Informe:

Este proyecto tiene construcción modular con el objetivo de cumplir los principios de la filosofía SOLID. Con este propósito, se hace uso extenso de las interfaces, destacando como esenciales para el funcionamiento del proyecto los objetos Rules, Table y State.

Los dos primeros objetos son clases. Rules almacena los criterios que se utilizarán para el funcionamiento del juego y Table se encarga de poner a punto los elementos para iniciar el juego, respectivamente. En cambio, State es un struct y sirve como portador de la información del juego, este dato se suministrará a todo el que necesite conocer del estado del juego.

Por otro lado, la estructura desacoplada del juego permite un nivel muy alto de presonalización de las partidas.

Teniendo en cuenta lo anterior, las clases generales son las siguientes:

\*Reglas(Rules)

-Game: El algoritmo del juego de dominó

-Pack: Genera las fichas a utilizar en el juego

-Dispatcher: Despacha las fichas a los jugadores

-Update The Board: Actualiza el tablero

-IsValid: Valida una jugada como correcta o incorrecta

-EndGame: Decide el final del juego.

\*Mesa(Table)

-Time Of Distribution: Despacha las fichas de un jugador según el criterio de la clase Rules.

-Play : Inicia una ronda del juego.

-Resert: Inicializa todos los elementos necesarios para el desarrollo de una partida.

\*Jugador(Player)

-AddHand () :Agrega una mano.

-Have(): Analiza si el jugador puede hacer una jugada.

-Play(): Se realiza una jugada.

\*Estrategia(Strategy)

Es la manera de jugar. En el juego hay tres implementaciones de la interfaz IStrategy, Random\_Player,Fat\_Player,Standart\_Player.Todo jugador analiza qué fichas puede jugar.

-Random\_Player: Manera de jugar random.

-Fat\_Player: Bota gorda. Ordena las fichas que puede jugar según el "peso" de la ficha. Lanza la más pesada.

-Standart\_Player: Juega de manera random y botagorda.

\*Parejas(Couples)

Se encarga de conformar los equipos

-Teams\_in\_Smash : Devuelve los equipos que participan en el juego.

-Team: Conforma los equipos.

-ThisTeam: Identifica en qué equipo esta cada jugador.

\*Árbitro(Referee)

Se encarga de alamacenar la información de los jugadores, sus puntos, rondas ganadas.

\*IDomino, esta interfaz es implementada por dos clases: Domino\_for\_Rounds y Domino\_for\_Points. El primero recibe por el constructor la cantidad de rondas mínimas que declara a un equipo/jugador ganador y el segundo tiene como parámetro la cantidad de puntos que declara ganador a un jugador.

\*Score

Esta clase puntúa la mano de un jugador siguiendo determinado criterio.

Implementaciones:

Juego Clásico:

ClassicGenerator<T> Implementa la interfaz IGenerator. Esta clase genera fichas de dos caras, recibe como parámatro un array que contiene los valores de las fichas a generar.Tiene el método PieceGenerator() que devuelve una lista de IFichas que contiene el estuche con la distribución de fichas del juego usual.

ClassicDistribution<T> Implementa la interfaz IDistribution, emplea el método >DistributionOfPics<T> para despachar una mano de fichas a un jugador. Este método recibe las fichas del juego y un entero que representa la cantidad de fichas que repartirán por mano.

ClassicGame<T> Implementa el algoritmo del juego clásico y utiliza el delegado Standart\_Filter para informar que una jugada es válida.

ClassicScore<T> La forma clásica de puntuar una ficha.

Variantes:

\*FullGenerator<T> Implementa la interfaz IGenerator, genera un estuche con todas las combinaciones posibles de fichas que se pueden construir con todos los valores que van a tener las fichas en el juego.

\*CousinDistribution<T> Implementa la interfaz IDistribution, la cual es una variante a la modalidad clásica de despachar que utiliza números primos para repartir random.

\*LonganizaGame<T> Implementa el juego de la longaniza y utiliza el delegado Longaniza\_Filter para validar una jugada.

