**Jaime Andrés Fresneda Vásquez**

**A00359338**

**Análisis y diseño del aplicativo**

**Contexto:**

Hacer un prototipo de “Space invaders” el cual emplee hilos en los movimientos de sus elementos y mediante la consola lleve la contabilidad de los enemigos derrotados. Además, mediante excepciones debe enviar los mensajes de ganaste o perdiste

**Requerimientos Funcionales:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| RF | descripción | entradas | Salidas | precondición | Postcondición |
| RF1 | El programa debe generar un grupo de enemigos | Inicio del juego | Arreglo de enemigos | Definir la cantidad de enemigos y su distribución |  |
| RF2 | El programa debe mover al grupo de enemigos en una secuencia determinada en el mapa | El mapa tiene definido un camino para el movimiento de los enemigos en este | Los enemigos se mueven | El patrón del piso debe estar definido y los enemigos deben poder leerlo | Las posiciones de cada enemigo cambian |
| RF3 | El programa debe generar un personaje controlable por el usuario | Inicio del juego | Un personaje en la zona inferior del mapa | Definir la posición de origen de este personaje | 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 | El usuario puede disparar proyectiles desde la posición de su personaje hacia arriba | Flecha del teclado que apunta hacia arriba | Un proyectil(esfera)  Sale desde la posición del protagonista y sube en el mapa | La posición actual del protagonista | El proyectil sube en el mapa |
| RF6 | El usuario podrá eliminar enemigos usando los proyectiles que dispara | La distancia entre los proyectiles y los enemigos es lo suficiente mente poca para el contacto | Desaparece del lienzo el enemigo que hizo contacto | Debe existir la bala y el enemigo | El enemigo que hizo contacto desaparece de su arreglo |
| RF7 | Los proyectiles que toquen a los enemigos también deben desaparecer | La distancia entre los proyectiles y los enemigos es lo suficiente mente corta para el contacto | Desaparece del lienzo el proyectil que hizo contacto | Debe existir la bala y el enemigo | El proyectil que hizo contacto desaparece de su arreglo |
| RF8 | Al eliminar un enemigo se debe enviar un mensaje a la consola | Un enemigo hizo contacto con una bala | Mensaje txt en la consola | El enemigo bebe haber desparecido | Se contabiliza la cantidad de enemigos eliminados |
| RF9 | Cuando se eliminen todos los enemigos debe aparecer un mensaje de victoria, empleando una excepción | Tamaño del arreglo de enemigos=0 | Mensaje de “ganaste” en la pantalla | Debe haber un arreglo de enemigos | Se da la opción de reiniciar la partida |
| RF10 | Si los enemigos llegan a la posición del protagonista debe aparecer un mensaje de que se perdió la partida, empleando una excepción | La distancia entre el protagonista y los enemigos 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 |
| RF11 | 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 |
| RF12 | Las entidades del programa se ejecutarán empleando hilos |  |  |  |  |

**Requerimientos no funcionales:**

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

**RNF2**: el movimiento de los enemigos debe ser como en el juego original