

Das Flammenrätzel- Tutorial

(Version 1.0 – 10.05.2018)

für

RPG Maker MV & RPG Maker VX Ace

Für kommerzielle und nicht-kommerzielle
Verwendung im RPG Maker.

Jedoch bitte ich um Credits inkl. Webseitenlink:

Credits:

Game Alchemists
<http://gamealchemists.com>

Dieses Tutorial darf weitergegeben und vervielfältigt werden.
Eine Editierung oder Ausgabe ohne Credits bzw.
Änderung des Urhebernemens ist nicht gestattet.



COPYRIGHT BY GAME ALCHEMISTS

Vorwort

Das Tutorial ist für den *RPG Maker VX Ace* geschrieben, kann aber ohne Probleme für den *RPG Maker MV* adaptiert werden. Lediglich ein anderes Licht- Plug-in wird gebraucht. Ich empfehle hier das **Terrax Plugins - Lighting system**.

Weitere Tutorials und Fragen?

Habt ihr Interesse an weiteren Tutorials, werdet ihr hier sicher fündig: <http://gamealchemists.com>

Habt ihr Fragen zum Tutorial? Dann meldet euch einfach im Forum: <http://gamealchemists.com/forum>

Ihr habt Ideen zu Minispielen und wisst nicht, wie ihr sie umsetzen sollt oder sucht ein exclusives Minispiel? Dann schreibt mir einfach eine E-Mail: <http://gamealchemists.com/kontakt/>

Einleitung



Wo ist Laurenz hier nur gelandet?
Mitten in einem Dungeon scheint es nicht weiter zu gehen, die Tür ist versperrt!

Was nun?

Schauen wir uns doch einmal kurz um.
Links und rechts befinden sich je zwei Feuerschalen auf dem Boden.
Zudem vier blaue und ein roter Schalter. Stellen wir uns doch einmal auf einen der blauen Schalter.



Na so was! Die Schalter scheinen die Luftzufuhr zu den Flammen zu regulieren.
Aber noch etwas, jedes Mal, wenn man auf einem Schalter steht, verändert sich das Flammenbild.
Der rote Schalter dient dazu die Flammen zu löschen.

Um was geht es in diesem Tutorial?

In diesem Tutorial befassen wir uns mit einer anderen Art von Schalterrätsel. Wir haben vier Feuerschalen vor uns. Jeder blaue Schalter verkleinert oder vergrößert das Feuer von einer oder mehreren Feuerschalen. Das Ziel ist es, heraus zu finden, welche Schalter man drücken muss, um alle Flammen auf dieselbe Größe zu bekommen.

Wir benötigen

1. Gute Laune
2. Das Script: CSCA Light Effects von Casper Gaming
3. Eine Hand voll Events
4. Ein paar Event-Kenntnisse

Schwierigkeitsgrad: ★★☆☆☆

Benötigtes Script

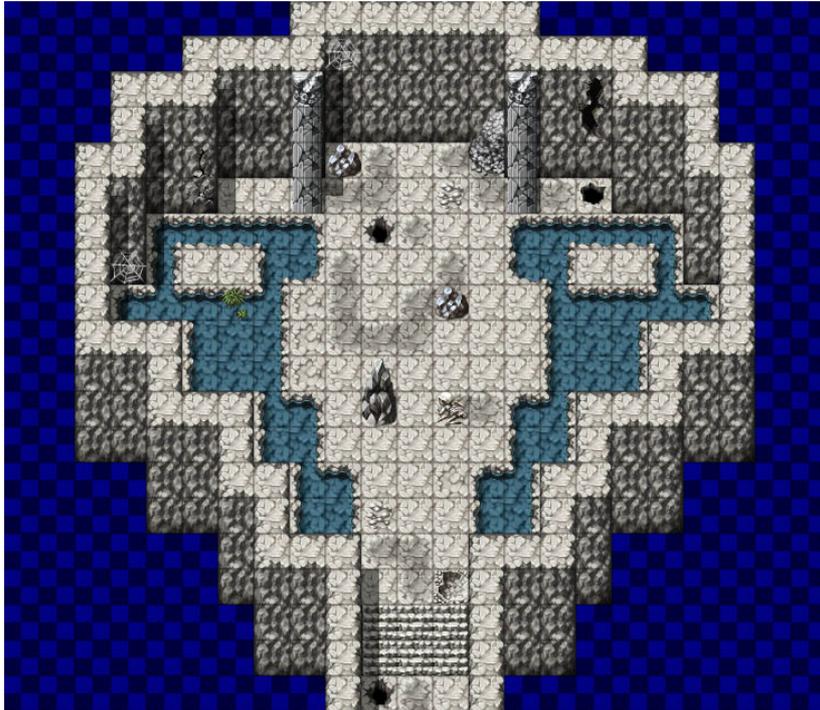
Ladet euch zuerst den Script für die Licht Effekte herunter.

Link zum Script CSCA Light Effects von Casper Gaming: <https://www.rpgmakercentral.com/topic/8589-csca-light-effects/>

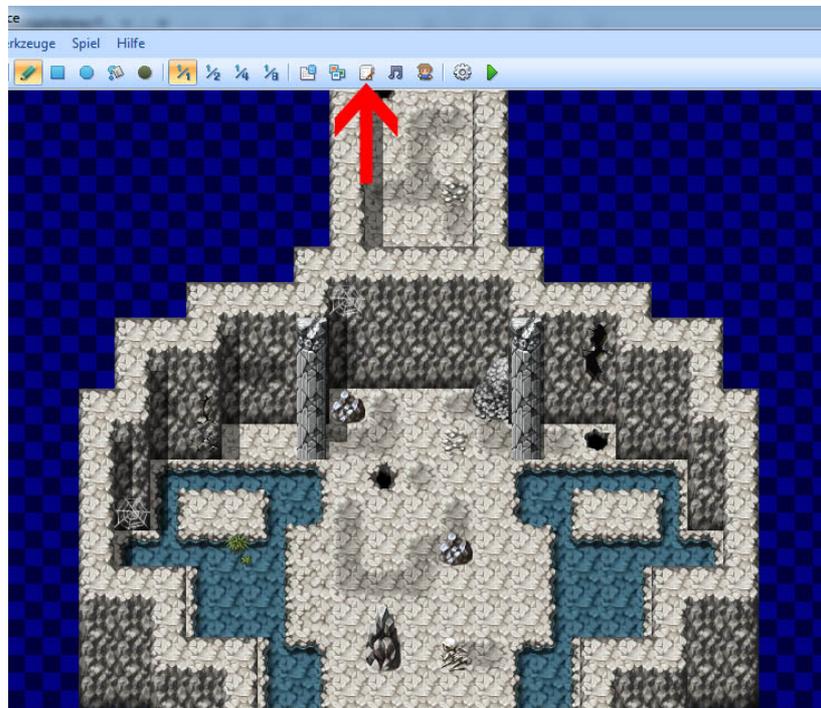
🚨 Wichtig: Die beiliegende Licht Grafikdatei einbinden unter: \Name eures Projekts\Graphics\Pictures

Das Script und die ersten Lampen

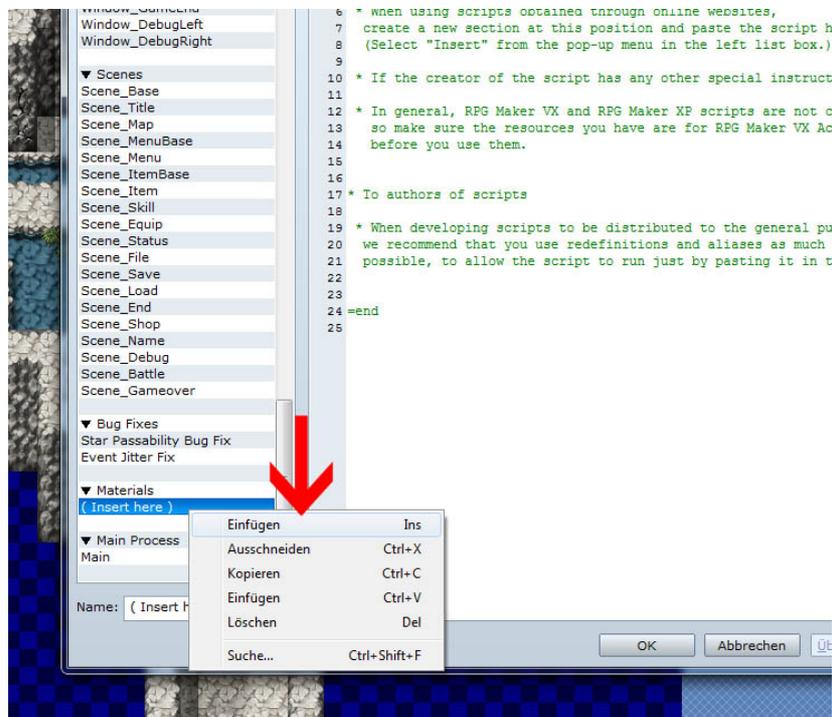
Das Erste, was wir brauchen, ist ein Dungeon, ob kompliziert, ob einfach. Es ist euch überlassen. Hier seht ihr meinen Dungeon in der Grundgestalt. Ohne Events und all den Schnickschnack:



Das Erste, was wir nun tun, ist den Script einzubauen. Dazu geht ihr auf das kleine Symbol in der Symbolleiste, was ich euch markiert habe. Es nennt sich **Script Editor** und ist ebenfalls über die Taste **F11** zu erreichen.



Es öffnet sich ein neues Fenster. Lasst euch jetzt nicht gleich abschrecken, ich erkläre alles Schritt für Schritt! Wir scrollen ganz nach unten, bis wir auf der linken Seite den Eintrag „**Materials**“ finden. Hier klicken wir mit der rechten Maustaste auf „**Insert here**“ ein kleines Fenster öffnet sich, wo wir Einfügen auswählen.



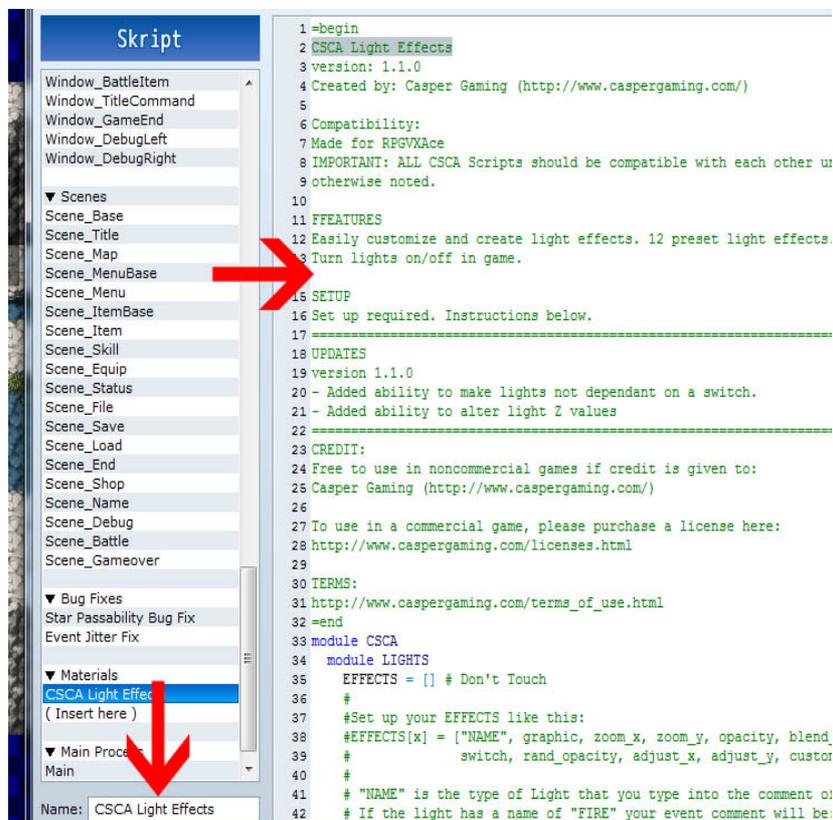
Zwischen „Materials“ und „Insert here“ wird nun ein Leerbereich geschaffen. Etwas weiter unten findet ihr den Eintrag „Name:“ und ein kleines Feld rechts daneben. Hier wird der Script benannt. Wie ihr ihn nennt ist egal, es hat keine Auswirkungen auf den Script und dient nur zur Übersicht. Der Script, den ich euch in einem anderen Reiter hier im Tutorial schon eingetragen habe, wird nun einfach wie ein normales Textdokument **komplett** kopiert. Klickt in den rechten großen weißen Bereich des Scriptfensters und fügt den Text ein. Wahlweise über „STRG+V“ oder über die rechte Maustaste und da auf Einfügen gehen.

➔ Zur Kontrolle, das Script fängt an mit:

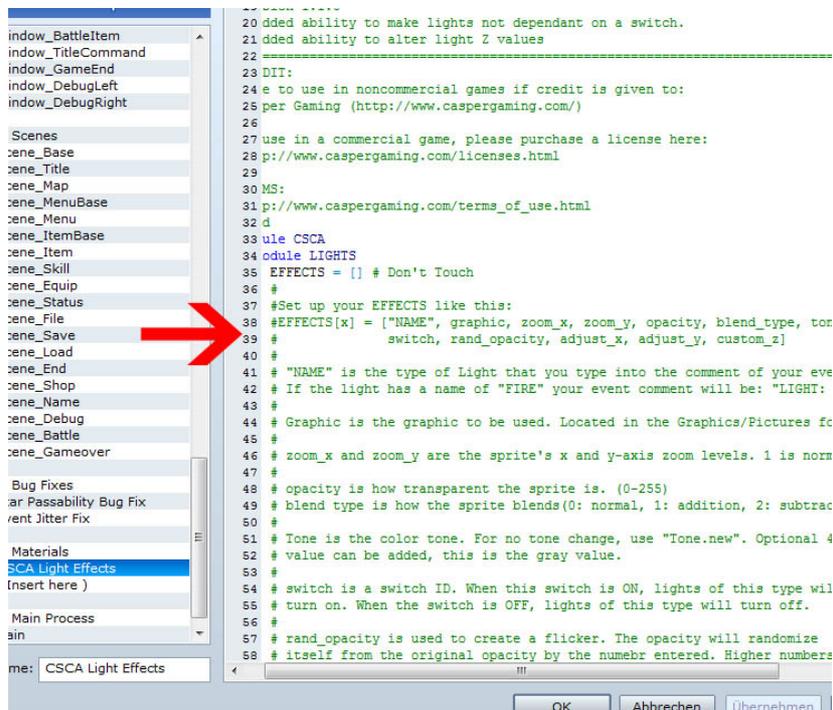
=begin
CSCA Light Effects

Und endet mit:

end
end



Bisher war es ja noch ganz einfach oder? Zwischendurch klicken wir unten rechts auf **Übernehmen**, um unseren bisherigen Fortschritt im Script Editor zu speichern. Nun wird es aber schwieriger. Ich verliere ein paar Worte über den Script. Dieser ist eigentlich sehr einfach gestrickt. Er arbeitet rein über Schalter. In Zeile 38 und 39 wird der Aufbau des Scripts erklärt, den wir ab Zeile 88 einstellen können.