\*DoubleScore<T> Implementa la interfaz IScore<T> y solamente puntúa según los dobles que tenga un jugador ingnorando el resto de fichas.

Desarrollo del juego:

El juego ocurre en el método Game() de la clase Rules, el cual devuelve un IEnumerable de State. Funciona con un ciclo que se ejecuta simpre que no se cumplan las condiciones para finalizar el juego . Se le informa a los jugadores del estado del tablero por un State y cada jugador comprueba si puede jugar utilizando el método booleano Have(). Este recibe como parámetro el State con el estado actual del juego y un delegado que informa si una jugada es válida, que devuelve true en caso de que el jugador tenga una ficha en su mano que pueda jugar. De poder jugar, el jugador devolverá una tupla ficha y posición a jugar. Esta tupla es recibida por el método booleano IsValid() y devuelve el true si la jugada es válida. Además, se actualiza el tablero con el método Update\_The\_Board(). Por último se devuelve un State con la condición del juego postjugada. Despúes de esto si la condición para terminar el juego es válida, se ejecuta el método EndGame() y este determina al ganador.

Interfaz Gráfica:

La interfaz gráfica, aprovechando las herramientas que ofrece Unity como motor de desarrollo, fue diseñada en este programa, en conjunto con las clases y métodos proporcionados por el namespace “DominoEngine”, cuyas componentes fueron previamente explicadas. Para ello se utilizó C# como lenguaje de programación y Visual Studio Code como entorno de desarrollo.

La interfaz gráfica está subdividida en 2 escenas: el menú principal, y el juego. La primera, engloba varios paneles de interacción, cada uno con botones y sliders (deslizadores) que permiten al usuario realizar modificaciones al juego, desde el aspecto visual y sonoro hasta aquellas relacionadas con la lógica interna del juego. En cada botón o slider se puede apreciar un pequeño texto que describe su funcionalidad. De igual forma, cada panel consta de un título ubicado en la parte superior. La segunda escena, el juego, muestra visualmente las partidas de dominó así como los ganadores. Además, en el extremo superior derecho de la pantalla de dicha escena se encuentra el botón “Pausa” que, al oprimirlo, permite acceder a las opciones de “Volumen” y “Brillo” para modificarlas, así como volver al Menú Principal.

Respecto al juego en sí, son modificables los siguientes aspectos:

1. *Modalidad de juego:* este hace referencia a la forma de juego, es decir, al total de valores disponibles para jugar en la mesa. En nuestro caso se puede jugar el juego “clásico” (solo 2 valores) o la conocida “longaniza” (4 posibles valores en la mesa).

2.*Generador de fichas:* se refiere a la manera en la que las fichas son creadas dado un conjunto de posibles valores (estos valores no son modificables, por ahora). Tenemos el generador “clásico” que crea las fichas en orden, según la lista de valores, y, por otra parte, el generador “crazy” que genera fichas repetidas, por lo que crea un efecto casi impredecible en la jugada del contrario.

3.*Distribuidor de fichas:* este aspecto está relacionado con la forma en la cual se le reparten las fichas a cada jugador. Sus variantes son: “clásico”, consiste en repartir las fichas al azar a cada jugador, respetando el número máximo de piezas que debe recibir cada uno; “cousin” que basa su distribución en números primos.

4.*Puntuación por ficha:* se refiere al conteo de puntos según los valores de las fichas que quedaron en las manos de cada jugador al final de una partida. Tiene 2 variantes: la “clásica”, donde el total de puntos acumulados por jugador es igual al valor de cada una de sus fichas; la “double” donde al final solo van a acontarse los puntos de las fichas que sean “dobles”.

5.*Condición de Victoria:* este aspecto decide el criterio de victoria: si es por mayor cantidad de rondas ganadas, o por mayor cantidad de ountos acumulados.

Con esta interfaz gáfica se pretendía mostrar, a través de un entorno de videojuego, las funciones modificables del namespace “DominoEngine” así como la puesta en marcha de su lógica. ¡Esperamos sea de su agrado!