Universidad Nacional ARTURO JAURETCHE

TRABAJO PRÁCTICO FINAL

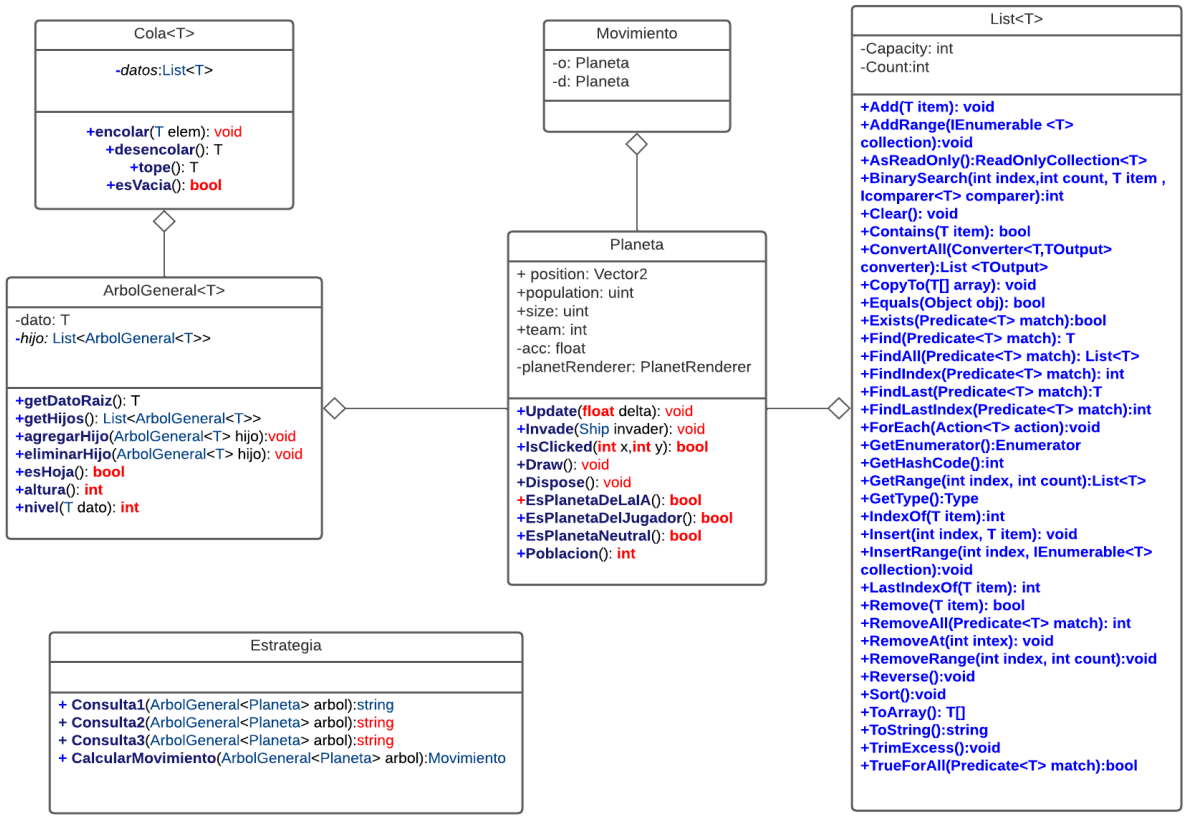
CONQUISTA PLANETARIA

- MATERIA: COMPLEJIDAD TEMPORAL, ESTRUCTURAS DE DATOS Y ALGORITMOS
- ALUMNO: CLAUDIO BERNAL
- COMISIÓN N°3
- FECHA: 27-11-2023

Introducción:

En este trabajo practico tengo que implementar la estrategia del bot en un videojuego. El objetivo del juego es apoderarse de todos los planetas del rival. Los planetas tienen una cantidad específica de naves y se dividen en tres colores: rojo para el jugador, azul para el bot y blanco para los que son neutrales. Además, cada planeta tiene un conjunto de rutas interplanetarias que los interconecta y que sirven para enviar flotas de un planeta hacia otro. Los planetas cuentan con una tasa de crecimiento que indica cuantas naves pueden generar durante cada asalto, que luego serán agregadas a la flota que el jugador posee en el planeta. Los planetas neutrales no agregan nuevas naves a la flota que alojan hasta que sea conquistado por el jugador o el bot. El jugador debe elegir el movimiento que realizará una flota que partirá de un planeta propio a cualquier otro planeta destino, pudiendo ser este último propio o no. Los movimientos serán posibles siempre y cuando exista una ruta interplanetaria directa entre un origen y un destino. Cuando la flota llega un planeta enemigo o neutral, tiene lugar un enfrentamiento, en el cual cada nave es sacrificada para destruir una nave enemiga, o sea que gana la flota que tenga más naves. En caso de que el planeta de destino sea del propio jugador, ambas flotas se unen, sumando sus naves, lo mismo se aplica al bot. En cada turno, el número de naves alojadas en los planetas del jugador y del bot (no los neutros), se incrementa de acuerdo con la tasa de crecimiento de cada planeta.

Las clases y sus relaciones están representadas por el siguiente diagrama UML:

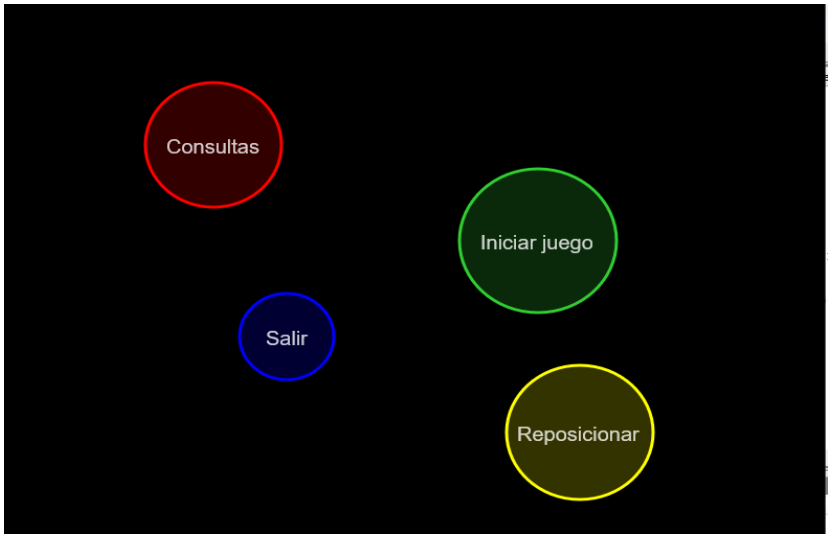


Problemas encontrados y su solución

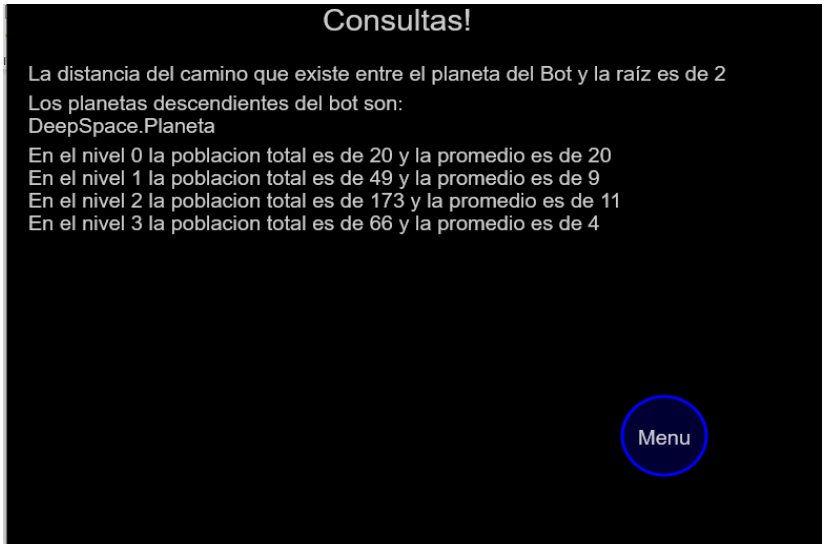
En el método "consulta2()" cuando el planeta del bot era una hoja me devolvía solo el texto "Los planetas descendientes del bot son: ", entonces cuando evalúa si el árbol auxiliar es planeta de la IA, además, dentro le agregue otro condicional para que retorne otro texto en caso de que este árbol sea hoja.