Wie schon im oberen Bild zu sehen, haben wir mehrere Einstellmöglichkeiten die der Reihe nach wären:

- name**
Der Name des Lichteffectes, den wir später aufrufen, z. B. „GROUND1“ oder „LIGHT2“.

graphic

Der Name der Grafikdatei für den Lichtkreis. Bitte noch einmal unter Script nachsehen. Da habe ich erklärt, wie ihr die Grafikdatei eintragen müsst. Der Name hier muss mit dem Namen der Grafikdatei identisch sein.

zoom x

Die Größe des Lichtkreises auf der X-Achse (horizontal). Umso größer die Zahl, umso größer der Lichtkreis. Mehr als 6 habe ich bisher nicht genutzt.

zoom y

Die Größe des Lichtkreises auf der Y-Achse (vertikal). Umso größer die Zahl, umso größer der Lichtkreis. Mehr als 6 habe ich bisher nicht genutzt.

opacity

Die Deckkraft des Lichtes. Umso kleiner der Wert, umso geringer die Deckkraft. 255 ist der maximale Wert, dabei ist die Deckkraft 100%.

blend type

Für was der Blend Typ zuständig ist, kann ich euch nicht sagen. Noch nie gebraucht, noch nie genutzt.

ton

Drei Zahlen, hier stellt ihr die RPG-Farben eures Kreises ein.

⚠ Diese Farben stimmen im Übrigen nicht mit normalen RPG-Farben überein! Wenn ihr eine Farbe sucht, öffnet euch am Besten ein neues Event. Geht da auf **Bildschirm färben** und sucht euch eine Farbe aus. Notiert die drei oder vier Werte und tragt sie dann im Script ein.

switch

Etwas sehr Wichtiges! Dies ist der Schalter, mit dem das Licht ein- und ausgeschaltet werden kann, z. B. ist hier die Zahl 23 der Schalter Nr. 23 in eurem Event. Ist der Schalter auf AN, ist das Licht an. Ist der Schalter auf AUS, ist das Licht aus.

⚠ Tipp: Ihr könnt mehreren Lichtern den gleichen Schalter geben. Es braucht also nicht jedes Licht einen eigenen Schalter.

rand opacity, adjust X, adjust y und custom z

Diese 4 Werte fasse ich einmal zusammen. Zum einen kann ich euch nicht sagen, für was sie da sind, zum anderen noch nie gebraucht.

Effects

Ganz am Anfang jeder Zeile findet ihr den Namen **Effects** und eine Zahl dahinter. Diese Zahl muss immer fortlaufend sein. Eine Zahl darf nie doppelt sein!

Noch eine kleine Erklärung im Allgemeinen:

Jede Zeile beinhaltet eine Lichtquelle mit verschiedenen Eigenschaften. Oben findet ihr eine Erklärung für die vorgegebenen Lichter.

EFFEK1 – LIGHT1 ist ein kleines weißes Licht.

EFFEKT9 – GROUND3 ist ein kleines rotes Licht.

```
66 #
67 # Effects have already been set up for you to be similar to Thomas Edison
68 # VX. You can customize them or create new effects yourself, too!
69 # The preset light effects are:
70 # GROUND1 - Medium steady white light.
71 # GROUND2 - Medium white light with slight flicker.
72 # GROUND3 - Small steady red light.
73 # GROUND4 - Medium steady green light.
74 # GROUND5 - Medium steady blue light.
75 # FIRE - Large red light with slight flicker.
76 # LIGHT1 - Small steady white light.
77 # LIGHT2 - X-Large steady white light.
78 # LIGHT3 - Small white light with slight flicker.
79 # TORCH1 - X-Large red light with heavy flicker.
80 # TORCH2 - X-Large red light with slight flicker.
81 # TORCH3 - Large white light with slight flicker.
82 #
83 # TITLEFIRE - Same as FIRE, but not dependant on a switch (always on) for
84 # use in CSCA Title System (Get that script here: )
85 #
86 # TITLE_GROUND2 - Same as GROUND2, but not dependant on a switch.
87 #
88 EFFECTS[0] = ["FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 1, 10, 0, 0
89 EFFECTS[1] = ["LIGHT1", "Light", 1, 1, 150, 1, Tone.new, 2, 0, 0, 0, 1000]
90 EFFECTS[2] = ["LIGHT2", "Light", 6, 6, 150, 1, Tone.new, 3, 0, 0, 0, 1000]
91 EFFECTS[3] = ["LIGHT3", "Light", 1, 1, 150, 1, Tone.new, 4, 10, 0, 0, 1000]
92 EFFECTS[4] = ["TORCH1", "Light", 6, 6, 150, 1, Tone.new(255,-100,-255), 5, 30, 0,
93 EFFECTS[5] = ["TORCH2", "Light", 6, 6, 150, 1, Tone.new(255,-100,-255), 6, 10, 0,
94 EFFECTS[6] = ["TORCH3", "Light", 3, 3, 100, 1, Tone.new, 7, 10, 0, 0, 1000]
95 EFFECTS[7] = ["GROUND1", "Light", 2, 2, 100, 1, Tone.new, 8, 0, 0, 0, 1000]
96 EFFECTS[8] = ["GROUND2", "Light", 2, 2, 100, 1, Tone.new, 9, 10, 0, 0, 1000]
97 EFFECTS[9] = ["GROUND3", "Light", 2, 2, 100, 1, Tone.new(255,-255,-255,255), 10,
98 EFFECTS[10] = ["GROUND4", "Light", 2, 2, 100, 1, Tone.new(-255,255,-255,100), 11,
99 EFFECTS[11] = ["GROUND5", "Light", 2, 2, 100, 1, Tone.new(-255,255,255,100), 12,
100
101 EFFECTS[12] = ["TITLE_FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 0, 1
102 EFFECTS[13] = ["TITLE_GROUND2", "Light", 1, 1, 150, 1, Tone.new, 0, 10, 0, 0, 5]
103 #
104 # Script Calls:
105 #
106 # To refresh lights(if an event was erased or changed pages and should no
```

Nun werden wir an diesen Werten Mal ein klein wenig herumspielen. Eigentlich nur an zwei Werten, wir wollen ja klein anfangen.

➡ Wir gehen in folgende Zeile: „EFFECTS[5] = [„TORCH2“, „Light“, 6, 6, 150, 1, Tone.new(255,-100,-255), 6, 10, 0, 0, 1000]“

Das Licht ist uns zu groß mit **X=6** und **Y=6**. Darum ändern wir dies in **X=4** und **Y=4**.

Zudem ist uns der Schalter 6 nicht genehm. Ich möchte, dass Schalter 3 für dieses Licht zuständig ist.

Drum wird die Zahl auf 3 geändert.

➡ Nun müsste eure Zeile so aussehen: „EFFECTS[5] = [„TORCH2“, „Light“, 4, 4, 150, 1, Tone.new(255,-100,-255), 3, 10, 0, 0, 1000]“

Wir merken uns den Namen „TORCH2“, den Namen der Grafik „Light“ und die Nummer des Schalters „3“.

Wir gehen rechts unten auf **OK** und das Fenster wird geschlossen.

```
pt
mand
d
ft
ght
ug Fix
S
# GROUND1 - Medium steady white light.
# GROUND2 - Medium white light with slight flicker.
# GROUND3 - Small steady red light.
# GROUND4 - Medium steady green light.
# GROUND5 - Medium steady blue light.
# FIRE - Large red light with slight flicker.
# LIGHT1 - Small steady white light.
# LIGHT2 - X-Large steady white light.
# LIGHT3 - Small white light with slight flicker.
# TORCH1 - X-Large red light with heavy flicker.
# TORCH2 - X-Large red light with slight flicker.
# TORCH3 - Large white light with slight flicker.
#
# TITLEFIRE - Same as FIRE, but not dependant on a switch (always on) for
# use in CSCA Title System (Get that script here: )
#
# TITLE_GROUND2 - Same as GROUND2, but not dependant on a switch.
#
EFFECTS[0] = ["FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 1, 10, 0
EFFECTS[1] = ["LIGHT1", "Light", 1, 1, 150, 1, Tone.new, 2, 0, 0, 0, 1000]
EFFECTS[2] = ["LIGHT2", "Light", 6, 6, 150, 1, Tone.new, 6, 0, 0, 0, 1000]
EFFECTS[3] = ["LIGHT3", "Light", 1, 1, 150, 1, Tone.new, 4, 10, 0, 0, 1000]
EFFECTS[4] = ["TORCH1", "Light", 6, 6, 150, 1, Tone.new(255,-100,-255), 5, 30,
EFFECTS[5] = ["TORCH2", "Light", 4, 4, 150, 1, Tone.new(255,-100,-255), 3, 10,
EFFECTS[6] = ["TORCH3", "Light", 3, 3, 100, 1, Tone.new, 7, 10, 0, 0, 1000]
EFFECTS[7] = ["GROUND1", "Light", 2, 2, 100, 1, Tone.new, 8, 0, 0, 0, 1000]
EFFECTS[8] = ["GROUND2", "Light", 2, 2, 100, 1, Tone.new, 9, 10, 0, 0, 1000]
EFFECTS[9] = ["GROUND3", "Light", 2, 2, 100, 1, Tone.new(255,-255,-255,255), 1
EFFECTS[10] = ["GROUND4", "Light", 2, 2, 100, 1, Tone.new(-255,255,-255,100), 1
EFFECTS[11] = ["GROUND5", "Light", 2, 2, 100, 1, Tone.new(-255,255,255,100), 1
#
EFFECTS[12] = ["TITLE_FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 0
EFFECTS[13] = ["TITLE_GROUND2", "Light", 1, 1, 150, 1, Tone.new, 0, 10, 0, 0,
#
# Script Calls:
#
# To refresh lights(if an event was erased or changed pages and should no
# longer be lit up), make the following script call:
# $game_map.cscs_light_need_refresh = true
#
# To dispose specific lights, use the following script call:
```

Geschafft! Nun wird es wieder etwas einfacher. Wir setzen uns ein Event in den Raum und gehen bei der Grafik des Events unter „!Other2“ und wählen uns die Fackel aus.

Als Optionen stellen wir ein:

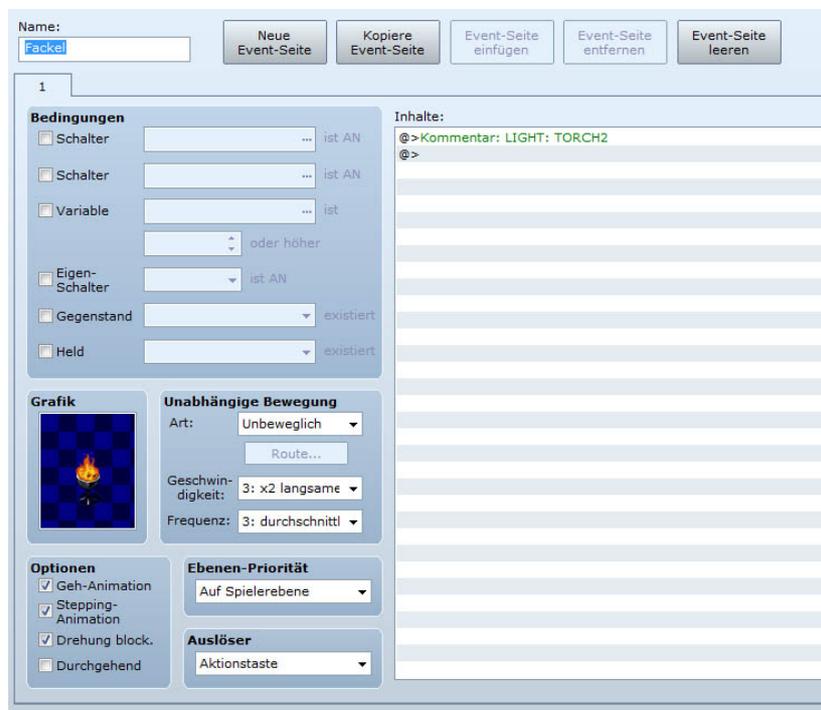
- ➔ Stepping Animation
- ➔ Drehung Blockieren

Ansonsten brauchen wir nichts ändern. Im Inhalt des Events legen wir uns einen Kommentar an. Diesen findet ihr auf der ersten Seite für Eventbefehle ganz links unten.

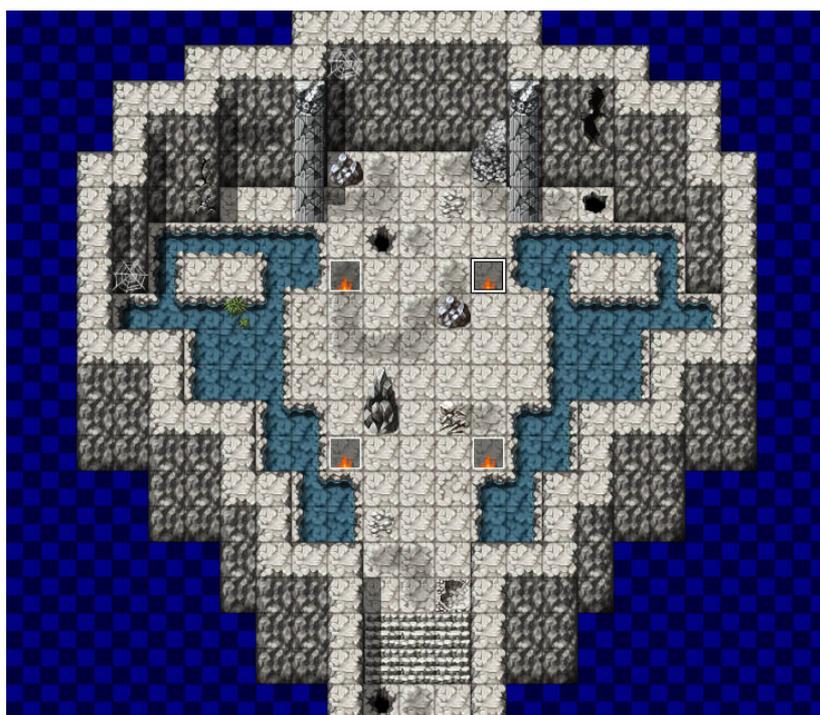
Hier schreiben wir nun folgendes hinein:

➔ LIGHT: TORCH2

Müsste euch ja bekannt vorkommen, oder? Es stammt aus der Zeile, die wir uns gemerkt haben. „Light“ ist die Grafikdatei und „TORCH2“ die Einstellung der Lampe. Das war auch schon der gesamte Zauber der Event-Datei für ein Licht. Wir haben so zu sagen die Glühbirne in die Fassung geschraubt.



