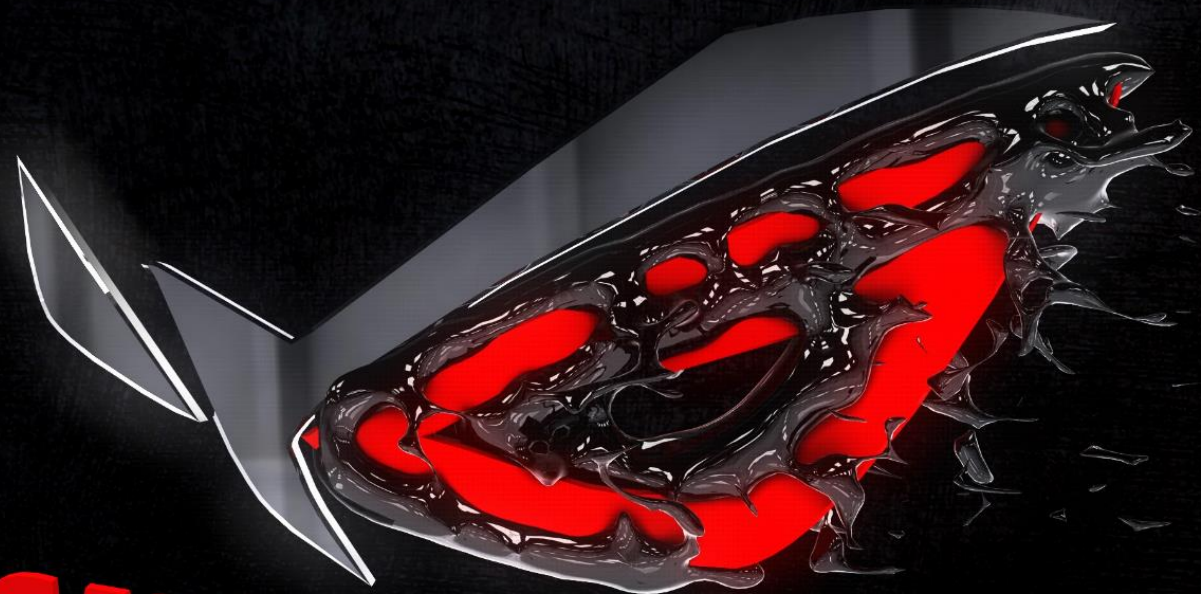
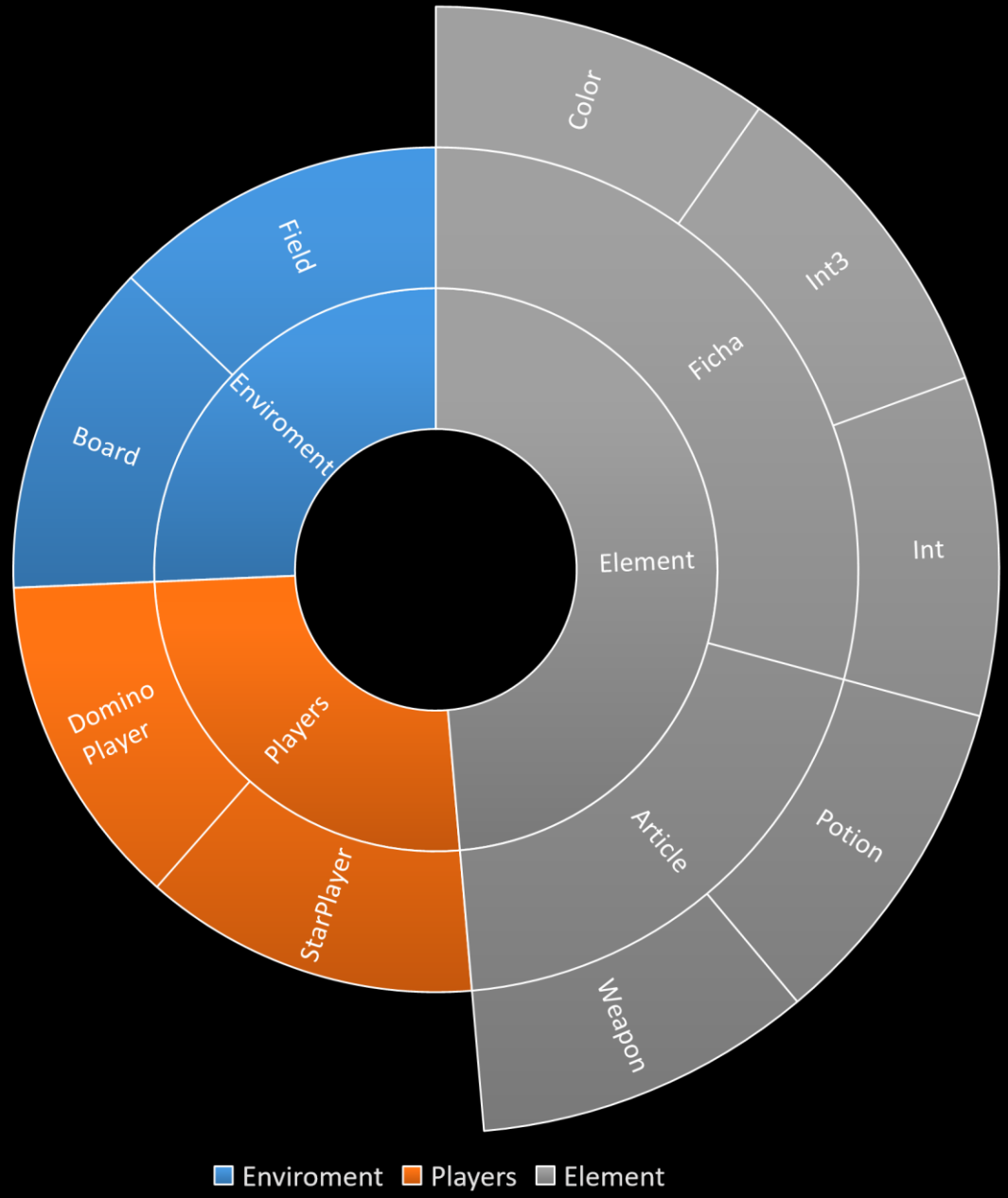


