Avance 3 del Proyecto - Pal Tinto
Descripción detallada del proyecto - Juego de Parques modificado

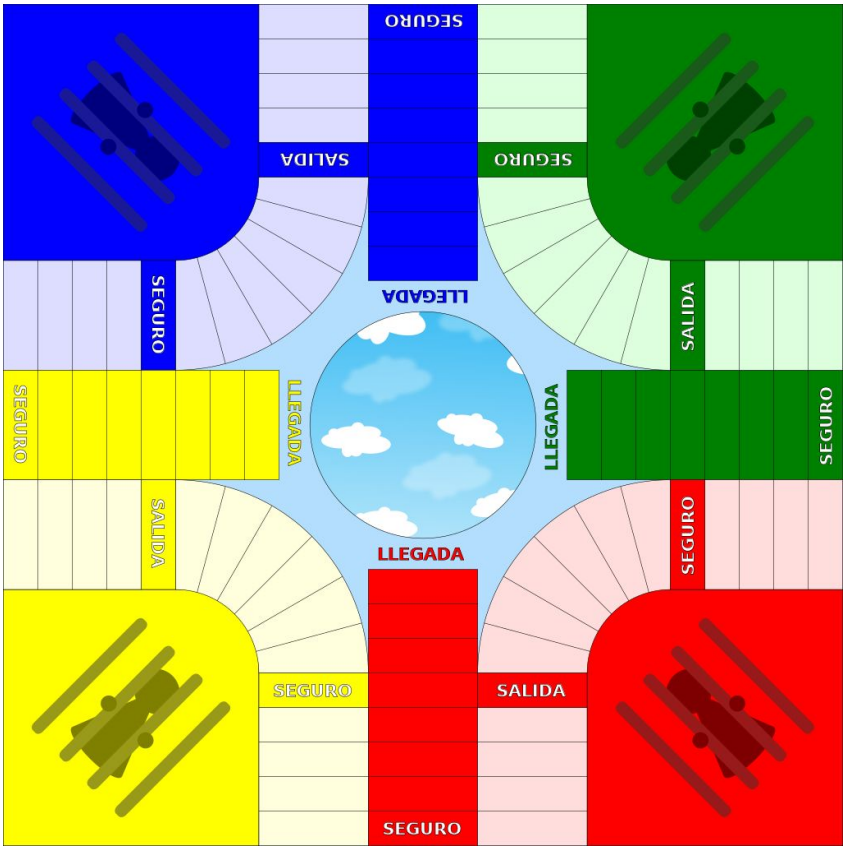
Fabian Stiven Abreo Cubides
Julian Eduardo Ariza Arevalo
Ivonne Nathalia Uribe Lara

INTRODUCCIÓN

El Parqués es un juego de mesa colombiano de complejidad media, adaptado del juego de mesa americano Parchessi que a su vez es adaptado del juego de mesa Indio Parchisi, el objetivo es llevar las fichas de cada jugador a una meta común, mediante el avance determinado por el valor de un par de dados. La escalera es un juego de mesa de complejidad baja similar al parques, pero que cuenta con atajos y trampas que permiten adelantar o atrasar un jugador. El parques modificado en este proyecto será en esencia un parques con los atajos y trampas del juego de la escalera, a su vez, cada equipo cuenta con una habilidad especial que le permitirá adicionar mayor carácter estratégico al juego.

1. Descripción detallada: A continuación se detallan las reglas del juego a implementar.

- La esencia del juego es un parques de 4 equipos.
- El tablero tiene la siguiente forma:



https://en.wikipedia.org/wiki/Parqu%C3%A9s#/media/File:Tablero_de_parqu%C3%A9s.svg

- Cada equipo tiene cuatro fichas.
- Cada equipo tiene un color diferente.
- Cada color tiene una habilidad que solo se puede usar una sola vez.
- El tablero tiene casillas aleatorias que adelantan o retroceden la ficha del jugador.
- El tablero está dividido en cuatro regiones para los cuatro equipos.
    - Cada región tiene 24 casillas y una cárcel.
    - 7/24 casillas son camino a la victoria de uso exclusivo del equipo de la región.
    - 2/24 son seguro.
    - 1/24 es salida.
    - 14/24 son casillas regulares.
    - 2/14 casillas regulares tienen efectos. Uno adelanto y uno atraso.
    - La zona de victoria es común para los cuatro equipos.
- El jugador que saque el mayor valor con un dado empieza el juego.
- El juego continúa en el sentido antihorario del jugador con primer turno.
- Cada jugador puede lanzar tres veces los dados para poner fichas en juego si no tiene ninguna.
- Si el jugador tiene al menos una ficha en juego, solo puede lanzar una vez.
- Por cada par con los dados se repite turno.
- Tres pares seguidos envían una ficha deseada por el jugador a la meta.
- Con un par de dados saca dos fichas de la cárcel a juego.
- Con un par de 6 o 1 se sacan todas las fichas de la cárcel a juego.
- Si un jugador está en una casilla y el resultado del turno de otro jugador es caer en la misma casilla, el jugador que estaba primero se va a la cárcel.
- Existen casillas con efecto seguro que no permiten enviar a la cárcel.
- Las casillas salida tienen efecto seguro, únicamente el propietario del color puede enviar a la cárcel cuando hay otro jugador en su casilla salida.
- El jugador puede usar el resultado respectivo de los dados para una o dos fichas respectivamente.
- El poder de la ficha roja es no ser afectado una única vez por casillas de retroceso.
- El poder de la ficha amarilla es aumentar una única vez en 1.5 redondeado por arriba el valor de una casilla de adelanto.
- El poder de la ficha azul es relanzar un solo dado una única vez a su gusto de nuevo antes de efectuar el movimiento.
- El poder de la ficha verde es no poder ser enviado a la cárcel por el siguiente jugador que pueda caer en su casilla.
- Los poderes sólo pueden ser usados en los respectivos turnos.
- El jugador que tenga todas las fichas en la meta gana el juego.
- El juego termina cuando hay un ganador.

