

Grundlagen - Kapitel 5: Variablen & Datentypen

Liebe Leser/innen,

das Weihnachtsfest kommt immer näher, das Jahr neigt sich dem Ende. Draußen ist es kalt und nass und somit einfach perfekt, um den Abend oder einen Nachmittag mit Trainz zu verbringen und etwas neues zu lernen.

Darum möchte ich euch heute ein weiteres Kapitel unserer Scripting-Reihe schenken. Ich weiß, es ist im Moment recht trocken, was Beispielscripte angeht, allerdings muß ich erst einmal zusehen, daß ich eine gewisse Grundlage schaffe. Im neuen Jahr wird es dann richtig los gehen.

Es kamen Wünsche aus dem Auran Forum zum Thema Animationen, denen ich natürlich gerne nachkommen werde, sobald ich hier alles wichtige vorab besprochen habe.

Heute wird es darum gehen Datentypen, also im Allgemeinen Variablen (Member!) zu verstehen

Dann gehts mal los!

Variablen

Wer sich mit dem Wort ?Variable? wieder an seine Schulzeit in den Mathematikunterricht zum Thema ?Algebra? zurück erinnert fühlt, liegt richtig. Egal, wie schwer oder leicht es damals gefallen ist, die Werte ?x? und ?y? aus einem linearen Gleichungssystem zu berechnen, gibt es keinen Grund jetzt zu resignieren. Es ist nämlich ganz einfach:

Eine Variable ist in der Programmierung und im Auran Game Script nichts anderes als eine Bezeichnung im Klartext für einen Wert. Das bedeutet nichts anderes, daß eine Variable ein Platzhalter für einen Wert ist. ?Variable? kommt von ?variieren? und bedeutet hier ganz richtig: ?Ein nicht fest definierter sich ändern könnender Wert?.

Im Auran Game Script unterscheidet man grundsätzlich unter vier verschiedenen Typen von Variablen, den sog. ?Datentypen?. Jeder Datentyp ist dazu da, eine besondere Art eines Wertes zu speichern.

Wir unterscheiden zwischen den Typen:

- int (Englisch: integer, Deutsch: Ganzzahl)
- float (Englisch: floating point, Deutsch: Gleitkommazahl)
- string (Englisch: string, Deutsch: Zeichenkette)
- bool (Englisch: boolean, Deutsch: Boolesche Variable; Wahrheitswert)

Gehen wir einmal auf die Datentypen genauer ein:

int:

Wie oben bereits erwähnt ist eine int-Variable dafür zuständig eine ganze Zahl zu speichern.

Eine ganze Zahl definiert sich dadurch, daß sie kein Komma und somit keine Nachkommastellen enthält (z.B.: 1, 2, 4, 7, 1123, etc). Eine int-Variable kann auch negativ sein (-45, -2, 0, 1, 125, 645).

float:

Eine float-Variable ist eine Kommazahl. Diese kann auch negativ sein (z.B.: -1.25478, -0.05478, 0.0, 2.478, 48948569.01). Wichtig ist: Ein tatsächliches Komma in der Programmierung und im Auran Game Script wird dann verwendet, wenn man Werte voneinander trennen möchte, bzw. diese aufzählt. Ein Komma bei einer Zahl wird als Punkt gesetzt. Außerdem ist es üblich bei einer float-Variablen am Ende des Wertes ein 'f' zu setzen, wie: 0.145f, was aber keine Pflicht ist.

string:

Eine string-Variable beinhaltet eine Zeichenkette. Ein Wort, eine Phrase, was auch immer gewünscht ist. Als Beispiel: 'Ich bin ein Scripter!' wäre ein gültiger Wert für eine string-Variable.

Eine Zeichenkette wird wie man sieht immer in Anführungszeichen gesetzt, damit das Programm, in unserem Fall Trainz, erkennt, daß es sich um eine Zeichenkette handelt.

bool:

Der Datentyp bool ist eine kleine Besonderheit. Er kann nur zwei Werte annehmen:

'true' (wahr) oder 'false' (false). True und false sind keine Zeichenketten, sondern ein fester Bestandteil der Programmierung und sind somit eigene fest definierte Werte.

Wie verwende ich nun Variablen in meinen Scripten:

Man unterscheidet in der Programmierung und im Auran Game Script zwischen der Deklaration und der Initialisierung einer Variablen. Das bedeutet nichts anderes als:

Deklaration:

Eine Variable wird benannt, 'erstellt', ohne ihr bereits einen Wert zu geben.

Initialisierung:

Einer bereits deklarierten Variable wird ein Wert zugewiesen.

Es geht auch beides in einem:

Wir deklarieren eine Variable und initialisieren diese zugleich.

Wie sieht das ganze aus?

Variablen deklariert man, in dem man zuerst angibt, welchen Typs sie angehören, anschließend wird direkt dahinter durch ein Leerzeichen getrennt die gewünschte Bezeichnung angegeben. Wer seine Variable jetzt schon initialisieren möchte, kann dann mittels Zuweisungsoperators ('=') ihr einen Wert zuweisen. Das geht wie in der Schule: $x = 1$.

