Grupo N.º 6

Desarrolladores: Muiña Sebastián

Steib Cristian

Colaboradores: Gaspar Dukart (Diseñador)

Opción de Juego: 2 “Mi Robot”

Reglas del Juego:

Debe llegar al fin para pasar de nivel.

No se debe caer del mapa.

Se estipula una cantidad mínima de movimientos por nivel. En base a la cantidad hecha se suma o resta puntaje.

Si choca los obstáculos, se resta vida y puntaje.

Si se queda sin vida, se reinicia el nivel.

Si no completa el nivel en el tiempo preestablecido se va restando puntaje cada x tiempo. x se puede configurar desde las opciones del juego

Detalles de la implementación:

Al ejecutar la aplicación aparece una lista con los nombres de los usuarios guardados (los que al menos pasaron el primer nivel), esta lista inicialmente iba a ser la lista de selección del framework de pilas, pero al mostrarla se veía muy chico por la resolución que usamos (2571 x 1306) y no se podía escalar. Solución: cada nombre es un actor Texto, el cual podíamos elegir tipografía, tamaño y color; creamos un actor Mono (no visible), el cual se posiciona donde se clickea, entonces produce una colisión entre el Mono y el Texto y se cambia de escena con los datos del jugador seleccionado.

Solo se permite el ingreso de letras para la creación de un nuevo jugador y no se permite que el nombre este en blanco ni que sea uno existente. Esto se hizo para mostrar explícitamente el nombre, y que no quede en blanco al mostrar los datos.

La mayoría de las animaciones están hechas con interpolaciones, hubo que tener en cuenta la sincronización. Otras animaciones son grillas, como la del cursor o el televisor; el engranaje se maneja con un evento que va reduciendo la velocidad. Se usan tareas con determinado tiempo de repetición para ir controlando la sincronización de todas las tareas en simultaneo, (tarea = (tiempo\_de\_repetición, función\_a\_llamar)), de esta forma logramos un conjunto de animaciones más fluidas.

En la escena del juego, nos encontramos con unos problemas, #el robot no debería moverse hasta que se terminen de realizar las interpolaciones de los botones y demás animaciones, porque el robot no aparece en pantalla hasta que no se terminen todas las animaciones; esto se solucionó agregando una tarea que es llamada después de tantos segundos (el tiempo que tarda en terminar de hacer las interpolaciones) para habilitar la acción de los botones del panel.   
 El mapa está compuesto por una grilla (matriz de 12x15), el robot se mueve de casillero en casillero y no está permitido que se mueva en diagonal. A partir de esto generamos nuestros propios movimientos para el robot y no usamos los que venían en el actor Robot por default (que eran : forward,backward,turnleft,turnright….) porque se usaban con tiempo y velocidad, lo cual no nos aseguraba la precisión de que se iba a mover tal cual queríamos (de casillero en casillero) , esta decisión tuvo sus complicaciones , para implementarlo primero que nada tuvimos que generar una matriz de 12x15, donde cada elemento tiene un Punto de coordenadas de la posición real que maneja pilas . Luego se creó un módulo de conversión ('avanzar',2)==>(fila,columna) , donde fila y columna está dentro del rango 12x15 , entonces se manda una lista de movimientos y me devuelve otra lista con cada paso que debe realizar el robot . Ej: [(0,0),(0,1),(0,2)] , después se recorre esa lista y se convierten a valores de las posiciones reales del mapa para mover al robot . Se tiene en cuenta que se termine de ejecutar el movimiento para realizar el siguiente más allá de que se presione una tecla mientras se está moviendo, este movimiento se agrega al final de la lista, entonces realiza los movimientos de forma ordenada y no se interrumpe nada. Otro de los factores que hubo que tener en cuenta, fue que cuando se envía la instrucción de girar, debíamos saber la posición hacia donde apuntaba el robot en ese momento, entonces se le aplica un MOD 4 , ya que hay solo 4 posiciones válidas.

Las instrucciones que se van enviando desde el panel de comandos quedan visibles\* en un sector de la pantalla que lo denominamos “prompt”. El mismo almacena todos los movimientos que realizo el jugador en ese nivel a modo de historial. Esto serviría para “compilar” las instrucciones y enviarlas luego al robot si es que está conectado.

\* Solo se muestran las instrucciones que entran en pantalla, hay 2 botones para recorrer las que no son visibles.

En cuanto a la implementación del Prompt se manejaron 2 listas, una con todos los elementos (instrucciones) y otra con solo los movimientos visibles; cada vez que se agrega un elemento y ya no entra más nada en la lista de elementos visibles, se realiza un desplazamiento de todos los elementos, para ir mostrando los actuales. Estas funciones de desplazamiento hacia la derecha o izquierda son usadas también por los botones que controlan el cursor. Un detalle de la implementación es que podes posicionar el cursor en cualquier lado pero al agregar una instrucción siempre se agrega al final, esto es así para que se respete el orden de entrada.

################################################################################################################################

Jugador:

Los datos del jugador contienen información por nivel, cada nivel tiene la siguiente información:

+vidas perdidas

+vidas al comienzo

+puntaje máximo

+puntaje obtenido

+tiempo inicio

+tiempo fin

+tiempo minimo

-movimientos mínimos (limite predefinido)

-movimientos efectuados

+cantidad de choques

+cantidad de caídas

+dificultad

\*Vidas perdidas: realiza un conteo de cuantas vidas se perdieron en ese nivel

\*Vidas al comienzo : informacion con la cantidad de vidas al iniciar el nivel

\*Puntaje Maximo: informacion con el Puntaje maximo que se podria alcanzar ese nivel

\*Puntaje obtenido: informacion con el Puntaje obtenido en ese nivel.

\*Movimientos minimos: Es la cantidad de movimientos minimos que se supone que debe realizar en ese nivel para llegar al objetivo.

\*Movimientos efectuados: Es la cantidad total de movimientos realizados , si se

se hicieron los movimientos : avanzar ,9 ; derecha ; avanzar, 2 , seran 3 los movimientos efectuados .

\*Cantidad de choques: informacion con la cantida de colisiones en ese nivel.

\*Cantidad de caidas: informacion con la cantidad de veces que se cayo del mapa.

\*Tiempo inicio: info hora

\*Tiempo fin : info hora

\*Tiempo minimo: es en el menor tiempo que se podria llegar a realizar la jugada , el tiempo minimo es almacenado en segundos.

dificultad

mover robot:

Este módulo recibe una lista de movimientos ej.

[('avanzar',2),('der',1),('avanzar',3),('izq',2),('retroceder'),2] y le envía la orden al robot físico.

Opciones de configuración:

Se pueden configurar distintos aspectos. Del robot se puede cambiar el Id, setear el nombre, listar los robots que están conectados, setear el tiempo y la velocidad (para lograr una similitud con entre el robot virtual y el físico). Del juego se puede establecer el nivel en el cual se desea iniciar, la dificultad, el tiempo máximo para pasar el nivel (tiene un máximo de xx y un mínimo de xx), se pueden habilitar y deshabilitar los sonidos o la música.

Escenas implementadas:

escena\_ayuda:

escena\_estadisticas:

escena\_juego:

escena\_menu:

escena\_menu\_q:

escena\_usuarios:

escena\_opciones: