Informe del Proyecto
De
Programación

Aula: C-211

Integrantes:

Leonardo Amaro Rodríguez

Alfredo Montero López

Nuestro Proyecto consta de dos partes fundamentales: el DLL donde se encuentra la lógica fundamental del juego y la interfaz de usuario creada en Unity. Sobre las cuales profundizaremos a continuación.

DLL:

Como anteriormente se dijo, aquí se encuentra la lógica del juego, dígase las reglas, los tipos de jugadores, el funcionamiento del juego de dominó y el evaluador de jugadas.

Entre las clases principales se encuentran: DominoGame(el encargado del funcionamiento del juego), DominoPlayer(la clase que define a los jugadores), IRules(interfaz donde se definen las reglas del juego), Move(la jugada que realiza cada jugador), Piece(la ficha) y Observer(el que analiza y puntúa cada jugada)

IRules:

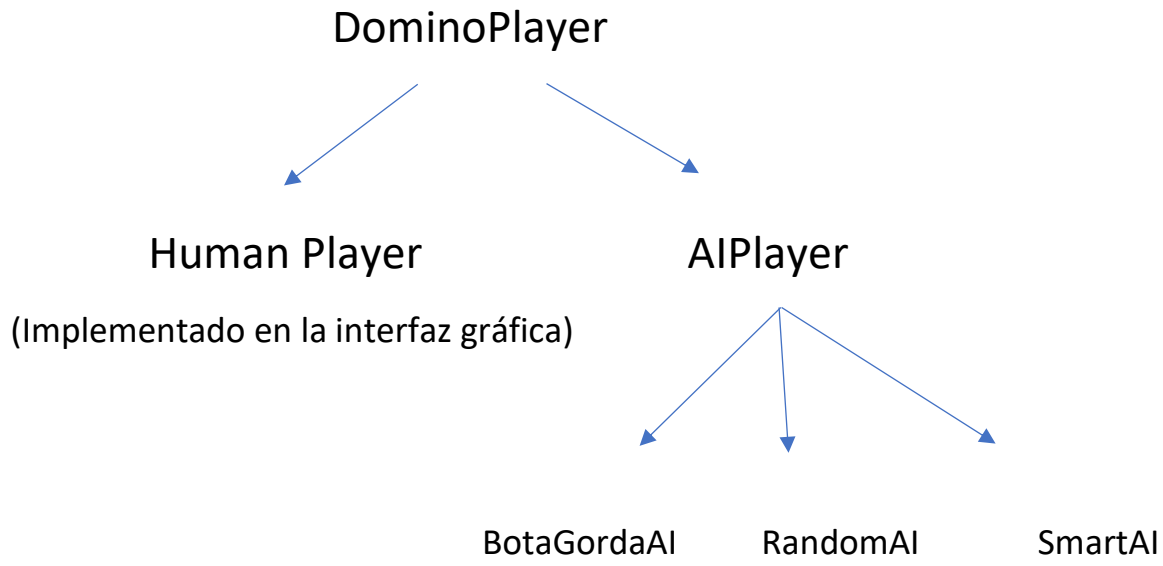
Interfaz donde se encuentra la declaración de las reglas que se pueden modificar. Nuestro proyecto permite la modificación de:

- Criterio de finalización del juego
- Criterio de cuando se pueden unir dos piezas
- Criterio del valor que tiene cada mano del jugador

Move:

Estructura que devuelven cada jugador como jugada. En ella se almacena la id del jugador, un bool que dice si se pasó, la ficha que jugó y otro bool que indica si jugó por la izquierda o por la derecha de la “mesa”. También consta de dos métodos que son los encargados de crear una jugada y un pase respectivamente.

Jerarquía de los tipos de jugadores:



DominoPlayer:

Cada jugador consta principalmente de una ID, que es lo que diferencia a cada jugador; una lista de piezas y dos diccionarios, cada uno donde se almacena la id de los jugadores y las fichas que no tienen sus parejas y adversarios respectivamente.

AIPlayer:

Cada AIPlayer está suscrito a un evento creado en DominoGame. El evento invoca a sus suscriptores cada vez que un jugador realiza una jugada. Esto se hace para llevar la cuenta de que jugador no lleva un valor determinado.

BotaGordaAI: Como su nombre lo indica su modo de juego es jugar siempre la ficha con mayor valor.

RandomAI: Literalmente juega una ficha aleatoria de su mano.

SmartAI: Este jugador le va asignando a cada ficha un valor dependiendo de ciertos parámetros. Cuando selecciona las fichas que se pueden jugar, crea un

duplicado en forma de matriz de n columnas y 2 filas, siendo n la cantidad de fichas que se pueden jugar. Luego va puntuando cada ficha al llamar a distintos métodos incorporados a SmartAI:

- PointsForTeamMate: si el(los) aliado(s) no tienen una ficha con un número específico, se aumenta el valor correspondiente a la cara de la ficha que le corresponda ese valor con una constante definida en el SmartAI, con el objetivo de jugar por el lado donde está la ficha que no tiene mi compañero.
- PointsForOpponent: Si el(los) oponente(s) no tienen una ficha con un número especifico, se le aumenta el valor correspondiente a la cara opuesta de la ficha que le corresponda ese valor con una constante definida en el SmartAI, con el objetivo de jugar una ficha por la cual el adversario no puede jugar.
- PointForDouble: Aumenta el valor de las fichas que en ambas caras tienen el mismo valor.
- PointForSameNumber: Aumenta el valor de las fichas que en una de sus caras tienen el mismo valor que las otras, para priorizar jugar las fichas de una misma “data”.
- PointForPlaced: Aumente el valor de las fichas que puedan ser colocadas por el lado de la mesa por la cual jugó un adversario.

Luego de obtener el puntaje juega la ficha que tiene el lado con mayor valor.

DominoGame:

Dentro de la clase se guardan los jugadores, las reglas definidas para el juego, el historial de jugadas, las piezas que quedaron sin distribuir a los jugadores y los extremos de la mesa.

Cuando se inicia el juego, se les asignan a los jugadores n fichas aleatorias generadas tomando el valor máximo de las fichas a crear definidas en las reglas, siendo n el máximo número de fichas que puede tener cada jugador. También se selecciona de manera aleatoria el jugador que va a empezar a jugar.

En nuestro proyecto, la llamada “mesa” donde van las fichas no es más que un historial de jugadas, donde cada jugada, definida como la estructura Move, almacena la id del jugador que la realizó, por donde jugó, si se pasó y la ficha que jugó. Por tanto, cada vez que un jugador juega, se actualizan los extremos de la “mesa”.

Cuando se realiza una jugada se invoca al evento OnMoveMade, que le pasa como parámetros a sus suscriptores la jugada (Move) realizada.

En resumen, la clase DominoGame, tiene definida la forma de obtener las piezas que se pueden jugar respecto a una lista de piezas y las reglas vigente, es la que se encarga de decirle a cada jugador cuando le toca jugar, si se acabó el juego y cuales son las piezas ubicadas a cada extremo de la “mesa”.

Observer:

Esta clase se encarga fundamentalmente de observar los movimientos realizados por cada jugador humano y le da una evaluación.

Su funcionamiento se basa en el razonamiento de un SmartAI, pero en vez de devolver la mejor ficha a jugar, devuelve todas las fichas que se pueden jugar de forma ordenada, siendo la primera ficha la mejor jugada posible. De ahí, al recibir que ficha jugó el jugador humano devuelve un valor calculado de la siguiente forma:

i= índice de la pieza con respecto a la lista de piezas ordenadas

t = cantidad de piezas que se pueden jugar

Retorna 1-(i/t)

De esta manera mientras mas cercano a 1 sea el valor devuelto, mejor es la jugada realizada.

La declaración del Observer se encuentra en el dll pero su implementación esta en la interfaz de usuario.

Interfaz de Usuario:

La interfaz de usuario de nuestro proyecto se creó en el motor de videojuegos ***Unity*** y su código fuente se encuentra en un repositorio apartado (link en el archivo Readme.md).

Nuestra interfaz permite darle uso a la funcionalidad ofrecida por el DLL en un ambiente agradable, además de varias opciones de configuración del juego y sus reglas sin necesidad de realizarle cambios al código de una aplicación de consola. La interfaz presenta una forma divertida de experimentar el juego, permite crear equipos, así como tantos jugadores humanos e *IA* como se desee; permite además configurar la forma en que se dibujarán las fichas, con dos opciones en su versión inicial (los clásicos Puntos o un Número representativo del valor), pero es muy sencillo adicionar más formas de dibujado con muy pocos cambios al código existente. Además de esto los "dibujadores" son adaptativos al valor de la ficha, es decir, están diseñados para dibujar cualquier valor que se configure para estas, por ejemplo, si nuestro valor máximo es 20, es decir, existe un "doble 20", ¡nuestro dibujador de puntos (por ejemplo) dibujará perfectamente 20 puntos en cada lado de la ficha!

Controles:

Dentro del propio juego, previsto que las partidas pueden tener una cantidad potencialmente infinita de fichas dadas las extensas opciones de configuración, el jugador puede mover la cámara alrededor del tablero con las teclas W, A, S y D, así como alejar o acercar la vista con la rueda del ratón.

En cada turno del jugador se le mostrarán las fichas que tiene en su mano, si estas no caben en la pantalla se puede utilizar el click izquierdo del ratón para desplazarse por la lista de piezas con comodidad; además saldrán resaltadas en distintos colores las fichas que el jugador puede jugar en el turno en particular (azul para jugar por la izquierda, rojo para la derecha y violeta para cualquiera de los lados), al seleccionar una ficha se le presentarán botones al jugador en dependencia de los lados en que se pueda jugar la ficha. En el caso de que no pueda jugar ninguna ficha en su turno, se le presentará un botón al jugador para

Pasar, este botón no estará disponible si el jugador tiene fichas para jugar (esto es posible que se adicione como opción configurable en una versión posterior del proyecto).