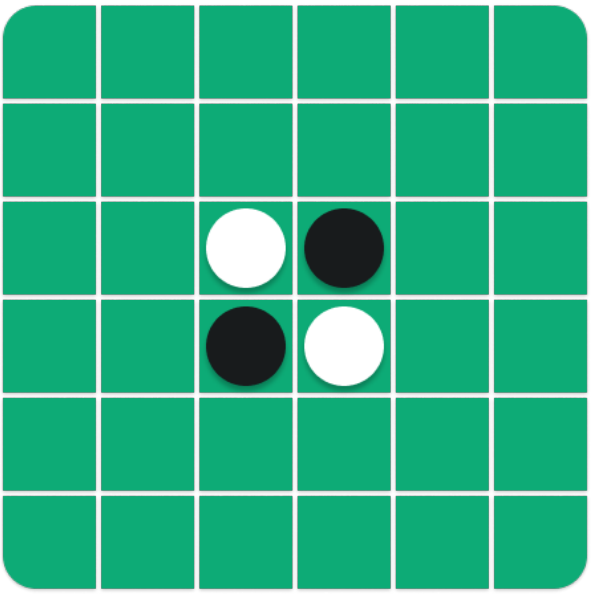
SISTEMAS INTELIGENTES
FACULTAD DE INGENIERÍA
UNIVERSIDAD ANDRES BELLO

Implementación y desarrollo de una IA en Juegos



Integrantes:

- Javier Almarza.
- Gonzalo Cañas.
- Nicolas Cruz.
- Vicente Salas.

Docente:

- Pablo Schwarzenberg

Índice

Índice **2**

Objetivos **3**

Introducción **4**

Aspectos técnicos del desarrollo **5**

Repositorio *5*

Dependencias *6*

Ejecución *8*

Desarrollo de la jugabilidad *9*

HITBOX (Área de colisión) *9*

Visualización celdas vecinas *10*

Implementación de la IA *14*

Direcciones artísticas del desarrollo **16**

Aspecto gráfico *16*

Interfaz gráfica *16*

Interacción con el usuario *17*

Paleta de colores *18*

Alineación de recursos 19

Uso de recursos 20

Conclusiones **21**

Bibliografía y referencias **22**

Objetivos

- Implementar algoritmos de inteligencia artificial en juegos.
- Desarrollar y adaptar mecánicas de juegos en softwares.
- Aplicar técnicas de búsqueda y razonamientos en juegos.

Introducción

En primer lugar, parece necesario hacer entrega de contexto acerca del juego y tema que se pretende abordar. *Reversi* es un juego de mesa que se originó en Inglaterra en el siglo XIX, a lo largo de los años fueron cambiando continuamente las reglas, hasta la llegada del año 1971, donde el japonés *Goro Hasegawa* definió las reglas modernas del juego, las cuales continúan siendo utilizadas hasta el dia de hoy. Reglas que fueron implementadas en el desarrollo del presente proyecto.

Por lo tanto, se darán a conocer todos los aspectos, perspectivas, visiones y directivas. Tomadas a lo largo de todo el desarrollo. En cuanto a los aspectos técnicos o específicos, acerca de la programación o diseño del proyecto, estos serán desarrollados con mayor claridad a lo largo del informe. Por lo que, se pretenden esclarecer los aspectos técnicos del desarrollo y compatibilidad con las dependencias.

Aspectos técnicos del desarrollo

Tal como se mencionó, para el desarrollo de este apartado, se busca esclarecer los aspectos técnicos referidos al desarrollo de: código, algoritmo, inteligencia artificial e implementación del juego Reversi en código.

Repositorio

Entonces, durante el desarrollo del proyecto se tuvo en mente trabajar con un controlador de versiones, como lo es Git. Por lo que, se hizo uso de Github, sitio para la gestión y distribución de proyectos que cuenten con Git. A causa de lo mencionado, se pudo trabajar de una manera más rápida en conjunto, cada uno de los integrantes hizo sus aportes enviando sus actualizaciones de proyecto por GitHub.

Figura (1): Repositorio de GitHub.

El repositorio se mantuvo en privado durante su desarrollo, de esta manera se protegía la propiedad intelectual del código y diseños desarrollados. Adicionalmente, como en la mayoría de proyectos informáticos, se desarrolló una documentación en el lenguaje MarkDown, dicha documentación está presente en el repositorio (archivo *readme.md*), la documentación cuenta con extractos textuales de código, para ofrecer un mayor entendimiento de los algoritmos utilizados.

Dependencias

Primero que todo, cabe mencionar que la sintaxis utilizada en el proyecto es diferente a lo que comúnmente se aprecia en proyectos basados en el lenguaje python, esto es debido a la posibilidad de utilizar tipado estático, presentado en versiones de Python superiores a la ***^3.6.x***. Esto quiere decir que, nuestra sintaxis incluye: puntos y comas, estandarización de tipos de datos, estructuración de parámetros y retornos. Se optó utilizar este modelo de tipado, para reducir los errores provocados por la falta de rigurosidad a la hora de definir variables, funciones y objetos.

Por otro lado, se manejaron las dependencias utilizando un entorno de desarrollo, los comandos para crear un entorno de desarrollo junto con las dependencias, quedaron mencionados en la documentación del repositorio. Por lo mismo, en cuanto a las dependencias externas del lenguaje, hicimos uso de las siguiente:

- *Pygame*.
- *Numpy*.

De igual forma, se hicieron participes múltiples dependencias incluidas en el núcleo (core) del lenguaje Python, estas dependencias son:

- *Time*.
- *Webbrowser*.
- *Typing*.
- *Copy*.
- *Sys*.
- *Os*.

Por lo tanto, se utilizaron múltiples archivos a la hora de trabajar en el proyecto. Por lo que, cada archivo tiene su función y propósito dentro del mismo. Tal como se puede ver en el diagrama presentado en la Figura 1.

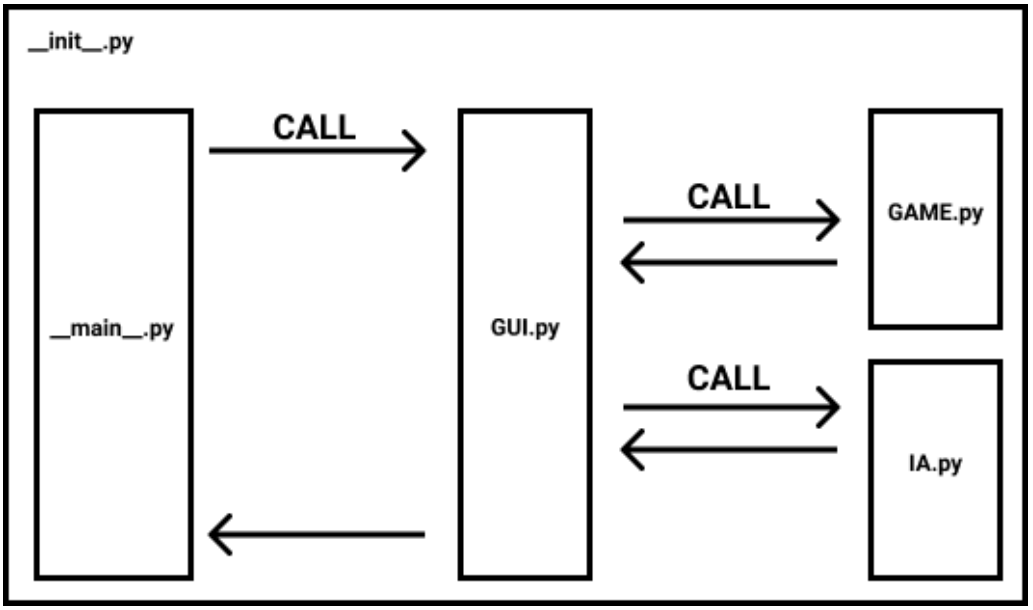


Figura (2): Estructura de consultas de los archivos.