Klickt rechts unten auf **OK** und kopiert dieses Event noch ein paar Mal, um den Dungeon etwas aus zu leuchten. Bei mir sieht es nun so aus:



Weiter geht es! Wenn es in einen Dungeon geht, werdet ihr ja sicherlich den Befehl „**Bildschirm färben**“ benutzen, damit es im Dungeon nicht taghell ist. Damit es etwas schöner aussieht und der Übergang von außen zum Dungeon fließend ist, macht man dies am besten zusammen mit dem Teleport. Dieses Event wird außerhalb der jetzigen Map gesetzt, von wo ihr her kommt. Ich nehme an ein schöner taghell erleuchteter Wald.

- ➔ Die erste Zeile ist der Standard-Sound
- ➔ Die zweite Zeile blendet auf komplett schwarz ab
- ➔ Die dritte Zeile teleportiert den Spieler in den Dungeon

⚠ Achtung: Bei Überblendung bei der Teleportersteuerung „**Kein**“ einstellen.

Somit hätten wir jetzt den Teleport zum Raum geschaffen. Würde man jetzt den Teleport nutzen, wäre es stockfinster. Ist ja auch logisch. Wir haben gesagt die Bildschirmfarbe auf komplett schwarz stellen.

Darum brauchen wir jetzt das erste Event in unserem Dungeon!

Im Übrigen werde ich jetzt nicht jeden Mausklick einzeln erklären. Das würde den Rahmen bei Weitem sprengen. Ich werde kleine Events im Ganzen präsentieren und erklären. Größere Events in Stücken aufzeigen und erklären.

Wir brauchen, sobald der Spieler auf die Karte kommt, ein Event, was sofort reagiert. Deshalb wird es hier ein Autostart-Event.

Das Erste, was geschehen muss, ist die Färbung des Bildschirms.

Dafür nutzen wir folgende Werte:

➡ Ton: **-100 -100 -100 0**

➡ Wartezeit: **20 Frames**

Bis hier alles normal.

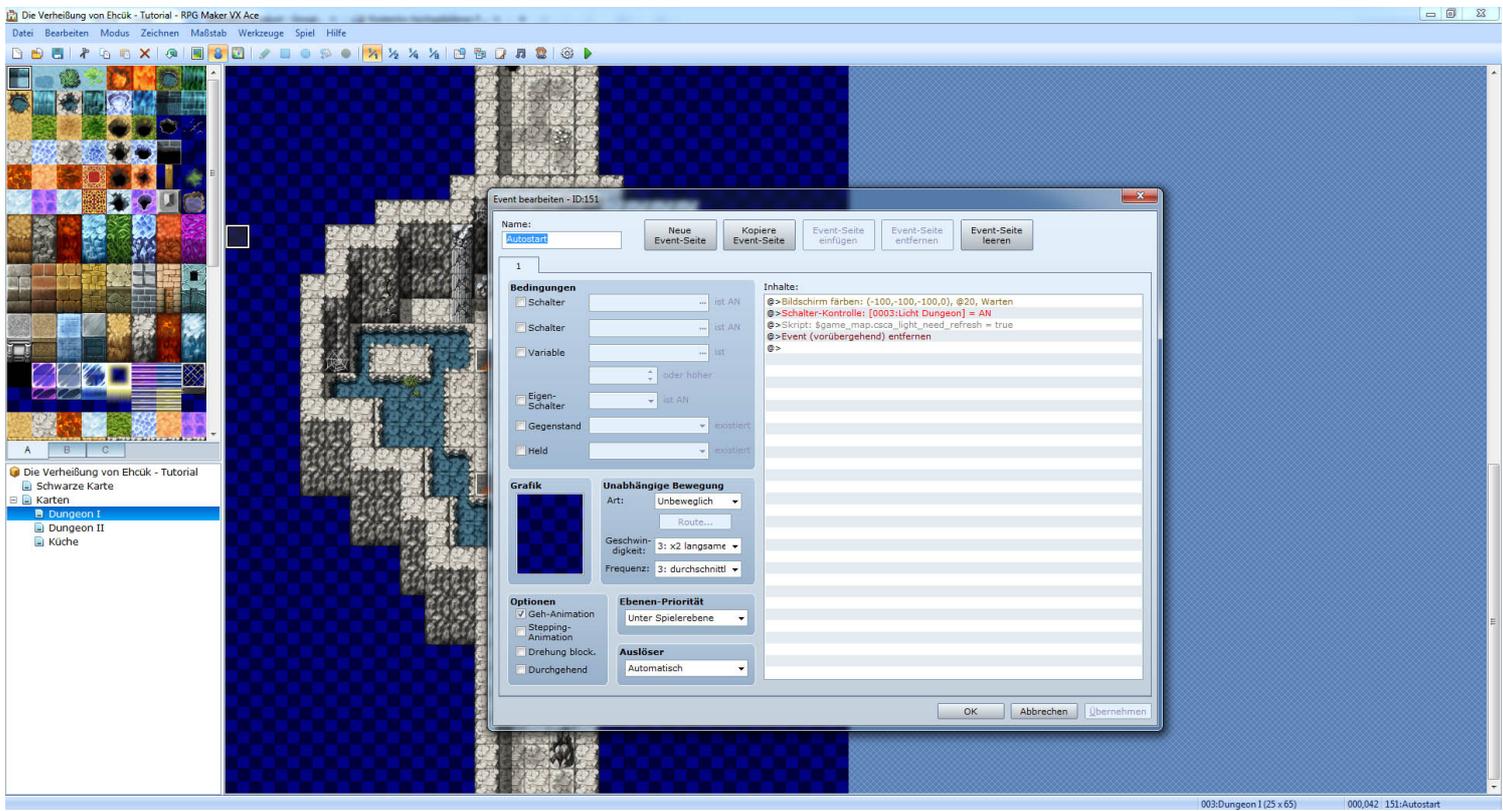
Jetzt schalten wir die Lichter der Fackeln an. Wie wir uns erinnern, ist dafür **Schalter 3** zuständig. Darum **Schalter 3** benannt und auf **AN** gestellt.

Das Nächste ist ein Scriptcode, den ihr eintragen müsst. Dieser ist für Folgendes zuständig:

Ein Event kann, wie ihr wisst, mehrere Seiten haben (Reiter). Vorhin haben wir ein Licht mit einen Kommentarbefehl aufgerufen. Nun könnte ich auf jede Eventseite einen Lichtkommentar machen. Damit wenn wir die verschiedenen Eventseiten aufrufen auch das richtige Licht angezeigt wird, kann man die Lichtquellen über diesen Scriptbefehl aktualisieren.

➡ Der Script Befehl lautet: **`$game_map.cscs_light_need_refresh = true`**

Als Letztes kommt der Befehl „**Event (vorübergehend) entfernen**“. Das Event wird also hier beendet und wird erst wieder neu gestartet, wenn der Spieler die Karte erneut betritt.

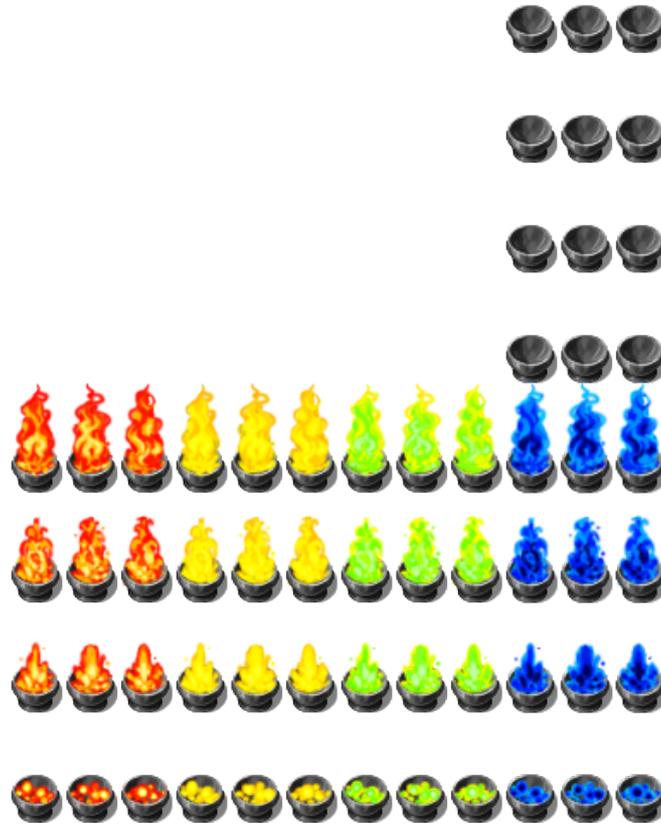


Nun testen wir mal das Ganze. Wenn alles gut gelaufen ist, müsste es im Spiel nun so aussehen.



Die Flammenschalen

Kommen wir zum nächsten Punkt des Tutorials – die Flammenschalen. Sicher werdet ihr am Anfang des Tutorials die Schalen gesehen haben. Diese gibt es so im Maker nicht, darum hier die Schalen als Download:



❗ Die Grafikdatei ist dem Tutorial in einer Zip-Datei beigelegt. Die Datei ist passend für den *RPG Maker VX Ace* und darf nur mit einer *RPG Maker VX Ace* Lizenz verwendet werden!

➡ Diese Grafikdatei müsst ihr einbinden unter: `\Name eures Projekts\Graphics\Charakters`

Fertig? Dann geht es sofort weiter. Auf der Karte habt ihr ja gesehen, dass ich links und rechts zwei kleine Inseln gemacht habe. Wie ihr euch erinnert, stehen da die Feuerschalen. Je zwei links und zwei rechts. Wir werden nun die Feuerschale ganz links als erstes eventen.

Event Flamme 1 – Reiter 1

➡ Wir erstellen uns ein neues Event welches wir „**Flamme 1**“ nennen.

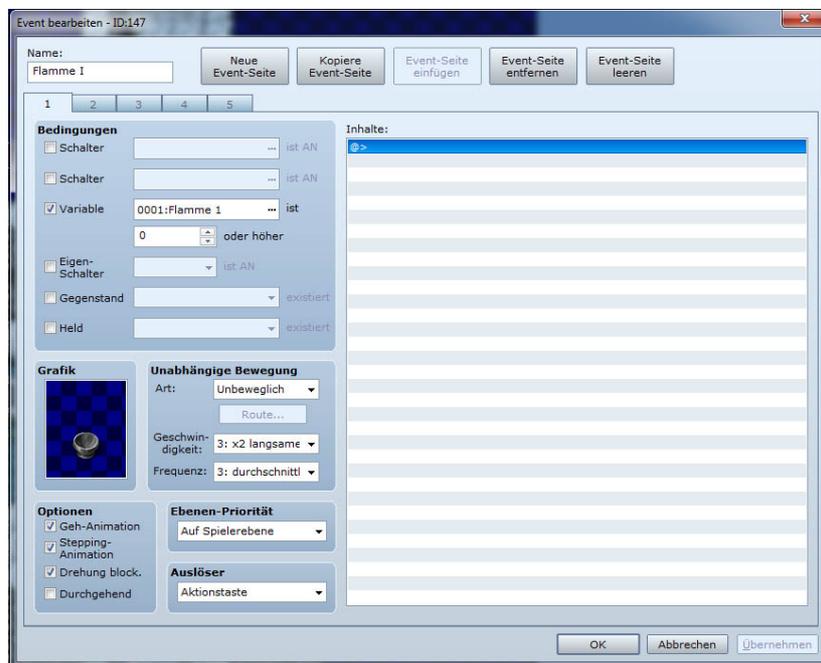
➡ Als Bedingung erstellen wir uns eine neue Variable, die wir „**Flamme 1**“ nennen. Die Größe der Variable ist auf der ersten Reiterseite 0!

➡ Als Grafik nutzen wir die leere Flammenschale aus der Datei, die ihr gerade in euer Projekt eingebaut habt.

Als Optionen stellen wir ein:

➡ **Stepping-Animation**

➡ **Drehung blockieren**



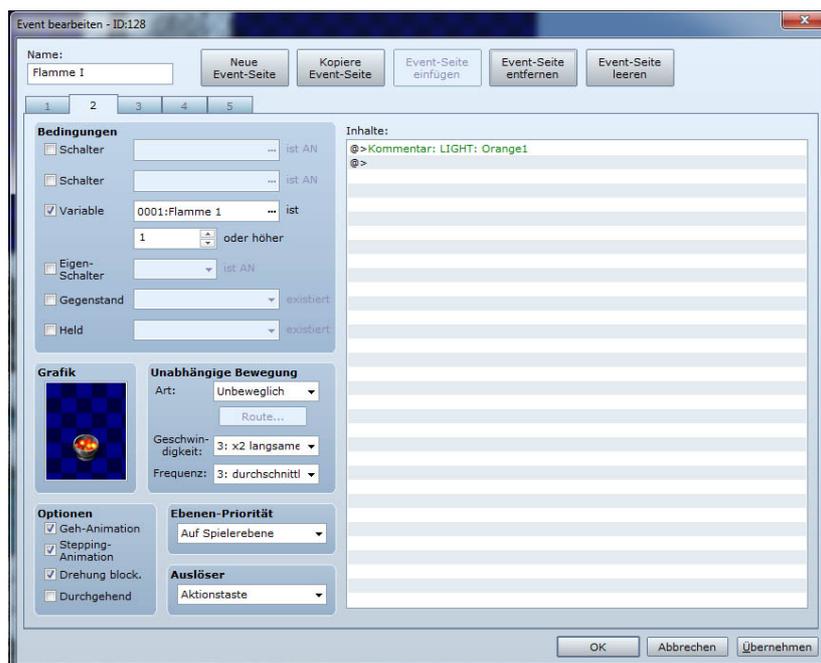
Event Flamme 1 – Reiter 2

Wie ihr schon auf meinem Bild seht, gibt es im Event weitere Reiter. Nehmen wir uns also den zweiten Reiter vor. Am besten, ihr kopiert euch gleich die erste Seite.

- ➔ Als Bedingung setzen wir die Variable, die wir „**Flamme 1**“ nannten, auf die Größe 1.
- ➔ Als Grafik nutzen wir die Flammenschale mit dem kleinen roten Glühen.
- ➔ Als Kommentar setzen wir hier: „**LIGHT: Orange1**“

Als Optionen stellen wir ein:

- ➔ **Stepping-Animation**
- ➔ **Drehung blockieren**

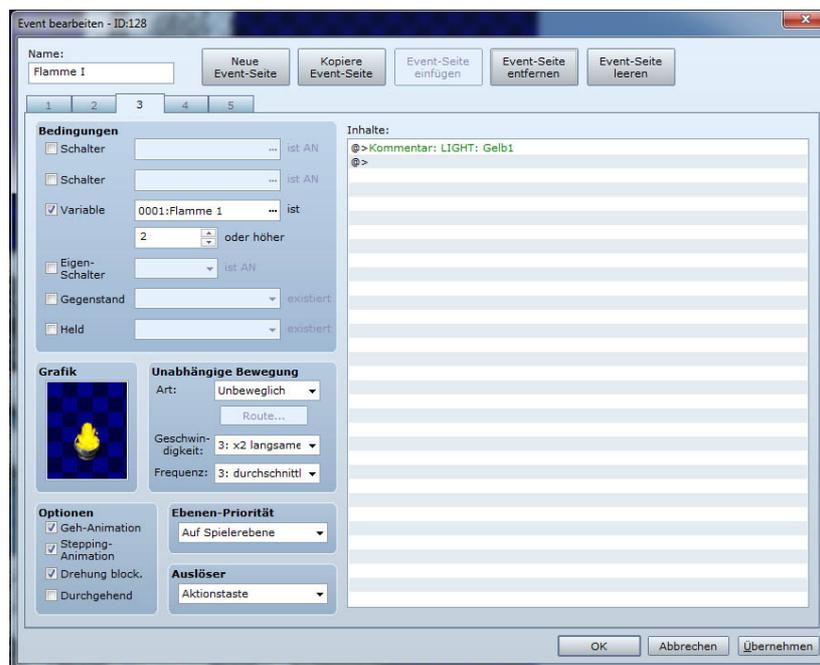


Event Flamme 1 – Reiter 3

- ➔ Als Bedingung setzen wir die Variable, die wir „**Flamme 1**“ nannten, auf die Größe **2**.
- ➔ Als Grafik nutzen wir die Flammenschale mit dem kleinen roten Glühen.
- ➔ Als Kommentar setzen wir hier: „**LIGHT: Gelb1**“

Als Optionen stellen wir ein:

- ➔ **Stepping-Animation**
- ➔ **Drehung blockieren**

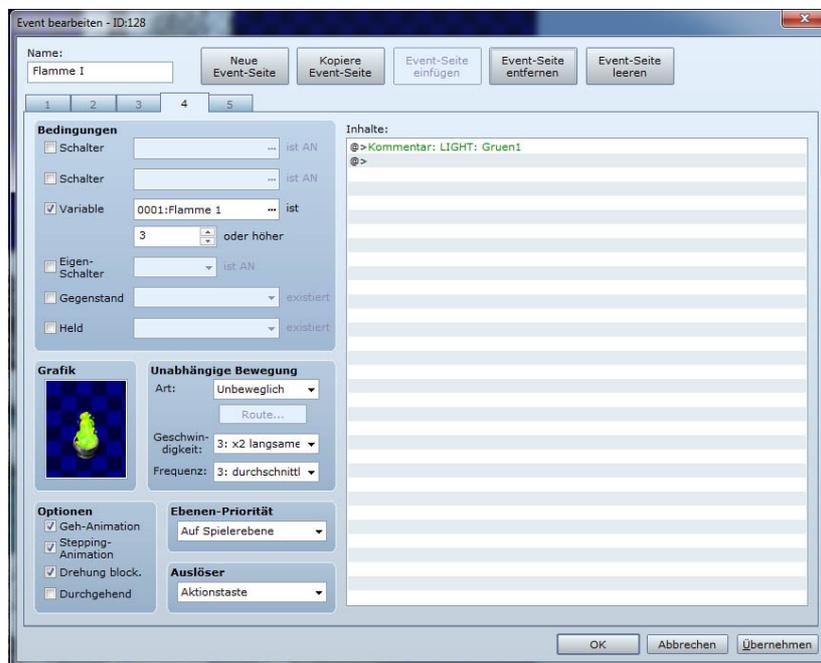


Event Flamme 1 – Reiter 4

- ➔ Als Bedingung setzen wir die Variable, die wir „**Flamme 1**“ nannten, auf die Größe **4**.
- ➔ Als Grafik nutzen wir die Flammenschale mit dem kleinen roten Glühen.
- ➔ Als Kommentar setzen wir hier: „**LIGHT: Gruen1**“

Als Optionen stellen wir ein:

- ➔ **Stepping-Animation**
- ➔ **Drehung blockieren**

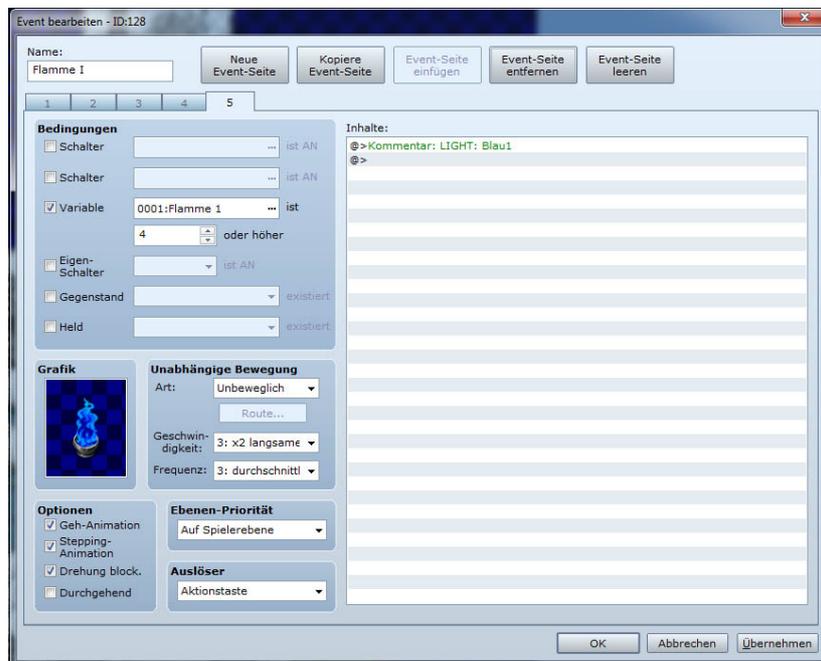


Event Flamme 1 – Reiter 5

- ➔ Als Bedingung setzen wir die Variable, die wir „**Flamme 1**“ nannten, auf die Größe **4**.
- ➔ Als Grafik nutzen wir die Flammenschale mit dem kleinen roten Glühen.
- ➔ Als Kommentar setzen wir hier: „**LIGHT: Blau1**“

Als Optionen stellen wir ein:

- ➔ **Stepping-Animation**
- ➔ **Drehung blockieren**



Damit sind wir hier fertig und klicken auf **OK**.

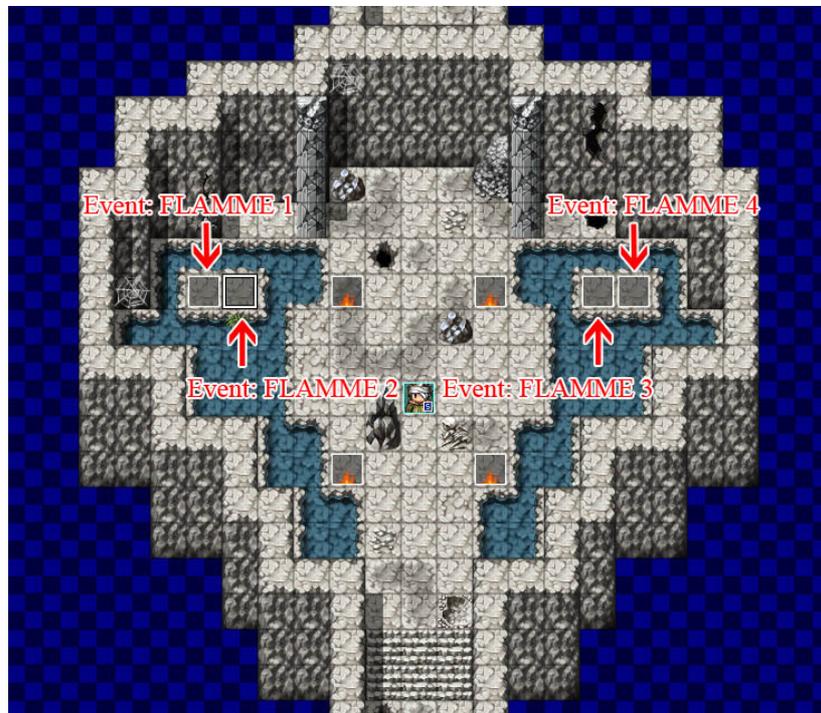
❗ Moment, werden jetzt einige sagen, die Kommentare wurden zwar jetzt eingetragen, aber die waren doch überhaupt nicht im Script drin!

Richtig, dazu aber gleich mehr, es folgt noch ein weiterer Schritt bis dahin.

Wir haben bisher nur ein Flammenevent erzeugt, jedoch brauchen wir noch drei weitere.

Wir haben gerade eben das Event „**Flamme 1**“ ganz links auf der Insel erstellt, das Event rechts daneben nennen wir „**Flamme 2**“.

Auf der rechten Inseln nennen wir die Flamme links „**Flamme 3**“ und rechts daneben „**Flamme 4**“.

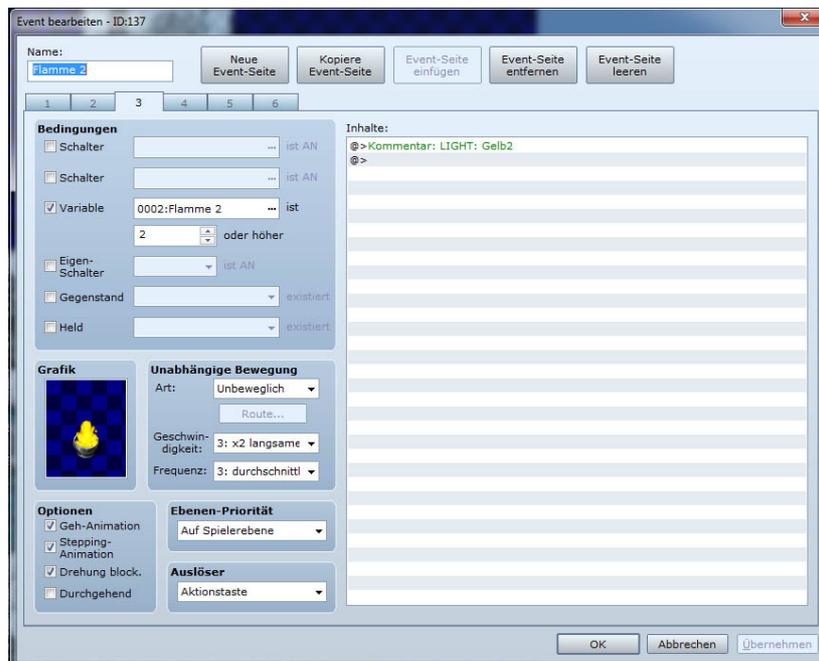


Alle Events an irgendein Platz? Dann geht es sofort weiter, denn es sind an den Events „**Flamme 2**“, „**Flamme 3**“ und „**Flamme 4**“ ein paar wenige Änderungen vor zu nehmen.

Wie ihr wisst, haben wir ja beim Event „**Flamme 1**“ die Bedingung „**Variable Flamme 1**“ eingetragen. Dies müssen wir nun bei den Events „**Flamme 2**“, „**Flamme 3**“ und „**Flamme 4**“ ändern.

- ➔ Wir ändern beim Event „**Flamme 2**“ die Bedingung „**Variable Flamme 1**“ auf „**Variable Flamme 2**“.
- Wir legen also eine neue Variable an. Dies macht ihr in jeder Reiterebene des Events „**Flamme 2**“ so.
- ➔ Die Zahlen bleiben überall erhalten und werden nicht verändert!
- ➔ Ebenfalls sind die Kommentare im Event „**Flamme 2**“ etwas zu verändert. Statt der **1** im Kommentar wird eine **2** eingetragen.

Ein Beispiel:



Ihr ahnt vielleicht schon, was nun kommt:

➔ Dasselbe, was wir nun beim Event „**Flamme 2**“ gemacht haben, wiederholen wir nun bei den Events „**Flamme 3**“ und „**Flamme 4**“. Dabei werden noch zwei weitere Variablen erzeugt, „**Variable Flamme 3**“ und „**Variable Flamme 4**“.

Jetzt kommen wir zu dem Punkt, der vielleicht erst einmal aufhorchen ließ. Wir haben ja in jedem Event auf mehreren Reitern verteilt Lichtkommentare gesetzt, die es nicht gab. Das holen wir jetzt nach. Wir öffnen noch einmal den Script-Editor und gehen noch einmal in den Licht-Script und fügen Folgendes unter den bestehenden Lichtern ein:

```
EFFECTS[14] = ["GROUND6", "Light", 3, 3, 150, 1, Tone.new(255,-100,-255), 37, 10, 0, 0, 1000]
```

```
EFFECTS[15] = ["Orange1", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 4, 10, 0, 0, 1000]
```

```
EFFECTS[16] = ["Gelb1", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 5, 10, 0, 0, 1000]
```

```
EFFECTS[17] = ["Gruen1", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 6, 10, 0, 0, 1000]
```

```
EFFECTS[18] = ["Blau1", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 7, 10, 0, 0, 1000]
```

```
EFFECTS[19] = ["Orange2", "Light", 2, 1, 150, 1, Tone.new(255,-255,-255,255), 8, 10, 0, 0, 1000]
```

```
EFFECTS[20] = ["Gelb2", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 9, 10, 0, 0, 1000]
```

```
EFFECTS[21] = ["Gruen2", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 10, 10, 0, 0, 1000]
```

```
EFFECTS[22] = ["Blau2", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 11, 10, 0, 0, 1000]
```

```
EFFECTS[23] = ["Orange3", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 12, 10, 0, 0, 1000]
```

```
EFFECTS[24] = ["Gelb3", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 13, 10, 0, 0, 1000]
```

```
EFFECTS[25] = ["Gruen3", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 14, 10, 0, 0, 1000]
```

```
EFFECTS[26] = ["Blau3", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 15, 10, 0, 0, 1000]
```

```
EFFECTS[27] = ["Orange4", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 16, 10, 0, 0, 1000]
```

```
EFFECTS[28] = ["Gelb4", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 17, 10, 0, 0, 1000]
```

```
EFFECTS[29] = ["Gruen4", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 18, 10, 0, 0, 1000]
```

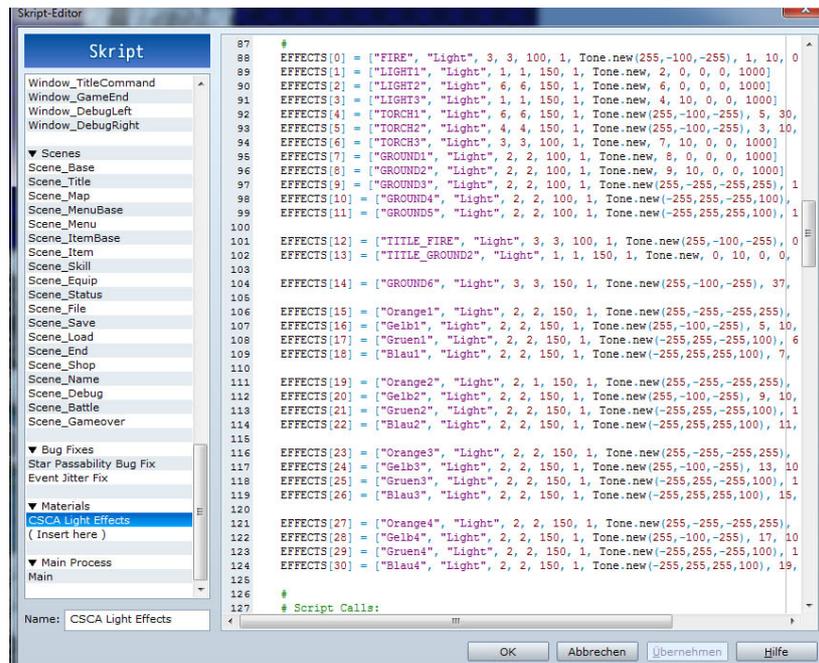
```
EFFECTS[30] = ["Blau4", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 19, 10, 0, 0, 1000]
```

Dies sind die Lichter für die verschiedenen Flammen. Ein blauer Schein für ein blaues Feuer, ein roter Schein für ein rotes Feuer etc.

