# TESORO BINARIO V2.0

## TRABAJO PRÁCTICO 2

*Algoritmos y Programación II - 75.41*

*Cátedra Ing. Patricia Calvo - 2do cuatrimestre 2023*

*Facultad de Ingeniería - Universidad de Buenos Aires.*


*Alumnos:*

1) *Facundo Tomás Fernández - 109085 - ftfernandez@fi.uba.ar*
2) *Gianfranco Turco - 110073 - gturco@fi.uba.ar*
3) *Mauricio Uribe - 105971 - muribe@fi.uba.ar*
4) *Martín León - 109138 - mfleon@fi.uba.ar*
5) *Franco Boggia - 109175 - fboggia@fi.uba.ar*
6) *Gaspar Derisi - 108586 - gderisi@fi.uba.ar*


*Profesores correctores:* *Gustavo Schmidt y Sergio Sirotinsky.*

*Fecha límite de entrega:* *10/11/23*

# Índice

# INFORME GENERAL

## OBJETIVO

Generar una pieza de software que simule el funcionamiento del juego Tesoro Binario en su versión multijugador Jugador.

## ENUNCIADO

Tesoro Binario es un juego de mesa para N jugadores en el que se introducen M tesoros por cada jugador en un tablero de X por Y por Z con el fin de que los tesoros no sean encontrados hasta el final. En cada turno el jugador puede:

A. Sacar una carta del mazo, para jugarla o guardarla. Ver las opciones de cartas.
B. Atacar una posición del tablero con un tesoro mina, eliminando si hay un tesoro en ella y dejando la casilla inactiva por tantos turnos dependiendo el poder de la mina, o dejándolo a la espera de ser descubierto y el jugador que lo descubre, pierde el turno.
C. En cada turno el jugador deja un espía en un casillero del tablero, recuperando el tesoro si hay uno en ella o dejando la casilla con el espía esperando que se mueva un tesoro. Cuando el espía encuentra el tesoro, tarda 5 turnos en recuperarlo dejando la casilla inactiva por esos turnos. Si el jugador pone un espía en una casilla con un espía contrario, ambos espías son eliminados del juego.
D. Mover un tesoro de lugar. Si un tesoro se mueve a una casilla con un tesoro, el juego le avisa al jugador para que mande un espía y lo pueda recuperar. El jugador no puede mover el tesoro a una casilla en proceso de que el espía está recuperando el tesoro.

El juego debe tener 6 cartas disponibles, 3 dadas por el enunciado y 3 definidas por el grupo. Las cartas dadas por el enunciado son:

1) Blindaje: bloquea la captura del tesoro por una cantidad de turnos.
2) Radar: colocado en una posición, detecta 3 casilleros a la redonda si hay o no tesoros.
3) Partir tesoro: permite partir un tesoro en 2 casilleros.

El juego termina cuando todos los jugadores se quedan sin tesoros salvo uno, ganando este jugador.

# INTERFAZ DE USUARIO

Toda la interfaz de usuario debe estar basada en texto, la entrada y salida y el estado del tablero tiene que mostrarse utilizando un Bitmap (ver librería) de una manera ideada por el grupo.

No es necesario que se limpie la pantalla, simplemente escribir el estado del tablero luego de cada jugada.

# DESARROLLO

Al comenzar con el desarrollo del juego, luego de leer el enunciado, nos reunimos para establecer las bases del juego, definir las reglas y la estructura fundamental.

Se definieron los siguientes puntos clave:

1) Un turno (Desde la perspectiva del usuario) se completa cuando todos los jugadores tienen la opción de jugar (Se completa una vuelta).
2) El juego termina cuando: Un jugador acumula cierta cantidad de tesoros; Todos los jugadores (excepto uno, el ganador) se quedan sin tesoros.
3) Al inicio se colocan los tesoros en forma aleatoria.
4) Orden de acciones en el turno: Robar carta del mazo -> Jugar carta (opcional) -> Mover tesoro (opcional) -> Colocar espía ó Colocar mina.
5) Cartas personalizadas:
    a) Doble turno: le permite al jugador jugar dos turnos seguidos. (resta el contador de captura de un tesoro) (no se puede jugar carta en el segundo turno).
    b) Roba carta a otro jugador.
    c) Rastrear casillero: Le muestra a un jugador una ubicación del mapa donde hay una

alta probabilidad de hallar un tesoro, pero también puede ser la coordenada de una mina o un espía enemigo.