En primer lugar, tal como se puede apreciar en la *Figura 1*, se tiene el archivo ***__init__.py***, archivo destinado a estructurar el directorio ***src/*** como un paquete o módulo de *Python*. Por lo mismo, el archivo ***__main__.py***, tiene la simple función de iniciar todas las características presente en el resto de archivos.

En segundo lugar, el archivo ***GUI.py*** está destinado para: manejar, crear y definir la interfaz gráfica del proyecto, este archivo hace uso de la dependencia *Pygame* para crear todo el ecosistema gráfico.

Continuando con lo mencionado, el archivo ***GAME.py***, controla la jugabilidad, ya sean las posiciones de las fichas en el tablero, como la validez de los movimientos de cada jugador (siendo uno de los jugadores humano y otro la inteligencia artificial).

Por último, los algoritmos de la inteligencia artificial, son manejados por el archivo ***IA.py***, donde se aplican múltiples algoritmos dependiendo de la dificultad elegida.

Ejecución

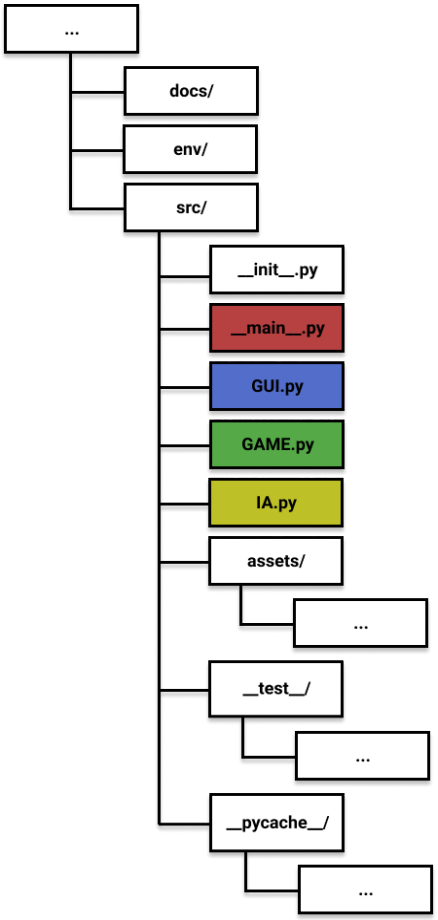


Figura (3): Estructura de archivos.

Recapitulación, la ejecución del proyecto es determinada por el archivo ***__init__.py*** que transforma al directorio ***src/*** en un paquete de *Python*. Por lo mismo, para ejecutar el proyecto, se debe utilizar el comando:

$*env\scripts\activate.bat*
$*py src/*

Por otro lado, en la Figura 2, se puede apreciar la estructura de archivos y carpetas. La creación de la carpeta es necesaria para la ejecución del proyecto. El archivo ***requirements.txt***, es un archivo destinado a almacenar los nombre de las dependencias necesarias para el correcto funcionamiento del proyecto. Por lo tanto, es absolutamente necesario ejecutar los comandos para crear el directorio e instalar las dependencias:

$py -m venv *env/*
$pip install -r requirements.txt

Cabe mencionar que, el directorio ***docs/*** contiene una copia del presente informe junto con archivos acerca de las necesidades y requerimientos del proyecto. Por otro lado, el directorio ***assets/*** contiene todos los recursos utilizados para crear la interfaz gráfica del proyecto, ya sean: fichas, tableros, botones e iconos.

Desarrollo de la jugabilidad

Durante el desarrollo del siguiente apartado, se tratarán temas con respecto a la funcionalidad o procesos detrás de cada uno de los aspectos del juego Reversi.

HITBOX (Área de colisión)

Continuando con lo mencionado, uno de los principales aspectos a la hora de utilizar la librería pygame, fue hacer uso de las colisiones y detección de coordenadas mediante el uso de polígonos, tal como se puede ver a continuación.

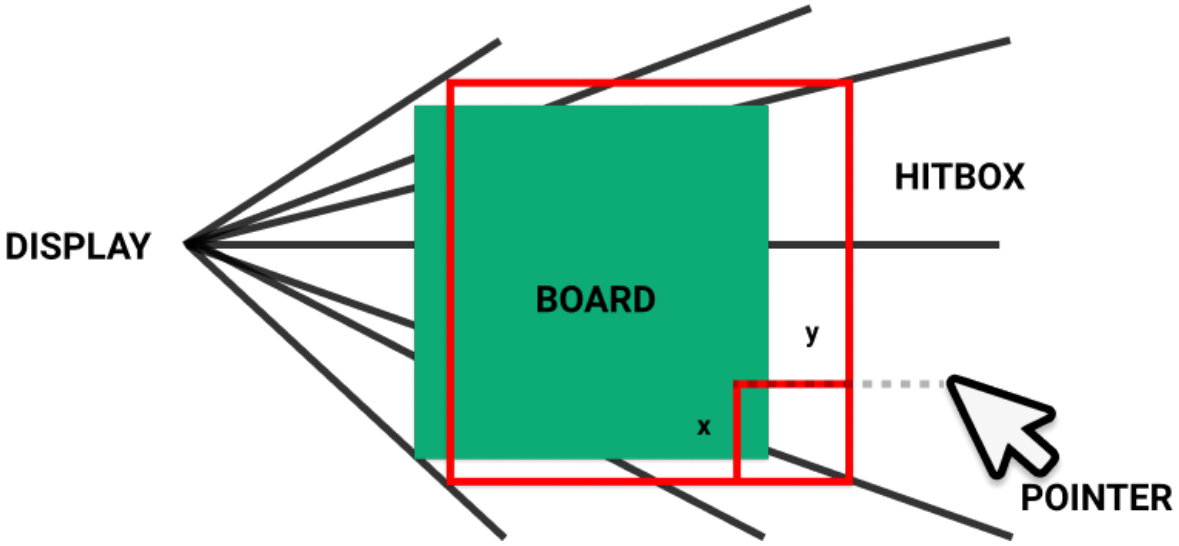


Figura (4): Implementación y uso de la Hitbox en el tablero.

Por lo tanto, se puede ver en la Figura (4), un diagrama acerca de cómo es que funciona la detección del cursor y la implementación de la Hitbox (área de colisión). Al usar una Hitbox, es posible determinar la casilla seleccionada por el usuario y representar dicha casilla en una matriz. La determinación de la casilla se basa en el uso de las coordenadas del cursor y la posibilidad de conocer si el cursor pulsa o no en dicha casilla.

