TP PROGRAMACIÓN 1 “El camino de gondolf”

Enzo Manuel Portorreal
47308499
enzo.portorreal2006@gmail.com
Leonardo Marcelo Cardozo
45889499
leonardocardozo699@gmail.com
Eugenio Ezequiel Segovia Mancuello
45929937
uni.segovia.ezequiel@gmail.com

Introducción

El TP consiste en un juego llamado “el camino de gondolf” donde un mago tiene que pelear contra murciélagos, esquivar rocas y cuenta con diferentes poderes, logra la victoria cuando acaba con todos los murciélagos.

Descripción

CLASE JUEGO:
En esta clase se inicializa el entorno, el tick y las variables y métodos necesarios para iniciar el juego.

En esta clase contamos con las variables:

- entorno
- gondolf
- murciélago[]
- int total murciélagos
- int murcielagosActivos
- int maxMurcielagosEnPantalla
- boolean murcielagoEliminado

- Image fondo, botonera, menu, gameOver, botoneraFuego
- Botonera explosion1
- Botonera explosion2
- Botonera explosion3
- int tipoExplosionSeleccionada
- int tipoExplosionActiva
- double explosionX
- double explosionY
- int duracionExplosion
- int duracionMaximaExplosion
- Obstaculos[] rocas
- boolean juegoIniciado
- int RADIO_FUEGO
- int RADIO_AGUA
- int RADIO_VENENO
- Image victoria
- boolean juegoGanado
- Image botoneraSeleccionFuego
- Image botoneraSeleccionAgua
- Image botoneraSeleccionVeneno
- image botoneraActual
- boolean mostraImagenMana
- int contadorMostrarImagen
- int DURACION_IMAGEN_MANA
- Image manaInsuficiente

Métodos de la clase:

- reiniciarJuego()

CLASE GONDOLF:
En esta clase se definen las variables y métodos del mago.

En esta clase contamos con las variables:

- double x
- double y
- int ancho
- int alto
- double velocidad
- Image imagenAbajo
- Image imagenArriba
- Image imagenDerecha
- Image imagenIzquierda
- Image imagenActual
- int ANCHO_PANTALLA = 1400
- int ALTO_PANTALLA = 1000
- int ANCHO_MENU
- int vida
- int mana

Metodos de la clase:

- moverArriba(Obstaculos[] rocas)
- moverAbajo(Obstaculos[] rocas)
- moverIzquierda(Obstaculos[] rocas)
- moverDerecha(Obstaculos[] rocas)
- colisionaConRoca(Obstaculos[] rocas, double nuevaX, double nuevaY)
- dibujar(Entorno entorno)
- colisionaEnemigo(Murcielago[] murcielagos)
- restarVida(int cantidad)
- sumarMana(int cantidad)
- restarMana(int cantidad)
- getMana()
- setMana(int mana)
- getVida()
- getX()
- setX(double x)
- getY()
- setY(double y)
- getAncho()
- setAncho(int ancho)
- getAlto()
- setAlto(int alto)

CLASE MURCIÉLAGO:
En esta clase se determinan las variables y métodos de los murciélagos.

En esta clase contamos con las variables:

- double x
- double y
- double ancho
- double alto
- String salida
- boolean activo
- Image imagenMurcielago

Métodos de la clase murcielago:

- Murcielago(double x, double y, int ancho, int alto, String salida)
- activar()
- estaActivo()
- desactivar()
- dibujar(Entorno entorno)
- moverHacia(double objetivoX, double objetivoY)
- getX()
- setX(double x)
- getY()
- setY(double y)
- getAncho()
- setAncho(double ancho)
- getAlto()

- setAlto(double alto)

CLASE OBSTÁCULOS:
En esta clase se encuentran las variables y métodos de las rocas.
<u>En esta clase contamos con las variables:</u>

- double x
- double y
- double ancho
- double alto
- Image roca
- double bordeInferior
- double bordeSuperior
- double bordeIzq
- double bordeDer

<u>Métodos de la clase:</u>

- Obstaculos(double x, double y)
- dibujarRoca(Entorno entorno)
- double getX()
- setX(double x)
- getY()
- setY(double y)
- getAncho()
- setAncho(double ancho)

- getAlto()
- setAlto(double alto)

CLASE BOTONERA:
En esta clase se definen las variables y métodos de la botonera.
<u>En esta clase contamos con las variables:</u>

- double x
- double y
- double ancho
- double alto
- Image explosion1
- Image explosion2
- Image explosion3
- double radio

<u>Métodos de la clase:</u>

- Botonera(double x, double y)
- dibujarExplosion1(Entorno entorno)
- dibujarExplosion2(Entorno entorno)
- dibujarExplosion3(Entorno entorno)
- getX()
- setX(double x)
- getY()
- setY(double y)
- getAncho()

- setAncho(double ancho)
- getAlto()
- setAlto(double alto)
- getRadio()

Implementación

Juego:
Al comenzar, se crea un objeto entorno, con un tamaño de 1400 píxeles de ancho por 1000 de alto. A continuación, se instancia el personaje principal del juego, llamado Gondolf, y se lo ubica en el centro de la pantalla con las coordenadas (550, 500).