En un principio necesitaba elegir una estructura para el tablero que, preferiblemente, ya sea una matriz tridimensional a base de listas con la posibilidad de que sean de distinto tamaño (ancho, alto, profundidad) pero comenzamos probando que además se creen solamente las listas necesarias para los casilleros ocupados y no TODO el tablero. Esa idea al poco tiempo fracasó por desconocimiento de ciertos manejos del lenguaje. Por lo que decidimos crear todos los caminos a casilleros posibles, al inicializar el tablero.

Una vez creado el tablero había que definir cómo iban a ser los casilleros y qué información iban a contener, además debíamos ver cómo guardar esa información. En principio lo habíamos manejado con una lista de datos que podía variar en su tamaño dependiendo del tipo de casillero, pero luego implementamos una clase que se encargue de manipular y guardar estos datos para facilitar el acceso. Posteriormente, se trabajó en características más específicas, como las interacciones del jugador con los casilleros, la validación de posiciones en el tablero y las consecuencias de intentar colocar un espía, mover un tesoro, etc.

En paralelo, se integraron elementos visuales al juego para mejorar la experiencia del usuario. Se diseñaron gráficos para representar los tesoros y se trabajó en la presentación visual del tablero para hacer el juego más atractivo.

Durante el proceso de desarrollo, se realizaron optimizaciones para mejorar el rendimiento del juego y se corrigieron errores que surgieron durante las pruebas. Cada nueva funcionalidad implementada fue documentada, proporcionando explicaciones detalladas sobre el funcionamiento del código para facilitar la comprensión y futuras modificaciones. Cualquier error encontrado fue abordado con ajustes finales en la jugabilidad y, por lo general, de manera grupal para acordar estrategias de solución en conjunto con quienes estaban trabajando en lo mismo.

Todo el desarrollo fue registrado mediante git y github para no perder el historial de cambios realizados y lograr trabajos en conjunto que de otra manera implicaría esperar a que cada uno termine con lo suyo y pasar los archivos de manera local.

En resumen, en el grupo, desde un principio se dividieron tareas y enfoques para facilitar el desarrollo y, una vez hubo un avance considerable, se comenzaron a integrar y probar todas las partes para corregir nuevos errores surgidos y crear nuevas y mejores funcionalidades requeridas por la mayor complejidad del código.

# CUESTIONARIO

## ¿Qué es un SVN?

Es un sistema de control de versiones que permite tener un registro de cambios en archivos y directorios a lo largo del tiempo, lo que facilita la colaboración en proyectos de desarrollo de software. Cada usuario solo tiene acceso al historial completo de versiones si está conectado al servidor central.

## ¿Qué es Git?

Es un sistema de control de versiones que permite tener un registro de cambios en archivos y directorios a lo largo del tiempo, lo que facilita la colaboración en proyectos de desarrollo de software. Cada usuario que trabaja con un repositorio Git tiene una copia completa del historial de versiones del proyecto en su sistema local.

## ¿Qué es Github?

Es una plataforma de alojamiento y colaboración para proyectos de desarrollo de software y utiliza el sistema de control de versiones Git. Permite crear repositorios (Tanto privados como públicos) para almacenar código, llevar un registro de los cambios realizados, colaborar con otros desarrolladores, proponer y revisar cambios, entre otras herramientas.

## ¿Qué es un Valgrind?

Es una herramienta de análisis de código y depuración para sistemas operativos basados en Unix. Se utiliza principalmente para encontrar, en el código, accesos a áreas de memoria no asignadas, fugas de memoria y problemas de inicialización de variables.

