Trabajo Final
Conquista Planetaria

COMPLJIDAD TEMPORAL, ESTRUCTURA DEDATOS Y ALGORITMOS.

2020 2C
Alejandra Prado
Leg: 12866
Comisión 5

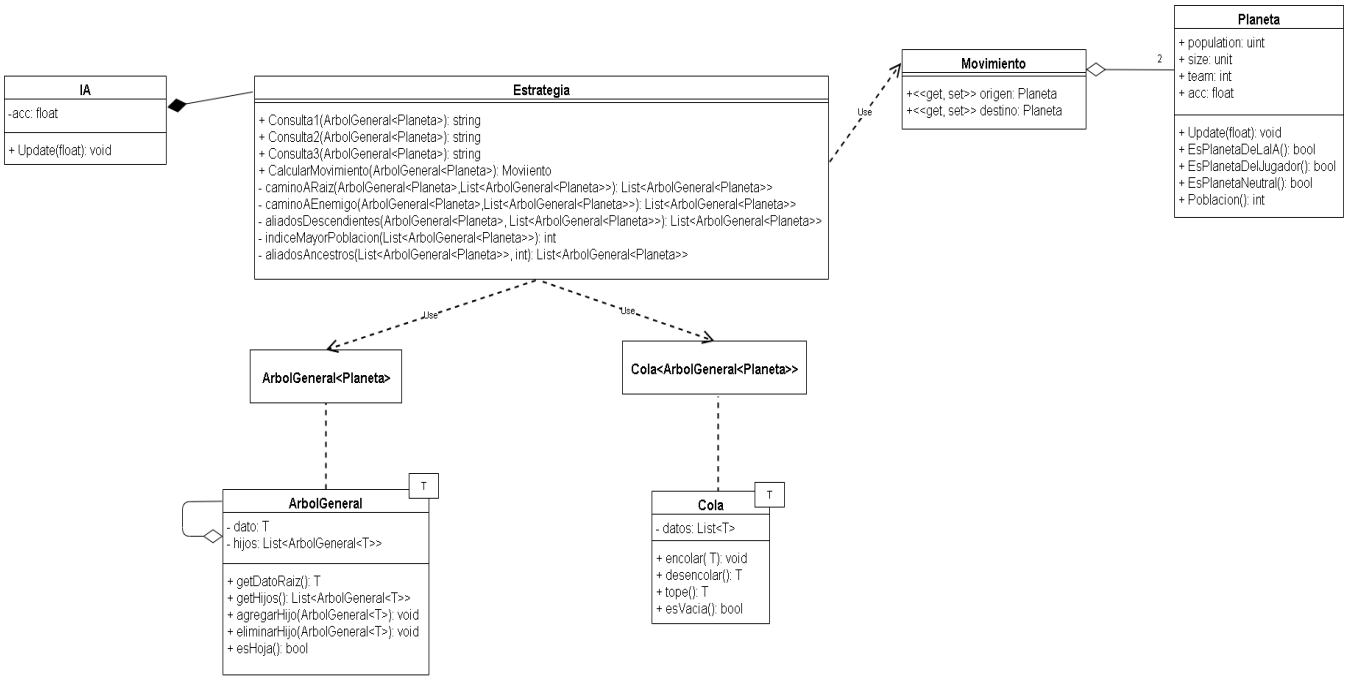
Universidad Nacional
ARTURO JAURETCHE

Introducción

El siguiente proyecto consiste en un juego de computador que se desarrolla como una conquista planetaria donde el enfrentamiento se da únicamente entre dos jugadores. Estos últimos están representados por el usuario y por una entidad de software que posee codificada cierta estrategia destinada a ganar el juego (Bot). El objetivo del juego es apoderarse de todos los planetas del rival.

En este proyecto se implementa la estrategia de ataque que utiliza el Bot cuando en el mapa se disponen los planetas jerárquicamente o en forma de árbol general.

Diagrama de clases



Implementación de métodos

Consulta1 (ArbolGeneral<Planeta> arbol):

Este método privado recibe como parámetro un árbol de tipo ArbolGeneral<Plantea> y retorna el nivel en el que se encuentra el enemigo más cercano

Detalles de implementación:

Comienza con una variable “nivel” de tipo int inicializada en 0. Luego, se recorre el árbol por niveles. En caso de encontrar un planeta perteneciente al usuario, se retorna un mensaje con el contenido de la variable nivel. Si se encuentra un null, que indica el fin de un nivel, la variable nivel aumenta en uno. Si el árbol no fue recorrido completamente, se sigue con el mismo procesamiento. En caso contrario, se retorna un string vacio (no fue encontrado ningún planeta perteneciente al usuario).

Consulta2(ArbolGeneral<Planeta> arbol):

Devuelve un mensaje con la cantidad de planetas por nivel con población mayor a 10.