En el método "consulta3()" solo me devolvía la población total y el promedio del nivel 0, 1 y 2. Esto era porque, una vez que terminaba de recorrer el nivel 3, la cola quedaba vacía. Entonces lo que hice fue agregar un condicional después de que encole los hijos, que evalúa si la cola está vacía y si lo está que calcule el promedio y agregue el resultado de este nivel.

Interfaz gráfica:



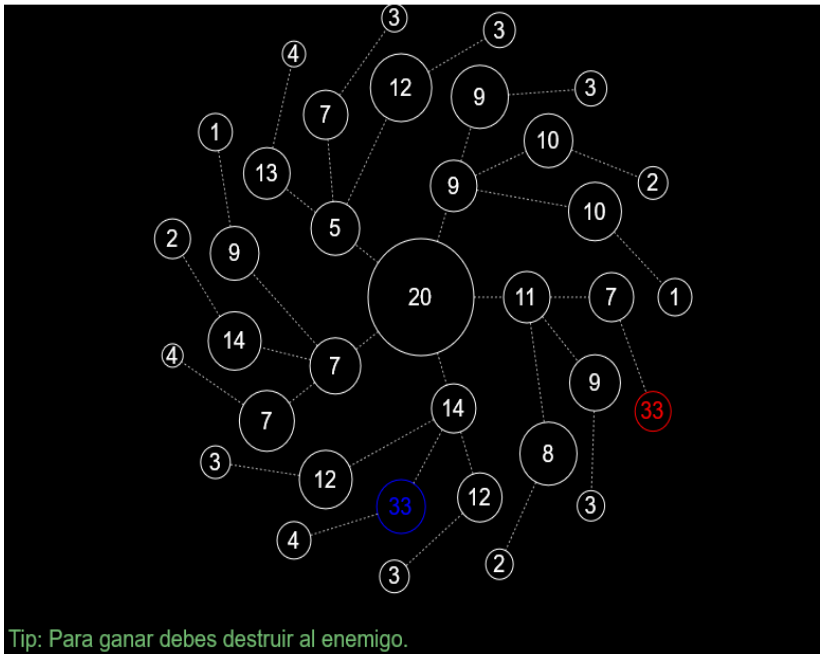
Este es el menú principal, donde se pueden seleccionar las opciones: "consultas", "iniciar juego", "reposicionar" y "salir".



Esto es lo que se muestra al seleccionar la opción "Consultas" del menú principal.