# MANUAL DE USUARIO

## REGLAS DEL JUEGO

Tesoro Binario 2.0 es un juego que se desarrollará en un tablero de 3 dimensiones personalizadas, el objetivo principal es encontrar y obtener 5 tesoros para ganar la partida mientras defendes tus propios tesoros de los demás jugadores. Se puede jugar con un mínimo de 2 jugadores y un máximo de 10 jugadores.

## COMIENZO DEL JUEGO

El juego comenzará pidiéndole al usuario una serie de datos para poder iniciar la partida. En principio, se le pedirá al usuario que ingrese la cantidad personas que jugaran la partida, la cantidad tiene que ser como mínimo 2 jugadores y máximo 10 jugadores, luego, se le pedirá ingresar un alto, ancho y profundidad personalizables para el tablero en el que se desarrollara toda la partida.

## DESARROLLO DE PARTIDA

La partida comienza solicitándole al usuario si desea exportar sus casilleros bajo control, en caso de elegir “si”, se le pedirá que elija uno de los casilleros, y este será exportado.

Luego se le pedirá una ubicación para **colocar a un espía** en el tablero, dependiendo de la posición seleccionada se desencadenarán diversos efectos en el desarrollo de la partida:

- Si el Espía es colocado en un casillero vacío simplemente se colocará en tal casillero.
- Si el Espía es colocado en un casillero que tenía un Espía enemigo ambos serán eliminados del tablero, dejando el casillero vacío.
- Si el Espía se coloca en un casillero que anteriormente era ocupado por un Tesoro

enemigo, el propietario del Tesoro perderá tal tesoro. Además, el casillero quedará inactivo durante los próximos 5 turnos. Una vez transcurridos estos 5 turnos, el jugador que ejecutó la acción de colocar al Espía en ese casillero obtendrá el Tesoro.

- Si el Espía se coloca en un casillero que anteriormente era ocupado por Tesoro Partido, el jugador deberá buscar la otra parte correspondiente para iniciar el proceso de robo de dicho tesoro.

- Si el Espía es colocado en un casillero que anteriormente era ocupado por una Mina, el casillero explotará instantáneamente dejando el casillero vacío.

- Si el Espía es colocado en un casillero propio del jugador actual se le pedirá volver a ingresar otra ubicación para colocar a su espía.

- Si el Espía es colocado en un casillero Blindado, Destruido o Inactivo perderá la capacidad de ubicar al Espía, el casillero permanece intacto.

Después de completar la colocación de un Espía en el tablero, se solicitará al usuario ingresar una ubicación adicional en el tablero para colocar una Mina. Los efectos en la partida variarán según la ubicación seleccionada:

- Si la Mina es colocada en un casillero Vacío simplemente se colocará en tal casillero.

- Si la Mina se coloca en un casillero que contiene un Tesoro enemigo, este último será destruido, causando la pérdida del tesoro por parte del enemigo. Además, el casillero quedará en estado de destrucción, lo cual lo mantiene inaccesible durante un tiempo determinado, el tiempo depende de la potencia de la mina.

- Si la Mina es ubicada en un casillero ocupado por un Espía enemigo, este será eliminado del tablero, dejando el casillero en estado de destrucción durante un tiempo determinado por la potencia de la mina.

- Si una Mina es colocada en un casillero ocupado por otra Mina enemiga, ambas minas explotarán simultáneamente, resultando en que el casillero quede vacío.

- Si una Mina es colocada en un casillero con un Tesoro Partido se dejará el casillero destruido durante una cantidad determinada de tiempo dependiendo de la potencia de la Mina convirtiendo a la otra parte del Tesoro Partido como un Tesoro normal.

- Si una Mina es colocada en un casillero Blindado el blindaje se eliminará del tesoro, dejando el casillero como un Tesoro normal sin blindaje.

