Marcos Bianchi - 108921

Compilación

Edite el makefile para poder compilar y correr escape_pokemon.c. Para compilar es “make escape” y para correr es “make valgrind-escape”. No use “make valgrind-escape” porque me tira errores de memoria antes de correr el programa, no sé por qué. Además le puse distintos flags a la compilación del escape porque me tiraba warnings que no me servían para nada, precisamente saque -Wconversion y -O2. Estuve corriendo el programa con “./escape ejemplo/objetos.txt ejemplo/interacciones.txt”.

Mi estructura sala_t

```c
struct sala {
        hash_t *objetos;
        lista_t *interacciones;

        abb_t *conocidos;
        abb_t *poseidos;

        bool escapo;
};
```

La variable escapo es solo un bool que si es true indica que el jugador ganó.

TDAs utilizados

Para los objetos utilice el diccionario para hacer más rápido el proceso de búsqueda, fue casi perfecto porque me hubiera gustado tener una función que itere la tabla llenando un vector como implementamos en el abb. Fue por eso que para mis objetos conocidos y poseídos utilice un abb, solo por esa función además de que es más rápido que una lista. Para las interacciones use una lista porque de las interacciones hay varias que comparten varios campos y hubiera sido raro designar la clave de una interacción al mensaje que muestra o un campo que no sea el del objeto con el que interactuamos para un diccionario. No use un árbol por el mismo punto pero en este caso no implementamos una manera de tratar cosas con nombres iguales, mi implementación no hubiera funcionado. Por último, las interacciones las iba a tener que iterar todas igualmente porque así es como funciona el juego, tengo que verificar el input del usuario con todas las interacciones y ejecutar las que coincidan.

Marcos Bianchi - 108921

Pruebas

En cuanto a las pruebas, me fallaron las que rompen la encapsulación, en especial la de la cantidad de interacciones porque agregue más interacciones al archivo de chanu/int.csv, solo cambie esas 2 y saque la que verifica que todos los nombres de los objetos sean los esperados porque al implementar un diccionario se desordenaron, nunca iba a dar bien esa prueba a menos que haya utilizado una lista.

Para los objetos cambie el sala->cantidad_objetos por hash_cantidad(sala->objetos) y para las interacciones cambie sala->cantidad_interacciones por lista_tamanio(sala->interacciones), aparte de verificar la cantidad con 13 en vez de 9.

Consideraciones

Para mostrar los errores de input al usuario decidi usar una flechita en la interfaz que se pone de color rojo en caso de ser un input o interaccion invalida y verde de haber sido valida. Aclaro esto porque en el diagrama de escape_pokemon hay texto explicando al usuario cuando una acción era inválida.

Puede ser que las pruebas sean un poco escasas y eso es porque primero hice la interfaz y fui testeando casi todo con eso. Casi para el final del trabajo empecé a hacer pruebas, además casi toda la implementación se testea desde sala_ejecutar_interaccion(). Están en el archivo de pruebas.c.

Agregue objetos e interacciones a los archivos de ejemplo y una nueva forma de ganar al juego.