Softwareplanung

Kurzbeschreibung:

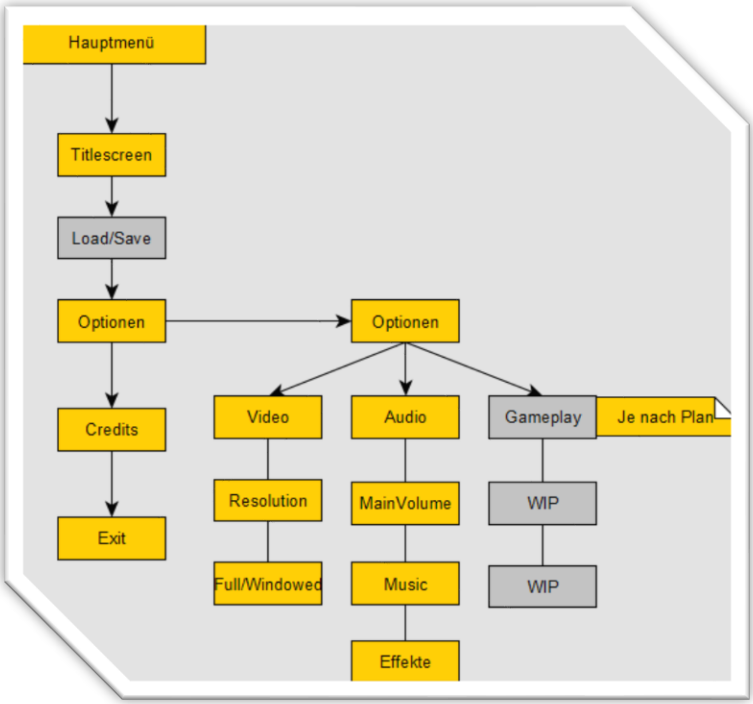
Das Projekt „2D-Aktion-RPG“, dem ein UML-Graph beiliegt, ist ein Freizeitprojekt. Die Handlung ist dem Genre der Isekai-Anime angelehnt. Zum Start des Spiels stirbt der Protagonist in der uns bekannten Welt und wird in einer ihm/ihr fremden Welt mit Magie wiederbelebt. Dies ist durch einen Fehler Gottes zu verantworten. Als Wiedergutmachung ermöglicht es Gott dem Spieler, in der Rolle des Protagonisten, besonders starke Magie zu wirken, wenn dieser Monster tötet, welche die Zivilisation bedrohen. Als finales Ziel gibt Gott dem Spieler die Aufgabe einen wiedererwachenden Dämonenkönig zu töten.

In Anlehnung an die klassische Heldenreise aus der griechischen Mythologie beginnt der Spieler als Protagonist auf Level 1 und muss durch generierte Dungeons laufen, die jeweiligen Bosse am Ende bezwingen und weiter aufsteigen. Als „Rückschlag“ oder „Hindernis“ muss der Spieler bestimmte Fähigkeiten benutzen, um diese Bosse zu töten, um mögliche Rückschläge zu vermeiden. Jedoch sind einige dieser Fähigkeiten erst nach einem geskripteten Event verfügbar in Form eines Todes. So kann der Spieler immer wieder Dungeons bezwingen und nach einem möglichen Tod seine gesammelten Erfahrungspunkte weiter verteilen, um so die Dungeons in aufsteigender Schwierigkeit leichter zu bezwingen.

Anforderungen und Features:

Als Anforderungen zu dem Projekt steht zunächst ein passendes Hauptmenu mit den Unterpunkten, die je nach Entwicklung des Projektes noch erweiterbar sind.

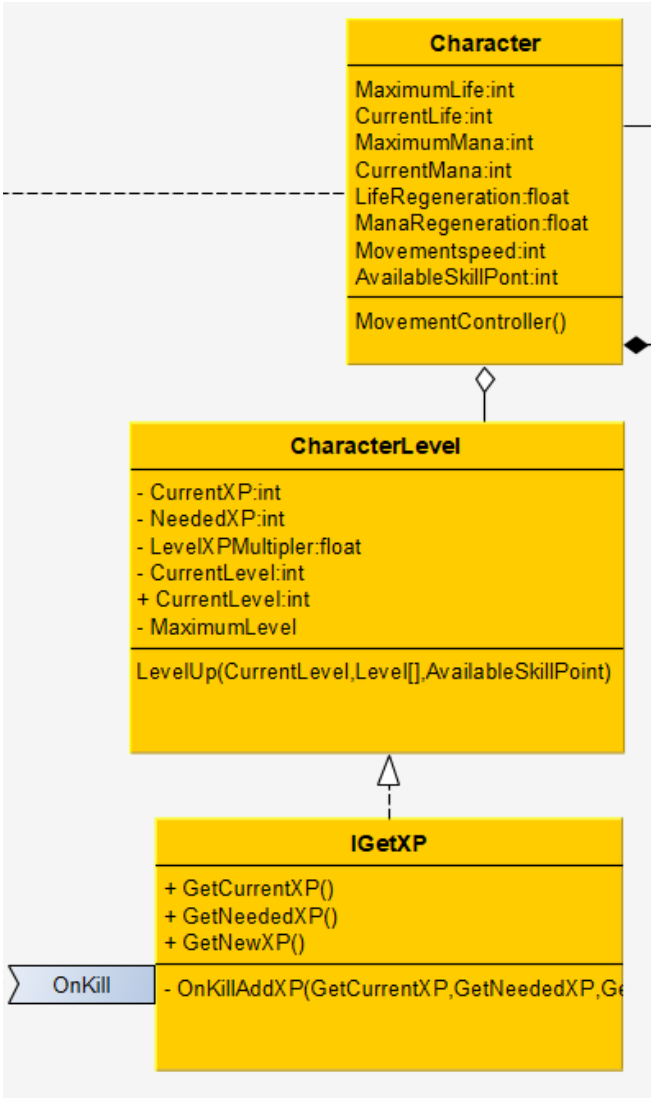
- Laden
- Optionen
    - Audio
        - Gesamtlautstärke
        - Musiklautstärke
        - Effektlautstärke
    - Video
        - Auflösung
        - Grafikdetails
        - VSync
        - Vollbild / Fenstermodus
- Credits
- Spiel verlassen