- Si una Mina es colocada en un casillero Inactivo se interrumpe la captura del tesoro, dejándolo destruido por una cantidad determinada de tiempo dependiendo de la potencia de la mina.

- Si una Mina es colocada en un casillero Destruido se destruirá nuevamente el casillero con una nueva cantidad de tiempo dependiendo de la potencia de la mina.

Una vez colocados el Espía y la Mina se le pedirá al usuario **mover un tesoro propio**, dependiendo a donde se mueva sucederán distintos efectos en la partida:

- Si el Tesoro se mueve a un casillero Vacío simplemente cambiará su ubicación a ese casillero.

- Si el Tesoro se mueve a un casillero con un Espía enemigo el espía enemigo roba el tesoro y el casillero pasa a estar Inactivo durante 5 turnos.

- Si el Tesoro se mueve a un casillero con una Mina enemiga el casillero explota dejándolo destruido durante una cantidad determinada de turnos dependiendo de la potencia de la Mina y el jugador que movió el tesoro perderá su tesoro.

- Si el Tesoro se mueve a un casillero con un Tesoro enemigo, un Tesoro Partido o un Blindado se le notifica al jugador que hay un tesoro en tal casillero para que pueda agarrarlo más adelante, el Tesoro se mantiene en su posición inicial sin moverse.

Cuando ya colocamos a nuestro Espía, Mina y movimos nuestro Tesoro se le ofrecerá al jugador si quiere **jugar una carta**, del caso de elegir “no” se avanzara al siguiente turno, pero del caso a elegir “si” se le mostrará al usuario las cartas que puede elegir para jugar, dependiendo de la carta se realizarán distintas acciones:

- **Blindaje:** Le agrega una protección al tesoro, si un Espía intenta robar el tesoro blindado el blindaje no se lo permitirá, blindaje permanecerá activa durante 5 turnos.

- **Radar:** Se revelan 3 casilleros alrededor del casillero elegido, si en los casilleros revelados es encontrado algún tesoro se te notifica por pantalla para que lo puedas ir a buscar.

- **Partir Tesoro:** Se pide ingresar la ubicación de un Tesoro propio, este mismo se parte en 2, a diferencia de un tesoro normal para poder recuperar un tesoro partido el enemigo deberá encontrar las 2 partes del tesoro.

- **Trackear casilla:** Le muestra a un jugador una ubicación del mapa donde hay una probabilidad de hallar un tesoro, pero también puede ser la coordenada de una mina o un espía enemigo.

- **Robar carta:** Se le roba a un jugador enemigo una carta de su mazo.

- **Doble turno:** Te permite jugar dos turnos seguidos.

