Mit string.gsub mehrere, unterschiedliche Informationen aus einem (1) Text auslesen

Lua-Grundlage

`string.gsub(Zeichenkette, Muster, Ersatz)` - bzw. wenn man es lieber in Englisch mag `string.gsub(string, pattern, replace)` - ist eine sehr leistungsstarke Funktion, die auf verschiedene Arten verwendet werden kann. Alles aufzuführen, würde den Rahmen hier sprengen. Ich beschränke mich auf das Nötigste, was erforderlich ist, damit auch Lua-Einsteiger oder sogar Lua-Unkundige nachvollziehen können, wie man mit dieser Funktion mehrere, unterschiedliche Angaben aus einem (1) Text auslesen kann und wofür man das verwenden kann.

Bei einfacher Verwendung der Funktion werden alle Vorkommen des `Musters` in `Zeichenkette` durch `Ersatz` ausgetauscht. `string.gsub("Papa küsst Mama und Paula küsst Hans", "küsst", "liebt")` liefert als Ergebnis: "Papa liebt Mama und Paula liebt Hans".

Einige Zeichen, so genannte magische Zeichen (*magic characters*), haben besondere Bedeutungen, wenn sie in `Muster` verwendet werden:
`( ) . % + - * ? [ ^ $`
Diese Zeichen ermöglichen beim Mustervergleich spezielle Aktionen durchzuführen.

- Das grundlegendste davon ist der Punkt "**.**", der ein beliebiges Zeichen darstellt.

- Das Minus-Zeichen "-" ist ein Modifikator für Wiederholungen und entspricht null oder mehreren Zeichen, wobei diese Wiederholungselemente immer auf die kürzest mögliche Sequenz passen.

- Durch eine sogenannte Erfassung (engl.: *capture*) wird es einem Muster ermöglicht, Teile der `Zeichenkette`, die mit Teilen des `Musters` übereinstimmen, zur weiteren Verwendung herauszuziehen, indem Sie die Teile des Musters, die Sie "erfassen" möchten, zwischen runden Klammern "**( )**"schreiben.

Mit diesen 4 *magischen* Zeichen haben wir die für dieses Projekt entscheidende Mustererfassung: "(.-)". So bedeutet z.B. "{(.-)}", dass alle Zeichen zwischen den geschweiften Klammern { } in *fauler* Übereinstimmung (engl.*: lazy match*) erfasst werden, d. h. mit so wenig Zeichen wie möglich. Ein Beispiel:

`string.gsub("Hans fährt {BMW}, Paul fährt {VW} und Peter fährt {Ford}.", "{(.-)}, "Goggomobil")` liefert als Ergebis: "*Hans fährt Goggomobil, Paul fährt Goggomobil und Peter fährt Goggomobil.*"

In diesem Beispiel war `Ersatz` eine Zeichenkette. `Ersatz` kann in der Funktion string.gsub aber unter anderem auch eine Funktion enthalten. Nehmen wir an, wir wollten die Autos der drei Freunde in einer Tabelle speichern.

```
Text = ("Hans fährt #BMW#, Paul fährt #VW# und Peter fährt
#Ford#."
Autos = {} -- Tabelle für alle Autos
Muster = "#(.-)#"
i = 1
string.gsub(Text, Muster, function(car) Autos[i] = car; i = i
+ 1 end)
```

Nun werden alle Zeichen innerhalb des 1. #-Zeichenpaares # # an den Funktionsparameter `car` übergeben, dann `car` in den 1. Tabellenplatz von Autos geschrieben und der Zähler i um 1 auf 2 erhöht. Danach werden alle Zeichen innerhalb des 2. #-Zeichenpaares # # an den Funktionsparameter `car` übergeben usw. für alle # #-Bereiche. Danach ist

```
Autos[1] == "BMW"
Autos[2] == "VW"
Autos[3] == "Ford"
```

Ein zweites Beispiel:

```
Text = "Franz fährt mit {80} Sachen quer durch Bayern."
speed = 0
string.gsub(Text, "{(.-)}", function(zahl) speed =
tonumber(zahl) end)
```

Hier werden alle Zeichen zwischen den geschweiften Klammern { } an den Funktionsparameter `zahl` übergeben, dann `zahl` in einen numerischen Wert umgewandelt und in der Variablen `speed` gespeichert, so dass `speed == 80` ist.