Was die einzelnen Werte zu bedeuten haben, hatte ich ja schon erzählt, darum rolle ich es nicht noch einmal auf.

❗ Lasst euch nicht von „Effekt 14“ stören, den brauchen wir nicht. Ist noch übrig von meinem testen. Kann es aber auch nicht einfach entfernen, da, wie ich schon sagte, die Effekt-Zahlen fortlaufend sein müssen.

Bei euch müsste es nun so aussehen:



```
Skript-Editor
Skript
Window_TitleCommand
Window_GameEnd
Window_DebugLeft
Window_DebugRight
▼ Scenes
Scene_Base
Scene_Title
Scene_Map
Scene_MenuBase
Scene_Menu
Scene_ItemBase
Scene_Item
Scene_Skill
Scene_Equip
Scene_Status
Scene_File
Scene_Save
Scene_Load
Scene_End
Scene_Shop
Scene_Name
Scene_Debug
Scene_Battle
Scene_Gameover
▼ Bug Fixes
Star Passability Bug Fix
Event Jitter Fix
▼ Materials
CSCA Light Effects
( Insert here )
▼ Main Process
Main
Name: CSCA Light Effects
87 #
88 EFFECTS[0] = ["FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 1, 10, 0
89 EFFECTS[1] = ["LIGHT1", "Light", 1, 1, 150, 1, Tone.new, 2, 0, 0, 0, 1000]
90 EFFECTS[2] = ["LIGHT2", "Light", 6, 6, 150, 1, Tone.new, 6, 0, 0, 0, 1000]
91 EFFECTS[3] = ["LIGHT3", "Light", 1, 1, 150, 1, Tone.new, 4, 10, 0, 0, 1000]
92 EFFECTS[4] = ["TORCH1", "Light", 6, 6, 150, 1, Tone.new(255,-100,-255), 5, 30,
93 EFFECTS[5] = ["TORCH2", "Light", 4, 4, 150, 1, Tone.new(255,-100,-255), 3, 10,
94 EFFECTS[6] = ["TORCH3", "Light", 3, 3, 100, 1, Tone.new, 7, 10, 0, 0, 1000]
95 EFFECTS[7] = ["GROUND1", "Light", 2, 2, 100, 1, Tone.new, 8, 0, 0, 0, 1000]
96 EFFECTS[8] = ["GROUND2", "Light", 2, 2, 100, 1, Tone.new, 9, 10, 0, 0, 1000]
97 EFFECTS[9] = ["GROUND3", "Light", 2, 2, 100, 1, Tone.new(255,-255,-255,255), 1
98 EFFECTS[10] = ["GROUND4", "Light", 2, 2, 100, 1, Tone.new(-255,255,-255,100), 1
99 EFFECTS[11] = ["GROUND5", "Light", 2, 2, 100, 1, Tone.new(-255,255,255,100), 1
100
101 EFFECTS[12] = ["TITLE_FIRE", "Light", 3, 3, 100, 1, Tone.new(255,-100,-255), 0
102 EFFECTS[13] = ["TITLE_GROUND2", "Light", 1, 1, 150, 1, Tone.new, 0, 10, 0, 0,
103
104
105 EFFECTS[14] = ["GROUND6", "Light", 3, 3, 150, 1, Tone.new(255,-100,-255), 37,
106
107 EFFECTS[15] = ["Orange1", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255),
108 EFFECTS[16] = ["Gelb1", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 5, 10,
109 EFFECTS[17] = ["Gruen1", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 6
110 EFFECTS[18] = ["Blau1", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 7,
111
112 EFFECTS[19] = ["Orange2", "Light", 2, 1, 150, 1, Tone.new(255,-255,-255,255),
113 EFFECTS[20] = ["Gelb2", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 9, 10,
114 EFFECTS[21] = ["Gruen2", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 1
115 EFFECTS[22] = ["Blau2", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 11,
116
117 EFFECTS[23] = ["Orange3", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255),
118 EFFECTS[24] = ["Gelb3", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 13, 10
119 EFFECTS[25] = ["Gruen3", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 1
120 EFFECTS[26] = ["Blau3", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 15,
121
122 EFFECTS[27] = ["Orange4", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255),
123 EFFECTS[28] = ["Gelb4", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 17, 10
124 EFFECTS[29] = ["Gruen4", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 1
125 EFFECTS[30] = ["Blau4", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 19,
126
127
128 # Script Calls:
```

Speichert und beendet den Skript-Editor.

Die Schalter

Kommen wir zum nächsten Punkt – die Schalter. Wie ihr wisst, haben wir insgesamt fünf Schalter zu erstellen. Vier blaue, ein roter.

❗ Tipp: Die blauen Schalter haben diese Farbe, weil am Ende alle Flammen zur Lösung blau sein müssen.

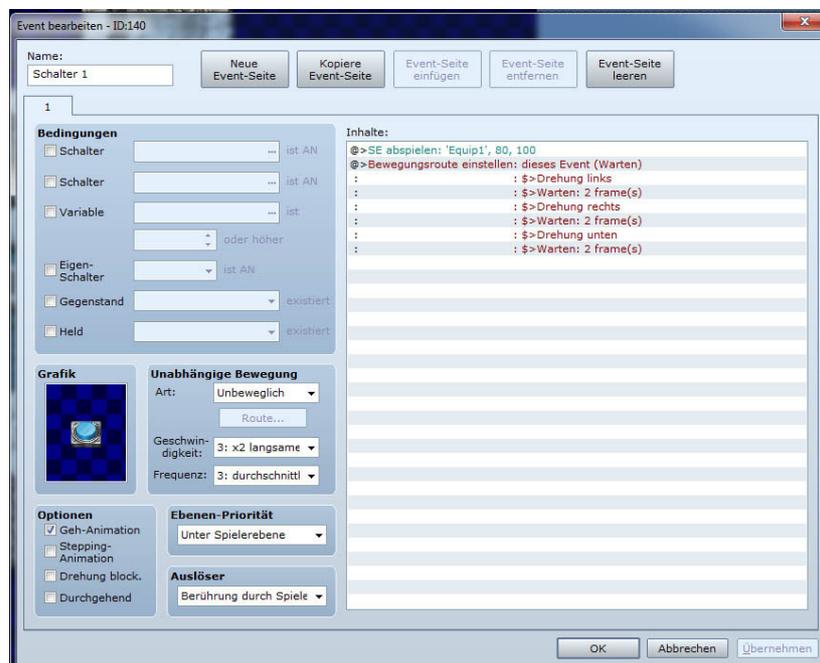
❗ Tipp: Rot wurde gewählt, da man alle Schaltereinstellungen damit wieder auf 0 setzen kann, der Vorgang wird also abgebrochen. Aber dazu später mehr.

Aber genug der Theorie, kommen wir zur Praxis. Jetzt müsst ihr besonders aufpassen, denn dies ist der schwierigste Teil des gesamten Rätsels.

Wir erstellen uns ein Event, einen ganz normalen Schalter und nennen ihn „Schalter 1“.

- ➔ Ebenen-Priorität: **Unter Spielerebene**
- ➔ Auslöser: **Berührung durch Spieler**
- ➔ Grafik: **Ein blauer Schalter**

Damit wir auch sehen, dass der Schalter etwas macht, brauchen wir erst einmal eine Bewegungsanimation und ein kleiner Sound, um es realistischer zu machen. Durch die Bewegungsrouten wird der Schalter gedrückt und geht dann in die Ausgangsposition zurück.



Nun kommt der schwerste Teil, ich hoffe ich erkläre es euch jetzt gut genug.

Wie ihr euch erinnert, haben wir im letzten Schritt vier Variablen erzeugt:

- ➔ Variable Feuer 1
- ➔ Variable Feuer 2
- ➔ Variable Feuer 3
- ➔ Variable Feuer 4

Diese müssen wir nun benutzen. Wenn wir einen Schalter betätigen, werden jetzt eine oder mehrere Variablen vergrößert oder verkleinert (addiert oder multipliziert), wie ihr im nächsten Bild seht.

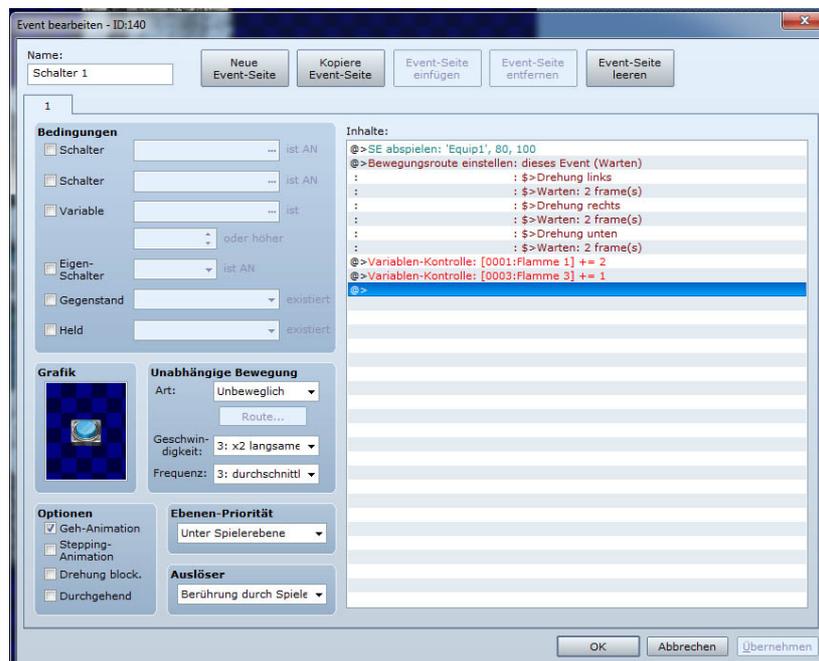
➔ Variable Flamme 1: +=2

➔ Variable Flamme 3: +=1

Event „**Flamme 1**“ hatte den Ausgangswert **0** in der Variablenbedingung. Wird nun der Wert durch **+2** auf **2** gesetzt, wird nun **Reiterseite 3** geladen. Seht am besten noch einmal nach. Event „**Flamme 1**“ wäre nun nicht mehr die leere Feuerschale sondern hätte nun die kleine **gelbe Flamme**.

Genau so gibt es jetzt bei Event „**Flamme 3**“ eine Veränderung. Die Variable wird um **1** erhöht. Die Feuerschale ist nun nicht mehr erloschen, sondern es ist nun ein **rotes Glühen** zu sehen.

⚠ Achtung: Bitte beachtet, **dass wir den Wert multiplizieren**. Verwechselt dies nicht mit „**Wert setzen auf**“. Dies sind zwei verschiedene Einstellungen in der Variablensteuerung!



⚠ Aber halt, werden jetzt einige sagen! Was ist, wenn ich mehrmals über die Schalter laufe, die Variable würde ja ewig erhöht werden oder ewig verringert. Es gäbe kein Ende!

Richtig, aus diesem Grund bauen wir uns nun einen kleinen Sicherheitsmechanismus ein. Wie ihr ja sicher noch wisst, hatten wir die Flammen bis zur Variablengröße **4** eingestellt und **0** war der geringste Wert. Also machen Werte über **4** und kleiner als **0** keinen Sinn.

Darum fragen wir jetzt per Bedingung ab:

➔ Ist die Variable „**Flamme 1**“ denn so groß wie 5 oder sogar größer?

Wenn Nein: Passiert gar nichts.

Wenn Ja: Geben wir einen Variablenbefehl aus. Variable Flamme 1 = 0

➔ Ist die Variable „**Flamme 3**“ denn so groß wie 5 oder sogar größer?

Wenn Nein: Passiert gar nichts.

Wenn Ja: Geben wir einen Variablenbefehl aus. Variable Flamme 3 = 0

➔ Ist die Variable „**Flamme 1**“ denn so groß wie -1 oder sogar kleiner?

Wenn Nein: Passiert gar nichts.

Wenn Ja: Geben wir einen Variablenbefehl aus. Variable Flamme 1 = 0

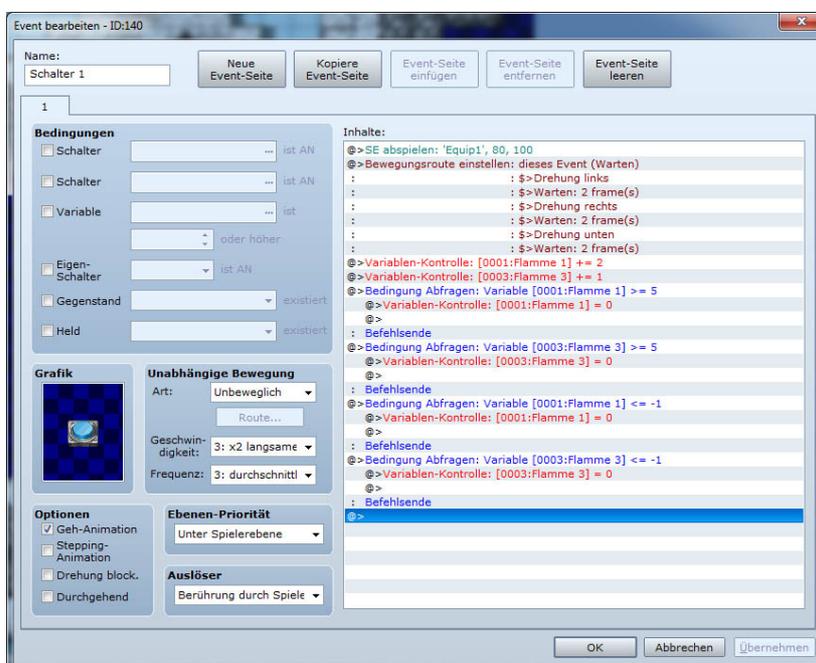
➔ Ist die Variable „**Flamme 3**“ denn so groß wie -1 oder sogar kleiner?