Visualización celdas vecinas

Por lo tanto, se tienen las representaciones de cada caso a la hora de seleccionar una casilla, en la matriz podemos observar que en la celda en rojo es la celda seleccionada y en amarillo son las celdas visitadas, en el mismo cuadro, las celdas verde claro son las celdas en las que se puede encontrar el caso que se muestra en las figuras, por otro lado está el cuadro de vistazos, que nos muestra las coordenadas de cada casilla con respecto a la casilla seleccionada, el color verde claro representa las casillas que son visitadas y en verde oscuro representa la casilla selecionada o las casillas que no se pueden visitar.

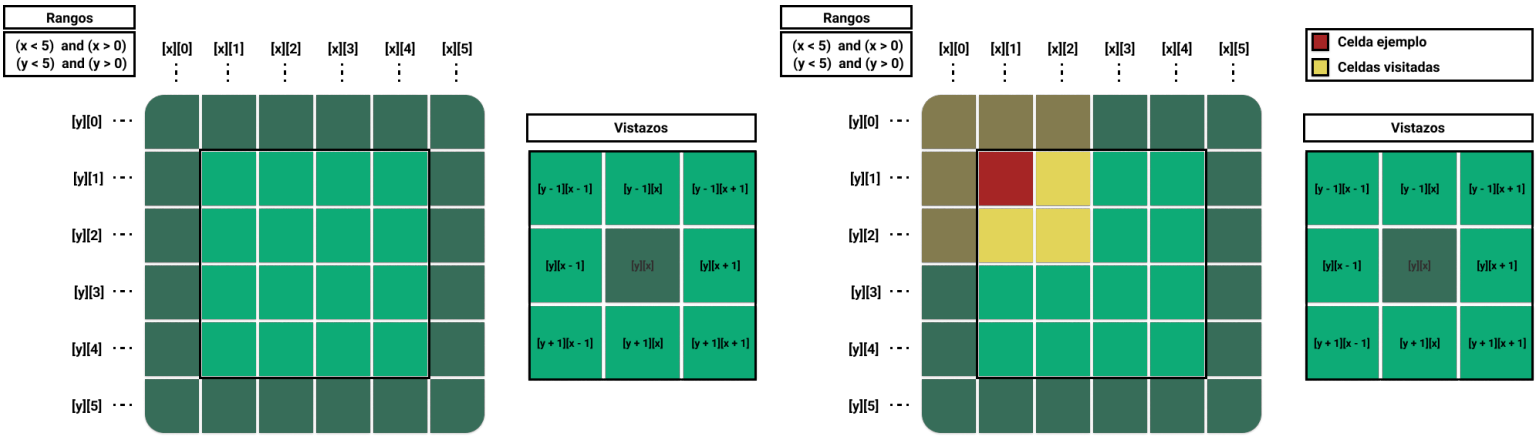


Figura (5): Representación vistazos de una celda, cuando se encuentra en el centro.

Con respecto a la figura (5), se puede apreciar que visita todas las celdas que están a su alrededor, este es el único caso que visitan todas las celdas.

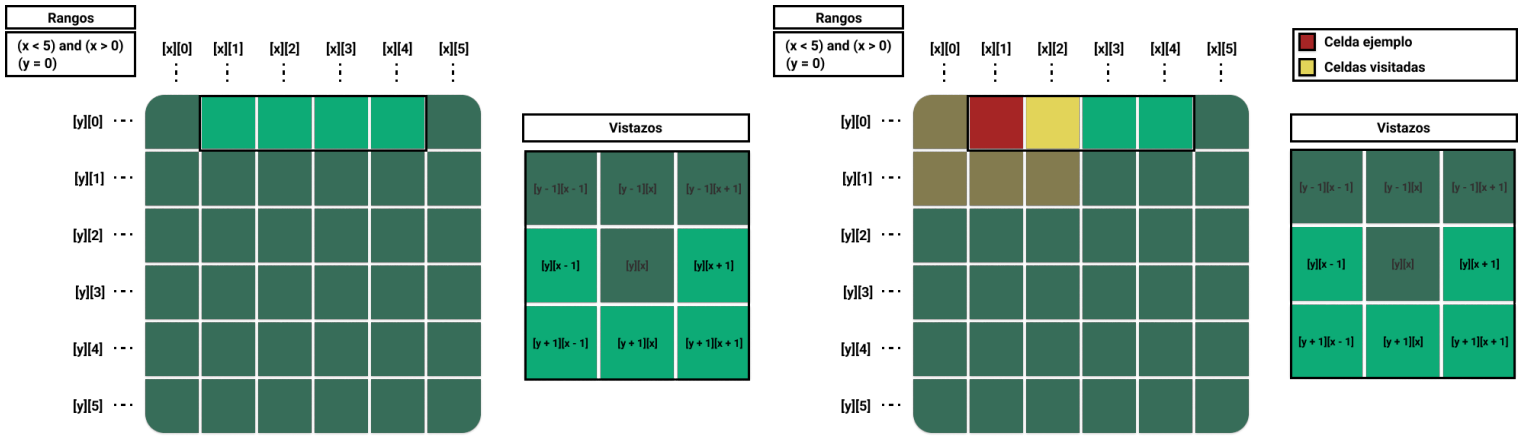


Figura (6): Representación vistazos de una celda, cuando se encuentra en el borde superior.

En el caso de la figura (6), solo visita 5 de las 8 casillas posibles, ya que esta se encuentra en el borde del tablero y por ende se queda con 3 casillas menos por visitar.

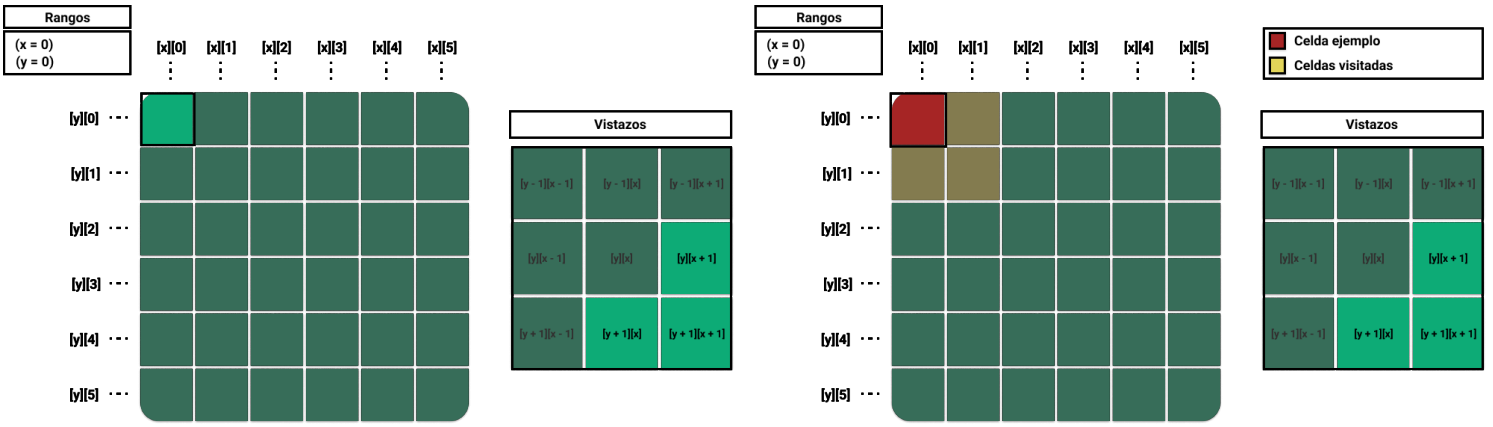


Figura (7): Representación vistazos de una celda, cuando se encuentra en la esquina superior izquierda

Con respecto a la figura (7), se puede decir que es el caso que recorre menos casillas, por lo que solo tiene 3 casillas posibles por visitar.