Weitere Anforderungen an das Projekt sind die Basisfunktionen über das Menu hinaus. Zum Laden der Spielszene mit einer möglichen Oberwelt, muss dieser auch die Fähigkeit besitzen über die einzelnen Szenen hinweg zu bestehen. Hierfür bietet sich die Implementierung eines Singletons-Patterns an, als Szenenmanager. So kann dieser bei passender Konfiguration über eine Szene hinaus bestehen und bietet die Möglichkeit als einzige Entität zu bestehen. Auch ist dieses Pattern für einen spielumfassenden Gamemanager möglich, der die Aufgaben von Szenenwechsel, Audioeinstellung und ähnlichen Elementen übernimmt, welche über den kompletten Spielverlauf verwaltet werden sollen.

Als weitere Anforderung kommen diverse Szenen die als Spielinhalte dienen. Beispielsweise eine Oberwelt zwischen den Dungeons und eine Szene für Dungeons. Diese sollten möglichst über einen performanten Algorithmus erstellt werden und dann abhängig vom Fortschritt des größer, sowie mit mehr und stärkeren Monstern befüllt werden.
Der Spieler benötigt auch zur Interaktion mit der Spielwelt entsprechend einen Charaktercontroller der den Input des Spielers ins Spiel bringt. Hierbei ist da drauf zu achten, dass mögliche Eingabegeräte nicht ausschließlich aus Maus und Tastatur bestehen, sondern auch gängige Controller (Xbox, Playstation oder Fremdhersteller) entsprechend abgedeckt sind, um eine größere Vielfalt an Vorlieben für Eingabegeräte einzufassen und diese Spieler nicht auszuschließen.

Für die Szenen der Dungeons werden durch einen Algorithmus Gegner in diesen verteilt. Daher benötigen diese als erste mögliche, aber notwendige, Implementation eine künstliche Intelligenz um eine ausreichende Bedrohung für den Spieler darzustellen. Hierfür bietet sich eine einfache State Maschine an. Mit dieser kann ein simples Verhalten simuliert werden. Diese bietet eine solide Basis zur Erweiterung. Für komplexere Verhaltensmuster sollte ein Behaviour Tree in Betracht gezogen werden, da dieser umfassendere Verhaltensmuster ermöglicht, aber zugleich auch anfälliger für fehlerbehaftet Verhalten durch fehlerhaften und komplexeren Code ist.



Da nun Spieler sich bewegen und Gegner angreifen können ist es auch notwendig für den Spieler eine Form des Angriffs einzubringen. Von einem simplen Angriffssystem, aber auch System für Talentbaum, bestehend aus einzelnen Fähigkeiten die aufeinander in Abhängigkeit aufbauen. Als Voraussetzung dafür ist ein System notwendig, dass es dem Spieler ermöglicht Werte zu besitzen, wie Leben, Statuswerte und Level.

Ein möglicher Ablauf wäre:

Spieler tötet Monster => Spieler erhält Erfahrungspunkte => Erhaltene Erfahrungspunkte >= benötigter Erfahrungspunkte

Als Resultat steigt der Spieler im Level auf und mögliche Statuswerte steigen abhängig vom Level auf. So kann der Spieler auf Level 1 nur 7 Stärke besitzen und nach dem Aufstieg in Level 2 nun 8 Stärke besitzen.

Dies bringt die Möglichkeit eines Item Systems mit sich, dass von Statuswerten abhängig bestimmte Verbesserungen in defensiver oder offensiver Natur mit sich bringt. So kann die Erhöhung des Statuswerts Stärke den physischen Angriff mit Waffen wie zum Beispiel Schwertern erhöhen. Die Einführung eines Item Systems und eines dazu passenden Inventar Systems ist eine umfangreiche Aufgabe, dessen Umfang nicht unterschätzt werden sollte. Eine ausführlichere Grafik einer möglichen Implementation ist der beiliegenden UML-Grafik zu entnehmen, sowie alle bisher genannten Systemen, dass auch Aspekte eines grafischen Benutzerinterface anschneidet.

Schlusswort:

Abschließend ist zu erwähnen, dass diese Systeme alle möglichst unabhängig voneinander gestaltet werden müssen, um so eine mögliche Erweiterbarkeit in der Zukunft zu gewährleisten. Dies ist prinzipiell als Grundvoraussetzung zu sehen und sollte bereits in der Planung erfasst werden. Eine gute Voraussetzung dafür bietet die Orientierung an den fünf Solid Prinzipien des objektorientierten Designs.

Das Projekt „2D-Aktion-RPG“ ist immer noch ein Freizeitprojekt und befindet sich in einer Planungsphase. Auf Grund des Studiums und privaten Verpflichtungen ist es da durch noch immer „work in progress“ und hier genannte Systeme oder Umfang können sich ändern. Es ist schlussendlich davon auszugehen, dass sich diese Systeme im Lauf der Entwicklung ändern können.