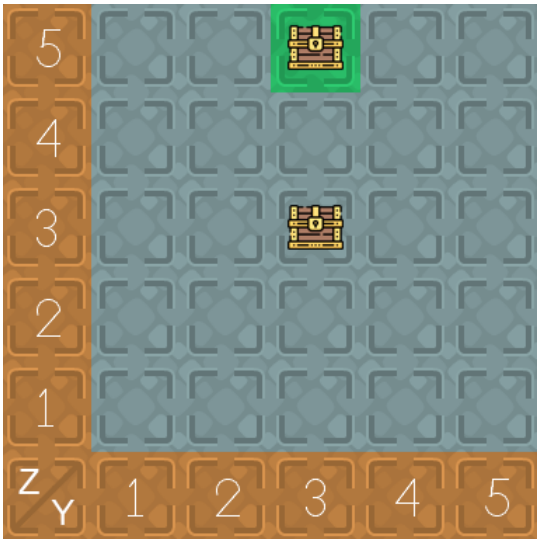
# FINALIZACIÓN DEL JUEGO

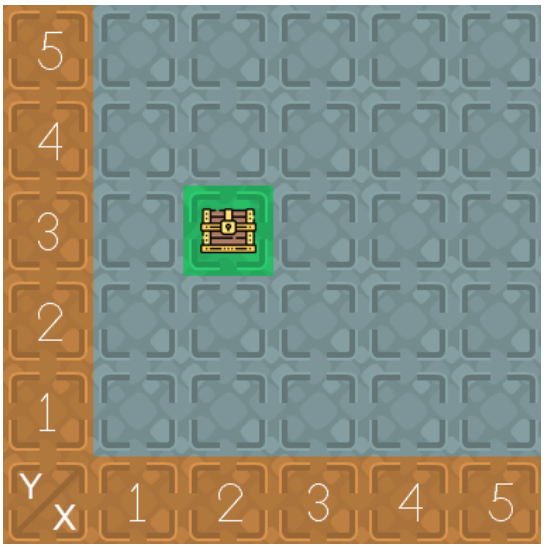
El juego termina si uno de los jugadores juntó los 5 tesoros pedidos, transformándose en el ganador o si es el último jugador en pie. Si un jugador perdió todos sus tesoros este es eliminado de la partida.

# INTERFAZ GRÁFICA

Al comienzo de una partida se creará una carpeta “tablero” dentro de ella, cada jugador tendrá su propia carpeta cuyo nombre será el del jugador al que le pertenece. En ella se mostrarán las tres perspectivas (XY, XZ e YZ) del tablero correspondiente al jugador. No se puede revisar los tableros de otros jugadores.

A continuación, se mostrarán cómo se ve un tablero de 5x5x5 y cómo se ve cada tipo de casillero.



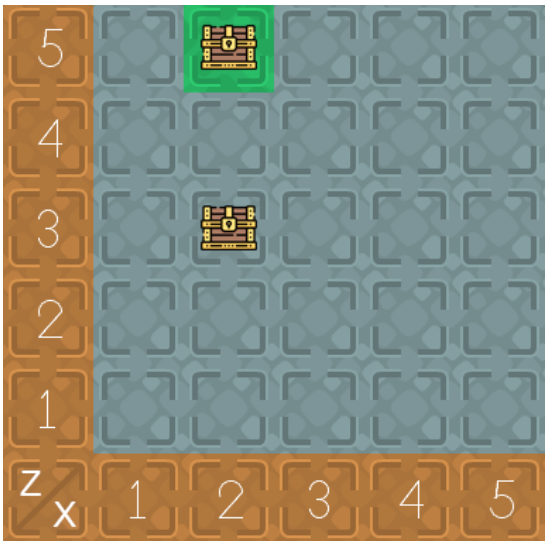




Fig. 1, 2 y 3: En estas imágenes se muestra cómo se ve un tablero de 5x5x5. Se exportan en planos YZ (Fig. 1), XY (Fig. 2) y XZ (Fig. 3).

En la esquina inferior izquierda se muestra la perspectiva del tablero que se está mostrando y a sus costados están numerados los casilleros.

Notar que en este tablero hay 2 tesoros. Uno de ellos está seleccionado (el que tiene un fondo verde) y el otro no.

El tesoro seleccionado está en el casillero de coordenadas X = 2, Y = 3 y Z = 5.

El tesoro sin seleccionar está en el casillero de coordenadas X = 2, Y = 3, Z = 3.

Notar que ambos tesoros tienen las mismas coordenadas X e Y, por eso en el plano XY se muestran superpuestos y se ve uno solo de ellos.

Fig. 4: Así se observa un casillero vacío.

Fig. 5: Así se observa un casillero con un tesoro.

Fig. 6: Así se observa un casillero donde hay un fragmento de un tesoro partido.

Fig. 7: Así se observa un casillero donde hay una mina.

Fig. 8: Así se observa un casillero inactivo.

Fig. 9: Así se observa un casillero donde hay un espía.

Fig. 10: Así se observa un casillero destruido.

Fig. 11: Así se observa un casillero blindado.

Estás mismos íconos con un fondo verde indican el casillero seleccionado.