GAME PLATFORM

Estructura de Clases

- Structure(GamePlatform): Permite modular cualquier juego estableciendo conceptos como ambiente, jugadores y elementos, que son comunes a todo tipo de juego.
- Aplication (DominoPlatfom, StarcraftPlatform): Establece la lógica del juego en cuestión, concretizando las funcionalidades básicas e imprescindibles.
- Aplication Management(Domino.Net, Starcraft): Define como va a funcionar el juego y es la capa de interacción con el usuario.

Esqueleto del programa



Environment

```
public class Environment<T, B, BT, P> where B: Player<BT, P>
{

    24 references | enzor, 32 days ago | 1 author, 1 change
    public T Collection { get; protected set; }
    55 references | enzor, 32 days ago | 1 author, 1 change
    public List<B> Players { get; protected set; }

    73 references | enzor, 32 days ago | 1 author, 1 change
    public int ActualPlayer { get; protected set; }
```

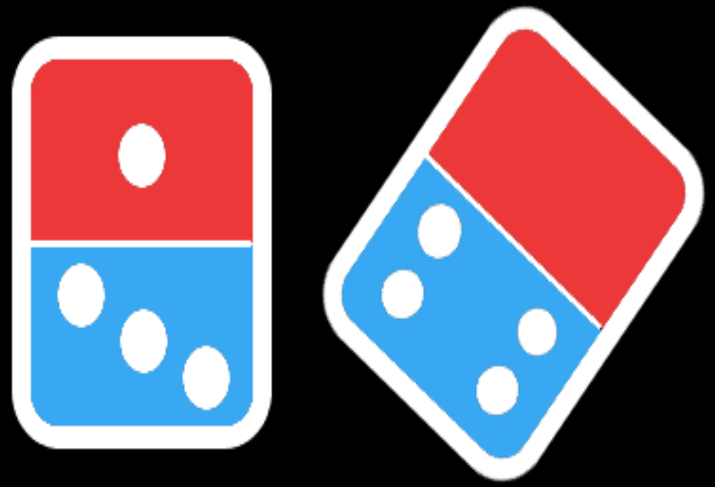
Board

```
public class Board<T, P> : Environment<TreeN<T, P>, DominoPlayer<T, P>, List<Ficha<T, P>>, P>
{

    List<Ficha<T, P>> FichasAfuera;

    //play log
    List<Ficha<T, P>>[] rounds;

    //scalar that defines the direction of the game
    int pass;
```

