**Método de la ingeniería**

**FASE 1:**

**Contexto del problema**

La empresa "Juegos Épicos" está buscando contratar a un par de estudiantes de informática para diseñar y desarrollar su juego titulado "Space Adventure". Este juego es un juego de exploración planetaria y gestión de recursos en el que el personaje principal, Ramiel, un explorador espacial, está en busca de su gato llamado Naranjita. Ramiel debe viajar a través de muchos planetas en su nave espacial mientras administra sus recursos, como el combustible y el oxígeno, para asegurarse de encontrar a su gato. Para desarrollar el juego de la mejor manera, la empresa busca que los desarrolladores implementen estructuras de datos, como grafos, y métodos de búsqueda y búsqueda de caminos mínimos en el juego.

*Más información:* Space Adventure

**Definición del problema**

El problema se centra en el desarrollo de un juego llamado "Space Adventure", donde el objetivo principal es ayudar a Ramiel a encontrar a su gato, Naranjita, en un universo compuesto por 50 o más planetas. El objetivo es diseñar e implementar una estructura de datos basada en grafos para gestionar la conectividad entre planetas, así como algoritmos de búsqueda y búsqueda de caminos mínimos para calcular rutas eficientes para Ramiel. Además, es importante tener en cuenta las reglas del juego, la gestión de recursos y la interacción del personaje con el entorno.

A continuación

| | |
|---|---|
| **Cliente** | Juegos Épicos |
| **Usuario** | Usuario |
| **Contexto del Problema** | *Juegos Épicos desea que su juego Space Adventure se implemente con la estructura de datos grafos. Los nodos del grafo serán las estaciones y los planetas del universo, las aristas representarán los caminos entre estaciones y planetas. El juego debe permitirle al jugador viajar entre planetas, explorarlos o visitarlos, para esto se utilizaran recorridos de búsqueda en grafos. Debe permitirle administrar sus recursos y representar sus estados de salud y oxígeno. Con un recorrido de caminos mínimos, se le debe permitir al usuario utilizar el modo automático de navegación entre planetas.* |
| **Requerimientos Funcionales** | El sistema debe permitirle al usuario:<br>**RF1.** Iniciar juego.<br>**RF2.** Navegar entre planetas o estaciones.<br>**RF3.** Visitar el planeta o estación.<br>**RF4.** Moverse.<br>**RF5.** Tomar objetos.<br>**RF6.** Disparar.<br>**RF7.** Interactuar con enemigos.<br>**RF8.** Comprar un objeto.<br>**RF9.** Activar piloto automático.<br>**RF10.** Manual de instrucciones<br>**RF11.** Salir. |

| | |
|---|---|
| **Requerimientos No Funcionales** | - La interfaz de usuario debe ser intuitiva y fácil de usar.<br>- El diseño de los niveles, personajes y objetos deben tener un patrón de color y formas que sean “agradables” de ver. |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF1. Iniciar juego* | | |
| Resumen | *El sistema deberá poder iniciar un juego nuevo para el jugador cuando él lo desee. El juego nuevo debe crear 40 planetas y 10 estaciones de repostaje además, debe interconectarlos con unos caminos y asignarle a esos caminos un peso. Dentro de cada planeta, se debe generar los objetos de recolección, mapeo y enemigos. Por otro lado, al jugador se le debe poner sus atributos de vida y oxígeno y a la nave se genera con su combustible al completo. Además, el sistema deberá generar los objetos que venderán las estaciones junto a sus precios. Por último, se deberá poner en una ubicación random (un planeta) al jugador y al gato.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | | | |
| Resultado o postcondición | El sistema generó los planetas y estaciones, además, agregó conexiones entre los planetas y estaciones junto a sus pesos de viaje. Por otro lado, se generaron los objetos dentro de los planetas y los objetos dentro de las estaciones. Además, se generó al jugador junto a sus atributos y su cohete con sus atributos en un nodo random. Por último, el gato también se generó en un planeta random. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | *El sistema muestra al jugador el universo explorable.* |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF2. Navegar entre planetas o estaciones* | | |
| Resumen | *El sistema debe permitirle al jugador poder navegar entre planetas o estaciones. El jugador deberá pulsar con el mouse el nodo que desea visitar, cuando lo pulsa, se obtiene el nombre del planeta o estación y se calcula el costo de viaje y se le informa al usuario para que el decida si desea viajar o no o, se le informa si el planeta elegido está muy lejos del actual.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | planetName | String | *Ingresa el nombre del planeta, se permitirán caracteres y números.* |
| Resultado o postcondición | El sistema encuentra o no el planeta, si no lo encuentra, le informa al usuario que el planeta no está suficientemente cerca para viajar, si lo | | |