Detalles de implementación:

Se comienza con una variable “msj” de tipo string que contiene un mensaje inicial y dos variables de tipo int: una que guarda el nivel y otra que acumula la cantidad de planetas encontrados con población mayor a 10, ambas inicializadas en 0.
Se recorre el árbol por niveles, aumentando el acumulador cada vez que se encuentre un planeta con población mayor a 10. Cuando se llega al final del nivel (indicado por un null) se agrega al mensaje inicial el número de nivel más el contenido del acumulador. Luego, se aumenta en uno el nivel y el acumulador vuelve a 0.

Si no se recorrió el árbol por completo, sigue el mismo proceso con el próximo nivel. En caso contrario, finaliza el método retornando el contenido de la variable de tipo string “msj”.

Consulta3(ArbolGeneral<Planeta>):

Devuelve un mensaje con la cantidad poblacional por nivel del árbol pasado como parámetro.

Detalles de implementación

Se comienza con una variable “msj” de tipo string que contiene un mensaje inicial y tres variables de tipo int: una que guarda el nivel, un acumulador de planetas por nivel (“acPlanetas”) y otro que acumula la población total por nivel (“acPoblacion”). Todas estas variables se inicializan en 0.

Se recorre el árbol por niveles, aumentando en uno acPlanetas cada vez que se encuentre un planeta y se suma su población a acPoblación. Cuando se llega al final del nivel (indicado por un null) se calcula el promedio poblacional y se concatena este valor al mensaje inicial junto con el número de nivel. Luego, se aumenta en uno el nivel y los acumuladores vuelven a 0.

Si no se recorrió el árbol por completo, sigue el mismo proceso con el próximo nivel. En caso contrario, finaliza el método retornando el contenido de la variable “msj”.

CalcularMovimiento(ArbolGeneral<Planeta> arbol)

Este método devuelve un objeto Movimiento compuesto por un planeta origen y un destino. Este movimiento es utilizado por la IA para atacar y conquistar los demás planetas.

Detalles de implementación

Este método se divide en dos flujos de ejecución principales: si la raíz del árbol pasado como parámetro no pertenece a la IA (conquista de raíz) y si pertenece a la IA (ataque a enemigo/planeta del jugador).

Conquista de raíz:

Se define el recorrido/camino que debe realizar el planeta de la IA más cercano a la raíz para llegar a esta.

El siguiente paso es indicar cuál es el movimiento/ataque que debe realizar la IA para conquistar la raíz. Si se selecciona el planeta más cercano a la IA como planeta de origen del movimiento/ataque se producen algunos inconvenientes. En primer lugar, como siempre se ataca con la IA más cercana y constantemente está reduciendo su población a la mitad, a la IA

atacante le conlleva mucho tiempo conquistar otro planeta, otorgándole ventaja al usuario. En segundo lugar, los planetas aliados están constantemente aumentando su población, la cual no es utilizada o aprovechada por la IA.
Para resolver estos problemas se decidió la siguiente implementación:
Se determina cuáles son los aliados del planeta de la IA más cercano a la raíz.
En el caso de que tenga aliados, se determina cuál de todos ellos tiene mayor población. Luego, se compara la población del planeta de la IA más cercano a la raíz con la del aliado más poblado.
Si la población del aliado es mayor, se lo emplea como planeta de origen del movimiento/ataque y como planeta destino se utiliza su posterior en la lista de aliados (en dirección a la raíz). En caso contrario, el planeta de la IA más cercano a la raíz es determinado como planeta de origen del movimiento/ataque y su posterior (en el camino hacia la raíz obtenido al comienzo) como destino. Esto también ocurre en el caso de que el planeta de la IA más cercano a la raíz no tenga aliados.

Ataque a enemigo/planeta del jugador

Se define el recorrido/camino que debe realizar la IA desde la raíz al enemigo más cercano a esta.
El siguiente paso es indicar cuál es el movimiento/ataque que debe realizar la IA para llegar y conquistar al planeta del usuario. Para ello se determina cuál es el planeta perteneciente a la IA más cercano al enemigo ("últimoIA") que se encuentra dentro del camino obtenido. Si se selecciona siempre este planeta como origen del movimiento/ataque además de producirse los mismos inconvenientes que en el caso de la conquista de la raíz, a la IA se le hace imposible ganar el juego ya que a la hora de atacar al enemigo éste siempre tiene una población mayor.
Para resolver estos problemas se decidió la siguiente implementación:
Se busca dentro del camino obtenido los aliados de últimoIA.
Si últimoIA tiene aliados, se determina cuál de todos ellos tiene mayor población. Luego, se compara la población de últimoIA con la del aliado más poblado.
Si la población del aliado es mayor, se lo emplea como planeta de origen del movimiento/ataque y como planeta destino se utiliza su posterior en la lista de aliados. En caso contrario, últimoIA se determina como planeta origen del movimiento/ataque y su posterior (en el camino hacia el enemigo más cercano obtenido al comienzo) como destino. Esto también ocurre en el caso de que últimoIA no tenga aliados.

caminoARaiz(ArbolGeneral<Planeta> arbol, ListArbolGeneral<<Planeta> > camino):

Método privado que devuelve la lista de árboles que contienen los planetas que se deben recorrer desde la raíz hasta el planeta de la IA más cercano.