# MANUAL DEL PROGRAMADOR

## HERRAMIENTAS DE PROGRAMACIÓN

El juego fue desarrollado en C++ en su estándar C++98. Se utilizan TDAs y templates que permiten modularizar el programa. Como entornos de desarrollo se emplearon Visual Studio Code, CLion y Eclipse. Para debuggear se empleó GDB y Valgrind.

## MODULARIZACIÓN

No se explicarán todos y cada uno de los TDAs y sus métodos porque se encuentran debidamente comentados y descriptos en sus respectivos archivo. Se explicarán las implementaciones más complejas y las relaciones entre los distintos TDAs.

El programa se puede separar en dos grandes grupos. Uno de ellos se encarga de implementar el tablero de juego junto con los casilleros. Por otro lado, se implementan los TDAs relacionados a las dinámicas de juego y la interacción con el usuario.

A continuación, se muestran las dependencias entre los TDAs en un esquema. Algunos de ellos aparecen repetidos, pero se implementaron una única vez.

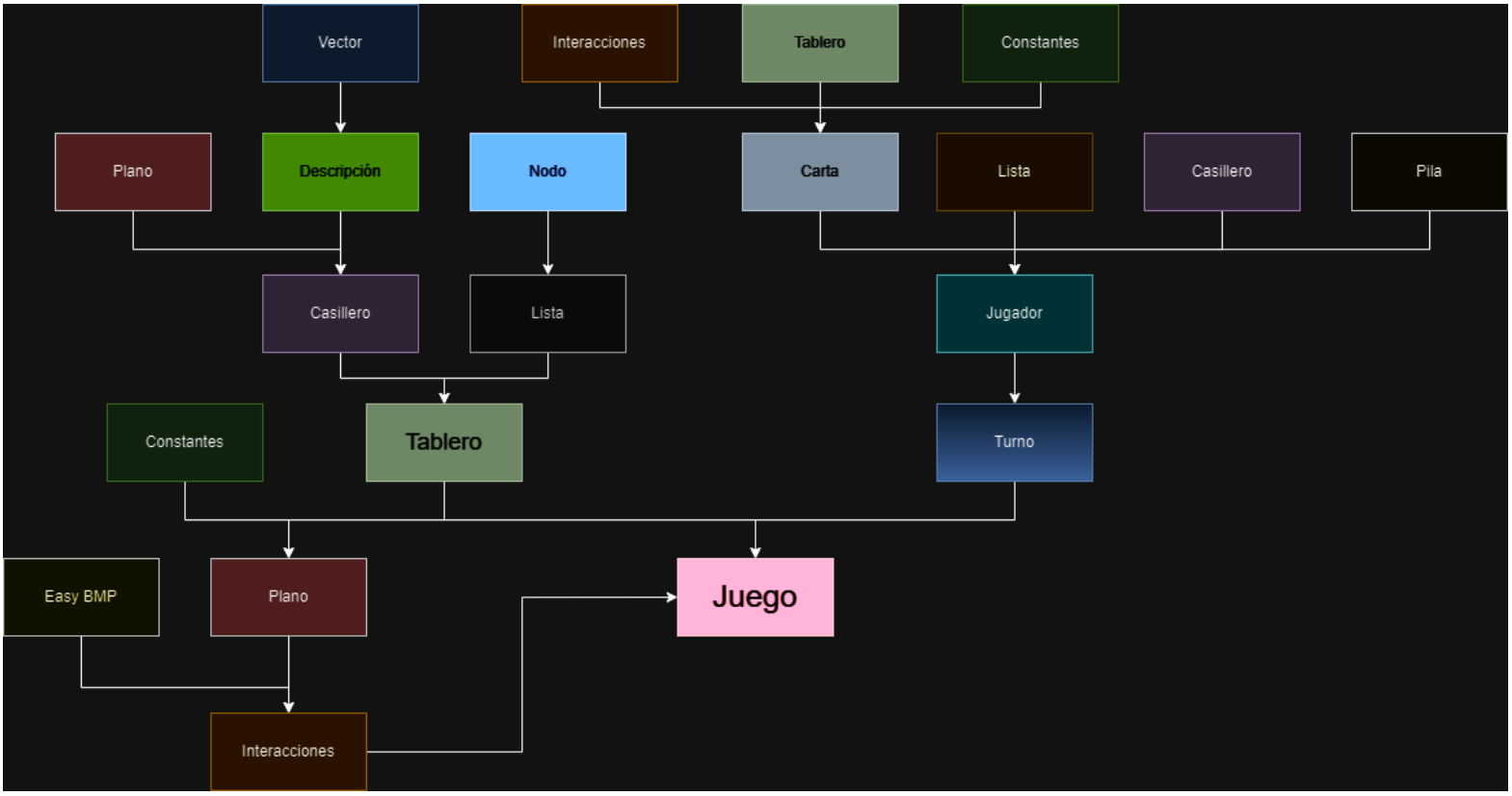



Fig. 12: En esta figura se muestran las dependencias de los TDAs.

El TDA Tablero es uno de los más importantes. Consiste en una lista de listas de listas de casilleros; un cuboide cuyas dimensiones las establece un jugador al comenzar la partida. El tamaño mínimo del tablero es de 5x5x5 para poder implementar los planos de casilleros vecinos (explicados más adelante).