Field

```
public class Field<P> : Environment<Element<P>[,], StarPlayer<P>, Article<P>[], P>
{

    46 references | enzor, 32 days ago | 1 author, 1 change
    public List<(int, int)> PlayersPositions { get; private set; }

    1 reference | enzor, 32 days ago | 1 author, 1 change
    public Field(Element<P>[,] Collection, IEnumerable<StarPlayer<P>> Players) : base(Collection, Players)
    {

        PlayersPositions = new List<(int, int)>();

        for (int i = 0; i < Players.Count(); i++)
            PlayersPositions.Add((2, 2));
```

DOMINO



Comienzo del juego

- Al usuario se le da la posibilidad de escoger como quiere que funcione el juego. El mismo puede realizar una serie de cambios, que son controlados para que sean válidos, que van a determinar como se ejecuta el juego.
- Cuando se presiona el botón para jugar, si hay la cantidad de jugadores necesarios, se inicializa el juego, a partir de las decisiones del usuario, que va a tener en sus manos una serie de cuestiones que van a tener que ver con el funcionamiento y desarrollo del juego.

Qué se puede cambiar

- Los jugadores (verdaderamente no se escogen los jugadores, ya que todos son jugadores de domino, sino cómo juegan): First Player (juega lo primero que ve); Random Player; Bota Gorda; Bota Suave; Smart Player.
- El tipo de ficha: Ficha clásica de números; Ficha múltiplo de tres(la suma de los lados juntados tiene que ser múltiplo de tres); Ficha clásica de color.
- Condiciones de finalización(es obligada la condición de trancarse): condición de Pegarse y con cuantas fichas se considera pegado el jugador; condición de terminar el juego si alguien se pega.
- Robar ficha cuando no se tiene una jugada valida.
- Plin: Si el jugador pone el cinco, pasa al que le tocaba el turno.
- Generar las fichas: Generarlas todas o generar una cantidad menor al azar.

Qué se puede cambiar (cont)

- Repartir las fichas: Random o en orden.
- Pase de turno: Clásico o si se pone una ficha con todos sus lados iguales se invierte el orden.
- Quien gana: el que menos puntos tenga al acabar el juego(clásico) o el que tiene la mayor cantidad de lados iguales.
- Mano inicial del jugador.
- Número (o cantidad de colores) tope en las fichas.
- Cantidad de lados de la ficha.
- Jugadas válidas por cada cara de la ficha.

Inicialización del Juego luego de pasar por el filtro lo pedido por el usuario

```
/// <summary>
/// Start the Board and store the necesary stuff for later
/// </summary>
/// <typeparam name="F"></typeparam>
/// <typeparam name="T"></typeparam>
/// <param name="generate"> how to generate the fichas for the game </param>
/// <param name="distribute"> how to distribute the fichas to players at the beginning </param>
/// <param name="robar"> play with this rule? </param>
/// <param name="players"></param>
/// <param name="print"> how to print the game </param>
/// <param name="pass"> how to pass the turn </param>
/// <param name="winner"> how to choose the winner </param>
/// <param name="InitialHand"> how many fichas at the start hand of each player </param>
/// <param name="conditions"> game rules </param>
1 reference | Enzo D'toste, 4 days ago | 1 author, 4 changes
public void Initialize<F, T>(GenerateFichas<T, Image> generate, Distribute<T, Image> distribute, bool robar, List<PlayFicha<T, Image>> players,
    IBoardPrint<T, Image> print, PassTurn<T, Image> pass, Winner<T, Image> winner, int InitialHand, params IConditions<T, Image>[] conditions) where F: Ficha<T, Image>
{

    List<DominoPlayer<T, Image>> dominoPlayers = new List<DominoPlayer<T, Image>>();

    foreach(var player in players)
    {
        dominoPlayers.Add(new DominoPlayer<T, Image>(player));
    }

    board = Board<T, Image>.StartGame(dominoPlayers, generate, distribute, InitialHand, pass);
    this.print = print;
    this.pass = pass;
    this.winner = winner;
    this.conditions = conditions;
    this.robar = robar;

    dimGame = new DimGame(pictureBox1.Width, pictureBox1.Height, new DimFicha(40, 40));

    timer1.Start();
```