Detalle de implementación:

Se agrega a la lista (de tipo List<ArbolGeneral<Planeta>>) pasada como segundo parámetro el árbol pasado como primer parámetro (de tipo ArbolGeneral<Planeta>).

Si el árbol que se agregó a la lista contiene un planeta perteneciente a la IA, se retorna la lista. En caso contrario, hay dos posibles flujos de ejecución:

1- Si el árbol que se agregó a la lista tiene hijos, se ejecuta un foreach que procesa cada subárbol hijo recursivamente (se utiliza el mismo método pero pasándole como primer parámetro el subárbol hijo y la lista obtenida hasta ese momento). El resultado que devuelve esta llamada se guarda en una variable/lista auxiliar de tipo List<ArbolGeneral<Planeta>>. Si esta variable no contiene un null (segundo posible flujo de ejecución) se retorna la lista auxiliar deteniendo el procesamiento de los demás subárboles hijos.
2- Si el árbol no tiene hijos (es una hoja) o si la totalidad de sus hijos fueron recorridos por el foreach, es eliminado de la lista. Luego, se llega al final del método retornando un null.

caminoAEnemigo(ArbolGeneral<Planeta> arbol, List<ArbolGeneral<Planeta >> camino)

Método privado devuelve la lista de árboles que contienen los planetas que se deben recorrer desde la raíz hasta el planeta del jugador más cercano.

Detalles de implementación

La implementación de este método es la misma del método caminoARaíz() con la única diferencia que retorna la lista cuando el árbol agregado contiene un planeta perteneciente al jugador.

aliadosDescendientes(ArbolGeneral<Planeta> arbol, List<ArbolGeneral<Planeta>> lisAliados)

Busca entre los hijos del árbol pasado como primer parámetro y añade a una lista (que luego es retornada) aquellos que contengan planetas pertenecientes a la IA.

Detalles de implementación

Se agrega a la lista pasada como segundo parámetro “lisAliados” el árbol pasado como primer parámetro. Luego, se comprueba si cada hijo contiene un planeta perteneciente a la IA. Si esto no sucede, se prosigue con el siguiente hijo. En caso afirmativo, se llama recursivamente al método pasándole como primer parámetro el subárbol hijo y como segundo la lista existente hasta el momento. La lista devuelta por esta llamada se guarda en una variable auxiliar. Por último, se almacena en la variable “lisAliados” la lista auxiliar y se la retorna.

indiceMayorPoblacion(List<ArbolGeneral<Planeta>> lista):

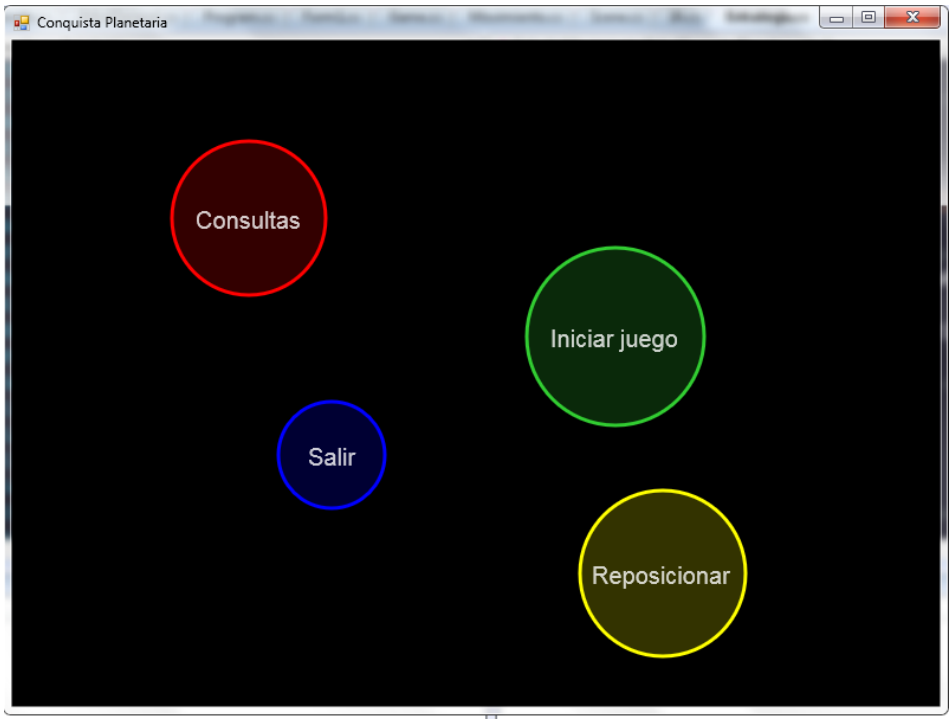
Este método privado recibe como parámetro una lista de tipo ArbolGeneral<Planeta> y devuelve el índice del árbol que contiene el planeta con mayor población.