Am Ende der Zeile muss dann ein Semikolon stehen.

Dazu einfach mal ein kleines Beispiel. Wir haben ein Script wie dieses hier, ein einfaches Szenerie-Objekt und spielen einfach mal ein wenig mit Variablen herum, ohne ein bestimmtes Ziel, bis auf das des Demonstrationszweckes, zu verfolgen:

Code

```
include "MapObject.gs"
class CDemoClass isclass MapObject
{
    // Wir deklarieren und initialisieren jeweils eine Variable jeden Typs
    string m_sEineZeichenkette = "Ich bin ein bool"
    // Wir deklarieren nur eine Variable jeden Typs

    public
    {
        inherited(pAsset);
        // Erst nach dem unser Objekt von Trainz im Spiel erstellt wurde,
        // initialisieren wir die letzten Variablen
        m_sZweiteZeichenkette = "Hey, ich bin ein bool"
    }
};
```

Alles anzeigen

Das Beispiel zeigt deutlich, wie es funktioniert, Variablen zu erstellen und diese mit Werten zu füllen. Außerdem wird in den Zeilen 21 - 24 ersichtlich, daß man wirklich einfach nur die Bezeichnung der Variable angeben muß um ihren Wert später wieder manipulieren zu können. Wir benötigen dann keine Angabe des Typs mehr, da durch die Deklaration der Variablen bereits bekannt ist, welchen Typs diese angehört.

Kann man Datentypen konvertieren?

Ganz klar: Ja! Darauf werde ich aber noch einmal gesondert eingehen. Nicht jeder Datentyp lässt sich in jeden anderen konvertieren. Der Grund dafür geht sehr tief in die Materie der Softwareentwicklung hinein. Auch wenn ich einfach sagen könnte, wie und was sich in was konvertieren lässt, ist es an dieser Stelle aufgrund fehlender Zielführung noch zu früh sich damit zu befassen.

Der Gültigkeitsbereich

Wer hat es geahnt? Variablen sind natürlich nicht überall gültig. Wir haben schonmal über den sog. globalen Gültigkeitsbereich gesprochen, den es in Auran Game Script ja gar nicht gibt. Das ist auch egal und nicht weiter schlimm und hat auch nicht viel mit dem Gültigkeitsbereich von Variablen zu tun. Es ist einfach nur das gleiche Thema.

Wer sich den Aufbau eines Scriptes schon einmal angesehen hat, hat sicher bemerkt, daß geschweifte Klammern eine sehr große Rollen spielen. Das tun sie in der Tat!
Wer genau hinsieht, erkennt sog. Blöcke. Ein Block wird durch eine geöffnete geschweifte Klammer geöffnet, `{`, und durch eine geschlossene geschweifte Klammer wieder geschlossen, `}`.
In jedem Block können neue Blöcke auftauchen, die dem vorherigen Block untergeordnet sind.

Sieht man sich da mal unser letztes Beispiel an, so hat man als `Hauptblock`, den Block der Klasse `CDemoClass` (Zeilen 002 bis 026). In diesem Block gibt es noch einen Block und zwar den der Init-Methode (Zeilen 016 bis 025). Der Block der Init-Methode ist dem Block der Klasse `CDemoClass` untergeordnet.

Wir haben nun alle Variablen in den Block der Klasse `CDemoBlock` deklariert. Damit sind unsere Variablen in allen Blöcken innerhalb der Klasse `CDemoClass` gültig und können in der gesamten Klasse verwendet werden.

Man kann in allen Blöcken im gesamten Script Variablen deklarieren. Wir könnten sogar hingehen und in unsere Init-Methode ebenfalls eine Variable deklarieren. Schreiben wir also etwas wie:

```
int m_iZahlFuerInit = 2;  
unter inherited(pAsset);
```

So haben wir eine Variable `m_iZahlFuerInit` erstellt, die nur in der Init-Methode gültig ist. Also nur ab der Zeile, in der sie deklariert worden ist, bis zur Zeile 025, also dort wo der Block der Init-Methode endet. Versuchen wir nun von irgendwo anders im Script diese Variable anzusprechen, also ihr beispielsweise einen neuen Wert zuzuweisen, bekommen wir in Trainz einen Scriptfehler an den Latz geknallt. Verständlich: **Denn, die Variable `m_iZahlFuerInit` existiert nirgendwo anders im Script, also im genannten Gültigkeitsbereich (Also innerhalb der Init-Methode).**

Damit mache ich für heute Schluss! Es waren wieder viele Informationen, die erstmal verstanden werden wollen. Als ich mit allem anfang, habe ich mich auch sehr schwer getan, manche Dinge zu verstehen. Es ist also kein Beinbruch, sich diesen Beitrag auch mehrere Male durchzulesen. Ich beantworte auch gerne Fragen, die ihr mir in den Kommentaren stellt in einem nächsten Beitrag.

Also bis demnächst
euer Pascal