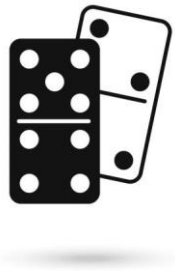
Reporte del Proyecto de Programación



En este proyecto como es conocido se pretende dar a luz a una biblioteca de clases que permita modelar diferentes variantes del juego Dominó y diferentes estrategias de jugadores virtuales, además la creación de una interfaz gráfica que permita visualizar los resultados de los juegos.

Narremos pues, a la vez que pasamos por diminutos fragmentos de códigos todo lo pensado en la realización del proyecto.

Primera pregunta que nos hicimos fue, qué queremos modelar, la respuesta era evidente, el juego del dominó ,nos vino de repente la pregunta qué es dominó, pues después de un tiempo de que nos hicimos esa pregunta fue que en verdad entendimos que el dominó fuera de cualquier contexto es esa parte arraigada de la cultura cubana, que nunca pasa de moda por lo divertido y simple que es jugarlo, por lo que con este proyecto quisiéramos honrarlo tratando en todo momento de respetar todas sus reglas clásicas para no manchar la belleza y simplicidad del mismo. En esencia es pasarla bien encajando caras iguales.

domino

Todo juego de dominó esta compuesto por jugadores ,las fichas y la mesa (aunque en la vida real hasta el piso es mesa para jugar). Pues empecemos con las funcionalidades de estos 3 elementos en nuestro código.

\\Token

Tiene su enfoque en la modelación de la funcionalidad de la ficha clásica del dominó, con sus dos caras. Es capaz de saber si es igual a otra ficha, y tiene la capacidad de girarse para acomodarse al juego.

Dichas fichas estarán creadas por un generador ,el cual también inicializa las estadísticas que se utilizan para el jugador inteligente. De dicho generador se tienen dos variantes implementadas:

\\ITokenGenerator

- Double Six: Tal como su nombre lo indica ,la variante del doble 6
- Double Nine: Variante doble 9

\\Player

En la implementación de lo que viene siendo un jugador, aunque mas bien en la lógica de nuestro código los jugadores son como la cáscara ,pues son capaces de jugar pero solo si les dices con que estrategia quieres que jueguen, por lo que en términos de la vida real son solos leales soldados. De esta forma no hacemos que un jugador se apegue a una única estrategia, sino que es capaz de en todo momento jugar cualquier estrategia. Todo jugador tiene su nombre como identificador ,una puntuación, tiene su mano de fichas, y alguna que otra estadística que se utiliza en la toma de decisiones de las estrategias.

\\Table

Esta es la mesa donde ocurre el juego, donde la magia de la programación encaja a todas las piezas con sus respectivas caras, además posee las fichas del juego, y algunas estadísticas importantes que son utilizadas por el jugador inteligente en la toma de decisiones.

Pues listo, ya tenemos la mesa, las fichas y los que la juegan. La cuestión ahora es jugar, hacer que el flujo entre estos elementos se de. Al inicio de cualquier partida en la vida real se da agua al dominó y cada uno escoge las fichas que desee boca abajo, o sea dejándole al factor suerte la posiblidad de que te toque una buena data. Pues este comportamiento intentamos simular en nuestro código con la distribución de fichas, de la cual hicimos dos variantes:

\\IDistribution

- Classic: Consiste en repartir fichas para cada jugador aleatoriamente dependiendo de la variante escogida.
- Doubles To Trash: Tiene la misma funcionalidad del Classic, con la excepción de que en caso de que algún jugador le hayan tocado 5 dobles o más, le permite devolverlos ,y volverle a repartir aleatoriamente la cantidad desechada.

Ya tenemos los jugadores con sus respectivas fichas, necesitamos ahora escoger uno de ellos para romper con el juego,la responsabilidad de esto se la dejamos a:

\\IFirstPlayer

- Random First Player: Se escoge un jugador aleatorio de entre todos para iniciar.

Una vez escogido el 1er jugador a jugar, arranca la maquinaria, se le entrega al jugador una lista de jugadas válidas, `ValidTokensToPlay` de la cual este se encargará de decidir una para jugar, de esta forma hacemos que no se tolere la trampa, aunque si modelamos lo que pudiera ser permisible o no , por ejemplo las fichas que son válidas aún cuando no cumplen las reglas clásicas del dominó están modeladas por:

\\IActionModeratorToAdd

- Classic: Las reglas básicas del domino

- Double White is a Valid Token always: Consiste en la permisión de poner la ficha del doble blanco en cualquier momento del juego siempre que alguno de los jugadores la contenga en su mano y este esté en su turno.