Wenn Nein: Passiert gar nichts.

Wenn Ja: Geben wir einen Variablenbefehl aus. Variable Flamme 3 = 0

So können keine Werte mehr größer als 4 oder kleiner als 0 sein.

⚠ Achtung: Hier benutzen wir den Befehl „**Wert Setzen auf**“ in der Variablenkontrolle! Wir multiplizieren oder dividieren hier nicht!



Soweit so gut, damit ist dieses Event fertig.

Jetzt kommt der Rechenteil der Aufgabe! Wir haben mit Hilfe von Schalter 1 nun die Variable 1 auf 2 erhöht und die Variable 3 auf 1. Nun machen wir so etwas Ähnliches auch bei den blauen Schaltern 2, 3 und 4. Nur können wir dies nicht irgendwie machen. Irgendwann muss ja einmal ein Zustand erreicht werden, bei dem eine Lösung möglich ist.

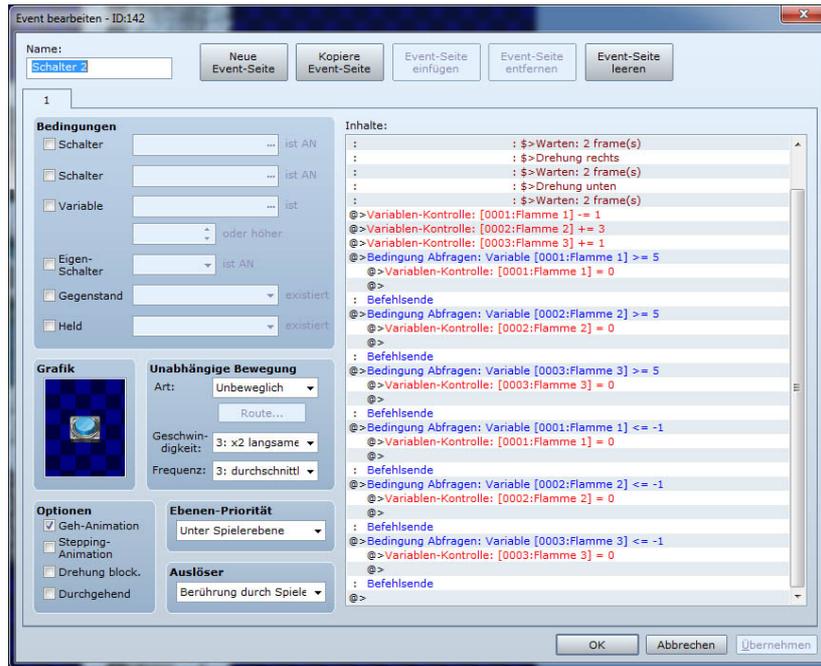
⚠ Unsere Lösung ist, alle Flammen blau zu bekommen, sprich alle Variablen auf 4 zu setzen. Da wir aber die Variablen mit den Schaltern immer wieder ändern, müssen wir uns einen Lösungsweg ausdenken, der bei mir wie folgt aussieht:

Schalter	Variable Feuer 1	Variable Feuer 2	Variable Feuer 3	Variable Feuer 4
Schalter 1	+ 2	/	+ 1	/
Schalter 2	- 1	+ 3	+ 1	/
Schalter 3	+ 1	+ 1	/	+ 3
Schalter 4	+ 2	/	+ 2	+ 1

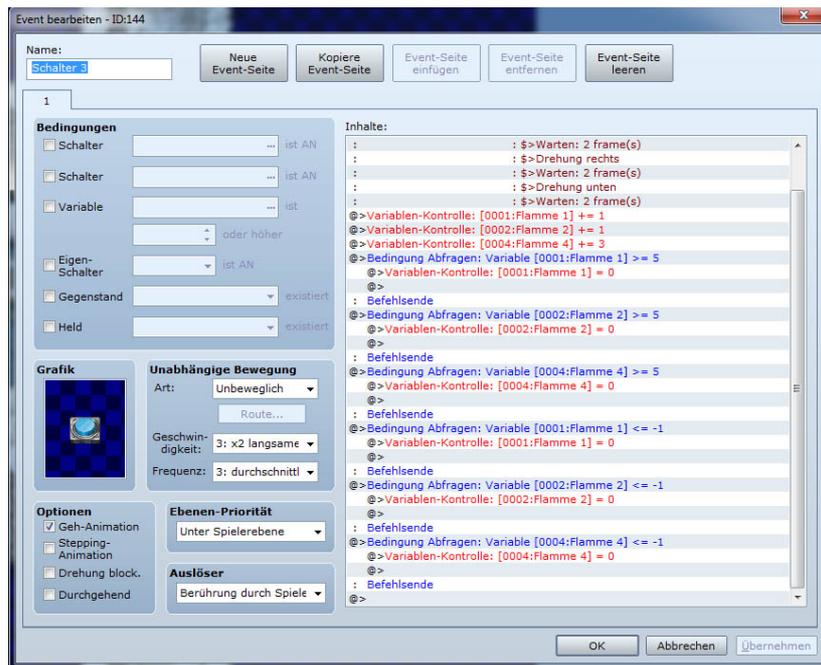
⚠ Es wurde ja schon beim Event „**Schalter 1**“ eingestellt, dass die Variablen nicht höher als 4 oder kleiner als 0 sein können. Darum folgt, was folgen muss, falsche Reihenfolge der Schalter ergibt einen falschen Lösungsweg. Man muss die richtigen Schalter in der richtigen Reihenfolge aktivieren, damit es klappt.

Jetzt braucht ihr nur noch die Schalter 2, 3 und 4 einbauen. Dies sieht dann so aus:

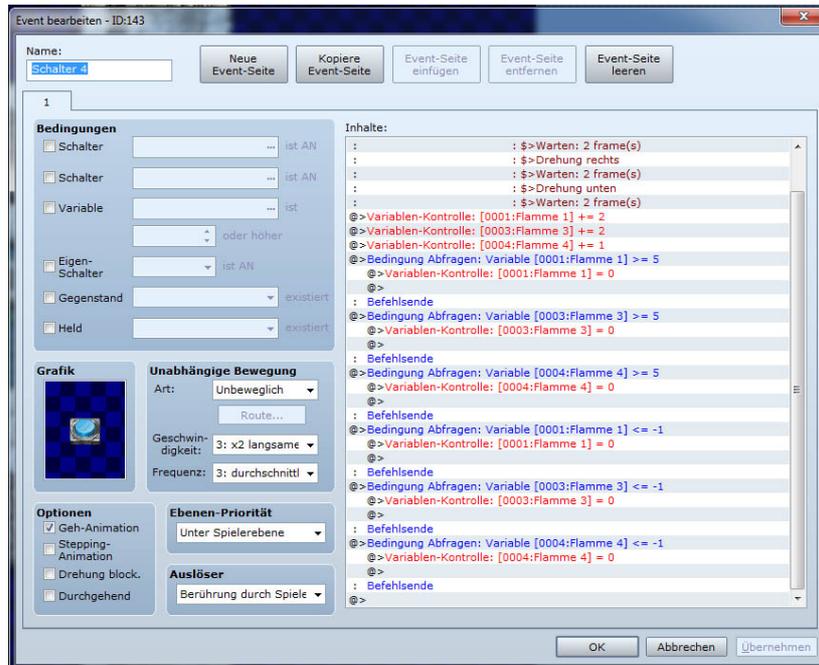
Schalter 2



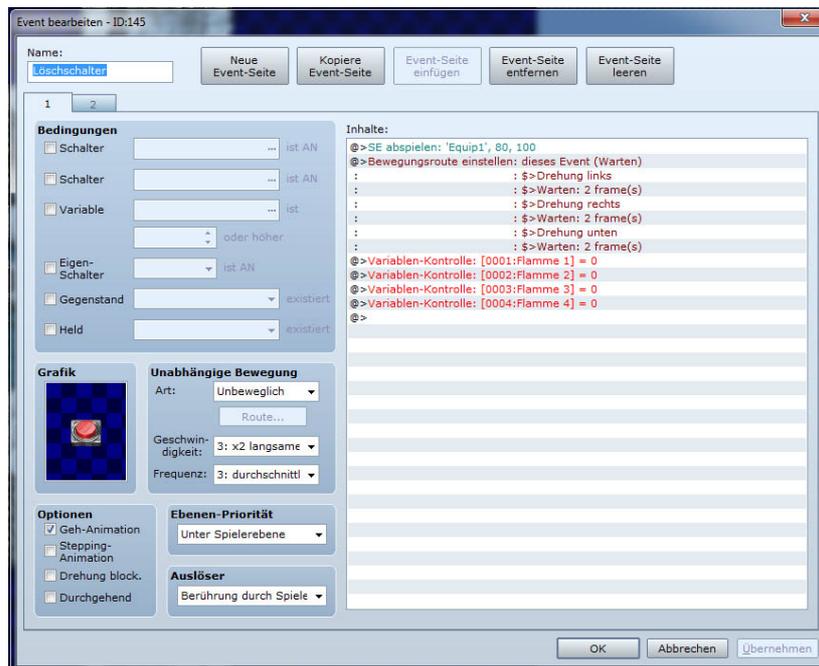
Schalter 3



Schalter 4



Wir ändern mit den Schaltern nun ständig die Einstellungen unserer Flammen. Jedoch, wenn man noch mal von vorn beginnen will, ist es schon schwer, die richtigen Schalter zu drücken, um zum Ausgangswert 0 zurückzukehren. Dies erledigt dann unser **Schalter 5, der rote**. Dieser setzt, wie man sieht, alle Variablen zurück auf 0.

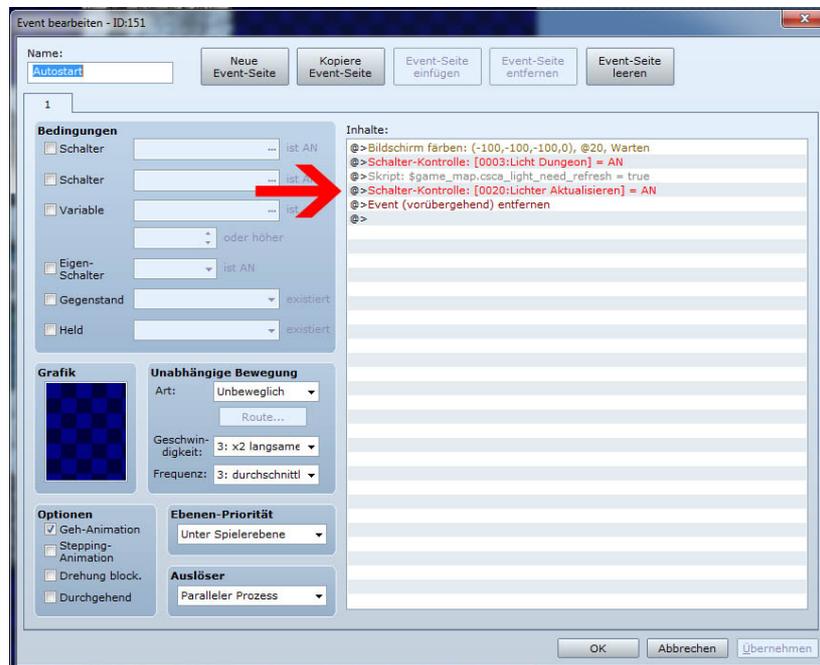


Damit sind wir mit den Schaltern durch.

⚠ Tipp: Um es etwas komplizierter zu machen, tauscht die Positionen der Schalter aus!

Die Lichtsteuerung

Willkommen bei der Lichtsteuerung. Da der schwierigste Teil hinter uns ist, kann es nun nur noch einfacher werden. Wir öffnen noch einmal unser erstes Event, den Autostart. Hier müssen wir nun noch einen neuen Schalter setzen, nämlich den Schalter für die Lichtsteuerung der Flammen. Damit sind wir in diesem Event schon fertig.



Nun kommt ein neues Event auf die Map, welches ich schlicht und einfach „Licht“ genannt habe.

- ➔ Bedingung ist hier unser gerade erstellter Schalter: „Lichter aktualisieren“
- ➔ Auslöser: **Paralleler Prozess**

Rufen wir uns noch einmal etwas ins Gedächtnis:

```
EFFECTS[15] = ["Orange1", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 4, 10, 0, 0, 1000]
EFFECTS[16] = ["Gelb1", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 5, 10, 0, 0, 1000]
EFFECTS[17] = ["Gruen1", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 6, 10, 0, 0, 1000]
EFFECTS[18] = ["Blau1", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 7, 10, 0, 0, 1000]
```

```
EFFECTS[19] = ["Orange2", "Light", 2, 1, 150, 1, Tone.new(255,-255,-255,255), 8, 10, 0, 0, 1000]
EFFECTS[20] = ["Gelb2", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 9, 10, 0, 0, 1000]
EFFECTS[21] = ["Gruen2", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 10, 10, 0, 0, 1000]
EFFECTS[22] = ["Blau2", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 11, 10, 0, 0, 1000]
```

```
EFFECTS[23] = ["Orange3", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 12, 10, 0, 0, 1000]
EFFECTS[24] = ["Gelb3", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 13, 10, 0, 0, 1000]
EFFECTS[25] = ["Gruen3", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 14, 10, 0, 0, 1000]
EFFECTS[26] = ["Blau3", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 15, 10, 0, 0, 1000]
```

```
EFFECTS[27] = ["Orange4", "Light", 2, 2, 150, 1, Tone.new(255,-255,-255,255), 16, 10, 0, 0, 1000]
EFFECTS[28] = ["Gelb4", "Light", 2, 2, 150, 1, Tone.new(255,-100,-255), 17, 10, 0, 0, 1000]
EFFECTS[29] = ["Gruen4", "Light", 2, 2, 150, 1, Tone.new(-255,255,-255,100), 18, 10, 0, 0, 1000]
EFFECTS[30] = ["Blau4", "Light", 2, 2, 150, 1, Tone.new(-255,255,255,100), 19, 10, 0, 0, 1000]
```

Dies sind unsere Schalter zu den Lampen. Es betrifft hier die Schalter 4 – 19. Mit diesen 16 Schaltern steuern wir die Lichter der Flammen.