En el diagrama se puede observar una dependencia circular entre Plano, Tablero y Casillero. Esta dependencia se resolvió implementando “forward declaration” (declaración anticipada) de las clases necesarias en cada TDA.

Cada casillero posee una descripción del contenido del casillero (Vacío, Tesoro, Espía, Inactivo, Blindado, Mina o Partido), el id del jugador que está usando ese casillero, tres Planos que guardan sus casilleros vecinos desde distintas perspectivas (XY, XZ, YZ) y las coordenadas asociadas con ese casillero (x, y, z). En este punto, cabe aclarar que las coordenadas pueden estar dadas como un puntero a un vector de tres elementos (las tres coordenadas) o como tres unsigned int dados por separado. Según el caso, resultó conveniente trabajarlo de una forma u otra.

Un Plano consiste en una lista de listas de casilleros que almacena un plano (valga la

redundancia) de tamaño 5x5 con todos los casilleros vecinos a un casillero indicado, es decir, almacena los vecinos a dos casilleros a la redonda del casillero seleccionado.

A continuación, se muestra una ilustración de cómo sería un tablero y un plano en perspectiva XY con los casilleros contenidos por ese plano. En amarillo se ve un tablero de 10x10x10 (no se marcan todos sus casilleros para facilitar la comprensión). En rojo vemos un plano cuyo casillero central es el punto verde y los puntos azules son sus vecinos.