| | | | |
|---|---|---|---|
| | encuentra, transporta al personaje al mundo o estación y le resta al combustible del cohete la cantidad de combustible gastado. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | *El sistema informa si el jugador pudo viajar o no a el nuevo nodo.* |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF3. Visitar el planeta o estación* | | |
| Resumen | *Cuando el jugador se encuentre en un planeta o estación, el sistema debe permitirle al jugador visitar el planeta o estación. El jugador le da al botón visitar, entonces se toma el tipo de nodo que es(planeta o estación) y se abre una nueva ventana donde se ven los detalles del lugar, además, se le resta 5 al combustible del cohete. Si el lugar es un planeta, se le muestra al usuario un mapa con los objetos descritos anteriormente, si es una estación, se le muestra al usuario lo que puede comprar y el valor del objeto.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | placeType | PlaceType | *Solo se permiten los valores: "PLANET" o "SERVICE_STATION"* |
| Resultado o postcondición | El sistema abrió una nueva pestaña con los detalles del lugar, si fuera un planeta, abrió una nueva pestaña con un mapa generado. Si fue una estación, abrió una nueva pestaña con una interfaz de tienda, mostrando los objetos y los precios. Si el combustible no alcanzó para poder visitar el planeta, se le avisó al jugador o si es necesario, se terminó la partida. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | visiting | String | El sistema despliega el mapa o la interfaz de la tienda en otra ventana. |
| | gameStatus | String | El sistema despliega si el jugador no tenía suficiente combustible o termina el juego. |

| | |
|---|---|
| Identificador y nombre | *RF4. Moverse* |
| Resumen | *Cuando el jugador se encuentre en un planeta, el jugador podrá moverse con unas teclas key, las teclas key serán:*<br>*- Flecha hacia arriba, para moverse hacía arriba.*<br>*- Flecha hacía abajo, para moverse hacía abajo.* |

| | | | |
|---|---|---|---|
| | *- Flecha hacía la izquierda, para moverse hacía la izquierda.*<br>*- Flecha hacía la derecha, para moverse hacía la derecha.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | keyBoardButton | KeyBoard | *Solo se permiten los valores: “UP” o “DOWN” o “LEFT” o “RIGHT”* |
| Resultado o postcondición | El sistema movió al icono del jugador hacía donde el jugador quería moverse. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | moved | String | El sistema despliega en el mapa la nueva posición del jugador. |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF5. Tomar objetos* | | |
| Resumen | *Cuando el jugador se encuentre en un planeta, el jugador podrá recoger objetos cuando el icono del jugador colisione(los toque) con alguno de ellos. Cuando el jugador colisiona con alguno se deberá saber que objeto es para que interactúe según sea el caso. Los objetos serán un arma, un corazón (vida), oxígeno, gasolina,módena y gato.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | object | ObjectType | *Solo se permiten los valores: “GUN” o “HEART” o “OXYGEN” o “COIN” o “CAT” o “GASOLINE”.* |
| Resultado o postcondición | El sistema tomó el objeto e interactúa con el jugador, si recogió una moneda a sus monedas se le sumaron 1, si recogió un arma, se le suman 5 balas, si recogió oxígeno, se le sumaron 3 segundos de oxígeno, si recogió un corazón, se le sumo 1 corazón a sus corazones, si recogió gasolina se le suman 5 de gasolina o, si recogió al gato, se le mostró que ganó el juego. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | collectedObject | String | El sistema realizará los cambios necesarios en la interfaz de jugador. |
| | gameStatus | String | El sistema despliega que el jugador ganó el juego. |

| | |
|---|---|
| Identificador y nombre | *RF6. Disparar* |

| | | | |
|---|---|---|---|
| Resumen | *Cuando el jugador tiene balas, el jugador podrá disparar su arma con la tecla key, “espacio”. La bala irá en dirección de la última tecla key de movimiento del jugador de manera recta hasta que sucedan alguna de las siguientes cosas:*<br>*- Si la bala choca contra una pared, desaparece.*<br>*- Si la bala choca contra un enemigo, lo elimina y lo hace desaparecer junto a la bala.*<br>*Por último, se le restará una bala al contador del jugador.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | gunAction | Action | *Solo se permiten los valores: “SPACE”.* |
| Resultado o postcondición | El sistema género la forma de la bala y la hizo viajar por un camino recto hasta que chocó con una pared o enemigo. Además, le resto una bala a las balas del jugador. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | El sistema muestra la bala viajando por el mapa y despliega los cambios en la interfaz del jugador. |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF7. Interactuar con enemigos* | | |
| Resumen | *Cuando el jugador se va moviendo por el mapa, puede que se encuentre a un enemigo y choque con él, si colisiona con él, el jugador perderá una vida y será regresado al mapa de planetas. El enemigo se irá moviendo por el mapa por una ruta, si el jugador le dispara y acierta, el enemigo desaparecerá.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | collisionStatus | boolean | *Solo se permiten los valores verdadero o falso para colisión.* |
| Resultado o postcondición | El sistema verifica reiterativamente las colisiones del jugador con los enemigos, si en algún momento colisionaron, el sistema debe restarle una vida y sacarlo a la ventana del mapa de planetas, si ya no le quedan vidas, el sistema muestra una pantalla de juego terminado. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | El sistema muestra a el jugador y enemigos y cambia la interfaz de usuario cuando ocurre una colisión. |
| | gameStatus | String | El sistema despliega si el jugador ya no le quedan más vidas y perdió el juego. |