Luego, se cargan varias imágenes del juego. Se carga una imagen de fondo y se escala a un tamaño de 1100x1000 píxeles, y se carga también la imagen de la botonera lateral, escalada a 300x1000. Adicionalmente, se cargan imágenes para distintas pantallas del juego: el menú principal, la pantalla de "Game Over" y la pantalla de victoria, todas ajustadas al tamaño total del entorno (1400x1000 píxeles).

Además, se cargan versiones alternativas de la botonera que muestran el estado actual de selección del hechizo (fuego, agua o veneno), cada una con sus respectivas imágenes. También se incluye una imagen que representa una advertencia por maná insuficiente, escalada a 150x250 píxeles, que se usará para indicar que el jugador no puede lanzar un hechizo por falta de recursos.

A continuación, se inicializa un arreglo de enemigos del tipo Murcielago, con un total determinado por la variable totalMurcielagos. Se generan en grupos de cuatro por iteración: uno entrando por la izquierda, otro por la derecha, uno por arriba y otro por abajo. Las posiciones iniciales son aleatorias en el eje correspondiente, y están ubicadas fuera de los límites visibles del entorno.

También se crean tres objetos Botonera adicionales, llamados explosion1, explosion2 y explosion3.

Luego, se genera un conjunto de obstáculos (rocas), representados por un arreglo de 15 objetos Obstaculos. Cada roca se coloca en una

posición aleatoria dentro del área jugable, respetando una distancia mínima entre ellas de 100 píxeles. Además, se impone una restricción para que no aparezcan cerca del personaje principal (a menos de 100 píxeles de distancia).

Finalmente, se inicia el juego con la llamada al método this.entorno.iniciar(), el sistema comienza a dibujar el entorno, detectar entradas del usuario y ejecutar las lógicas de juego correspondientes (movimientos, colisiones, hechizos, etc.).

tick:

Si el juego aún no ha comenzado (juegoIniciado == false), se dibuja una imagen de menú. Luego, si el jugador hace clic sobre el área que representa el botón de "Jugar", el juego se inicia estableciendo juegoIniciado = true. Si no se hace clic, se sale del método, deteniendo toda la lógica de juego.

Luego, si el personaje principal (gondolf) es null (es decir, gondolf == null), se muestra la pantalla de "Game Over". Si el jugador hace clic sobre el

botón de reinicio, se llama al método reiniciarJuego() para volver a comenzar desde cero.

Además, si se han eliminado todos los murciélagos (totalMurcielagos y murcielagosActivos en cero), el juego se da por ganado (juegoGanado = true). Se muestra una imagen de victoria, y si el jugador hace clic en el botón correspondiente, también se reinicia el juego.

También se dibuja el fondo del escenario y todas las rocas que existen. Las rocas sirven de obstáculo para el personaje y se renderizan en pantalla.

Por otro lado, se verifican las teclas presionadas (WASD o flechas) y se mueve el personaje gondolf en la dirección correspondiente, teniendo en cuenta las colisiones con rocas. También se dibuja al personaje y se controla si colisiona con enemigos (murciélagos), en cuyo caso se reduce su vida y se eliminan enemigos.

Luego, si la vida del personaje llega a cero, se lo elimina (gondolf = null), lo que lleva al estado de "Game Over" en el siguiente ciclo.

Posteriormente se detectan clics del mouse sobre ciertos botones que seleccionan el tipo de explosión (fuego, agua o veneno). Si se hace clic en otro lugar y se tiene maná suficiente, se lanza una explosión en la posición del mouse y se consume el maná correspondiente. Si no hay maná suficiente, se muestra una imagen avisando.

Luego, si hay una explosión activa, se dibuja visualmente y se reduce su duración. Durante este tiempo, cualquier murciélago dentro del radio de la explosión es eliminado y gondolf gana algo de maná.

Además se activa un número limitado de murciélagos en pantalla. A medida que el total restante disminuye, el límite máximo de enemigos en pantalla aumenta progresivamente. Los murciélagos activos se mueven hacia gondolf y se dibujan.

Y por último, según el tipo de explosión seleccionada, se muestra una botonera específica. También se muestran indicadores visuales del juego: la vida del jugador, el maná restante y la cantidad de enemigos totales restantes. Si el jugador intentó lanzar una explosión sin suficiente maná, se muestra una advertencia.

Conclusiones

En conclusión es un juego que nos plantea diversos desafíos, algunos más complejos que otros y ponen a prueba nuestra habilidad para resolverlos así como también nuestra lógica de programación. Algunos problemas que hemos tenido han sido, por ejemplo, elegir una resolución correcta para todos los integrantes del grupo, implementar una nueva función y que aparezca un error en otras funciones, pero los hemos resuelto mirándolo desde otros ángulos e intercambiando ideas entre nosotros.