Darum müssen wir nun als nächsten Arbeitsschritt 16 Schalter erstellen, die da heißen:

Schalter Nr.	Schalter Name
4	Licht Orange 1
5	Licht Gelb 1
6	Licht Grün 1
7	Licht Blau 1
8	Licht Orange 2
9	Licht Gelb 2
10	Licht Grün 2
11	Licht Blau 2
12	Licht Orange 3
13	Licht Gelb 3
14	Licht Grün 3
15	Licht Blau 3
16	Licht Orange 4
17	Licht Gelb 4
18	Licht Grün 4
19	Licht Blau 4

Wenn ihr fertig seid, kommt nun etwas Bedingungsabfragenchaos auf uns zu, aber keine Angst, es geht wieder Schritt für Schritt.

Wir haben ja noch unser neues Event mit dem Namen „**Licht**“ auf.

➔ Am Anfang kommt erst einmal ein „Wait = 2 Frames“ hin. Dieses beugt Lags vor.

Der nächste Schritt ist ein Abfrageblock für je eine Flamme.

Darum fragen wir nun ab:

➔ Variabel Flamme 1 = 0

Wenn ja:

Schalter Licht Orange 1 = AUS

Schalter Licht Gelb 1 = AUS

Schalter Licht Grün 1 = AUS

Schalter Licht Blau 1 = AUS

Dies ist logisch. Kein Feuer brennt, darum brauchen wir auch keinen Feuerschein. Also jegliches Licht für das Event „**Flamme 1**“ aus.

Sollte die Variable nicht 0 sein, kommen wir zur nächsten Abfrage:

→ Variabel Flamme 1 = 1

Wenn ja:

Schalter Licht Orange 1 = AN

Schalter Licht Gelb 1 = AUS

Schalter Licht Grün 1 = AUS

Schalter Licht Blau 1 = AUS

Wenn die „**Variable Flamme 1**“ gleich 1 ist, dann ist die kleine rote Flamme an, darum müssen alle anderen Schalter aus sein, außer der Schalter für das kleine orange Licht.

Sollte die Variable nicht 1 sein, kommen wir zur nächsten Abfrage:

→ Variabel Flamme 1 = 2

Wenn ja:

Schalter Licht Orange 1 = AUS

Schalter Licht Gelb 1 = AN

Schalter Licht Grün 1 = AUS

Schalter Licht Blau 1 = AUS

Wenn die „**Variable Flamme 1**“ gleich 2 ist, dann ist die schon etwas größere gelbe Flamme an, darum müssen alle anderen Schalter aus sein, außer der Schalter für das gelbe Licht.

Sollte die Variable nicht 2 sein, kommen wir zur nächsten Abfrage:

→ Variabel Flamme 1 = 3

Wenn ja:

Schalter Licht Orange 1 = AUS

Schalter Licht Gelb 1 = AUS

Schalter Licht Grün 1 = AN

Schalter Licht Blau 1 = AUS

Wenn die „**Variable Flamme 1**“ gleich 3 ist, dann ist die schon recht große grüne Flamme an, darum müssen alle anderen Schalter aus sein, außer der Schalter für das grüne Licht.

Sollte die Variable nicht 3 sein, kommen wir letzten Abfrage:

→ Variabel Flamme 1 = 4

Wenn ja:

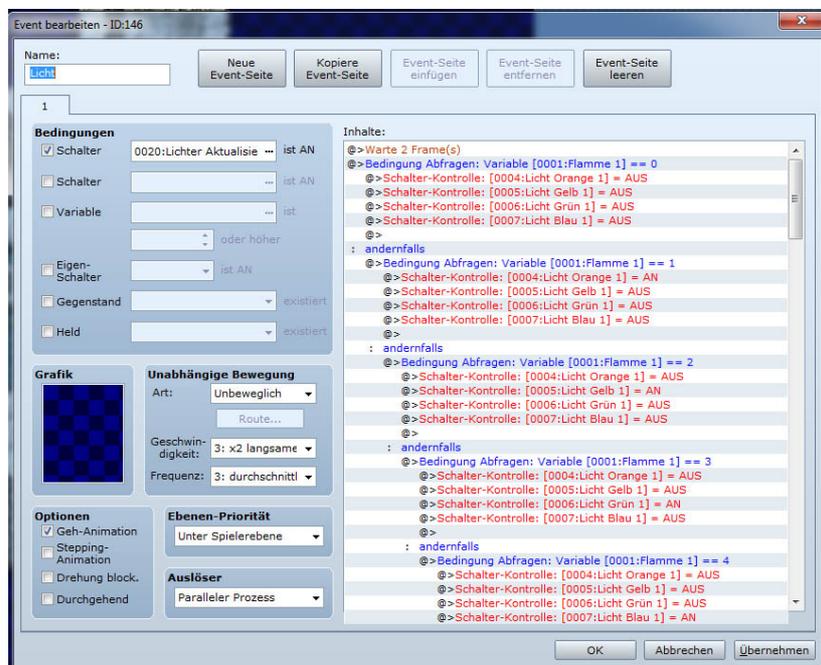
Schalter Licht Orange 1 = AUS

Schalter Licht Gelb 1 = AUS

Schalter Licht Grün 1 = AUS

Schalter Licht Blau 1 = AN

Wenn die „**Variable Flamme 1**“ gleich 4 ist, dann ist die große blaue Flamme an, darum müssen alle anderen Schalter aus sein, außer der Schalter für das blaue Licht.



Damit hätten wir aber nur die Lichtersteuerung des Events „Flamme 1“ fertiggestellt. Wir haben aber noch drei weitere Flammen. Darum kopieren wir uns diesen Teil der Bedingungsabfrage und ändern die Variablen und Schalterwerte ab:

Was sehen eure vom Tutorial entzündeten Augen denn da am Ende? Kennen wir diesen Befehl nicht?

Die Antwort lautet: JA!

Aber warum stellen wir hier diesen Scripteintrag noch einmal hin? War nicht die Rede davon, dass der Scripteintrag dafür zuständig sei, die Lichter bei Änderungen zu aktualisieren?

❗ **Jetzt habt ihr mich aber!** Ja ist auch so ... eigentlich sollte der Scriptbefehl, wenn er einmal ausgeführt wurde, auch seine Wirkung tun. Aus irgendeinem Grund macht er dies aber nicht, darum muss er am Ende stehen, damit die Lichter aktualisiert werden, wenn der Schalter geändert wird.

Hier noch einmal der Script Befehl: `$game_map.cscs_light_need_refresh = true`

Damit sind wir hier schon fertig!

Die Lösungsabfrage

Fast geschafft! Nun müssen wir nur noch die Bedingung festlegen, wann das kleine Rätsel gelöst ist. Darum öffnen wir uns noch einmal das Event „Licht“.

Alle vier Flammen sollten zur Lösung groß und blau sein. Dazu müssen alle Variablen auf 4 sein.

Fragen wir dies doch ab und legen mehrere Bedingungen direkt hintereinander.

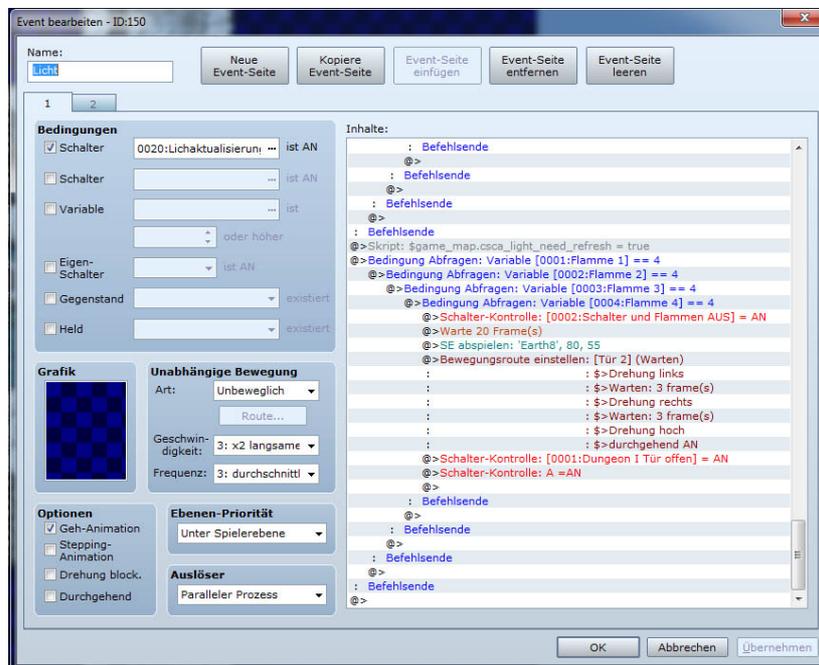
Also fragen wir nun ab:

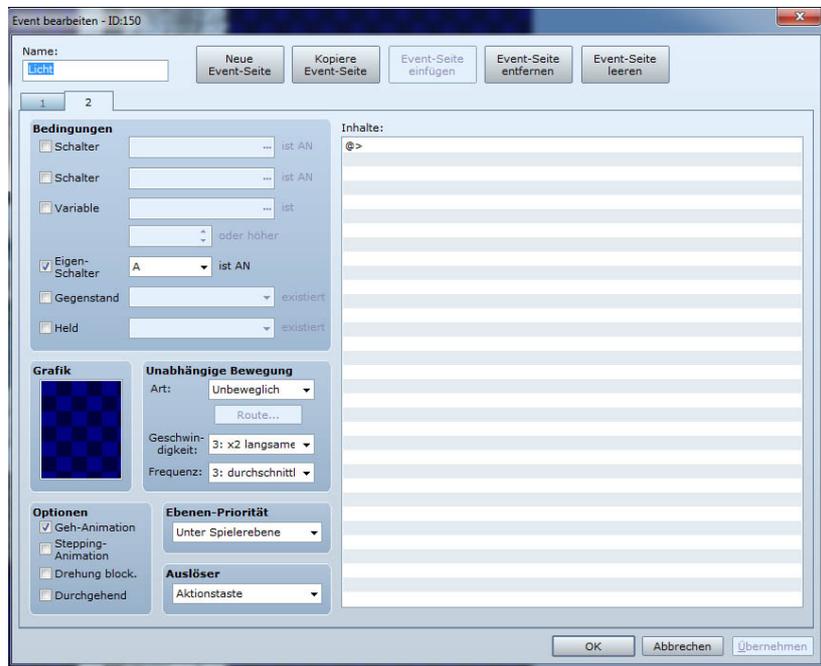
→ Bedingung Variable Flamme 1 = 4 → Wenn ja: Bedingung Variable Flamme 2 = 4 →
Wenn ja: Bedingung Variable Flamme 3 = 4 → Wenn ja: Bedingung Variable Flamme 4 = 4

Sollte dieser Zustand eintreten, ist das Rätsel gelöst und einiges wird nun aufgelöst:

- Ein neuer Schalter wird aktiviert: „**Schalter und Flammen AUS**“
- Wait = 20 Frames
- Ein Türknarren für Detailverliebte, bei mir **Earth 8, 80, 55**
- Türbewegung für eine Tür, die wir toller Weise noch nicht eingebaut haben
- Ein zweiter neuer Schalter wird aktiviert: „**Dungeon Tür I offen**“
- Als letztes nutzen wir den guten alten Eigenschalter, um das Event zu deaktivieren

Damit wären wir in diesem Event fertig und ich zeige euch noch kurz wie es bei mir aussieht. Eine Erklärung für die ganzen Schalter folgt danach.





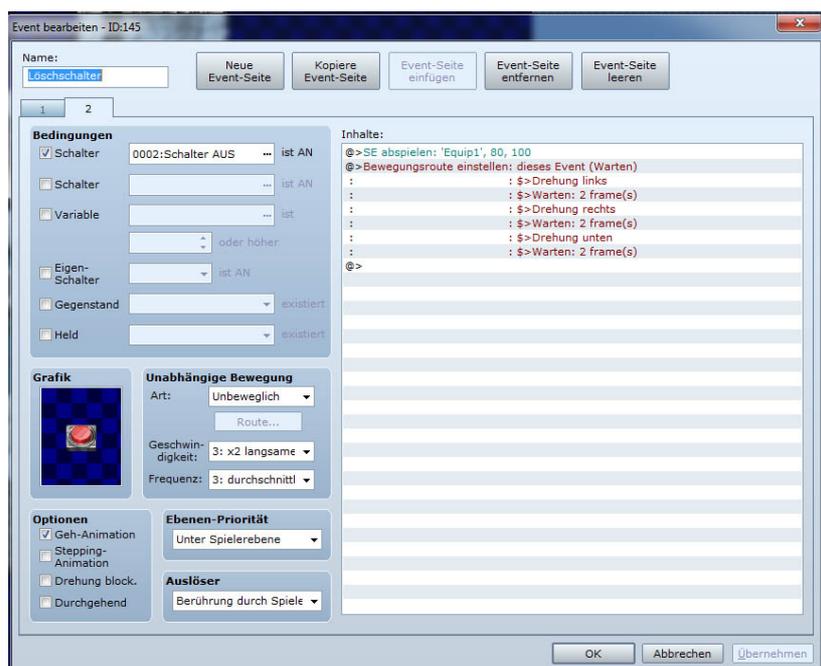
Damit sind wir mit diesem Event fertig. So, nun zur Erklärung der ominösen neuen Schalter. Da wir ja Einiges eingeschalten haben, müssen wir auch einiges wieder ausschalten. Ergibt Sinn, oder?

Schalter „Schalter AUS“

Mit diesem Schalter sorgen wir dafür, dass die Schalter keinen Blödsinn mehr machen können. Öffnet noch einmal alle fünf Schalter und erstellt noch eine Reiterseite, in dem der Schalter zwar gedrückt werden kann, aber nichts mehr passiert.

! Vergesst nicht die Bedingungen einzustellen!

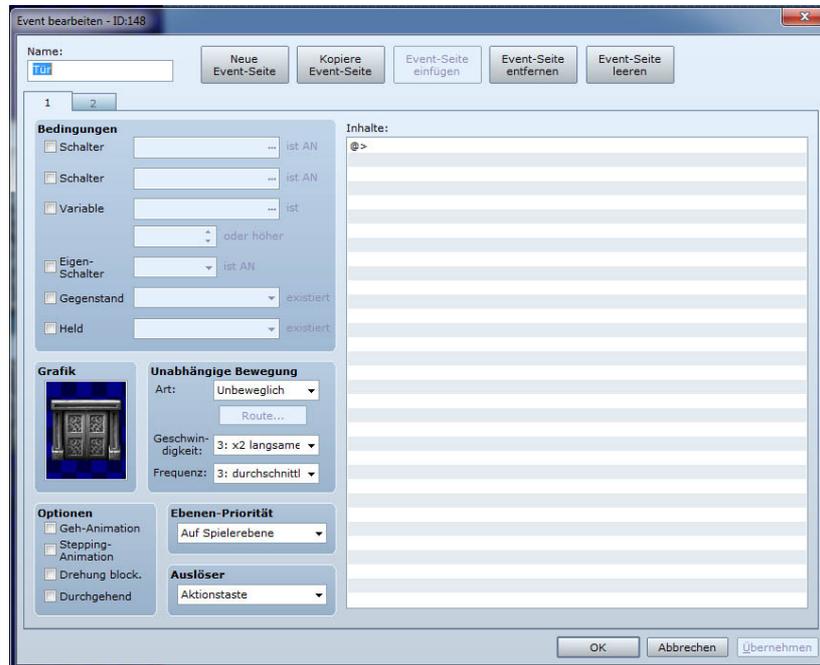
! Achtet auch auf die Farben der Schalter, wenn ihr Eventseiten kopiert. Es sieht doof aus, wenn die Schalterfarbe der Map plötzlich wechselt.