De aquí en adelante, los casos se repiten pero de distinta forma ya sea en el borde inferior, verticalmente en los bordes de derecha e izquierda o en las esquinas inferiores.

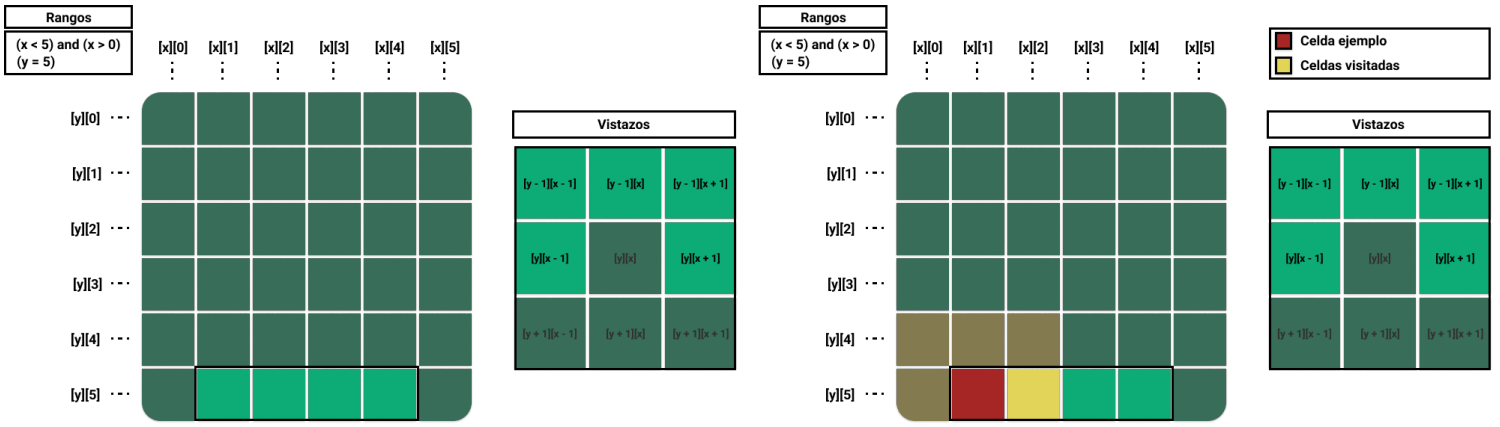


Figura (8): Representación vistazos de una celda, cuando se encuentra en el borde inferior.

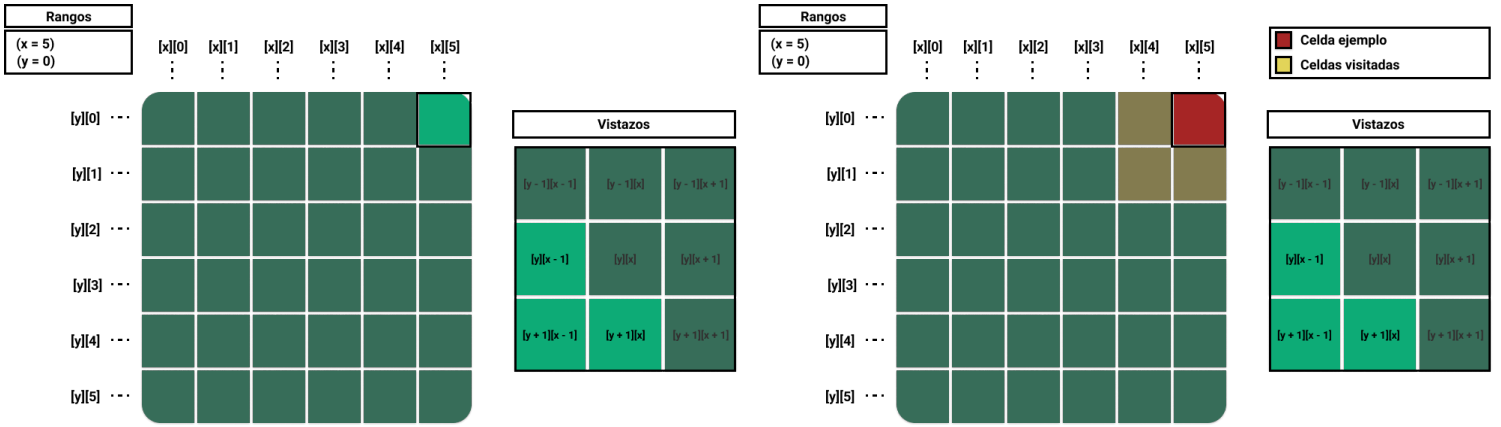


Figura (9): Representación vistazos de una celda, cuando se encuentra en la esquina superior derecha.

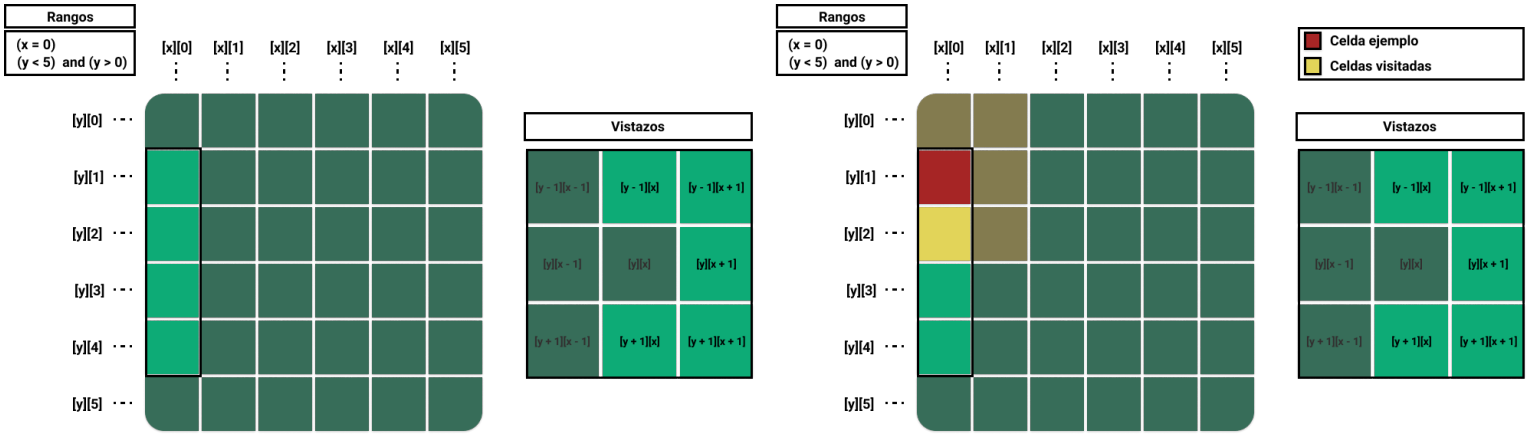


Figura (10): Representación vistazos de una celda, cuando se encuentra en el borde derecho.

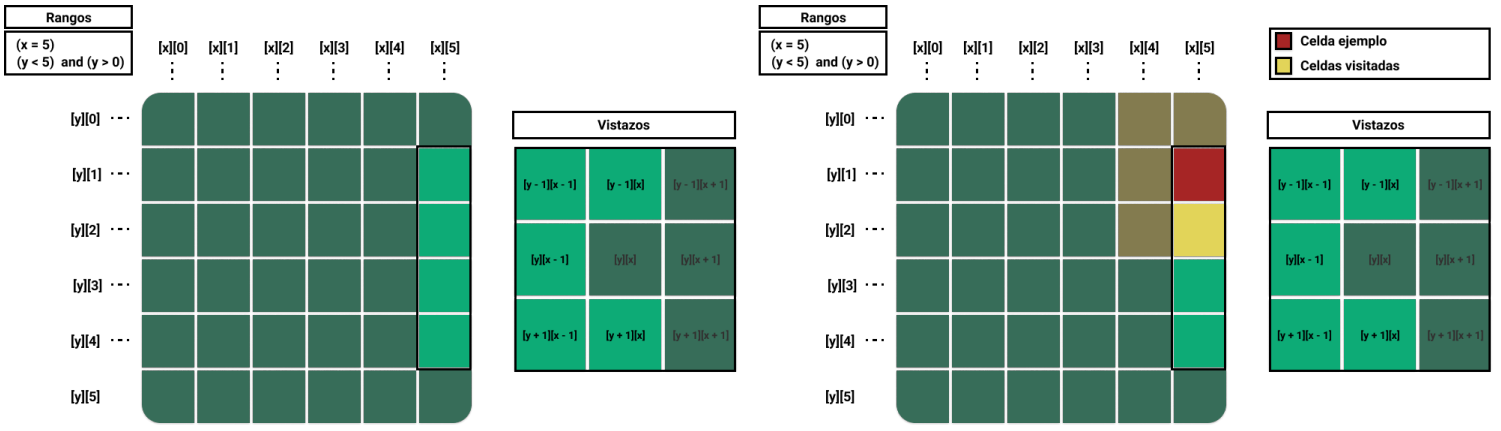


Figura (11): Representación vistazos de una celda, cuando se encuentra en el borde izquierdo.

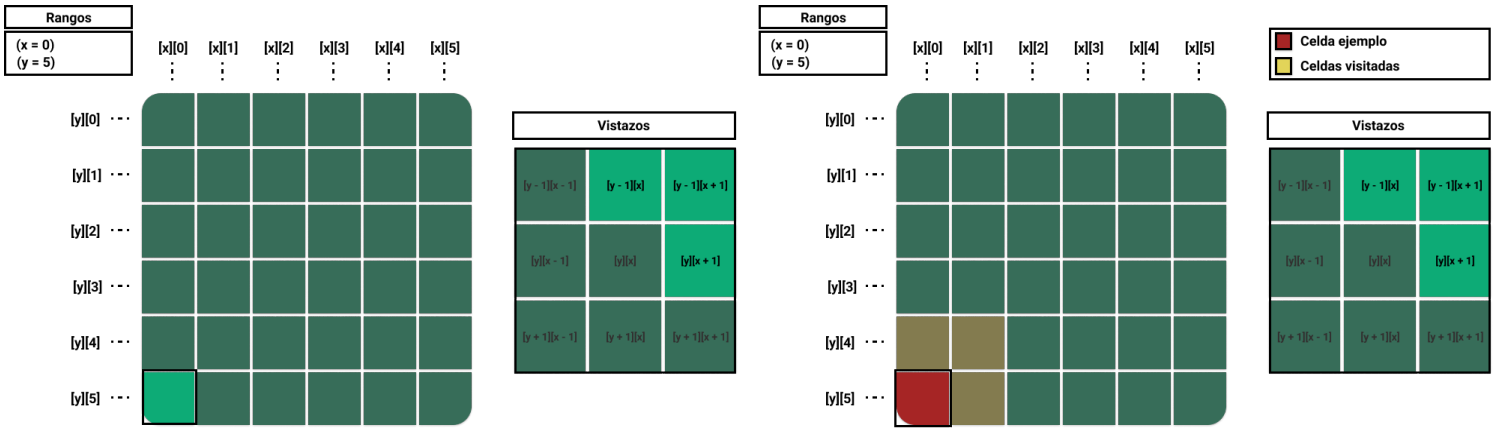


Figura (12): Representación vistazos de una celda, cuando se encuentra en la esquina inferior izquierda.

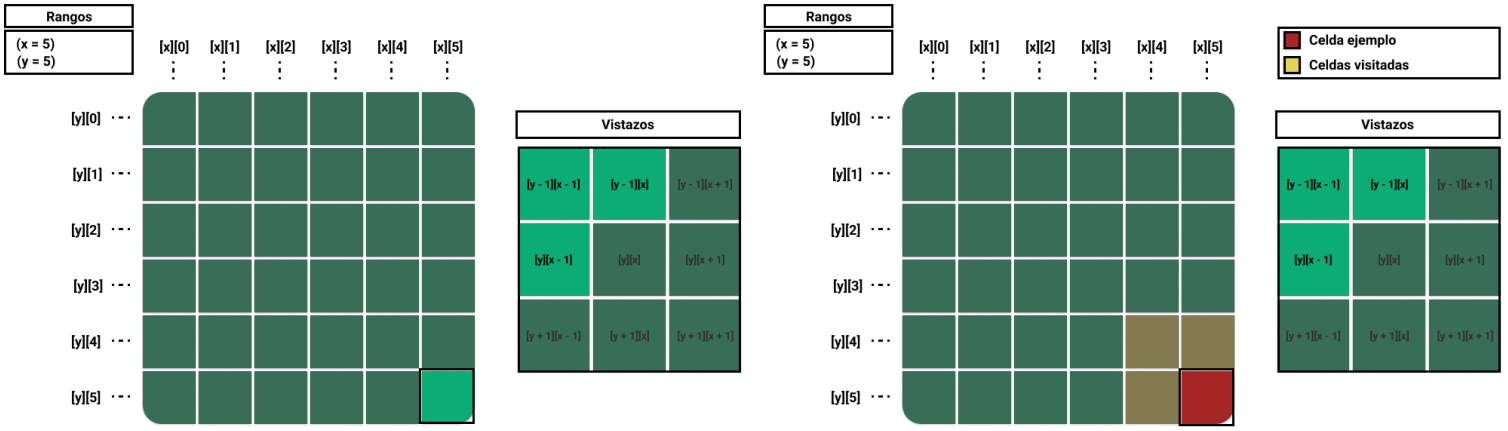


Figura (13): Representación vistazos de una celda, cuando se encuentra en la esquina inferior derecha.

