Evidencia de avance en el proyecto final de informática II.

Al comenzar creando una buena estructura en el diagrama de clases correspondiente al desarrollo del proyecto, creí que no habría mucha dificultad para implementar estas ideas estructuradas en la codificación, pero el tema de las interfaces gráficas está siendo un tipo de bloqueo para implementarlas, dado que, aunque es el mismo lenguaje y se usan los conceptos que se aprendieron este tiempo, es un “acento” diferente del lenguaje y cuesta implementarlo, pero no es ningún motivo de rendición, nada lo es.

Comienzo por el principio, creando una ventana donde hayan dos opciones: (1) Registrarse y (2) Iniciar sesión. Para lograr esto he creado una clase ventana que hereda de QmainWindow y representa lo que recién mencioné. Uso un Qlabel apuntando a un fondo para mostrar una imagen de inicio cargada de un archivo .qrc, esta imagen se adapta al tamaño de la ventana gracias a una función propia de la librería Qpixmap, y se va redimensionando automáticamente con otro método de la librería mencionada. En esta ventana se crean dos instancias de QPushButton, las dos instancias corresponden al botón de iniciar sesión y el de registro. Ambos están colocados en posiciones específicas con tamaños específicos, y estos dos botones son conectados a sus slots correspondientes, los cuales van a abrir interfaces de dialogo para inicio de sesión y registro, Esto último mencionado es importante porque tengo dos clases distintas, una que hace referencia al dialogo de registro y otra al dialogo de inicio de sesión, ambas interactúan con un archivo de texto que almacena el nombre del usuario y su contraseña correspondiente, la contraseña es almacenada de manera simple (no hay ningún método de codificación binaria). Dentro de la misma ventana, hay un pequeño audio de fondo, para esto debo usar QMediaPlayer configurado con un archivo de audio localizado en el .qrc, esta reproducción se realiza en un bucle infinito definido en setLoops () para que la música se mantenga activa durante toda la sesión en la ventana, y para este audio se crea un Objeto QAudioOutput que me va a permitir controlar aspectos de reproducción y salida de audio de la aplicación.

Luego de iniciar sesión correctamente llegamos a otra ventana, que corresponde a la clase base del nivel 1 y nivel 2, la cual la llamé NivelWindow. Esta clase que también hereda de QmainWindow, representa la interfaz ya mencionada, que va a dar a elegir al usuario para seleccionar el nivel 1 o el nivel 2, aquí también hay un fondo de una imagen de los Simpson con las mismas configuraciones de fondo que ya mencioné en la primer ventana, y también, como en la primer ventana, he instanciado dos botones como objetos de QPushButtom, cada botón está vinculado a un slot mediante connect (), además sigue estando presente la misma canción.

Luego de oprimir el botón para jugar el nivel 1 llego finalmente al primer nivel, donde al fondo se encuentra una imagen de alguna calle de Springfield, Bart, y la parca como personajes, donde hasta ahora Bart se puede mover y saltar libremente, pero debe evitar que la parca lance sus hachas a su dirección, debe hacer esto mientras intenta recolectar las gemas para destruir al villano.

Para lograr esto, construyo una clase base llamada personaje la cual hereda de objetos gráficos de Qt para ofrecer el movimiento, graficas de los personajes etc. En esta clase defino métodos públicos pertinentes a ser parte de un personaje, defino objetos de tipo QTimer, porque si Bart salta o es golpeado, las funciones que aplican a Bart un empuje, mediante slots son conectadas, a un par de timer´s que ubican a Bart en el suelo luego de ser golpeado, o de saltar.

En la clase de Bart defino su sprite, la escala de su sprite, un método que lo mueve horizontalmente según sea el evento de teclado, y también la función de ser golpeado la cual hace un llamado a las funciones de la clase padre donde se aplica un empuje, se cambia la velocidad, se actualizan los retrocesos, etc.

En la clase de la parca también defino su sprite y escala, tengo una función importante que retorna punteros a objetos de tipo Hacha, con ecuaciones de magnitud y normalizando el “vector” que apunta a la dirección de Bart, en esta función creo objetos de tipo hacha y los agrego a la escena solo si no ha terminado, debo hacer énfasis en que todas se guardan en una lista.

En la clase del Hacha también tengo unos atributos privados que muestran las velocidades de las componentes del Hacha, un booleano que me dice si el hacha está eliminado o no, un timer de movimiento claramente, un método que hace que el hacha se mueva, y un getter para el booleano antes mencionado. Cuando la implemento defino el sprite y su escala, un método que conecta las salidas del timer con el movimiento del hacha, y en la función de mover trato de hacer que el hacha se elimine cuando se sale de las dimensiones de la pantalla si Bart la esquiva, aunque me esta crasheando la memoria porque parece que estoy eliminando objetos tipo hacha que no existen o que no se han creado y está siendo un problema bastante grande por lo que pienso añadir unos rectángulos que estén en los bordes de la imagen de fondo y que se verifique cuando el hacha los colisione, porque el problema no lo he podido resolver en un muy buen rato.

También tengo una clase sencilla llamada objeto, la cual solo implementa un constructor que settea la posición del objeto, de esta clase padre, hereda la clase gema, la cual es uno de los objetivos de Bart, esta gema desaparece cuando Bart la colisiona, y cuando las recolecta todas, se gana el juego, según esto, se puede ver que uso una lista de gemas para almacenarlas.

Finalmente implementando la clase de Nivel1, en la que comienzo añadiendo a la escena los ítems de los objetos creados con sus posiciones respectivas, añadiendo los timers para verificar las colisiones, para lanzar hachas cada cierto tiempo, y en el destructor de la clase, elimino estos timers. También el método que inicializa la escena, con el suelo como rectángulo y con el fondo como imagen, aquí también está el método que maneja los eventos del teclado, cambiando las direcciones del Bart según la tecla, y el salto, y por ultimo verifico las colisiones, esta función verifica perfectamente si Bart colisiona con las gemas o con la parca, lo cual funciona excelente, el ciclo que hace que Bart retroceda si choca con el hacha, pero cuando el hacha sale de la pantalla estoy accediendo a una extraña posición de memoria a eliminar y el programa queda crashed, el error insiste de cualquier forma y es que quizá sigo sin verificar cuándo sale de la pantalla. Y por último, si se recolectan todas las gemas, sale un mensaje por pantalla de que se ganó.

Falta mucho por hacer realmente, pero en cada renglón redactado he tenido docenas de dificultades y dudas, espero llegar a un punto que el juego satisfaga, aunque se ha aplicado la mayoría del conocimiento teórico otorgado, y es lo que satisface. Así finalizo mi avance parcial para el 13/11/2024, con mucho más por implementar pero con el buen objeto de hacerlo.