Fortgeschrittener

Using-Direktiven vereinfachen

Inhaltsverzeichnis

1 Using-Direktiven über multiclass inheritance einbinden statt direkt zu callen:

Beim Thema Scripting gibt es oftmals die Schreibweise von "<Namespace>::<Methode>" --> zum Beispiel: "Game::GetTime()". Mir ist das über die Zeit hinweg ziemlich aufn Zeiger gegangen und deshalb habe ich ein wenig C++-Wissen genutzt und dachte mir einfach den Namespace direkt einzubinden per "using namespace <Name>;". Das ging bis zu einem gewissen Grat, aber dann kam die Emergency 4 interne Problematik dazu, dass das scheinbar nicht unendlich geht. Hier eine generische Funktion mit der einzelnen Vererbung der CommandScript-Klasse inkl using-Anweisung:

```
object GenericFunction : CommandScript
{
    using namespace Audio;
    using namespace Game;
```

Somit wird aus der vorherigen Schreibweise:

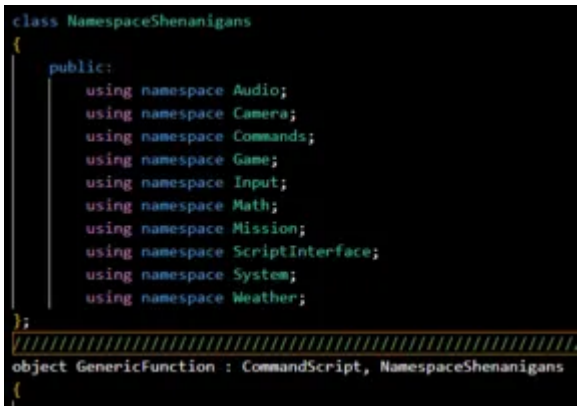
```
void PushActions(GameObject *Caller, Actor *Target, int childID)
{
    float uhrzeit = Game::GetTime();
```

... die "neue" Schreibweise ohne den Namespace davor:

```
void PushActions(GameObject *Caller, Actor *Target, int childID)
{
    float uhrzeit = GetTime();
```

Da aber die neue Schreibweise inklusive Einbindung der Using-Direktive zu dem oben benannten Fehler führt, müssen wir diesen mitigieren. Lange Zeit habe ich das getestet und eine Lösung gefunden:

Man kann eine ***Extra-Klasse*** errichten, die bereits die using-Direktiven in sich trägt und durch ***multiple Klassenvererbung*** in C++ ist es möglich diese zu nutzen:



Vorteil:

- Ihr müsst nicht ständig <Namespace>::<Methode> eintippen, sondern könnt die neue Schreibweise (s.o.) nutzen.
- Der Fehler von zu vielen Using-Direktiven taucht nicht mehr auf (zumindest bei meinem mehrfachen extensiven Testing mit rund *10.000 Calls**)**

:!: ~~**ACHTUNG:**~~
~~**JEDES Script checkt nur innerhalb seiner eigenen Datei, ob diese Klasse vorhanden ist und kann auch nur innerhalb der eigenen Datei diese Klasse finden.** Es ist also zwar ein wenig Arbeit alles zusammenzufassen und vielleicht nicht der extrem beste Codingstyle, jedoch spart man sich Arbeit beim Schreiben und jedes object (Methode) kann sowohl von CommandScript als auch eurer eigenen Klasse vererben (siehe oben).~~

~~** = Ob das bei multiplen .script-Dateien nicht irgendwann doch zu einer Querelei kommt, kann ich zum aktuellen Zeitpunkt noch nicht abschätzen, wird sich aber in meiner Entwicklungsarbeit an Hamburg relativ zügig zeigen.~~ 8|

Tante EDIT: *Dank Antiphonist es möglich über den **Pre-Compiler** die Klasse sogar global zugreifbar zu machen, d.h. ihr müsst gar nicht einen hässlichen Codingstyle oder eine wilde Einzeldatei erstellen. Es ist somit machbar diese Struktur nun über mehrere Scripte zur Verfügung zu stellen. Bitte meldet euch mal, wenn ihr das ausgiebig getestet habt, ob es nochmal zu einem Fehler kommt!*

* = Sollte das dennoch auftreten, bitte sofort Bescheid geben, denn dann muss ich nochmal schauen, wie man diese Problematik weiter umgehen kann. Vielleicht per Injection oder Sideload oder wie auch immer. :saint:

LG Korbi