DEFINICIÓN DE CLASES.

Este juego de parques modificado, consta de 8 clases que son:

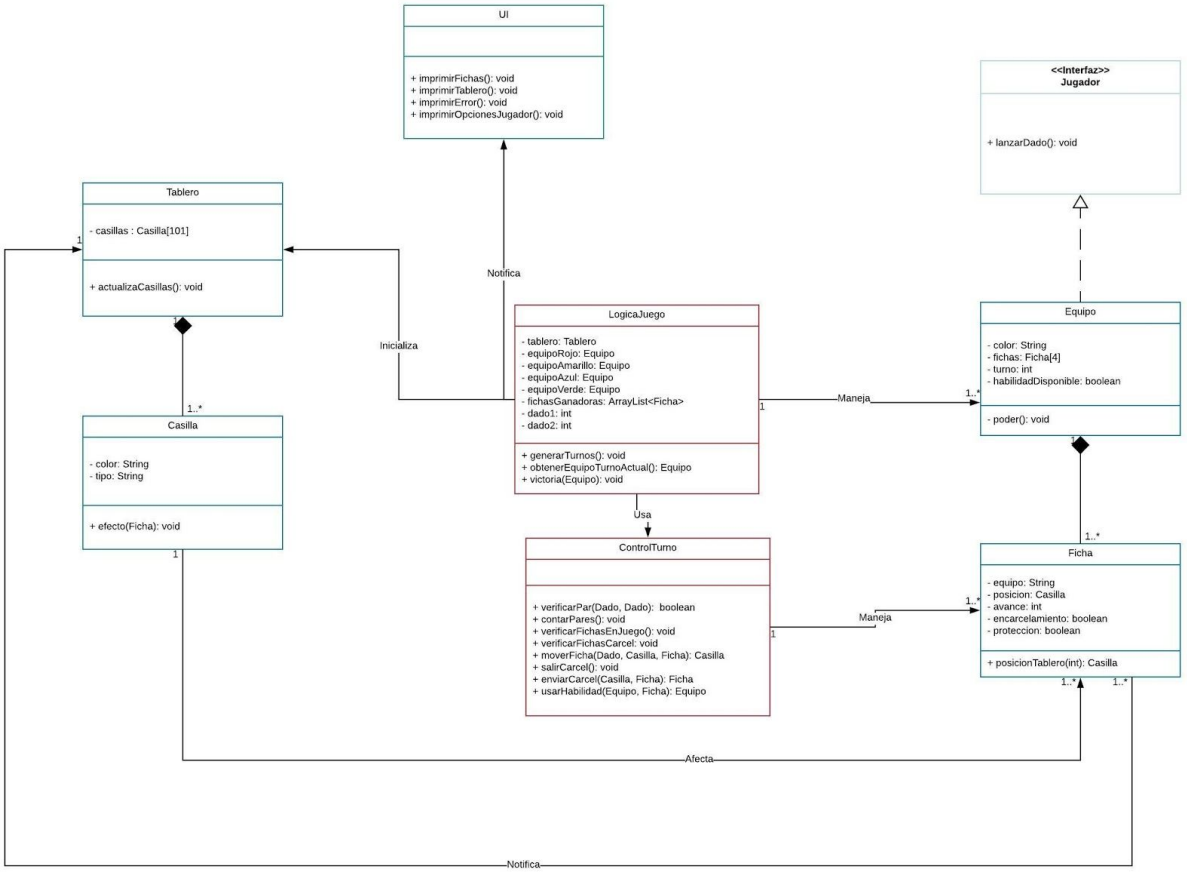
1. JUGADOR: el jugador no tendrá ningún atributo ya que es una interfaz y no necesitamos de ningún atributo de característica final pero sí tendrá un método que será lanzar los dados (+lanzarDado():void).