List<ArbolGeneral<Planeta>> aliadosAncestros(List<ArbolGeneral<Planeta>> camino, int limite)

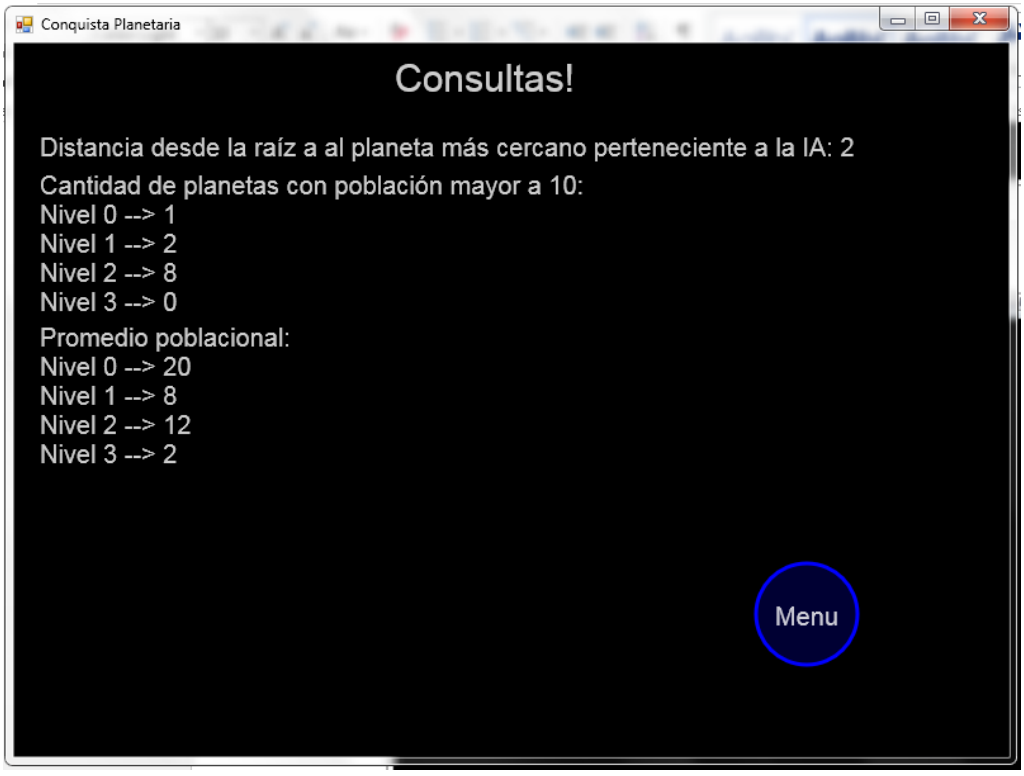
Método privado que recorre una lista de tipo ArbolGeneral<Planeta> hasta un índice determinado agregando en una lista, que luego es retornada, aquellos árboles que contienen planetas pertenecientes a la IA.

