ANÁLISIS TALLER INDIVIDUAL

JUEGO:
ANOMALY LEAK

AUTOR:
JUAN FELIPE BLANCO TIGREROS

DOCENTE:
FELIPE GARCÍA

UNIVERSIDAD ICESI
DISEÑANDO CON ALGORITMOS
9 DE MARZO DE 2021-1

ANÁLISIS:

Contexto:

Se debe crear un prototipo de un nuevo videojuego tipo Space Invaders mezclado con asteroids donde los enemigos sean infinitos y la dificultad vaya aumentando con el tiempo transcurrido. Adicionalmente, el juego debe mostrar el tiempo transcurrido y los puntos obtenidos. El juego debe tener al menos 4 pantallas. Este juego cuenta con algunas especificaciones extra como la implementación de habilidades especiales y una amplia variedad de enemigos con características distintas. El jugador contará con un número de vidas que perderá a medida que los enemigos crucen la frontera tras de él o al impactar con él. Si las vidas llegan a 0 es un game over. El jugador también cuenta con una serie de habilidades adicionales a su disparo básico que le permitirán formar estrategias más complejas.
Para una inmersión del usuario más satisfactoria se buscará que el juego sea lo más responsivo posible, es decir que haya una retroalimentación adecuada para algunas acciones del jugador, como la implementación de sonidos y animaciones. Igualmente el jugador debe poder navegar entre la pantalla de inicio y las instrucciones y debe poder reiniciar el juego desde la pantalla de resumen (En las que se desplegará su tiempo y puntos obtenidos en la partida).

ENTIDADES:

- Main
- Pantalla de inicio
- Pantalla de instrucciones
- Pantalla de juego
- Pantalla de resumen
- Nave (jugador)
- Enemigo (abstracta)
- Enemigo básico
- Enemigo fuerte
- Enemigo rápido
- Enemigo deslizante
- Enemigo tanque
- Enemigo dps
- Bala

REQUERIMIENTOS FUNCIONALES:

RF1: Cargar pantalla de inicio

Descripción: El programa debe generar una pantalla de inicio con un ancho y alto determinado. Adicionalmente debe cargar una imagen de fondo de las mismas dimensiones. Finalmente, esta

pantalla debe contar con dos botones con una variante cuando el mouse está sobre ellos y otra cuando no.

Entradas: Ancho, alto (enteros).

Salidas: No retorna

Precondición: La imagen (archivo.png) debe existir y estar referenciada.

Postcondición: Se debe mostrar la pantalla con sus respectivos elementos.

RF2: Detectar mouse

Descripción: El programa debe detectar el click del mouse y determinar si está dentro de una pantalla y área determinada para cambiar de pantalla.

Entradas: coordenada X y Y del mouse (enteros).

Salidas: No retorna

Precondición: -

Postcondición: Cambiar de pantalla (variable entera).

RF3: Cargar pantalla de instrucciones

Descripción: El programa debe generar una pantalla de juego con un ancho y alto determinado. Adicionalmente debe cargar unas imágenes de fondo que encajen dentro de dichas dimensiones. Adicionalmente se deben cargar todas las imágenes que serán usadas en esta pantalla.

Entradas: Ancho y alto.

Salidas: No retorna

Precondición: - Que las imágenes existan y estén referenciadas.

Postcondición: Cambiar de pantalla (variable entera).

RF4: Crear nave

Descripción: La Pantalla de juego debe crear un objeto jugador de tipo SpaceShip (o nave), atribuyéndole sus respectivos atributos, entre los que se encuentran las imágenes que este objeto usará.

Entradas: imágenes.

Salidas: No retorna

Precondición: - Que las imágenes existan y estén referenciadas.

Postcondición: Se creará un objeto de tipo SpaceShip, perteneciente a la Pantalla de juego.

RF4: Crear un arreglo de enemigos

Descripción: La Pantalla de juego debe crear una lista de enemigos.

Entradas: -

Salidas: No retorna

Precondición: -

Postcondición: Se iniciará una nueva lista vacía de enemigos.

RF5: Agregar enemigos a la lista de enemigos infinitamente

Descripción: La Pantalla de juego debe crear enemigos de forma aleatoria (de los 6 posibles)

cada cierto tiempo.

Entradas: temporizador.

Salidas: Nuevo enemigo.

Precondición: El arreglo de enemigos debe estar instanciado y la variable de temporizador debe estar en 0

Postcondición: Aumenta la lista de enemigos.

RF6: Pintar y mover enemigos

Descripción: La Pantalla de juego debe poder acceder a los métodos pintar y mover de cada enemigo y por medio de un ciclo pintar todos los que en ese momento sean “visibles”.

Entradas: -

Salidas: -

Precondición: El enemigo existe y su visibilidad sea verdadera

Postcondición: Aumenta la lista de enemigos.

RF7: Manejar impactos

Descripción: La Pantalla de juego debe reconocer cuando un enemigo cruza la frontera o impacta con el jugador para restarle las vidas equivalentes al daño del tipo de enemigo que impactó, igualmente, debe ajustar la visibilidad de este enemigo a falsa. Por otro lado debe revisar las interacciones entre las balas del jugador y los enemigos para quitarles la vida correspondiente.