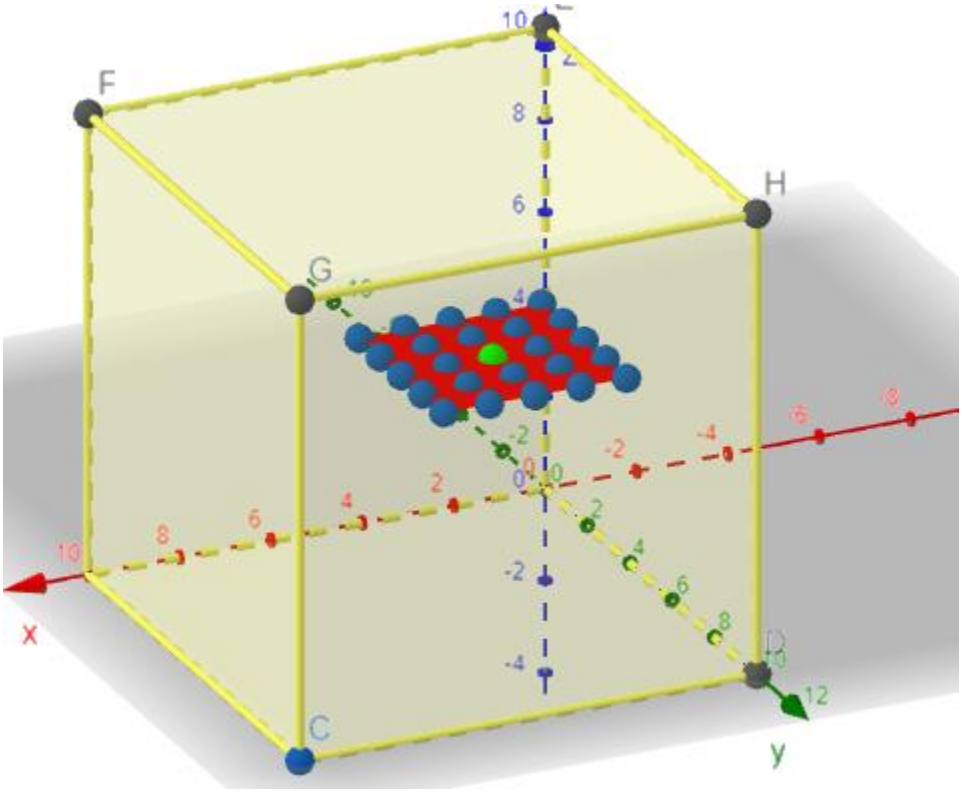



Fig. 13: Se muestra una representación gráfica del tablero con un casillero central (esferas verdes) y sus casilleros vecinos (esferas azules) en el plano de vecinos de perspectiva XY.

En el caso que el casillero central no tenga dos dos a los costados se corrige su posición y se corre para lograr formar un plano de tamaño 5x5.

Otra parte importante del juego es el TDA Interacción. Como su nombre lo indica, se encarga de interactuar con los usuarios. Tiene métodos para solicitar a los usuarios coordenadas, la dimensión del tablero, una opción entre una lista de opciones, entre otros. Además, se encarga de imprimir el tablero, las instrucciones para los usuarios, las opciones entre las cuales deben elegir, etc. Para imprimir el tablero de una forma gráfica se emplea una librería denominada Easy BMP.

El TDA más importante es el TDA Juego. Se encarga de relacionar todos los otros TDAs e implementar la lógica y dinámica del Juego. El mismo consiste en un tablero de juego, un turno, una lista con los jugadores de la partida, las interacciones y una variable booleana que indica si el juego terminó o no. A continuación, se explicará algunos de estos TDAs, el resto son intuitivos y fáciles de leer.

TDA Jugador: posee un Id que lo permite distinguir del resto de jugadores. Una lista con todos los casilleros donde hay algo que le pertenece al jugador (ya sea un tesoro, un espía o una mina), esto nos evita de tener que recorrer todo el tablero cada vez que queremos buscar algo que le pertenezca al jugador. Un contador de la cantidad de casilleros propios del jugador. Una lista con las cartas que el jugador posee en la mano y un contador de la cantidad de cartas que posee el jugador. Un vector de 3 dimensiones que almacena la cantidad de tesoros activos, inactivos y obtenidos (en ese orden) del jugador, lo que permite detectar rápidamente si el jugador perdió. Una pila que almacena todos los cambios que les ocurrieron a los casilleros del jugador mientras no era su turno. Estos cambios se le muestran al jugador al comenzar su turno.

Además, se mencionó el TDA Carta, el mismo es muy sencillo y únicamente almacena el tipo de la carta y permite obtenerlo o reemplazarlo. La implementación de las cartas se realiza en el TDA Juego con un método para cada tipo de carta y el método jugarCarta hace un llamado al que corresponda según el tipo de carta a jugar.

Por último, existe un TDA Constantes que almacena las constantes globales y los distintos tipos empleados en los diferentes TDAs, como el TipoCarta, TipoTesoro, TipoCasillero, etc. Estos tipos se implementan como Enum y al utilizarse en cada TDAs se los emplea como enteros sin signo (unsigned int).