Implementación de la IA

El agente computacional toma el lugar del jugador que utiliza las fichas blancas, considerando así que siempre iniciará jugando el jugador y será el agente el quien responda.

La jugada que realice el agente será calculada dependiendo de la dificultad que se elija para el juego, pudiendo elegir entre las dificultades fácil, normal y difícil.

Para los tres niveles de dificultad se utilizó el algoritmo de poda alfa beta, el cual se encarga de retornar la jugada que le genere la mayor utilidad a la máquina. Dado que el juego del reversi tiene una complejidad muy alta resulta imposible explorar todo el árbol de juego en cada turno de la máquina, por lo cual este algoritmo tiene implementada la mecánica de profundidad. Es en base a la dificultad del juego que se define un nivel de profundidad para el algoritmo de poda alfa beta siendo esta de 2 para el nivel fácil, 4 para el nivel medio y 6 para el nivel difícil. La profundidad indica cuántas jugadas a futuro examina el algoritmo para encontrar la jugada que genere la utilidad más alta.

La función de utilidad implementada para el algoritmo de poda alfa beta se basa en revisar si es que el tablero de juego ya ha sido completado o ya se ha alcanzado la profundidad deseada. De ser el caso, la utilidad se calculará como la diferencia entre el número de fichas negras con el número de fichas blancas (negras - blancas) siendo este un valor negativo en caso de ganar las blancas y positivo en caso de ganar las negras, por lo que la utilidad se multiplica por -1.

El agente resultó ser bastante eficiente en cada uno de los niveles de dificultad, no tardando demasiado en calcular la jugada óptima y en muchos casos terminando el juego en un número muy reducido de jugadas.

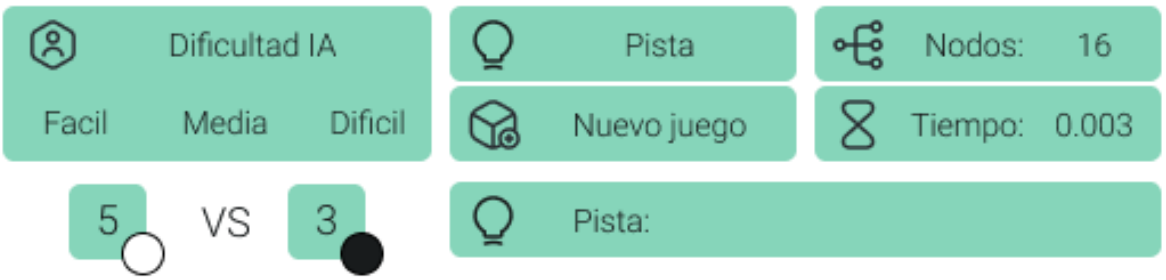


Figura (14): Extracto de la GUI.

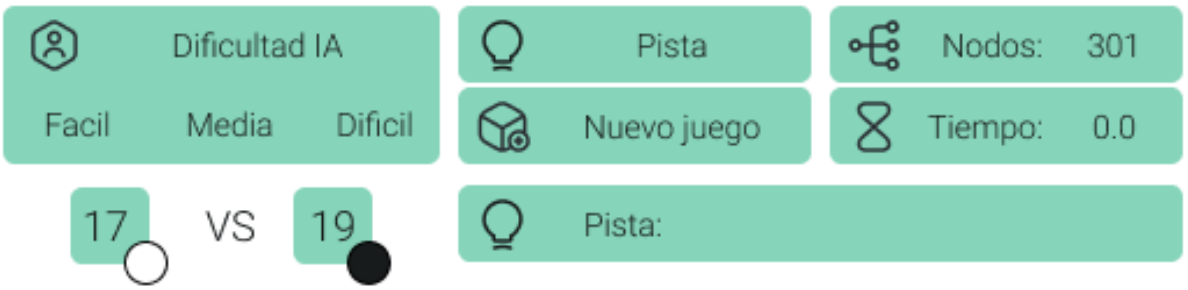


Figura (15): Extracto de la GUI.

Estos son los estados inicial y final del juego, en él se aprecia que a medida que el juego avanza el programa tarda menos tiempo en encontrar las jugadas.

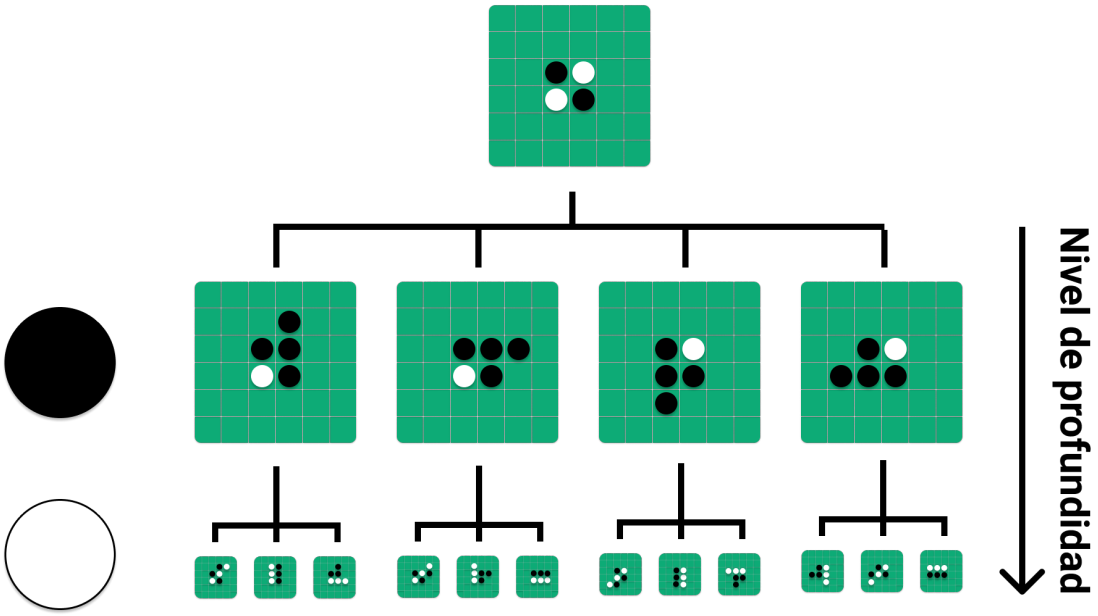


Figura (16): Representación de las jugadas posibles.