Imágenes

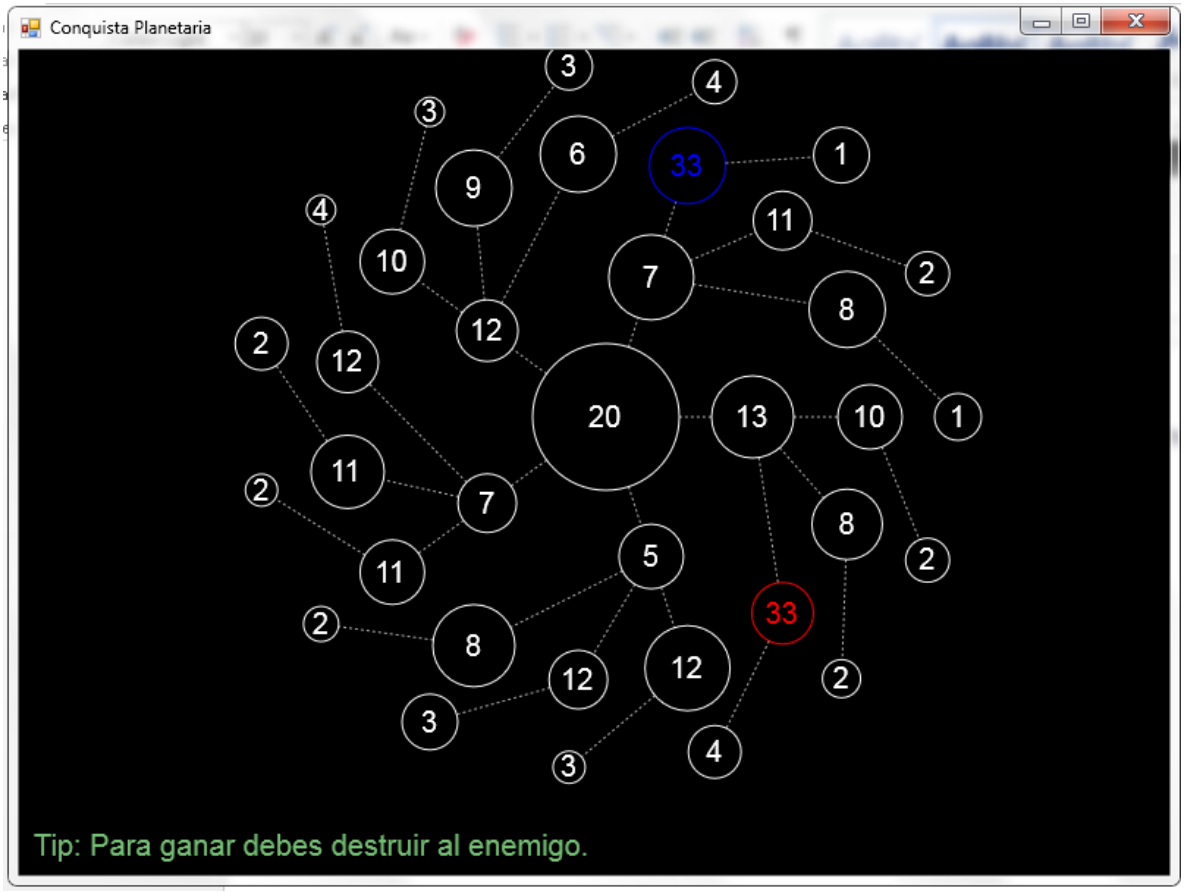
Pantalla de inicio:



Pantalla de consultas: Muestra información sobre el árbol principal

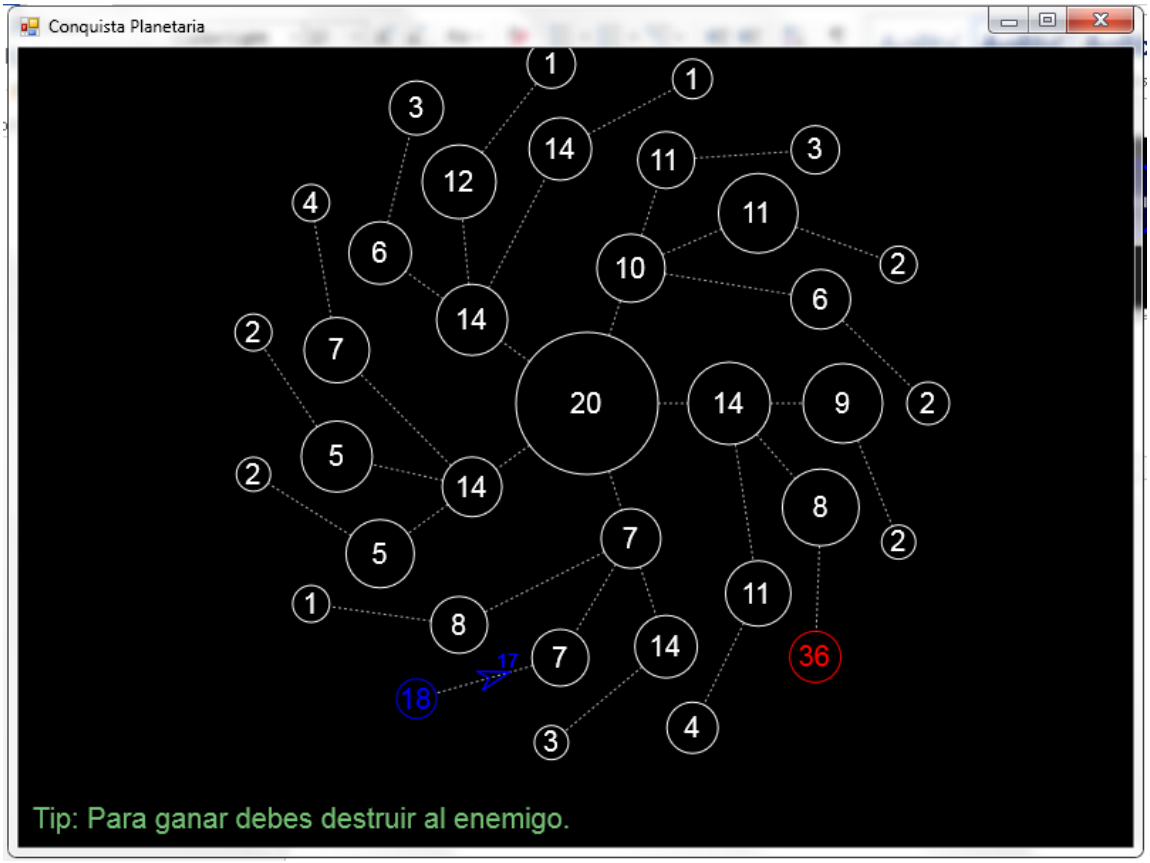


El inicio del juego coincide con la información mostrada en la pantalla de consultas

