Programmierung:

Grundlegende Struktur:

Programmieren in C++:

Warum C++?

C++ ist eine von [Bjarne Stroustrup](https://de.wikipedia.org/wiki/Bjarne_Stroustrup) bei [AT&T](https://de.wikipedia.org/wiki/AT%26T) entwickelte Programmiersprache, sie gilt als Erweiterung der Sprache C. (Wiki) C++ ermöglicht dem Nutzer eine Maschinennahe Programmierung und dadurch auch hohe Effizienz. Dies ist gerade bei der leistungsaufwändigen Spieleprogrammierung relevant.

C++ und OOP:

Die Sprache unterstützt, anders als C, objektorientierte Programmierung (OOP). Das bedeutet das der Code praktisch in kleinerer abschnitte unterteilt werden kann, diese abschnitte nennt man Klassen. (Bessere Erklärung einfügen)

Wenn der Code, der sich in den Klassen befindet, ausgeführt werden soll, muss eine Instanz (Kopie des Codes) erstellt werden. Je nach dem ob der Code im Stack oder im Heap allokiert werden soll, wird anderer Code verwendet sprich: „Class ClassInstanceAtStack;“ oder „Class\* ClassInstanceAtHeap = new Class()“

Wobei „new“ einen Pointer\* initiiert und die Größe der Klasse in Bytes im Heap freigibt. (Dabei wird in der Pointer Variable die Speicheradresse gespeichert). Um den Speicher wieder Freizugeben muss anschließend „delete“ aufgerufen werden um den Speicher freizugeben. Um den new/delete aufwand zu umgehen werde ich im Programm smart-pointer\* verwenden.

std::shared\_ptr und std::unique\_ptr: (Aus der offiziellen Dokumentation zitieren)

Um das Prozedere um new/delete bei der Erstellung von Pointern\* zu umgehen wurden in C++XX (Recherchieren) Smart-Pointer hinzugefügt diese automatisieren die Speicher zu- und Abweisung bei Pointern. Es gibt 3 verschiedenen Variationen dieser Pointer: unique\_ptr, shared\_ptr und weak\_ptr. std::unique\_ptr kann nicht kopiert werden. (z.B in einem Vector gespeichert werden oder als Parameter einer Funktion verwendet werden).

std::shared\_ptr hingegen kann kopiert werden. Dies kommt aber mit einem Overhead\* da Kontrollspeicher mit überblick über erstellte Referenzen initialisiert wird. Prinzipiell wird der Speicher erst freigegeben wenn keine Referenz zum shared\_ptr existieren. Sollte eine Referenz zu einem shared\_ptr benötigt werden welche nicht als eigenständige Kopie gelistet werden soll kann std::weak\_ptr verwendet werden.

(Evtl. C++ Logo einfügen?)

Da ich zu beginn der Arbeit schon grundlegende Kenntnisse in C++ habe, liegt die Wahl der Programmiersprache nahe.

Da ich bereits vor

C++ Spracheigenschaften:

SMFL:

Das Spiel:

Klassen:

GameWindow:

MenuWindow:

MapManager:

Player:

Enemy

Item

Gun

Perlin Noise:

* Theorie
* NoiseGenerator