**Paso 1. Identificación del Problema**

**Identificación de necesidades y síntomas**

* Modo multijugador: El juego debe admitir un máximo de dos jugadores.
* Presencia de enemigos: El juego debe incluir la presencia de enemigos hostiles.
* Escenario interactivo: El juego debe estar ambientado en un escenario interactivo.
* Recogida de armas: Los jugadores deben tener la capacidad de recoger armas dispersas por el escenario.
* Selección de avatar: Los jugadores deben poder elegir su propio avatar para jugar.
* Recarga de armas: Debe haber un sistema de recarga de armas en el juego.
* Inteligencia enemiga: Los enemigos deben mostrar comportamientos de movimiento inteligentes.
* Escenario destructible: El escenario del juego debe ser susceptible de sufrir daños o destrucción.
* Indicador de vida: El juego debe informar la vida que tiene el jugador para saber cuándo se muere.

**Identificación y definición concreta sin ambigüedad del proble**ma

El problema se centra en la creación de un videojuego interactivo que permita a dos jugadores participar en una experiencia basada en un grafo con 50 aristas y 50 vértices. El objetivo principal es implementar algoritmos gráficos fundamentales para enriquecer la jugabilidad. Estos algoritmos incluyen la búsqueda en anchura (BFS) y en profundidad (DFS) para la exploración del grafo, la determinación de caminos de peso mínimo utilizando los algoritmos de Dijkstra y Floyd-Warshall, así como la construcción de un Árbol de Recubrimiento Mínimo (MST) mediante los enfoques de Prim y Kruskal.

**TABLA DE ESPECIFICACIÓN DEL PROBLEMA DE INGENIERÍA DE SOFTWARE**

|  |  |
| --- | --- |
| CLIENTE | Icesi |
| USUARIO | Jugadores |
| REQUERIMIENTOS FUNCIONALES | RF1: Selección de jugadores  RF2: Recogida de armas  RF3: Recarga de armas  RF4: Indicar de la vida  RF5: Presencia de enemigos  RF6: Inteligencia enemiga  RF7: Escenario interactivo  RF8: Escenario indestructible |
| CONTEXTO DEL PROBLEMA | Se necesita el desarrollo de un videojuego en el cual sea multijugador, el desarrollador tiene permitido crearlo como el desee. |
| REQUERIMIENTOS NO FUNCIONALES | Debe tener 50 artista y 50 vértices. Además, debe tener 3 algoritmos de grafos. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | RF1: Selección de jugadores | | |
| Resumen | Se debe informar el sistema cuanto es el numero de jugadores que van a jugar. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Jugadores | Int | Debe ingresar máximo 2 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. Ingresa al juego. 2. Debe informar cuantos jugadores van a jugar. 3. Se registra el número de jugadores. | | |
| Resultado o postcondición | . Guardar la información de numero de jugadores | | |
| Salidas | Nombre salida | Tipo de dato | Condición de selección o repetición |
| ConfirmacionJugadores | Boolean |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | RF2: Recogida de arma | | |
| Resumen | El jugador o jugadores deben seleccionar un arma en que se encuentren en el mapa | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Bala | Int | Numero de bala 10 |
| Arma | Int | Tiempo de uso 6 segundos |
| Apuntado | String | Se debe mover el mouse donde indica el usuario |
| BotonIzquiero | String |  |
|  |  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. El jugador o jugadores deben apunta 2. Con el boton | | |
| Resultado o postcondición | Eliminar los enemigos | | |
| Salidas | Nombre salida | Tipo de dato | Condición de selección o repetición |
| ConfirmacionJugadores | Boolean |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | RF3: Recarga de arma | | |
| Resumen | El jugador o jugadores deben recargar la munición del arma. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| Bala | Int | Numero de bala |
| Arma | Int | Tiempo de uso 10 segundos |
| Apuntado | String | Se debe mover el mouse donde indica el usuario |
| BotonIzquiero | String |  |
|  |  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. El jugador o jugadores deben apunta | | |
| Resultado o postcondición | Eliminar los enemigos | | |
| Salidas | Nombre salida | Tipo de dato | Condición de selección o repetición |
| ConfirmacionJugadores | Boolean |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | RF5: Indicar de vida | | |
| Resumen | En jugador o lo jugadores podrán informa de cuantos es la cantidad de vida que le queda para morirse. | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| DañoRecibido | Int | Debe inicializar en la máxima cantidad disponible |
| TiempoRecuperacion | Float |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. Calcular el daño que hacen los enemigos 2. Tiempo de recuperación de la vida 3. Vida recuperada totalmente | | |
| Resultado o postcondición | 1. Calcular la vida del jugador | | |
| Salidas | Nombre salida | Tipo de dato | Condición de selección o repetición |
| IndicarlaVida | Int |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre o identificador | RF6: Inteligencia enemiga | | |
| Resumen | Los enemigos deben buscar de una manera eficiente al jugador o los jugadores | | |
| Entradas | Nombre entrada | Tipo de dato | Condición de selección o repetición |
| DañoRecibido | Int | Debe inicializar en la máxima cantidad disponible |
| TiempoRecuperacion | Float |  |
|  |  |  |
|  |  |  |
|  |  |  |  |
| Actividades generales necesarias para obtener los resultados | 1. Calcular el daño que hacen los enemigos 2. Tiempo de recuperación de la vida 3. Vida recuperada totalmente | | |
| Resultado o postcondición | 1. Calcular la vida del jugador | | |
| Salidas | Nombre salida | Tipo de dato | Condición de selección o repetición |
| IndicarlaVida | Int |  |
|  |  |  |  |

Paso 2. Recopilación de Información

Se buscan diferentes alternativas para poder encontrar una mejor solución en nuestro problema. En ese caso se buscaron aplicaciones o sistemas parecidos al desarrollo de nuestro video juego llamado skulbum en para poder entretener las personas. Las 3 principales que encontramos

**1 Call of duty Warzone:** Video juego desarrollado para multiplataforma como en celular, computador y consolas. Este juego es un género de acción además es multijugador lo cual permite jugar varios jugadores a la vez.

**2.Gran Turismo:** videojuego desarrollado para la consola de PlayStation es un género de carreras donde puedes encontrar varias pistas del mundo y coches, los cuales puedes manejar y te van enseñando de como los pilotos profesionales manejan en estas pistas.

**3.Skulbom** Un juego desarrollado con grafos en lenguaje de java con interfaz grafica sin demasiado requisito para poder correrlo y una animación realizada en 2D.

Para poder tener un mejor desarrollo en nuestro videojuego debemos saber que tipo de estructurar se van a usar para poder implementar los grafos de manera adecuadamente.

**Recorrido de Grafos DFS**

**Caminos de Peso Mínimo (Dijkstra, Floyd-Warshall)**

El algoritmo de Floyd-Warshall lo que hace la comparación de todos los caminos que se pueden encontrar a través del grafo entre cada par de vértices cada par de vértices . El algoritmo es capaz de hacer lo con 3 vértices de comparación lo hace mejorando paulatinamente hace un camino mas entre los dos vértices, hasta obtener la estimación mas optima.

Hay un camino el cual tiene dos candidatos un camino mínimo, se utiliza los vértices del conjunto (1..k) o un camino que existe desde i hasta k+1, y de k+1 hasta j. Se sabe que el camino mas optimo es de i a j se utilizan los vértices de 1 hasta k esto se define como el caminoMinimo(i,j,k) puede haber un camino mucho mejor que es i a k 1 a j , la longitud de este camino será la concatenación del camino minmo de i a k+1.

Podemos definir CaminoMinimo(i,j,k) de manera recursiva:

caminoMinimo(i,j,k)=min(caminoMinimo(i,j,k-1), caminoMinimo(i,k,k,-1)+caminoMinimo(k,j,k-1));

caminoMinimo(i,j,0)=pesoArista(i,j);

Este proceso continua hasta que k=n, y haberemos encontado el camino mas corto para todos los pares de vértices (i,j) usando algún verticte intermedio

En este caso en nuestra implementación de nuestro progam

**Árbol de Recubrimiento Mínimo -MST- (Prim, Kruskal)**

**Paso 3. Búsqueda de Soluciones Creativas**