**Fase 1: identificación del problema:**



Una reconocida empresa dedicada al desarrollo de videojuegos, conocida por su realismo en las físicas y hechos de la vida real, ha sido contratada por la NASA S.A.S. para desarrollar un software capaz de simular con precisión los viajes espaciales. El objetivo principal del software es calcular las rutas más cortas y eficientes para los viajes interplanetarios, intergalácticos y espaciales, evitando pérdidas materiales, monetarias y humanas. Sin embargo, se debe tener en cuenta que muchos de estos viajes solo serán realizados en simulaciones, ya que no se cuenta con los recursos ni la tecnología necesarios para llevarlos a cabo en la realidad.

La empresa planea desarrollar el software como un videojuego que también servirá para recaudar fondos y continuar con las investigaciones en el campo espacial. En el videojuego, se podrán simular los viajes utilizando diferentes medios de transporte, incluyendo agujeros de gusano y agujeros negros, aunque la NASA S.A.S. considera que estos métodos son extremadamente peligrosos e imposibles en la realidad. Por lo tanto, el uso de estos "medios de viaje" será una opción adicional en el juego para agregar emoción, pero no se considerarán parte de la realidad del software.

La empresa también tiene la intención de incluir la posibilidad de regresar de un punto dado al planeta Tierra o a una estación espacial internacional en caso de que ocurran problemas o fallas con la tripulación o la nave. Además, se busca calcular la ruta más corta y eficiente para llegar a un destino sin utilizar agujeros de gusano ni agujeros negros, para mantener la autenticidad y realismo del juego según los estándares establecidos por la NASA S.A.S.

**Requerimientos:**

|  |  |
| --- | --- |
| **Cliente** | NASA S.A.S. |
| **Usuario** | NASA and players |
| **Requerimientos funcionales** | El sistema debe permitir:  R1: Camino más corto entre la tierra y un punto de interés.  R2: Verificar la posibilidad de regreso a la Tierra  R3: Mostrar el camino más eficiente entre la tierra y un punto de interés sin usar agujeros de gusano o agujeros negros. |
| **Contexto** | Una empresa de desarrollo de videojuegos, reconocida por su realismo, ha sido contratada por la NASA S.A.S. para crear un software que simule viajes espaciales de forma precisa. El objetivo es calcular rutas eficientes y seguras para los viajes interplanetarios, intergalácticos y espaciales, evitando pérdidas materiales y humanas. El software se desarrollará como un videojuego para recaudar fondos y se enfocará en la realidad, excluyendo agujeros de gusano y agujeros negros como medios de transporte. También se contempla la opción de retorno a la Tierra o a una estación espacial en caso de problemas. |
| **Requerimiento de procesos:** | ➔ R1: El programa debe ser desarrollado en un grupo de 3 desarrolladores  ➔ R2. El sistema debe ser subido al sistema de control de versiones GitHub y mantenerse privado hasta el día del lanzamiento. |
| **Requerimientos no funcionales** | ● R1: La interfaz de la aplicación debe ser simple y fácil de entender para que los usuarios se sientan cómodos con ella.  ● R2. El programa debe ser desarrollado en Java. |

|  |  |  |  |
| --- | --- | --- | --- |
| Nambre | R1. Camino más corto entre la tierra y un punto de interés. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto entre la tierra y un punto de interés elegido por el usuario y podrá usar agujeros de gusano o agujeros negros | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
| interestPoint | Vertex | Si el punto de interés no existe. |
| Actividades generales o necesarias para obtener el resultado | El sistema recibe una base de datos que contenga las aristas y vértices, una vez cargado en el sistema y después de haber recibido el punto de interés, el programa debe utilizar el algoritmo de Dijkstra para encontrar el camino más eficiente e imprimir la manera de llegar hallada. | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar al destino. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nambre | R2. Verificar la posibilidad de regreso a la Tierra. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto de un punto a la tierra o y podrá usar agujeros de gusano o agujeros negros | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
|  |  |  |
| Actividades generales o necesarias para obtener el resultado | El sistema debe calcular y demostrar si es posible y como volver a la tierra partiendo del punto donde se encuentre. | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar de vuelta a la tierra o no, dependiendo de la ubicación inicial. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Nombre | R3. Mostrar el camino más eficiente entre la tierra y un punto de interés sin usar agujeros de gusano o agujeros negros. | | |
| Resumen | El sistema debe encontrar e imprimir cual es el camino más corto entre la tierra y un punto de interés elegido por el usuario, pero esta vez no podrá usar agujeros de gusano ni agujeros negros por apelación al realismo. | | |
| Entradas | Nombre de entrada | tipo de dato | Selección de condición o repetición |
| interestPoint | Vertex | Si el punto de interés no existe. |
| Actividades generales o necesarias para obtener el resultado | El sistema deberá usa Dijkstra condicionado por el hecho de no poder usar aquellos vertices que correspondan a los agujeros negros y agujeros de gusano | | |
| Resultado o poscondición | Se obtiene el camino más eficiente para llegar al destino. | | |
| Salidas | Nombre de salida | tipo de dato | Selección de condición o repetición |
| message | String |  |

**Fase 2: Recolección de información:**

Aquí puede ir información que aclare al cliente cómo se va a solucionar el problema, puede ser Dijkstra, Bfs y teoría de grafos.