**Jaime Andrés Fresneda Vásquez**

**A00359338**

**Análisis y diseño del aplicativo**

**Contexto:**

Hacer un prototipo del juego” Crossy Road”, el cual da instrucciones sobre la posición y movimiento de algunos de sus coches además de la del protagonista. Por otra parte, emplee hilos en los movimientos de sus elementos y cuenta con tres pantallas: una de juego, una de “perdiste”, “ganaste” y finalmente una en la cual muestre la duración de las partidas que se ganaron más la fecha en el momento, a la vez que permite ordenar los resultados con base en estos datos.

**Requerimientos Funcionales:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RF | descripción | entradas | Salidas | precondición | Postcondición |
| RF1 | El programa debe generar un grupo de carros | Archivo txt con instrucciones | Arreglo de enemigos | Tener el archivo txt en los archivos |  |
| RF2 | El programa debe mover al grupo de carros de una forma determinada en archivo txt | Las instrucciones del archivo txt | Los enemigos se mueven |  | Las posiciones de cada carro cambian |
| RF3 | El programa debe generar un personaje controlable por el usuario | Inicio del juego | Un personaje en la zona superior del mapa | Posición definida del personaje en archivo txt | Posición del personaje en el mapa |
| RF4 | el usuario podrá mover al personaje de izquierda a derecha usando el teclado | Las teclas del que son flechas del teclado | El personaje se mueve de izquierda a derecha | Que la posición siguiente a donde se moverá sea dentro del terreno del mapa | La posición del protagonista cambia |
| RF5 | Cuando el personaje cruce todo el mapa verticalmente gana la partida y pasa a la pantalla de ganaste | Posición en y del personaje mayor igual a lo largo del lienzo | Mensaje de “ganaste” en la pantalla |  | Se da la opción de reiniciar la partida |
| RF6 | Si los carros tocan al protagonista debe pasar a una pantalla de perdiste | La distancia entre el protagonista y los carros es lo suficiente mente corta para el contacto | Mensaje de “perdiste” en la pantalla | Debe existir el protagonista y los enemigos | Se da la opción de reiniciar la partida |
| RF7 | En caso de ganar o perder se dará la opción de volver a jugar | Se envían los mensajes de ganar o perder | Se vuelve a empezar la partida | Haber ganado o perdido | Se generar un nuevo arreglo de enemigos y el contador de enemigos destruidos baja a cero |
| RF8 | Las entidades del programa se ejecutarán empleando hilos |  |  |  |  |
| RF9 | El programa cuenta el tiempo que se demora el usuario en completar(ganar) una partida | El tiempo en partida | cronometro | Haber empezado el programa | Se hace un registro del tiempo que se demoro |
| RF10 | El programa graba la fecha y hora al momento que el usuario gana la partida | La fecha y hora | Registro de fecha y hora | Ganar la partida | Se registra cono información de la partida |
| RF11 | El programa ordena naturalmente las partidas realizadas con base en la fecha y hora al momento de terminar las partidas | Registro de fecha y hora; duración de partida además de seleccionar la opción “fecha” | Lista ordenada con base en las fechas que se tiene en el registro | Haber ganado amenos 1 vez | Cambio el ordenamiento del arreglo de registros |
| RF12 | El programa ordena parcialmente las partidas con base en el tiempo que le tomo al usuario terminar la partida | Registro de fecha y hora; duración de partida además de seleccionar la opción “duración” | Lista ordenada con base la duración de las partidas que se tengan en el registro | Haber ganado amenos 1 vez | Cambio el ordenamiento del arreglo de registros |

**Requerimientos no funcionales:**

**RNF1**: diferencia visual entre los carros y el protagonista

**RNF2**: el movimiento de los enemigos debe ser claro