2. EQUIPO: principalmente el equipo se compone de fichas, además de esto consta con los siguientes atributos:un color para diferenciarse(String), una lista de cuatro instancias de la clase ficha de las cuales se compone (array),un turno(int),un poder que cambiará de acuerdo a que equipo sea(String)y por último el atributo “habilidad disponible” que permite saber si el poder fue utilizado o no (boolean).esta clase no tendrá ningún método.
3. FICHA: consta de los siguientes atributos: el atributo equipo que será el color del equipo al cual pertenece la ficha(string), una posición que será la casilla en la cual la ficha está actualmente (Casilla),tendrá un avance que se encargará de sumar o restar un número entero dependiendo los movimientos que haga la ficha es decir que tantas casillas avanza o retrocede la ficha,tendrá un atributo encarcelamiento que permitirá saber si la ficha está o no en la cárcel (boolean) y otro llamado protección que dirá si la casilla en la cual está la ficha es un seguro o no lo es(boolean).además de esto la ficha tendrá el método posicion tablero que usando la información del atributo avance , determinara cual es la casilla en la cual se encuentra la ficha,de aquí se definirá el atributo posición.(+posición Tablero(int):Casilla).
4. CASILLA: la casilla tendrá dos atributos, un color que será importante para el juego (String)y un tipo de casilla que determina si la casilla es la cárcel, un seguro, una trampa , una ventaja o una casilla común y corriente. la casilla tendrá un único método que se encargará de afectar a la ficha que se encuentra allí dependiendo que clase de casilla es, si la casilla es un seguro , no permitirá que la ficha sea comida, si es la cárcel ,la ficha no podrá jugar , si la casilla es una ventaja , la ficha avanzara y si es una trampa la ficha retrocederá , por lo tanto este método tendrá un condicional que actúe de cierta manera dependiendo el tipo de la casilla (+efecto(Ficha):void).
5. TABLERO:el tablero se compone de casillas por lo tanto su único atributo es un arreglo de casillas específicamente 101 (casilla[101]) y también tiene un método que se encargará de actualizar las casillas , es decir, según el random que se haga, cambiará las trampas y las ventajas de posición(+actualizarcasillas():void).
6. CONTROLTURNO: el control del turno será una clase que no tendrá atributos pero sí métodos fundamentales para la función del juego, es importante aclarar que los métodos de esta clase se ejecutarán con el equipo que tenga el turno actual . primero tenemos el método verificar pares que como él mismo se explica . verificará si los dados lanzados sacaron el mismo número (+verificar Par(Dado,Dado):boolean),también tiene el método que cuenta que tantos pares seguidos ha logrado con los dados , ya que es necesario para las reglas del juego (+contar Pares():void),tiene el método que verifica las fichas que siguen el el juego y en algún lugar diferente a la meta oa la cárcel (+verificar Fichas En Juego():void), verifica las fichas que están en la cárcel(+verificarFichasCarcel():void) , mueve las fichas(+moverFicha(Dado,Casilla,Ficha):Casilla) ,dado el caso saca fichas de la cárcel(+ salirCarcel(): void) o envía fichas a la cárcel (+ enviarCarcel(Casilla, Ficha):

Ficha)y es el que ejecuta la habilidad de cada equipo cuando el jugador lo decida(+ usarHabilidad(Equipo, Ficha): Equipo).

7. LÓGICA DEL JUEGO: esta capa tiene relación tanto con la interfaz como con los datos por lo tanto en esta capa se instancia un tablero , cuatro equipos y sus respectivas fichas , una lista de fichas que son las que ya llegaron a la meta y dos dados. la importancia de esta capa recae , en que aquí es donde se darán todas las instrucciones y además tendrá tres métodos, el método que genera el turno(+ generarTurnos(): void),el método que obtiene el equipo que lleva el turno(+ obtenerEquipoTurnoActual(): Equipo)(estas dos para que la clase control turno pueda trabajar y por último , un método que verifica si algún equipo ya obtuvo la victoria(+ victoria(Equipo): void).
8. UI: en esta capa de interfaz de usuario, la única clase tendrá los siguientes métodos :imprimirá el tablero(+ imprimirTablero(): void) , las fichas(+ imprimirFichas(): void) , los errores en los input que se puedan dar (+ imprimirError(): void)y mostrará las opciones que tiene el jugador(+ imprimirOpcionesJugador(): void).

2. Abstraer todas las entidades y relaciones entre ellas y diseñe el diagrama UML.



3. Generar las definiciones de clases y encapsularlas para cada una de las entidades.

Se adjunta un archivo .zip con las clases encapsuladas. Estas clases pueden cambiar durante el transcurso del proyecto.

4. Diseño e implementación de la GUI.
este proceso no se ha podido implementar en el proyecto ya que hemos tenido dificultades, recolectando información y la manera de llevar esto a cabo.

Bibliografía

- Wikipedia. (2018, 05 14). *Parqués*. Retrieved from Wikipedia: https://en.wikipecdia.org/wiki/Parqu%C3%A9s