

Universidad Simón Bolívar

Departamento de Computación y Tecnología de la Información

Laboratorio de Algoritmos y Estructuras I CI-2691

Prof. Carmen Rosseline Rodríguez

Especificaciones del Proyecto

4 En Línea

Integrantes:

José Barrera. Carnet: 15 - 10123

Alfredo Cruz. Carnet: 14 – 10261

Sartenejas, abril de 2018

Introducción

El juego “4 En Línea” tiene como finalidad colocar cuatro fichas en línea (horizontal, vertical o diagonal) en un tablero vertical. En cada turno el jugador, deja caer una ficha en alguna de las 7 columnas. Cuando se encuentran alineadas 4 fichas del mismo jugador este queda ganador del juego y se concluye la partida actual. Sin embargo, puede llenarse el tablero y que no se logre ninguna conexión en cuyo caso se declara un empate. O puede que el jugador decida retirarse y automáticamente conceder la victoria al rival.

El problema a resolver consiste en lograr desarrollar en el lenguaje de programación Python y en el pseudo-lenguaje GCL, una adaptación del juego 4 En Línea. Para hallar la solución al problema implementaremos el código utilizando la estructura de análisis descendente, partiendo de una carta estructurada (que se anexará más adelante). Ésta permite tener una presentar una perspectiva más clara del código, pues divide y jerarquiza las tareas a realizar en pequeños procedimientos y funciones más fáciles de entender. Los cuales nos permitirán vislumbrar la solución del problema de forma más clara y organizada.

Descripción de la finalidad de las definiciones de tipos de datos:

Para hacer más legible el código y facilitar su lectura, decidimos establecer los siguientes tipos de datos al inicio del juego:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Descripción** |
| nombreusuario | String/Entrada | Nombre del Usuario |
| nivel | Integer/Entrada | Dificultad de la Máquina |
| partida | Integer/Entrada | Empezar Una Nueva Partida(0), Cargar Una Anterior(1) o Salir del Juego (else) |
| seguir | Bool/Entrada | Si se desea rendirse |
| x | Integer/Entrada | Fila donde el usuario desea jugar |
| y | Integer/Entrada | Columna donde el usuario desea jugar |
| A | List | Matriz Tablero de Juego |
| G | List | Tablero de Resultados |
| jugando | Bool | Controla el ciclo de jugar partida |
| turno | Integer | Contador de turnos de la partida |
| juegauser | Integer | Si le toca jugar al usuario |
| ganador | Integer | Ganador de la partida finalizada: Empate(0), Usuario(1) o Máquina (2) |
| dentrodeljuego | Bool | Controla el ciclo del juego |
| movida | Bool | Controla el ciclo de intentar una juagada para el Nivel 1 |
| i | Integer | Fila donde la Máquina juega |
| j | Integer | Columna donde la Máquina juega |
| datosdejuego | Class | Atributos: nombre, turno, nivel, A, i, j (valores de juego) |
| anterior | datosdejuego | Almacena los datos de juego de una partida anterior. |
| actual | datosdejuego | Almacena los datos de juego la partida actual. |

Objetivos de los subprogramas de la carta estructurada:

Juego/Obtener Jugada/jugadaUser : La función jugadaUser, recibe como argumento el tablero de juego A, solicita al usuario la fila y la columna que desea jugar, si son válidas modifica el tablero de juego lógico y gráfico.

Juego/Obtener Jugada/GuardarJuego : La función GuardarJuego, recibe como argumento un archivo de texto (guardado.txt es el predeterminado) donde escribir los datos de juego actuales, contenidos en el segundo argumento (una variable miembro de la clase datosdejuego, actual es el predeterminado). No tiene datos de salida.

Inicializar Partida/Cargar Partida/CargarJuego : La función CargarJuego, recibe como argumento recibe un archivo de texto (guardado.txt es el predeterminado) desde donde lee la información de una partida anterior y la escribe en una lista, donde cada elemento es un dato de juego.