Entradas: posiciones en X y Y del jugador, enemigo y balas.

Salidas: -

Precondición: El enemigo existe y su visibilidad es verdadera, el jugador es vulnerable y las balas del jugador (o enemigo) existen y son visibles.

Postcondición: Disminuir la vida del jugador o enemigo dependiendo del caso y ajustar la visibilidad del enemigo a false, si las condiciones se cumplen.

RF8: Mover la nave

Descripción: El programa debe reconocer cuando se oprimen las teclas a"w", "d" o "a" y debe alterar las variables necesarias de dirección del personaje para generar que este se mueva verticalmente hacia arriba o abajo.

Entradas: variable de dirección (booleano).

Salidas: -

Precondición: El jugador está dentro de los límites del lienzo y en el caso del dash su enfriamiento (CD) es 0.

Postcondición: Se altera el valor de la posición en Y del jugador.

RF9: Atacar y defender

Descripción: El programa debe reconocer cuando se oprimen las teclas a de espacio, "i", "o" u "p" y debe crear (y mover) una bala normal o pesada; volver la vulnerabilidad del jugador a falso (escudo) o detectar a todos los enemigos detrás de cierta coordenada en X para quitarles dos puntos de vida.

Entradas: variables booleanas para cada habilidad.

Salidas: -

Precondición: Los CD de cada habilidad son iguales a 0 (independientes el uno del otro).

Postcondición: Se efectúa alguno de los cambios mencionados en la descripción.

RF10: Crear arreglo de balas (o lista)

Descripción: La nave debe generar una nueva lista de balas vacías.

Entradas: -

Salidas: -

Precondición: La nave debe existir.

Postcondición: ""descripción""

RF11: Agregar balas y pintarlas

Descripción: La nave debe generar una nueva bala, pintarla y moverla cada vez que se presiona la tecla “espacio” u “o” si el CD de disparo es igual a 0.

Entradas: posX y posY de la nave, imágenes de las balas.

Salidas: -

Precondición: La nave debe existir al igual que el arreglo de balas.

Postcondición: agregar una nueva bala al arsenal de la nave, pintarla y moverla.

RF12: Eliminar balas

Descripción: La nave debe eliminar las balas cuya visibilidad sea falsa (aquellas que impactaron con un enemigo o salieron del lienzo).

Entradas: booleano de visibilidad.

Salidas: -

Precondición: “”descripción””

Postcondición: El arreglo de balas reduce su tamaño.

RF13: Eliminar enemigos

Descripción: La pantalla de juego debe eliminar las balas cuya visibilidad sea falsa (aquellas que impactaron con un enemigo o salieron del lienzo).

Entradas: booleano de visibilidad.

Salidas: -

Precondición: “”descripción””

Postcondición: El arreglo de balas reduce su tamaño.

RF14: Mostrar tiempo en pantalla de juego

Descripción: La pantalla de juego debe mostrar el tiempo transcurrido desde que se cargó esa pantalla.

Entradas: tiempo transcurrido, segundos, minutos.

Salidas: -

Precondición: Estar en la pantalla de juego

Postcondición: Mostrar el tiempo transcurrido en la pantalla.

RF15: Sumar puntos y mostrarlos

Descripción: La pantalla de juego debe reconocer cuando se asesina a un enemigo para tomar los puntos de dicho enemigo y sumarlos al contador total.

Entradas: Puntos enemigos

Salidas: -

Precondición: Impactar al enemigo y al hacerlo, que su vida sea igual o menor a 0.

Postcondición: Sumar los puntos del enemigos al contador total y mostrar dicho total..

RF16: Cargar pantalla de resumen

Descripción: El programa debe reconocer cuando las vidas del jugador sean iguales o inferiores a 0 para cambiar la pantalla a la de resumen. Al cargarla se cargan sus debidas imágenes, junto con el tiempo que se sobrevivió y los puntos obtenidos en la partida. Adicionalmente cuenta con un botón que permite reiniciar el juego.

Entradas: segundos, minutos, puntos.

Salidas: -

Precondición: Tener 0 vidas o menos.

Postcondición: Cargar la pantalla de resumen.

RF17: Reiniciar el juego

Descripción: El programa debe permitir iniciar otra partida, reseteando los CD, puntos y el tiempo.

Entradas: -

Salidas: -

Precondición: Oprimir el botón de reinicio.

Postcondición: Cargar la pantalla de juego.

REQUERIMIENTOS NO FUNCIONALES:

RNF1: Animaciones

Deben haber animaciones que potencien la experiencia del juego al permitir una mayor inmersión por parte del usuario.

RNF2: Sonidos

El juego debe cargar sonidos al detectar ciertas interacciones con el usuario.

RNF3: Botones responsive

Los botones deben cambiar cuando el usuario mueva su mouse sobre ellos para indicar que es posible una interacción.

RNF4: Feedback de vidas y CD

Debe haber una manera de comunicar al usuario cuantas vidas tiene y si los CD de sus habilidades son iguales a 0 para permitirle tener mayor sensación de control al momento de jugar.