Diese beiden Beispiele zeigen auch, dass die Abgrenzungszeichen vor und nach den runden Klammern der Mustererfassung sowohl gleich sein können wie im 1. Beispiel "#(.-)#" als auch unterschiedlich sein können wie im 2. Beispiel "{(.-)}".

Welche Zeichen kann man für solche Abgrenzungszeichen verwenden bzw. nicht verwenden? **Nicht** verwenden sollte man (obwohl es möglich wäre) alle Zeichen, die auch in dem zu erfassenden Teil der `Zeichenkette` vorkommen können, also `a` bis `z` sowie `A` bis `Z` und `0` bis `9`, da es damit leicht zu kürzeren Erfassungen

kommen kann, als man möchte. Daraus folgt, dass man Sonderzeichen wie z. B. **! § & = @ € ~ | { }** nehmen kann.

Was ist mit den sog. *magischen* Zeichen? Kann man die auch verwenden? Nicht so ohne weiteres, denn woher soll Lua wissen, ob z.B. das Minus-Zeichen ein normales Minus-Zeichen ist oder als Multiplikator verwendet werden soll. Nun da kommt ein weiteres *magisches* Zeichen ins Spiel:

- Das %-Zeichen hat als *magisches* Zeichen einen "Entzauberungs"-Charakter. Es nimmt dem nachfolgenden Zeichen seine Magie und wandelt es in ein normales Zeichen um, auch sich selbst.

So hätte man in den o. g. Beispielen anstatt `"#(.-)#"` z.B. auch `"%%(.-)%%"` oder `"%$(.-)%$"` verwenden können. Von einer Verwendung von `"%((.-)%)"` oder `"%-(.-)%-"` ist aber dennoch abzuraten, denn `( )` und `-` kommen auch häufig in zu erfassenden Texten vor (siehe oben).

Speichert man nun alle auszulesenden Angaben in einem (1) Text - zusammenhängende immer mit derselben Kombination von Abgrenzungszeichen, nicht zusammenhängende jeweils mit anderen Abgrenzungszeichen - so kann man sie mit den entsprechenden Mustern je nach Bedarf abrufen und verwerten. Dabei ist es egal, wo man den Text speichert, in einem Slot oder einem Tagtext:

```
ok = EEPSaveData(999, Text)
ok = EEPStructureSetTagText("#7", Text)
ok = EEPRollingstockSetTagText("sbbce6-8ii_14253_sm2", Text)
```

und natürlich entsprechend auslesen:

```
ok, Text = EEPLoadData(999)
ok, Text EEPStructureGetTagText("#7", Text)
ok, Text EEPRollingstockGetTagText("sbbce6-8ii_14253_sm2)
```

Züge suchen sich in der Realität kein freies Bahnsteiggleis. Sie halten in der Regel im Bahnhof an vorgegeben Gleisen, bei der Hinfahrt meistens an anderen als bei der Rückfahrt. So hält z. B. in Dortmund Hbf der ICE 543 nach Berlin Gesundbrunnen über Hannover - Wolfsburg - Berlin Hbf in Gleis 10. Nehmen wir an sein Gegenzug sei der ICE 846 nach Köln Hbf über Essen - Duisburg - Düsseldorf Flughafen - Düsseldorf Hbf. Der hält in Gleis 16. Weiterhin nehmen wir an, dass die Fahrplangeschwindigkeit des Zugpaares auf dem in EEP abgebildeten Streckenabschnitt 180 km/h beträgt. All diese Informationen kann man nun in einen Text schreiben:

```
Text = "!10! &Berlin Gesundbrunnen& &Hannover - Wolfsburg -
Berlin Hbf& &ICE 543& |16| @Köln Hbf@ @ Essen - Duisburg - D-
dorf Flgh - D-dorf Hbf@ @ICE 846@ {180}"
```

Auslesen kann man diese Informationen dann wie in den Beispielen 1 und 2 gezeigt.