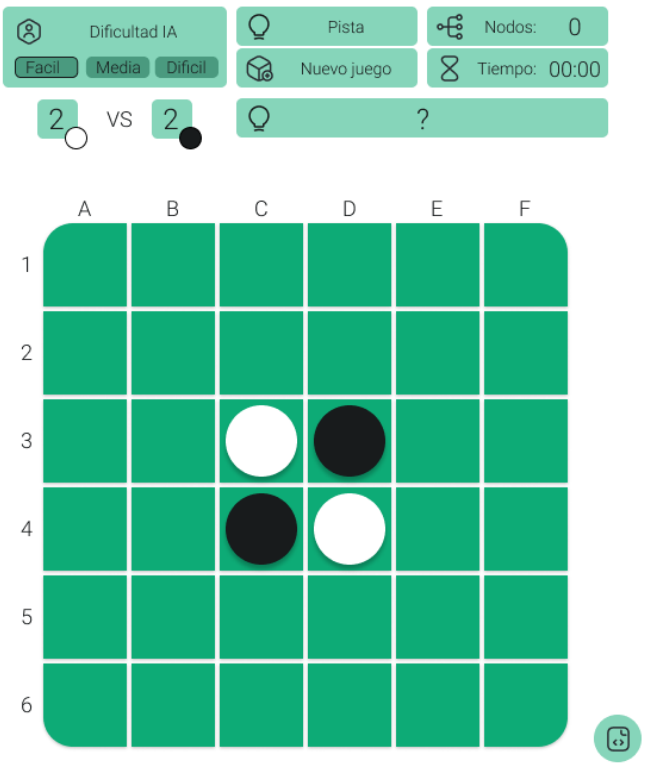
Direcciones artísticas del desarrollo

Cabe destacar que, al acrecentar este apartado, se buscan iluminar los aspectos relacionados con el enfoque cohesivo del apartado gráfico del proyecto. Apartado, que comúnmente es omitido o no tomado en consideración en muchos proyectos informáticos.

Aspecto gráfico

En general, se diseñó la parte gráfica del juego de manera minimalista, es decir minimizando los detalles y sólo exaltando lo más importante para que sea agradable y comprensible a simple vista mientras se juega.

Interfaz gráfica



En la figura (17), se puede apreciar el diseño completo del juego, como se dijo anteriormente, este diseño nos muestra un diseño minimalista en el que predomina en la gama de colores el color verde, sin embargo las fichas, son de los colores clásicos del juego, que son blanco y negro. Se puede observar que la interfaz gráfica del juego contempla muchos cuadros con palabras y números, estos elementos contienen cada uno, una función que van a ser explicados a continuación.

Figura (17): Vista del tablero.

Interacción con el usuario

En la siguiente figura se puede reconocer el control de la interfaz gráfica, explicando que es y para qué sirve cada parte de esta.

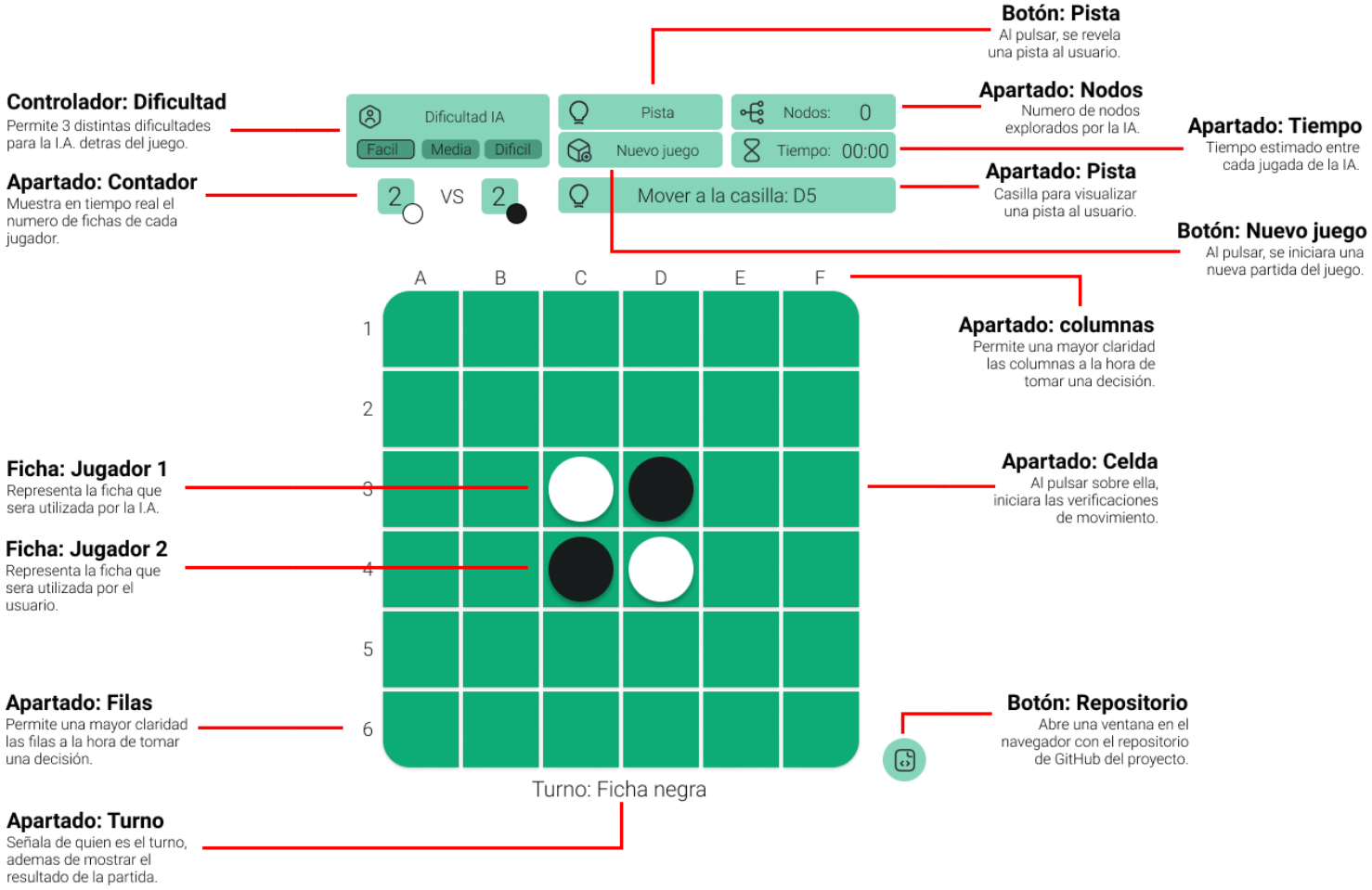


Figura (18): Estructura de archivos.

Paleta de colores

A continuación, se presenta la paleta de colores utilizada en la realización de la interfaz del juego Reversi (Figura 19):
Cómo es posible observar en la interfaz del juego, el color #0DAB76, que contiene principalmente color VERDE, usado para rellenar las celdas del tablero del Reversi, mientras que las fichas de los jugadores ocupan los colores blanco (#FFFFFF) y negro (#181B1C), además de este último se ocupó también en las letras de la interfaz gráfica. Por su parte, los botones de la interfaz fueron rellenados con el color #86D5BA y para los botones de control de dificultad fue utilizado el color #4A9A7F. Además de ocupar el color rojo #FF0000 para la hitbox del tablero.

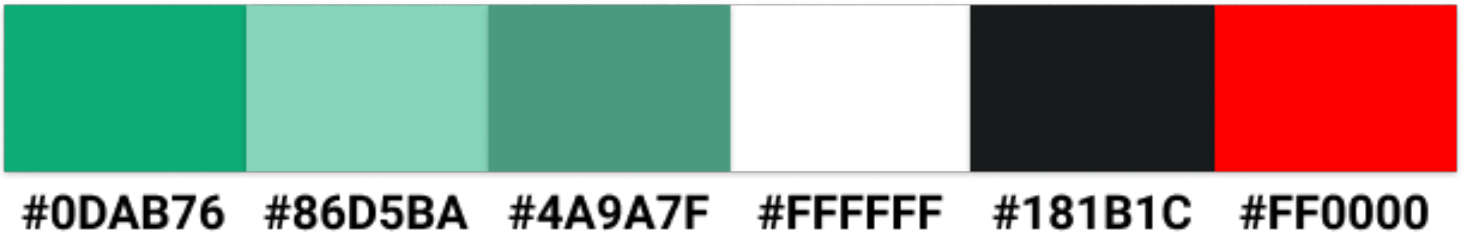


Figura (19): Paleta de colores utilizada en el diseño de la interfaz.