| Identificador y nombre | *RF8. Comprar un objeto* | | |
|---|---|---|---|
| Resumen | *Cuando el jugador se encuentra en un nodo que es de tipo estación y lo decide visitar, el sistema debe desplegar una ventana con una interfaz de tienda, donde aparecerán 3 tipos de objetos junto a sus precios. El jugador solo podrá comprar 1 corazón por 2 monedas y luego aparecerá como agotado, el jugador podrá comprar paquetes de 3 balas por 1 moneda, podrá comprar hasta 3 paquetes y por último, el jugador podrá comprar 5 de combustible por 1 moneda y no podrá comprar más de 50 de gasolina. El jugador le da click en lo que quiere comprar junto a su cantidad.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | objectAmount | int | *Solo se permiten números enteros positivos.* |
| | objectBought | ObjectType | *Solo se permiten los valores: "GUN" o "HEART" o "GASOLINE"* |
| Resultado o postcondición | El sistema hizo los cálculos de cantidad y tipo de objeto y se lo resto a las monedas del jugador, si las monedas no le alcanzan, el sistema debe avisar al jugador, si no, cambia el interfaz del usuario según sea la compra, sumandole la gasolina al cohete, sumándole una vida o sumándole la cantidad de balas compradas. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | El sistema muestra la nueva interfaz del jugador con los cambios realizados según la compra. |
| | noCoins | String | El sistema despliega un aviso que le avise al usuario que no tiene suficientes monedas |

| Identificador y nombre | *RF9. Activar piloto automático* |
|---|---|
| Resumen | *Cuando el jugador se encuentra en cualquier nodo, podrá activar el modo piloto automático con un botón , cuando lo activa el sistema deberá preguntarle cuánta gasolina desea que su cohete gaste en el viaje. Si la cantidad puesta por el jugador es mayor a la que tiene el cohete, se le deberá decir al jugador que no tiene gasolina suficiente, si la cantidad puesta por el jugador deja al jugador con menos de 10 de combustible, se le debe avisar al jugador que estos son sus últimos* |

| | | | |
|---|---|---|---|
| | *recursos. Si el jugador decide viajar, se le restará la gasolina según sea el viaje alcanzado por el cohete.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | planetName | String | *Ingresa el nombre del planeta, se permitirán caracteres y números.* |
| | gasolineAmount | int | *Solo se permiten números enteros positivos .* |
| Resultado o postcondición | El sistema le informó al usuario si se pudo realizar o no el viaje automático, si lo hizo, se mostró el viaje del cohete por la interfaz de jugador si no, se le mostró un aviso de que no se logró viajar. Si el viaje se realizó pero sobró combustible se lo informa al usuario y se suma en la cantidad de combustible del cohete. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | El sistema muestra la nueva interfaz del jugador con los cambios realizados en el movimiento del cohete. |
| | tripNoCompleted | String | El sistema despliega un aviso que avisa al usuario que no se pudo realizar el viaje. |

| | | | |
|---|---|---|---|
| Identificador y nombre | *RF10. Manual de instrucciones* | | |
| Resumen | *Cuando el jugador se encuentra en el menú principal del juego, el podrá elegir la opción de instrucciones del juego, se le mostrará una ventana con todos los detalles del juego. La ventana tendrá un scroll para ir leyendo el texto y tendrá un botón para ir al menú principal.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | | | |
| Resultado o postcondición | El sistema mostró la ventana con las instrucciones del juego y le mostró al jugador un botón para ir al menú principal. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | alert | String | El sistema muestra una ventana al jugador con instrucciones del juego. |

| Identificador y nombre | *RF11. Salir* | | |
|---|---|---|---|
| Resumen | *Cuando el jugador se encuentra en el menú principal del juego, el jugador podría elegir la opción salir que lo saque del juego y termine la ejecución del juego.* | | |
| Entradas | **Nombre de entrada** | **Tipo de dato** | **Condición de valor válido** |
| | | | |
| Resultado o postcondición | El sistema sacó al jugador de la ventana de menú principal y terminó la ejecución. | | |
| Salidas | **Nombre de la salida** | **Tipo de dato** | **Formato** |
| | | | |

**Fase 2:**
Con el objetivo de obtener una comprensión completa de los conceptos involucrados, se realiza una búsqueda de definiciones de términos tanto teóricos como prácticos.

- Teoría de grafos
- Grafo no dirigido
- Vértice
- Arista
- Representación de grafos
    - Lista de adyacencia
    - Matriz de adyacencia
- Recorridos sobre grafo
    - BFS
    - DFS
- Caminos de peso mínimo
    - Algoritmo de Dijkstra
    - Algoritmo de Floyd Warshall

**Fase 3:**

**Fase 4:**

**Fase 5:**