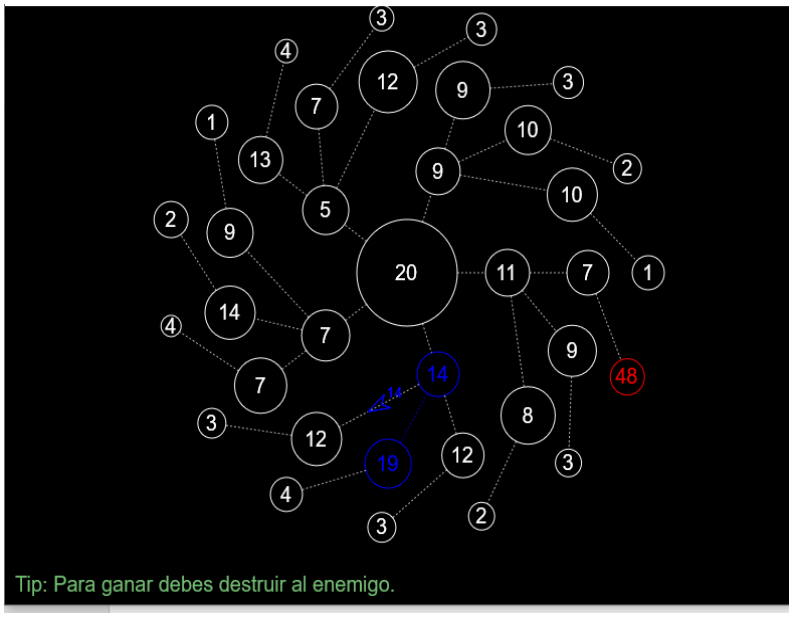
Estos son los strings que retornan los métodos "Consulta1()", que es la distancia del camino entre el planeta del bot y la raíz, "Consulta2(), los planetas descendientes del bot, y "Consulta3()" la población total y la promedio por cada nivel del arbol.

Esto se muestra al elegir la opción “Reposicionar” que lo que hace, como lo dice en pantalla, es reposicionar aleatoriamente el planeta del bot y el del jugador.



Así es como se ve el juego, se inicia al seleccionar la opción “iniciar juego”.

El jugador puede conquistar los planetas (nodos) que sean hijos o del nodo que sea hijo el planeta del jugador. Esto lo hace clickeando su planeta(nodo rojo) y el planeta que desea conquistar. En este caso, solamente puede conquistar el planeta con población de 7.



Como puede apreciarse, el bot(nodo azul), que tenía una población de 33, conquistó el planeta neutral que tenia población de 14 y, este envió sus tropas a el que tiene una población de 12.

Esto funciona gracias al método "CalcularMovimiento()" de la clase "Estrategia". Que lo que hace, es retornar una instancia de la clase "Movimiento", si la población del bot es mayor o igual al doble de la población de los planetas adyacentes.

Métodos implementados en la clase Estrategia en lenguaje C

- Consulta 1: En este método se me pide obtener la distancia entre el planeta del bot y la raíz. Opté por utilizar un recorrido por niveles con separación, en el árbol general recibido por parámetro, incrementando la distancia hasta encontrar el planeta(nodo) que pertenece al bot. Una vez terminado retorna la respuesta.

```
public String Consulta1(ArbolGeneral<Planeta> arbol)
{
    var c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> arbolAux;
    int distancia = 0;

    // encolamos raiz
    c.encolar(arbol);
    c.encolar(null);
    // procesamos cola
    while(!c.esVacia()){
        arbolAux = c.desencolar();

        // procesar el dato
        if(arbolAux==null){
            if(!c.esVacia()) c.encolar(null);
            dista
        }
        else{
            //si encuentra el planeta del bot, sale del bucle while
            if(arbolAux.getDatoRaiz().EsPlanetaDeLaIA()) break;
            // encolamos hijos
            foreach(var hijo in arbolAux.getHijos()) c.encolar(hijo);
        }
    }
    return "La distancia del camino que existe entre el planeta del Bot y la raíz es de " + distancia;

}
```

local variable Cola<ArbolGeneral<Planeta>> c;

- Consulta 2: Este método debe devolver los planetas que son descendientes del bot. Aquí también utilicé un recorrido por niveles, al igual que en la consulta 1, solo que sin separación.