Demo-Anlagen

Um zu demonstrieren, wie man in der Praxis aus diesen Informationen eine Gleisansteuerung, unterschiedliche Geschwindigkeitseinstellungen für verschiedene Züge und Zugzielanzeigen realisieren kann, stelle ich die von mir gebaute Demo-Anlage "**gsub-Demo-ZZA_V15**" allen MEF-Usern zur Verfügung. Hierin werden

Zugzielanzeiger aus dem Shop-Set V15NDB20015 genutzt. Da dieses Set die editierbaren Textfelder benutzt, ist die Anlage erst ab EEP15 betreibbar. Außer dem genannten Shopset wurden nur Grundmodelle von EEP15 verwendet.

Damit auch EEPler, die dieses Set nicht besitzen, sich die praktische Anwendung dieser Lua-Funktion ansehen können, habe ich zusätzlich die Anlage "**gsub-Demo-Infotext_V15**" erstellt. Anstatt der Anzeige in den ZZAs erfolgt hier die Anzeige über EEP-Infotext-Funktionen. Da diese Anlage dieselben Ressourcen nutzt, ist auch sie ohne Beeinträchtigung nur ab EEP15 nutzbar. Jedoch lässt sich der Bahnhof leicht gegen einen anderen austauschen. Und wie man Züge austauscht ist weiter unten beschrieben.

Speichermöglichkeiten gibt es - wie schon gesagt - viele. Da es sich aber jeweils um Daten zu einem Zug handelt, was liegt da näher, als es im Zug selber zu speichern. Ein Zug besteht aus mindestens 1 Rollmaterial. Also wählte ich immer das 1. Rollmaterial in positiver Fahrtrichtung (was dem letzten bei Rückwärtsfahrt entspricht).

In beiden Anlagen fahren 4 Züge durch einen 3-gleisigen Bahnhof. Auf der Hinfahrt (von Ost nach West) fahren alle über Gleis 1. Die Personenzüge halten kurz am Hausbahnsteig. Der Güterzug fährt durch. Kaum verschwinden sie im Westen am Horizont kommen sie auch schon als Gegenzug zurück. Der ICE hält immer an Gleis 2. Regional-Express (RE) und Regional-Bahn (RB) halten immer an Gleis 3. Während der RE weiter auf die Strecke fährt, endet die RB im Bahnhof und fährt anschließend ins BW. Der Güterzug fährt auch auf seiner Rückfahrt am Bahnhof durch. Doch hier darf der Lokführer sich bei jeder Durchfahrt selbst aussuchen, ob er durch Gleis 2 oder 3 fährt. Auch diese Anweisung ist im Tagtext gespeichert.

Mir ist klar, dass man den Lua-Code der Anlagen an vielen Stellen optimieren kann. Ich habe bewusst keine ineinander geschachtelten Funktionen benutzt, zu übergebende Parameterwerte häufig vorher spezifiziert und möglichst deutsche Variablen- und Funktionsnamen in ausgeschriebenen Wörtern verwendet. Ich weiß, dass nicht nur der Ritter von Berlichingen mir mit der eisernen Faust drohen wird, da (fast) alle Funktionen aus if-Anweisungen bestehen. Aber ich bin fest der Meinung,