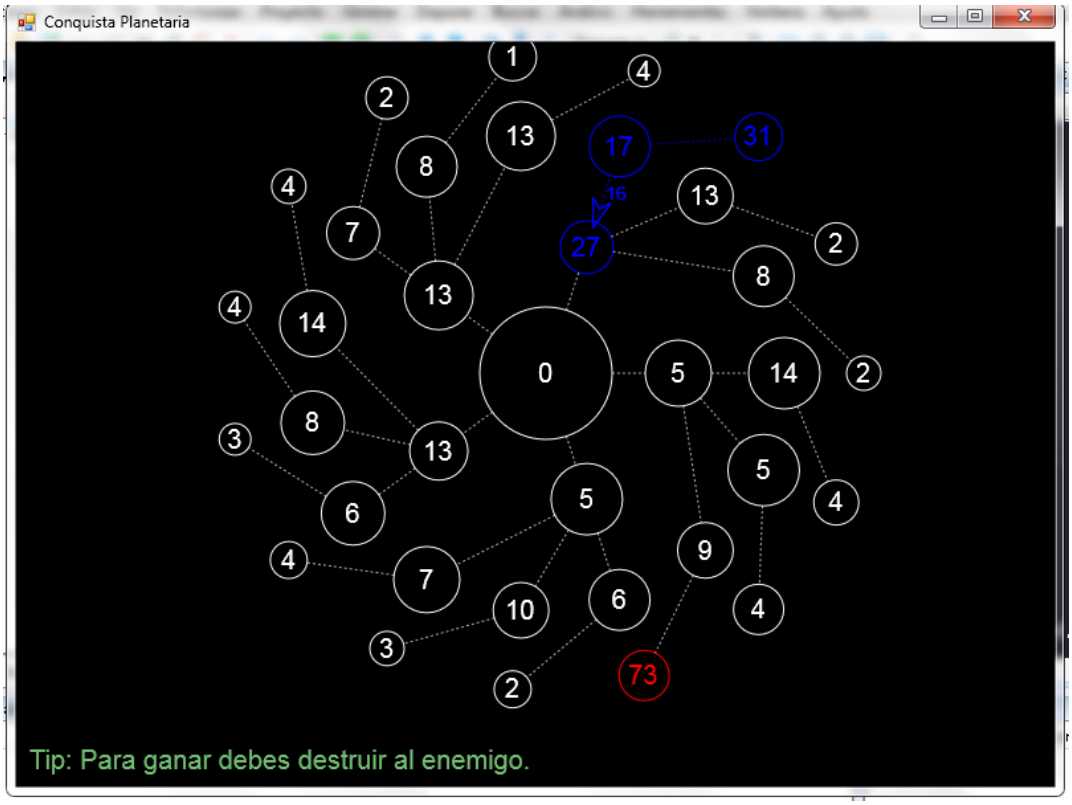


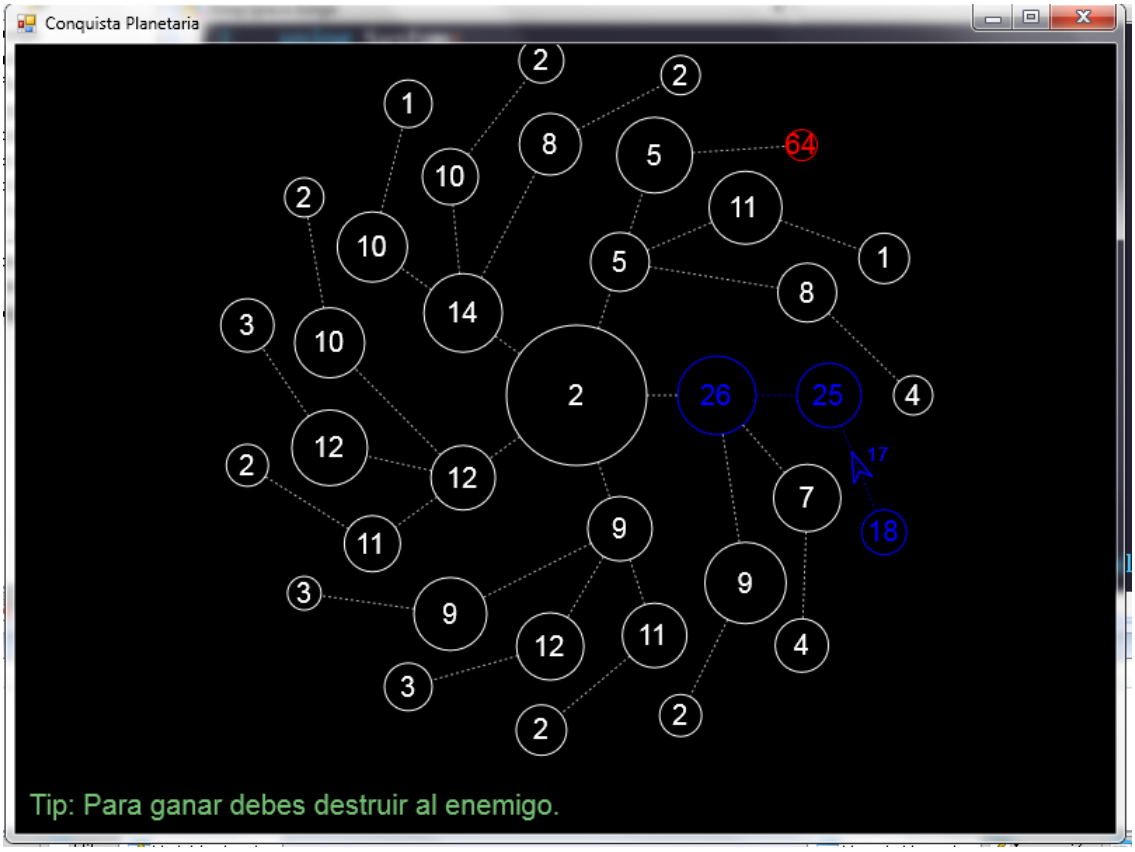
Pantalla de juego:

