en cada turno muestra el nombre del jugador.

Al iniciar los modos singlepayer y multiplayer se solicita el nombre de el o los jugadores que participaran en la partida.

Se implemento un algoritmo de IA que mediante la utilización de probabilidades determina el mejor movimiento que puede realizar el sistema en el modo singleplayer el cual funciona correctamente y juega a ganar y bloquear los movimientos del rival.

Cada vez que un jugador gana una partida del modo multiplayer este gana un punto (Toda la información de los puntajes se guarda en un txt.), si el jugador se encuentra entre los 10 jugadores con más puntos este aparecerá en la tabla de rankings

Se puede guardar una partida en el modo singleplayer y en el modo multiplayer para poder continuarla luego, al iniciar nuevamente el modo de juego se preguntará si se desea continuar la partida anterior o no en caso de seleccionar no esta partida no se podrá recuperar nuevamente. Al finalizar la partida esta no podrá ser recuperada tampoco.

Dado que la matriz es infinita y el algoritmo que se utiliza para determinar el siguiente movimiento del oponen ente revisa cada ficha de la matriz este puede sobreexceder la capacidad del sistema.

Conclusiones:

-El proyecto fue realizado correctamente según las instrucciones dadas, ambos modos de juego funcionan correctamente y la tabla de rankings también, se puede guardar partidas y continuarlas luego.

-Los rankings y las partidas guardadas son almacenados en la memoria secundaria, esto con el fin de poder recuperar la información en un futuro: mediante la implementación de un sistema de archivos txt.

-El algoritmo de para el singleplayer funciona correctamente y bloquea los movimientos del oponente mientras sea posible. Sin embargo, al utilizar un sistema que verifica toda la matriz para poder predecir cual es el mejor movimiento, este consume muchos recursos de memoria y procesamiento.

-Dado que el juego cuenta con una matriz infinita los recursos de procesamiento y memoria que puede llegar a ocupar el juego es muy grande y al expandir la matriz mucho el turno del sistema dura mucho tiempo en determinar el siguiente movimiento e incluso puede llegar a sobreexceder la capacidad del sistema.

Estadística de tiempos:

Análisis de requerimientos	5 horas
Diseño de la aplicación	20 horas
Investigación de funciones	25 horas
Programación	30 horas
Pruebas	11 horas
Elaboración documento	1 hora
TOTAL	**92 horas**