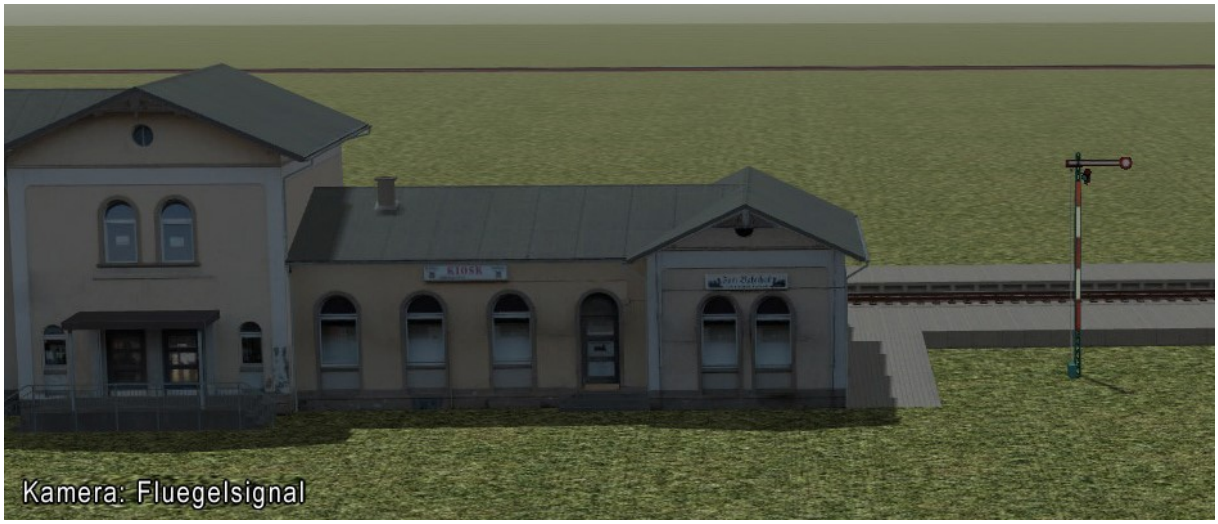
> **Wenn** *(if)* die Sonne scheint **dann** *(then)* fahre ich an die See,
> **ansonsten wenn** *(else if)* es schneit **dann** *(then)* fahre ich in die Berge,
> **ansonsten** *(else)* bleibe ich zu Hause. **Schluss**! *(end)*.

versteht jeder EEPler; denn auch bei Signal/Weichen-Verknüpfungen wird dieses Prinzip verwendet: **Wenn** Signal gestellt ist auf ... **dann** Signal/WeichX stellen auf …. Damit der weitere Code auch für Lua-Anfänger und selbst Lua-Unkundige verständlich ist, habe ich nahezu jede Codezeile ausführlich kommentiert.

Nach dem Download aus dem Kostenfrei-Bereich muss man den in der ZIP-Datei enthaltenen Ordner **gsub-Demo_V15** nur in den Anlagenordner von EEP kopieren. Darunter befinden sich zwei Unterordner mit den beiden Anlagen und diese Dokumentation sowie die Datei "Zugdaten.lua" mit den TagText-Einstellungen der in den Anlagen eingesetzten Züge als Beispiele.

Anlage starten

Neben dem Bahnhof steht als Relikt aus alten Zeiten ein Flügelsignal. Stellt man dieses auf Fahrt, beginnt das Spiel.



Anlage beenden und speichern

Natürlich kann man die Anlage jederzeit beenden. Solange man sie vorher nicht speichert, beginnt sie beim nächsten Mal wieder mit dem 1. Zug.

Möchte man die Anlage aber speichern, weil man etwas verändert hat, so empfiehlt es sich, die Anlage geordnet abzufahren. Wenn ein Zug auf der Rückfahrt (!) in den Bahnhof einfährt (also in Gleis 2 oder 3) wird automatisch auf die Kameraposition "Gleis 3" (s. Bild unten) umgeschaltet. Hierin ist das Flügelsignal neben dem Bahnhof gut zu sehen und man kann es mit einem Mausklick bei gedrückter Strg-Taste wieder auf Halt stellen.

Alternativ kann man jederzeit, wenn sich ein Zug auf der Rückfahrt befindet (auf keinen Fall vorher!!!), aber bevor er den Bahnhof wieder verlässt (auf keinen Fall später!!!), auf die Kameraposition "Fluegelsignal" umschalten und das Flügelsignal wieder auf Halt stellen.

Es wird kein neuer Zug mehr aus dem virtuellen Depot ausfahren. Aber man muss abwarten, bis der Zug am Horizont verschwunden ist, bevor man die Anlage abspeichert (vorzugsweise unter einem anderen Namen).

Einen Zug löschen

Einen Zug kann man jederzeit löschen, in dem man den Vollbildmodus mit der Taste F4 verlässt, auf Handbetrieb umschaltet und die P(ause)-Taste drückt. Nun kann man den Zug mit der linken Maustaste markieren und mit der rechten Maustaste im Kontextmenu "Zuggarnitur löschen" anklicken.