Primer objetivo: conquista de la raíz

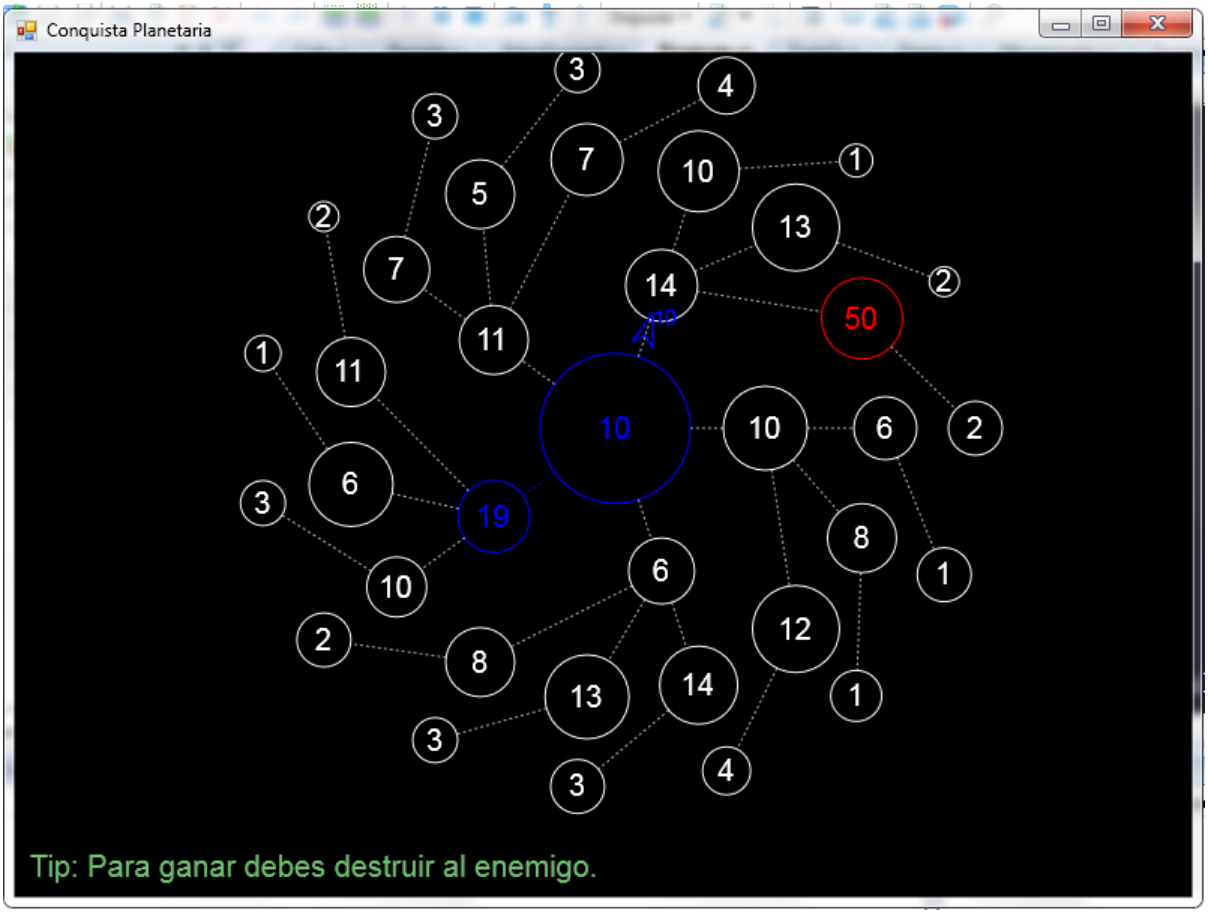


La IA ataca o envía refuerzos desde el planeta que tiene mayor población

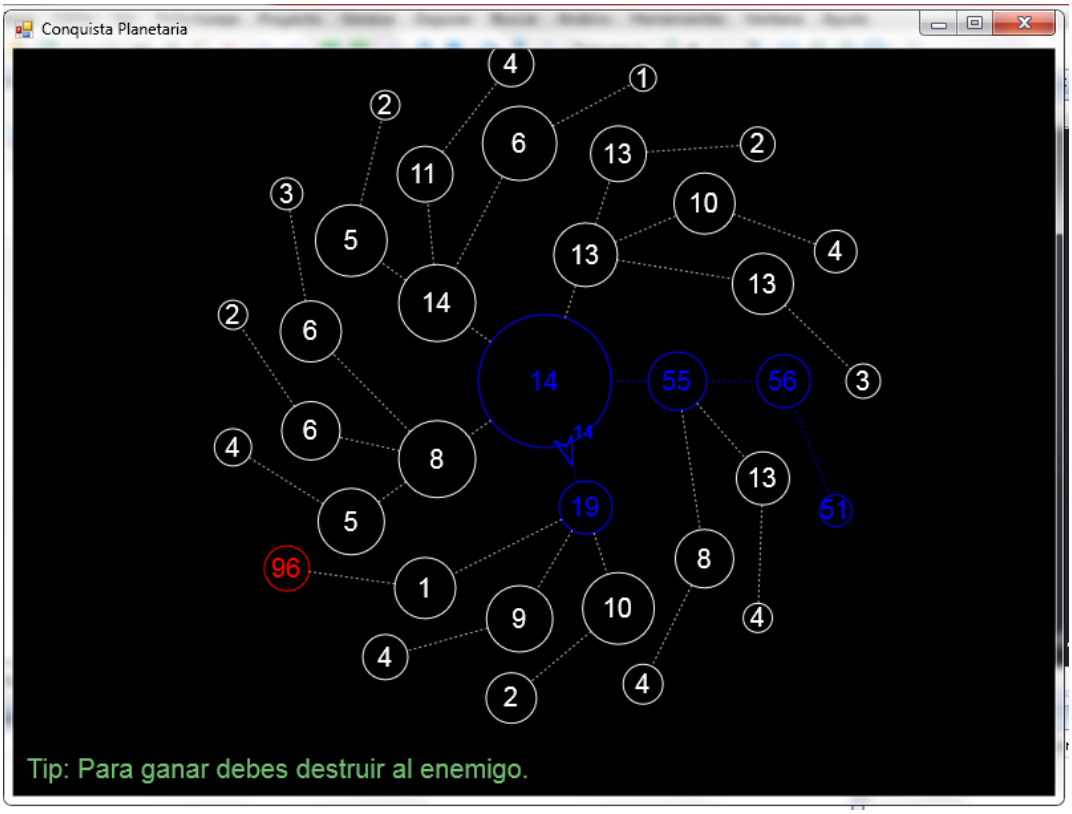




Una vez conquistada la raíz, dirige su ataque hacía el enemigo más cercano



Una vez conquistada la raíz, la IA ataca con el planeta aliado con mayor población que se encuentre entre la raíz y el camino al enemigo.



Pantalla de reposición: Al seleccionar esta opción se reposicionan los planetas reiniciando el juego.

Sugerencias/Ideas

Al momento de atacar al enemigo, la IA podría ser más eficientes si utiliza los planetas ya conquistados de la fase de conquista de la raíz. Para esto se debería, de alguna manera, realizar una única lista con todos los aliados y a partir de ella determinar con cuál planeta conviene atacar.

Cuando la raíz no pertenece a la IA, antes de iniciar con la conquista de la misma, se podría comprobar si el planeta del enemigo es descendiente del planeta de la IA inicial. Si esto es así, según la población de ambos, se podría calcular si a la IA le conviene atacar directamente al enemigo sin la necesidad de conquistar la raíz o proseguir con la fase de conquista de la raíz.