Juego/Obtener Jugada/Máquina/Dificultad Media/ IA : La función IA, recibe como argumento el tablero de juego A, la fila de la jugada anterior de la máquina i y la columna de la jugada anterior de la máquina j. Retorna el tablero de juego A con la jugada realizada, la fila de la jugada que acaba de realizar i y la columna de la jugada que acaba de realizar j.

Juego/Validar Jugada : Cuando se quiere saber si una posible jugada es válida, para luego ejecutarla, se deben evaluar sus coordenadas en la función valida(), la cual retorna verdadero o falso dependiendo de si dicho movimiento cumple o no las reglas del juego.

Juego/Verificar 4 en Línea : Cuando se quiere saber si se ha formado una línea ganadora en el tablero, se pasan como argumento a la función victoria(), la matriz del tablero A, el bool jugando, el entero ganador y las coordenadas de la última jugada i, j. Se retorna la matriz del tablero, el bool jugando y el entero ganador modificados si había efectivamente una línea ganadora en caso contrario se dejaban que como se recibieron.

Implementación del programa:

• Creamos una pantalla de inicio para que el programa se viera más organizado. • Decidimos separar al usuario participante de las máquinas creando un registro “Jugador” que lo represente y para la máquinas, tenemos un registro de ‘’Maquina” y un arreglo donde se almacenan todas las máquinas participantes. El arreglo de máquinas fue creado porque pensamos que sería más conveniente diferenciarlas del usuario. • Decidimos hacer una matriz de habitaciones, donde solo están las posiciones válidas puesto que los jugadores nunca pueden estar en los pasillos. Para calcular la manera en que el jugador puede caer en una posición, utilizamos un registro llamado “Punto” ya que al restar las coordenadas x e y de la posición inicial con la posición elegida y sumar sus valores absolutos se obtiene el número necesario para moverse a cierta habitación. Si el valor de ese alcance es menor al del dado se le muestra a los jugadores las habitaciones a las que se pueden mover para que elijan una correcta.

Manera en que la máquina hace la toma de decisiones:

Para el turno de la máquina, las jugadas siguen 2 estrategias bien diferenciadas de acuerdo al nivel de dificultad (1 o 2) que selecciona el usuario:

Nivel 1: Se intenta un par de coordenadas random hasta que estas se ubiquen en una casilla válida para jugar. Entonces se ejecuta la jugada.

Nivel 2: La máquina en su primer turno intenta hacer la jugada fila 5, columna 3. Si el jugador la realiza primero entonces hace la jugada fila 5, columna 2. A partir de la primera jugada, construye su estrategia en base a la jugada anterior. Busca las jugadas validas en las 7 direcciones posibles y ejecuta aquella dirección y sentido con más jugadas validas en un alcance de 3 espacios contiguos. Si la jugada anterior se encuentra rodeada, es decir, todas las 7 direcciones y sentidos posibles se evalúan 0, se busca y ejecuta la primera jugada valida obtenida recorriendo la matriz de abajo hacia arriba y de derecha a izquierda.

Conclusiones

La realización del presente proyecto de laboratorio, nos permitió implementar la estructura de análisis descendente para resolver un problema de complejidad sin precedentes dentro del curso. Durante su desarrollo, se hizo evidente que el dividir la tarea en pequeños problemas que trabajan de forma interconectada, realmente permite atacar los problemas de forma más eficaz. Esto debido a que se puede analizar cada componente como una nueva mini-tarea, enfocando la capacidad de resolución de problemas en algo más simple, que incluso pudiese aplicarse en futuras macro-tareas.

Nos dimos cuenta, además, de que el análisis descendente es un proceso que no es ajeno a los cambios, pues las primeras soluciones a cada mini-tarea, por lo general no escapaban de mejoras en pro de simplificar el código, hacerlo más legible e inclusive agregarle nuevas funcionalidades al programa.

En cuanto a los algoritmos y las estructuras de datos, podemos concluir que fijar las herramientas adecuadas es determinante para desarrollar soluciones elegantes a problemas concretos. Un aspecto básico que, de descuidarse, deviene en una pérdida de tiempo para comprender el código tanto por los lectores como por el propio desarrollador.

Referencias

* https://www.hasbro.com/es-es/customer-service/toy-game-instructions