  Para llegar al resultado esperado, primero debe encontrar al bot y le agregue un booleano para saber cuándo los próximos planetas procesados serán descendientes del bot, para no agregar al resultado los planetas que no lo son.

```
public String Consulta2( ArbolGeneral<Planeta> arbol)
{
    var c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> arbolAux;
    string desc = "";
    //creo "esDesc" para cambiarlo a true en los proximos planetas procesados luego de
    //encontrar al del bot
    bool esDesc=false;
    // encolamos raiz
    c.encolar(arbol);

    while(!c.esVacia()){
        arbolAux = c.desencolar();

        // si encuentro al planeta del bot...
        if(arbolAux.getDatoRaiz().EsPlanetaDeLaIA()){
            if(arbolAux.esHoja()) return "El planeta del bot no tiene descendientes";
            //cambio "esDesc" a true, ya que los proximos seran hijos del bot
            esDesc=true;
            //reseteo la cola, porque solamente deberá contener los planetas descendientes del bot
            c=new Cola<ArbolGeneral<Planeta>>();
        }
        //si no es planeta de la ia significa que bien, es hijo del planeta del bot o es descendiente
        //de este, por eso tambien evalúo si esDesc es true. Si no lo es significa que aún no se encontro el bot
        if(!arbolAux.getDatoRaiz().EsPlanetaDeLaIA() && esDesc) desc+=arbolAux.getDatoRaiz().Poblacion()+ ", ";

        // encolo a los hijos si es que todavia no agregué ningun descendiente, o si ya encontre al bot
        foreach(var hijo in arbolAux.getHijos())
            c.encolar(hijo);

    }

    return "Las poblaciones de los planetas descendientes del bot son: " + desc;
}
```

- Consulta 3: Este método debe retornar la población total y la promedio por cada nivel del árbol.

  Lo que hice fue, al igual que en la consulta 1, implementar un recorrido por niveles con separación. Esto lo hice para separar los resultados por cada nivel.

```
public String Consulta3( ArbolGeneral<Planeta> arbol)
{
    var c = new Cola<ArbolGeneral<Planeta>>();
    ArbolGeneral<Planeta> arbolAux;
    int pobTotal=0;
    int prom=0;
    int nivel=0;
    int cant=0;
    string resultado="";
    // encolamos raiz
    c.encolar(arbol);
    c.encolar(null);
    // procesamos cola
    while(!c.esVacia()){
        arbolAux = c.desencolar();
        //si el arbolAux es null...
        if(arbolAux==null){
            //si la cola no esta vacia, encolo null para que se separen los niveles
            if(!c.esVacia()) c.encolar(null);
            //calculo el promedio entre las poblaciones de todos los planetas del nivel
            prom=pobTotal/cant;
            //guardo el resultado de lo pedido por nivel
            resultado+="\n" + "Nivel " + nivel + ": población total de " + pobTotal + " y población promedio de "+ prom;
            //incremento nivel para el proximo nivel y seteo a 0 las demas variables para el proximo nivel
            nivel++;
            cant=0;
            prom=0;
            pobTotal=0;
        }
        else{
            //incremento la cantidad de planetas en este nivel
            cant++;
            //sumo la poblacion de cada planeta
            pobTotal+=arbolAux.getDatoRaiz().Poblacion();
            //encolo a los hijos
            foreach(var hijo in arbolAux.getHijos()) c.encolar(hijo);
        }
    }
    return resultado;
}
```

Posibles mejoras

Este videojuego podría mejorarse si se divide en niveles, aumentando la dificultad a medida que van pasando los niveles, de modo que, por ejemplo, por cada nivel se aumente la velocidad a la que ataca a otros planetas o que ataque a mas de uno a la vez. También podría mejorarse estéticamente, en vez de que se muestre una flechita moviéndose de un nodo a otro podría mostrarse alguna animación con naves, y en vez de que los planetas sean circunferencias, podrían verse como planetas. Además, podría ser agregada una banda sonora diferente por cada nivel.

Conclusión

Con este trabajo pude afianzar varios conceptos de la materia, entre ellos: el uso del TAD "cola" y del árbol general para realizar los métodos de la clase estrategia. Además, también me sirvió para repasar lo aprendido anteriormente en la carrera como, por ejemplo, el uso de clases, objetos, recursión, entre otros temas. En fin, este Trabajo y la cursada me ayudó mucho a repasar bastantes temas y también a utilizar el razonamiento lógico para la resolución de los problemas de código.