Y tambien contemplamos la posbilidad de que algunas fichas no estén permitidas para jugar, aunque no se haya realizado una implementación concreta de esto, que está dada por:

\\IActionModeratorToSub

-Classic: Todas las fichas se permiten siempre que pasen el filtro de `ValidTokensToPlay`

Luego de pasar por estos filtros se le entregan las posibles jugadas al jugador y este escoger cual desea hacer, teniendo en cuenta también una suposición de quien es el próximo jugador para que el actual pueda jugar de la mejor forma posible, el cual esta determinado por:

\\IStep

-Classic: Es la forma clásica de orden de juego a favor de las manecillas del reloj

-Change Direction With Pass: Como su nombre lo supone cambia el sentido del juego cada vez que alguien es pasado

Una vez jugado con esa suposición de quien iría después, se establece realmente quien será el próximo a jugar. De esta manera cíclica se va realizando la partida hasta que se determine el fin de esta y los ganadores de la misma en caso de haber ,el cual se modela mediante:

\\IEndRound

-Classic: Este fin ocurre si alguien se pegó o todos los jugadores se han pasado (se trancó el juego), o alguno de los jugadores se ha pasado seguidamente más veces que la cantidad de jugadores, esta última condición fue un extra que le añadimos a la version clásica para tener una mejor experiencia del juego.

En caso de concluir la ronda se inicia la próxima ronda si no se ha logrado terminar el juego ni se han determinado los ganadores o empate, ,todo este final del juego se permite gracias a:

\\IEndGame

- Classic: En caso de acabar la ronda se acaba el juego

- Twice Passes: De haber dos pases seguidos por alguien acaba el juego

- Crazy Chicken :Es un fin del juego ,creación nuestra ,en la cual en el momento de que alguno de los jugadores alcance una puntuación final de 100 o más habrá acabado el juego y entonces todos los jugadores que cumplan con esto y sus compañeros de equipo en caso de tener, no serán considerados ganadores de la partida ,el resto sí.

Hemos mencionado puntuación final ,pero aún no hemos definido su forma de calcularla, de esto se encargará:

\\IScore

-Classic: Consiste en sumar las caras de todas las fichas de un jugador y esta suma es precisamente la puntuación que se le asigna al jugador.

-Doubles Have Double Punctuaction: Es lo mismo que hace el Classic con la excepción de que las fichas que son dobles se tomarán con el doble de puntuación.

En esencia eso es todo lo que hace nuestro juego por detrás ,y el encargado de llevar a cabo todo, de lograr el flujo, de hacer que independientemente de la variante escogida de los elementos configurables sepa que hacer en cada momento, es el propio juego, el cual bautizamos con el nombre de `Game` para ser lo mas descriptivo posible.

Este además posee un `Log` en el cual ha medida que ocurre el juego se van registrando todos lo pormenores del mismo, para así poder visualizarse.

El total de elementos configurables como ya se ha mencionado antes son los siguientes:

- ITokenGenerator (2)
- IDistribution (2)
- IFirstPlayer (1)
- IActionModeratorToAdd (2)
- IActionModeratorToSub (1)
- IStep (2)
- IEndRound (1)
- IEndGame (3)
- IScore (2)

Con respecto a las estrategias de los jugadores que como ya hemos mencionado son las que hacen que el jugador juegue, tenemos:

```
//IStrategy
```

-Intelligent: Es una estrategia que juega en base a probabilidades, antes que todo este pregunta si se ha probado en juego que el jugador al que desea pasar no lleva determinado valor de ficha que este posee, en ese caso juega la ficha de este valor pues es seguro que lo va a pasar, (dígase probado en juego aquellas fichas que el jugador se pasó por ella con anterioridad) Aclarar que esta estrategia juega con la suposición de que el jugador al que pretende pasar ya esta determinado como próximo jugador. En caso de que no ocurra lo anterior dicho de que haya seguridad de pasarlo, jugamos la jugada mas probable que no tenga, es básicamente contar la mesa, y jugar el valor de ficha que más probabilidad tiene de no llevar.

-Botagorda: Juega la ficha mas alta de su mano siempre que se sea posible

-Random: Juega aleatoriamente

-Intelligent Botagorda: Estrategia enfocada en jugar el valor de ficha que se ha probado en juego que no tiene el jugador al que se desea pasar, en caso de que no ,jugar como Botagorda

-Intelligent Random: Análogo al anterior con jugar como Random

Hasta aquí toda la lógica del juego tratando lo menos posibles de usar tecnicismos, con respecto a la interfaz gráfica solo decir que en caso de agregar una nueva variación de los elementos configurables del juego, solo es agregar la nueva instancia a un array, manteniendo las buenas prácticas.

botagorda **da agua** **se pegó** **se trancó el juego**