**Identificación del problema y análisis de requerimientos**

|  |  |
| --- | --- |
| Cliente |  |
| Usuario |  |
| Requerimientos funcionales | R1. El sistema debe permitir que el jugador se mueva alrededor del tablero  R2. El sistema debe permitir que el jugador pueda poner obstáculos en el tablero  R3. El sistema debe mostrar las jugadas posibles |
| Contexto del problema | El contexto del problema se centra en el diseño y desarrollo de un juego que implemente grafos. Para esto se va a usar como referencia el juego “Quoridor”, el cual consiste en que los jugadores compiten para llegar al otro lado del tablero antes que sus oponentes. Cada jugador puede colocar barreras en el tablero para bloquear el avance de los demás. El tablero es cuadrado y tiene un diseño de cuadrícula. Se va modelar el tablero del juego como un grafo, donde cada casilla es un vértice y las conexiones entre las casillas representan los posibles movimientos permitidos. |
| Requerimientos no funcionales | R4. El juego debe ser fácil de usar y comprender para los usuarios, con una interfaz intuitiva y clara. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *R1 – Movimiento del jugador* | | | |
| Resumen | *El programa deberá permitir que el jugador pueda mover las piezas alrededor del tablero, los movimientos permitidos son verticales u horizontales. En caso de que haya un* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
|  |  | |  |
| Resultado o Postcondición | El programa moverá la ficha del jugador en la dirección seleccionada | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
| Pared | | EventTarget |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *R2 – Poner obstaculos* | | | |
| Resumen | *El programa deberá permitir que el jugador pueda poner paredes u obstáculos alrededor del tablero. Para que el jugador pueda poner una pared, se necesita que el jugador seleccione 4 celdas.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| Casilla1 | MouseEvent | |  |
| Casilla2 | MouseEvent | |  |
| Casilla3 | MouseEvent | |  |
| Casilla4 | MouseEvent | |  |
|  |  | |  |
| Resultado o Postcondición | El programa pondrá la pared en el lugar seleccionado | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | |  |  |

DFS-> Asegurarse que hay un camino hasta el otro lado del tablero

Dijkstra(encontrar el camino mas rapido) -> Calcular si el jugador hizo la mejor movida para calcular puntajes al final

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identificador y nombre | *R1 – Jugadas posibles* | | | |
| Resumen | *El programa deberá mostrar las jugadas posibles, para esto es necesario seleccionar al jugador.* | | | |
| Entradas | **Nombre entrada** | **Tipo de dato** | | **Condición valores válidos** |
| Jugador | MouseEvent | |  |
| Resultado o Postcondición | El programa mostrara las posibles jugadas | | | |
| Salidas | **Nombre salida** | | **Tipo de dato** | **Formato** |
|  | |  |  |