Desarrollo del juego

- Luego de inicializar comienza el juego, controlado principalmente por el método PlayTurn de la clase Board. Este manda a jugar al jugador, el cual juega de acuerdo a su forma. Si el jugador intenta meter forro, se es tomado como que paso turno. Se revisa en caso de tocar mesa si existe robar, e ese caso el jugador, si existen fichas afuera, roba y el turno le vuelve a tocar a él. Por último se chequea si se finalizó el juego, y si no, entonces se define a quien le toca jugar ahora, que va depender de las reglas de pase de turno.

Árbol del juego

- Su estructura es la de un árbol n-ario con n igual a la cantidad de lados de la ficha. Cada hijo es un lado de la ficha y tiene una lista de árboles que va a tener como tamaño máximo la cantidad de veces que se puede jugar por un solo lado de la ficha.
- Sabiendo esto, para saber las jugadas disponibles en la mesa basta con preguntar recursivamente, si la lista de cada uno de los árboles hijos no ha sobrepasado su capacidad máxima.
- Agregar una ficha, es recursivo y se hace teniendo en cuenta que lado de la ficha se juega(index_side) y cual es el lado de la mesa que hace match con ese lado jugado(board_side).

StarCraft

Como Jugar

- Arrows(→,←,↑,↓): Moverse.
- P : Pasar el turno.
- A : Atacar.
- Numpad # : Tomar el articulo #.
- Cada arma tiene su rango, fuerza, desgaste de energía y forma de ataque.
- Cada heroe tiene su propio tope de vida y energía, igual que sus propios artículos, los cuales puede cambiar por los que encuentre en el campo.
- Cada poción tiene su funcionalidad (ILife: actua sobre la vida; IEnergy: actua sobre la energía).

On Screen

- Rango
- Fueza
- Gasto de energía
- Mapa
- Objeto jugable
- Coleccion del jugador
- Jugador Actual

Por qué este otro juego

- Este juego muestra la versatilidad del Game Platform, el cual puede servir lo mismo para un domino, que para un juego de guerra. Dos juegos que en principio parecen no tener nada en común, parten de la misma raíz, y es que en sí todo juego tiene: un ambiente en el que se desarrolla, jugadores, y elementos tanto jugables, como ambientales.
- Este juego no es tan versátil como el dominó y lo sabemos, sin embargo tiene la estructura para serlo. En este proyecto nos hemos limitado a hacerlo lo mas sencillo posible, pero sin tener que cambiar casi nada de lo que esta hecho, estamos seguros de que se puede complejizar todo lo que se quiera.

Conclusiones

- En este proyecto hemos tratado de cumplir con los principios SOLID en la medida de lo posible, para hacer nuestro juego lo más extensible posible. Ya sea: crear nuevos juegos, agregar nuevos jugadores, elementos, reglas del juego para personalizarlo y adaptarlo a distintas culturas, se logra de manera rápida y sin muchos contratiempos, respondiendo de manera eficiente al mercado.
- Por ejemplo:

✓ Cambiar el jugador en el domino, es simplemente cambiar como va a jugar

✓ Cambiar el pase de turno sería simplemete proveer al método de un vector que defina a que jugador le toca.

✓ En caso de inventarse una nueva regla se agrega como ICondition y el juego solo, se encargará de ejecutar la regla.

✓ Para cambiar como se ve el juego basta con cambiar el PrintParameters(en caso de que fuese necesario) y definir la nueva forma de imprimir en pantalla.