Alineación de recursos

En las siguientes imágenes se puede apreciar de qué forma estaban alineadas las fichas dentro de las celdas y las celdas en sí.

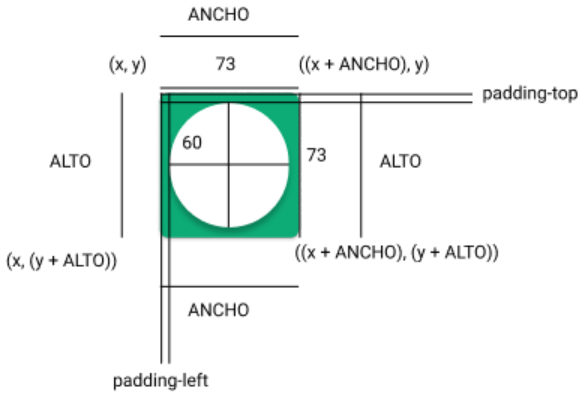


Figura (20).

En la figura (20) se puede apreciar la forma en la que una ficha estaba alineada simétricamente dentro de una celda.

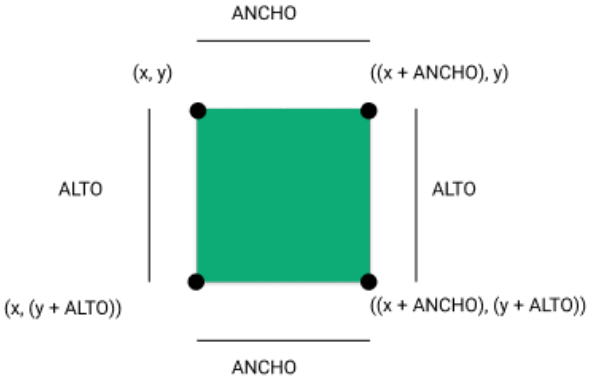


Figura (21).

En la figura (21) se puede apreciar como es la alineación de las celdas en el tablero del juego.

Uso de recursos

A continuación se muestran los recursos gráficos en detalle utilizados en el juego (Figura 22), cabe mencionar que estos recursos se pueden encontrar de manera “ensamblada" en la carpeta ***assets***.

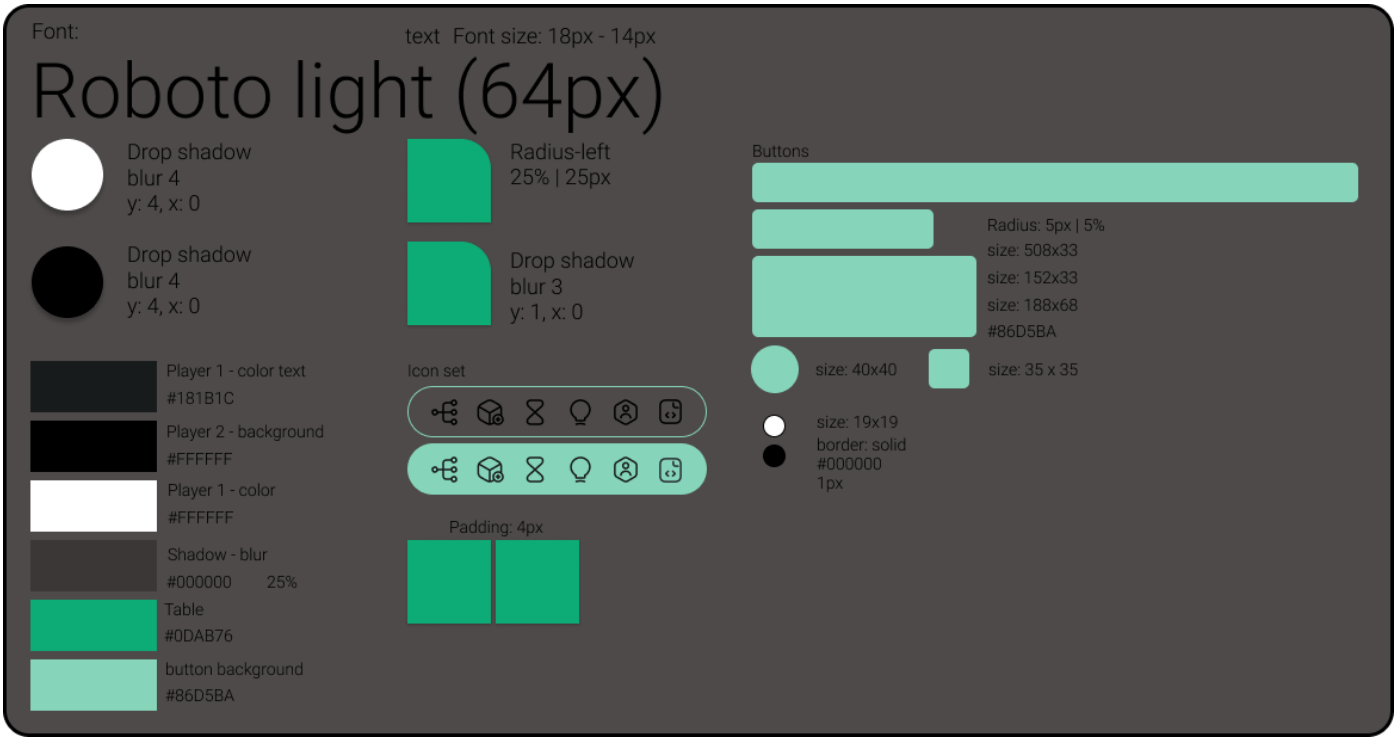


Figura (22): Resumen de recursos utilizados.

Conclusiones

Al momento de hacer un videojuego como Reversi se deben considerar ciertos aspectos generales, uno es el diseño de algoritmos en código para su funcionamiento, otro es el diseño gráfico del juego, ya que este va a mostrar cada uno de los movimientos que se realizan y otro aspecto que se podría encontrar en este juego es la implementación de la inteligencia artificial, esto quiere decir que se necesita un conocimiento previo de algoritmos de búsqueda y razonamiento para realizar una IA de este juego.

Bibliografía y referencias

colaboradores de Wikipedia. (2021, 27 julio). Reversi. Wikipedia, la enciclopedia libre. *https://es.wikipedia.org/wiki/Reversi*

Play Reversi Small. (s. f.). Maths Fun. Recuperado 21 de septiembre de 2021, de *https://www.mathsisfun.com/games/reversi-small.html*

Reversista - Reversi / Otelo - Historia del reversi. (s. f.). Google Sites. Recuperado 21 de septiembre de 2021, de https://sites.google.com/site/elreversista/historia