Università degli Studi di Bergamo
Scuola di Ingegneria
Dipartimento di Ingegneria Gestionale, dell'Informazione e della Produzione
Corso di Laurea Magistrale in Ingegneria Informatica

Spelonca Explorer

Progetto C++

Daniele Ravasio, mat. 1045934

Anno Accademico 2020-2021

Indice

1 Introduzione

Questo progetto è l'ideazione di un gioco, è stato realizzato con C++ e potrebbe essere successivamente implementato in unity.
L'obiettivo di questo gioco è quello di trovare l'uscita da una caverna, muovendosi usando i comandi WASD.
La dinamica di combattimento non è convenzionale, infatti quando si incontra un nemico bisogna semplicemente lanciare un dado, e se il numero è maggiore di quello del nemico allora il nemico perderà una vita, altrimenti la perderà il nostro eroe, inoltre c'è anche la possibilità di fuga, la quale ha una chance del 20% di successo.

2 Suddivisione e classi

Ogni file è diviso in file.h e file.cpp. Nelle classi abbiamo la presenza del distruttore e del costruttore, nella classe Enemy il costruttore prende un valore già pre-settato, mentre nella classe Hero, il costruttore viene inizializzato con un parametro che viene preso in input dall'utente. In ognuna delle classi le variabili sono private mentre i metodi sono pubblici, in quanto, facendo così potremo andare ad aggiornare il contenuto della variabile mediante i metodi. Abbiamo anche la presenza di membri virtual, questi ci permettono di ereditare e ridefinire il metodo, andando quindi a fare un overriding. Parlando di ereditarietà abbiamo la presenza sia di ereditariertà multipla che non multipla, per esempio la classe Hero ed Enemy che derivano da Personaggio, sono un caso di ereditarietà non multipla. Mentre il caso di Gioco che deriva da Spelonca e da Menu è un caso di ereditarietà multipla.
Inoltre è stata usata anche la STL, ovvero la standard template library, grazie al fatto che è basata su un approccio che consente, troviamo il suo utilizzo nei metodi dove si richiama (Character*), andando a richiamare questo verrà fatta una distinzione in fase successiva, andando a capire il tipo preciso, quindi si risolverà senza problemi, non originando il problema dello slicing, ovvero quando l'oggetto di un tipo di sottoclasse viene copiato nell'oggetto di tipo superclasse.