Nachdem der Zug entfernt wurde, schaltet man wieder auf den Automatikbetrieb um, drückt wieder die P(ause)-Taste und wählt die Kameraeinstellung "Fluegelsignal" aus. Nachdem man das Flügelsignal auf Halt gestellt hat, sollte man die Anlage speichern (vorzugsweise unter einem anderen Namen).

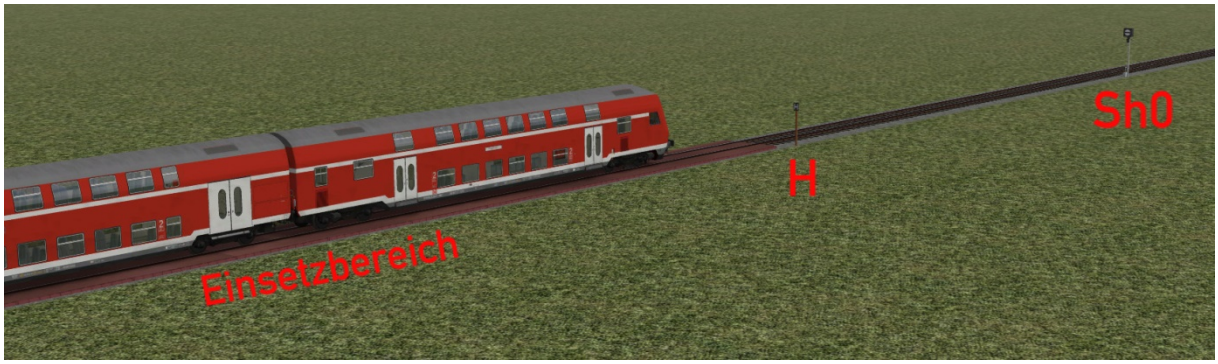
Nun kann man entweder den Betrieb beenden oder das Flügelsignal wieder auf Fahrt stellen und das Spiel geht weiter.

Einen neuen Zug einsetzen

Einen neuen Zug einsetzen kann man entweder bevor man die Anlage startet oder nachdem man sie ordnungsgemäß beendet hat. Das Flügelsignal steht dann auf Halt.

Man verlässt (wenn nicht bereits geschehen) den Vollbildmodus (F4) und wechselt auf die Kameraposition "Einsetzgleis".

Es ist ratsam als Erstes den Lua-Editor aufzurufen und den TagText für den neuen Zug einzutragen. Hierzu scrollt man im Lua-Editor nach unten bis hinter die Funktion EEP-Main(). Innerhalb der Funktion *Setze TagTextNeuerZug()* gibt es 2 Zeilen, die bis auf die Kommentarzeichen nur aus ###### bestehen. Dazwischen trägt man bei jeder **local** definierten Variablen den entsprechenden neuen Wert ein. (Als Beispiele sind die Script-Auszüge für die eingesetzten Züge in der Datei *Zugdaten.lua* enthalten.) Danach klickt man im Lua-Editor auf die Schaltfläche Script neu laden.



Nun kann man links von der Haltsignaltafel innerhalb des roten Gleisbereiches den neuen Zug einsetzen. Wenn das gesamte Rollmaterial aufgesetzt ist, setzt man entweder den Zug mit dem Handregler in Bewegung oder gibt ihm eine Geschwindigkeit im entsprechenden Feld vor. Dies muss nicht die "Fahrplangeschwindigkeit" sein. Hauptsache der Zugverband setzt sich Richtung

Haltesignal in Bewegung. Gegebenenfalls ist die Richtung zu ändern. Bei neu eingesetzten Zügen sind die Lokführer meistens noch so mit dem Lesen der Fahrinstruktionen beschäftigt, dass sie die Haltsignaltafel überfahren. Da schauen wir mal gnädig drüber hinweg, denn der Zug wird vom anschließenden Rangiersignal zum Halten gezwungen.

Nach Überfahren der Haltsignaltafel erscheint eine Warnung mit der Frage, ob man den TagText im Lua-Script geändert hat. Falls nicht, sollte man das spätestens jetzt tun. Die letzte Zeile der Warnung ignorieren wir, denn wir wollen ja den TagText eintragen.

Danach stellt man das mechanische Rangiersignal auf Fahrt. Der neue Zug setzt sich in Bewegung und nachdem der Lokführer die neuen Instruktionen fertig gelesen hat, gibt er fröhlich einen Pfeifton ab.

Nun heißt es aber noch etwas warten, bis der Zug im virtuellen Depot angekommen ist. In dieser Zeit wechselt die Kameraposition zweimal automatisch. Erst nachdem beim zweiten Wechsel auf die Position "Fluegelsignal" umgeschaltet wurde, kann man (und sollte auch) die Anlage (unter neuem Namen) speichern. Danach kann man den Betrieb beenden oder - in dem man das Flügelsignal wieder auf Fahrt stellt - neu beginnen.

Nur den Tag-Text eines Zuges ändern

Um nur den Tag-Text eines Zuges zu ändern, muss man nicht zuerst den Zug löschen und ihn dann neu einsetzen, sondern man leitet ihn auf das Einsetzgleis um und das geht folgendermaßen.

Wenn der entsprechende Zug auf der Hinfahrt (!) in den Bahnhof einfährt (also in Gleis 1) wird automatisch auf die Kameraposition "Gleis 1" umgeschaltet (s.Bild unten). Hierin ist das Flügelsignal neben dem Bahnhof gut zu sehen und man kann es schnell wieder auf Halt stellen.

Alternativ kann man jederzeit, bevor der Zug am Ausfahrsignal von Gleis 1 vorbeifährt (auf keinen Fall später!!!), auf die Kameraposition "Fluegelsignal" umschalten und das Flügelsignal wieder auf Halt stellen.

Automatisch wird die Weiche hinter dem Bahnhof umgestellt, der Zug fährt auf das Einsetzgleis und kommt vor der Haltsignaltafel zum Stillstand. Nun ändert man den Tag-Text wie unter *Einen neuen Zug einsetzen* beschrieben und lädt das Script neu.

Durch Betätigung der Haltsignaltafel setzt man danach den Zug in Bewegung. Auch hier erscheint die Erinnerung an die TagText-Eingabe (und die letzte Zeile der Warnung ignorieren wir auch diesmal). Nach Stellung des mechanischen Rangiersignals auf Fahrt wird der neue TagText in den Zugverband eingetragen und der Lokführer gibt einen Pfeifton ab. Wenn der Zug im virtuellen Depot angekommen ist, wird automatisch auf die Kameraposition "Fluegelsignal" umgestellt. Nun kann man wie gehabt speichern und durch Stellung auf Fahrt weitermachen.

Versehentlich einen Zug auf's Einsetzgleis geschickt?

Keine Panik! Man kann getrost die Haltsignaltafel im Einsetzgleis auf Fahrt stellen. Aber nur die! Nach Passieren der Haltsignaltafel erscheint wie gehabt die Warnung. Anstatt den Lua-Editor zu öffnen, machen wir das, was in der letzten Zeile steht. Wir wechseln in die Kameraposition "Fluegelsignal" und stellen dies auf Fahrt. Anschließend wechseln wir wieder zurück auf die Kameraposition "Einsetzgleis" und stellen das mechanische Rangiersignal auf Fahrt. Nun fährt der Zug ins virtuelle Depot ohne einen TagText einzutragen. Und da der Lokführer keine neuen Instruktionen hat, gibt er auch keinen Pfeifton ab. Man muss auch nicht mehr warten, bis der Zug im virtuellen Depot angekommen ist. Der nächste Zug fährt automatisch aus dem virtuellen Depot nach Gleis 1 des Bahnhofs.

Nachwort

Ich wünsche allen "Eine gute Fahrt" und viel Spaß beim Studieren des Lua-Codes. Vielleicht fällt dem ein oder anderen eine andere Anwendungsmöglichkeit für string.gsub in EEP ein und schreibt darüber im MEF.

Fried-liche Grüße