Schalter „Dungeon Tür I offen“

Als letztes kommen wir noch zur Türsteuerung. Setzt euch die Tür an die Mauer eurer Wahl. Als Grafik nehmt ihr eine beliebige Tür. Die Grafik der Tür sollte verschlossen sein.

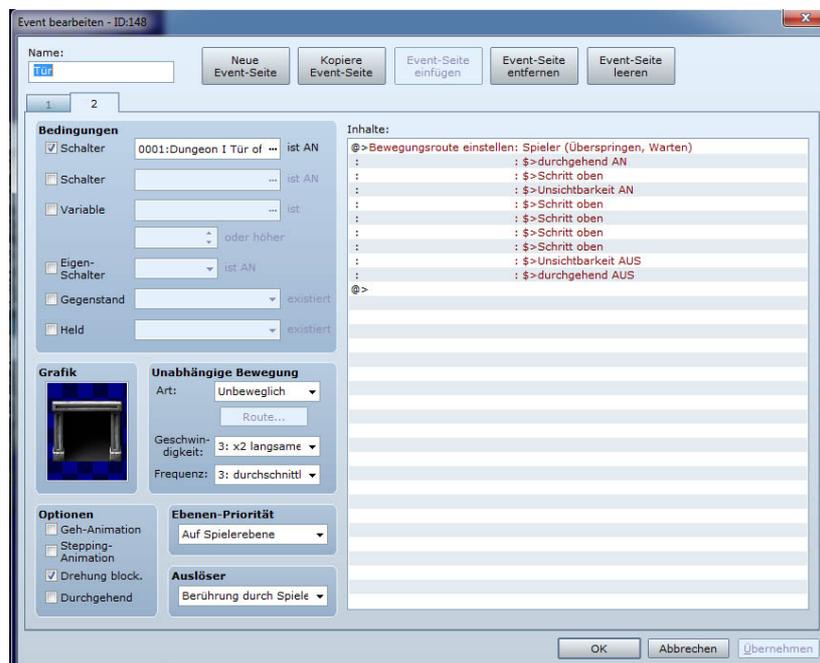
Dies erfolgt alles auf der ersten Ebene. Zu beachten ist also, dass die Tür auf der Spielerebene ist und der Auslöser auf Aktionstaste steht.



Auf der zweiten Seite könnt ihr die Tür nun einstellen, wie ihr es gewohnt seid. Bei mir ist die Tür immer auf Spielerebene und der Auslöser ist Berührung durch Spieler.

Bei den meisten dürfte jetzt ein Teleport im Inhalt stehen – bei mir nicht. Denn unser kleiner Protagonist läuft nur durch die Tür, um dem nächsten kleinen Rätsel geradewegs in die Arme zu laufen.

❗ Vergesst nicht, den Schalter zu setzen und eventuell im letzten Event, was wir bearbeiten haben, noch das richtige Event für die Türsteuerung einzustellen.



Damit sind wir am Ende des Tutorials angelangt.

Wenn ihr alles genau nach Vorlage gemacht habt, dürftet ihr nun ein schönes kleines Rätsel haben. An vielen Ecken könnte man einiges anders machen, auch durch Scriptbefehle vereinfachen. Aber wie sagt man so schön: „Viele Wege führen nach Rom.“ Dies war meiner und habe mich für ihn entschieden, da meine Tutorialreihe gerade an neue Makerfreunde gerichtet ist. Darum sollte es so einfach wie möglich sein. Ich hoffe sehr, dass alles verständlich war. Wenn nicht, dann scheut euch nicht, nachzufragen.

 Im Übrigen könnt ihr mit dieser Methode auch jeden anderen Licht-Script nutzen. Jedoch ist jeder Script anders und ich habe nur diesen hier erklärt. Bei anderen Scripten müsstet ihr dann selber die nötigen Anpassungen vornehmen.

Ein klein wenig Detailverliebtheit

Huch? Hier liest ja noch einer weiter...

Nun gut, dann gehörst du zu den Detailverliebten, wie ich es einer bin. Wenn wir schon Feuer mit Lichteffekten haben, warum nicht auch ein gemütliches Feuerknistern?

Höre ich da Sätze wie: „**Schalte doch einfach einen BGS-Sound an!**“?

Könnte man machen, jedoch hört man das Feuer überall gleich laut und dies ist doch arg unrealistisch. Darum erstellen wir uns noch ein Event mit dem Namen: „**Feuerknistern**“.

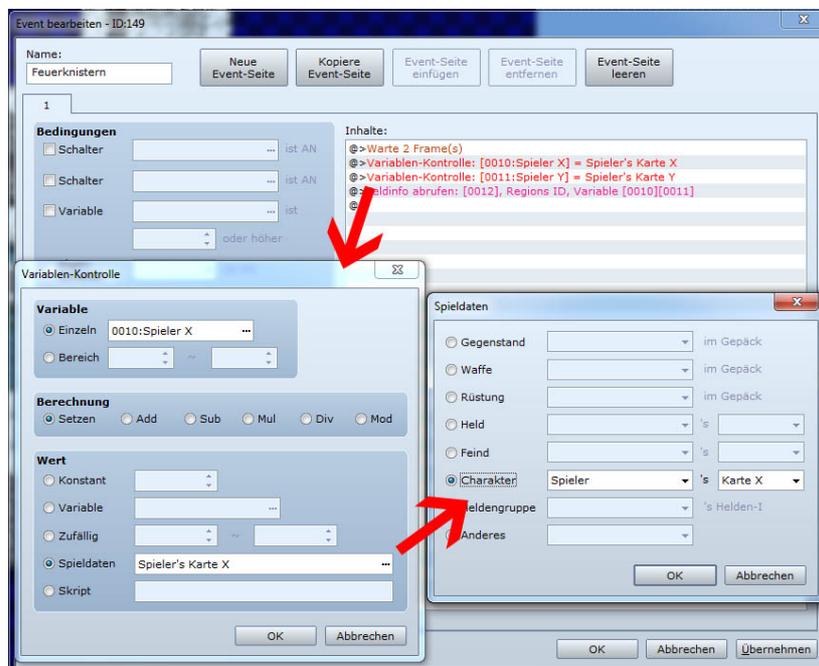
Einzigste Einstellung im Event:

➔ Auslöser: **Paralleler Prozess**

Der Inhalt:

- ➔ **Wait 2 Frames** (wieder eine Sicherung gegen Lags)
- ➔ Variable Koordinate **Spieler X**
- ➔ Variable Koordinate **Spieler Y**
- ➔ Variable **Feld ID**

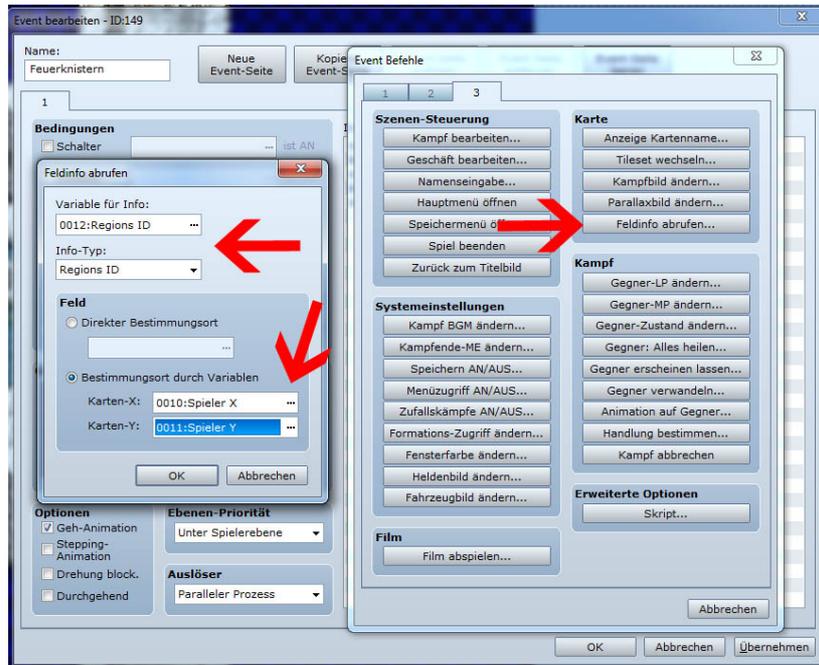
Dies sind die Variablen, die wir brauchen. Spieler X und Y ein zu stellen ist recht einfach. Am besten zu erklären mit einem Bild:



Feld-ID-Infos ruft man etwas anders ab. Feld-ID-Info ist auf Seite drei der Eventbefehle zu finden.

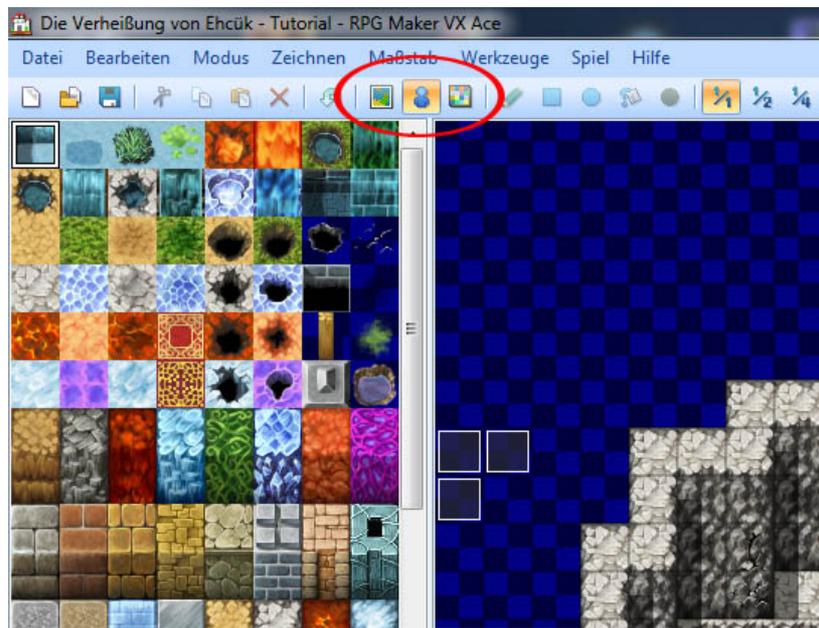
- ➔ Wir setzen den Variablennamen ein
- ➔ Info-Typ ist Regions-ID
- ➔ Bestimmungsort wird durch die Variablen X und Y des Spielers ermittelt

Wir fragen also ab, auf welchem Regions-ID Feld der Spieler ist.



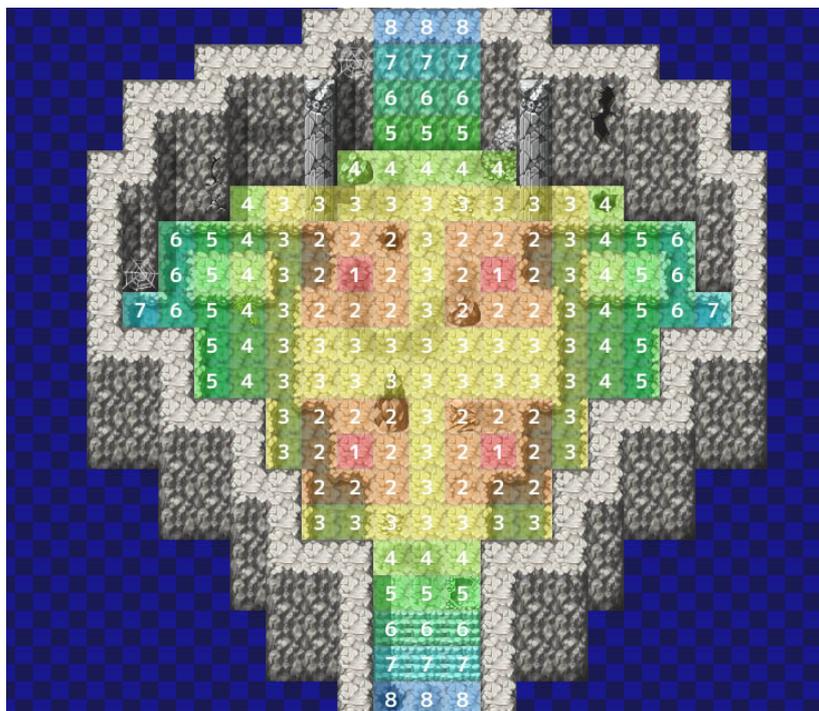
Wir gehen zwei Mal auf **OK** und schließen damit das Event.

Einige werden sich nun fragen, was sind Region IDs? Das ist recht einfach zu erklären. Zwei der eingekreisten Symbole dürfen euch bekannt vorkommen. „Karte“, „Events“ und daneben ist „Regions“. Klickt mal drauf.



Die Karte wird etwas blass und die Tilesets sind weg. Stattdessen sind da Nummern. Dies sind die Regions IDs. Klickt man sie an und „zeichnet“ damit auf der Karte herum, legt man sozusagen unsichtbare Markierungen aus. Diese nutzen wir nun zur Soundsteuerung. Dabei gilt, um so weiter weg vom Feuer, um so größer die Regions ID.

Bei mir sieht dies nun so aus:



Zurück in unser Event, die Regions-IDs sind verteilt. Nun fragen wir ab, wo der Spieler steht und was passiert, wenn er auf einem Regions-ID-Feld ist.

Dann fragen wir ab:

→ Variable Regions ID = 1

Wenn ja:

BGS „Fire“ abspielen, Lautstärke 80 %

Sollte die Variable nicht 1 sein, kommen wir zur nächsten Abfrage:

→ Variable Regions ID = 2

Wenn ja:

BGS „Fire“ abspielen, Lautstärke 70 %

Sollte die Variable nicht 2 sein, kommen wir zur nächsten Abfrage:

→ Variable Regions ID = 3

Wenn ja:

BGS „Fire“ abspielen, Lautstärke 60 %

Sollte die Variable nicht 3 sein, kommen wir zur nächsten Abfrage:

→ Variable Regions ID = 4

Wenn ja:

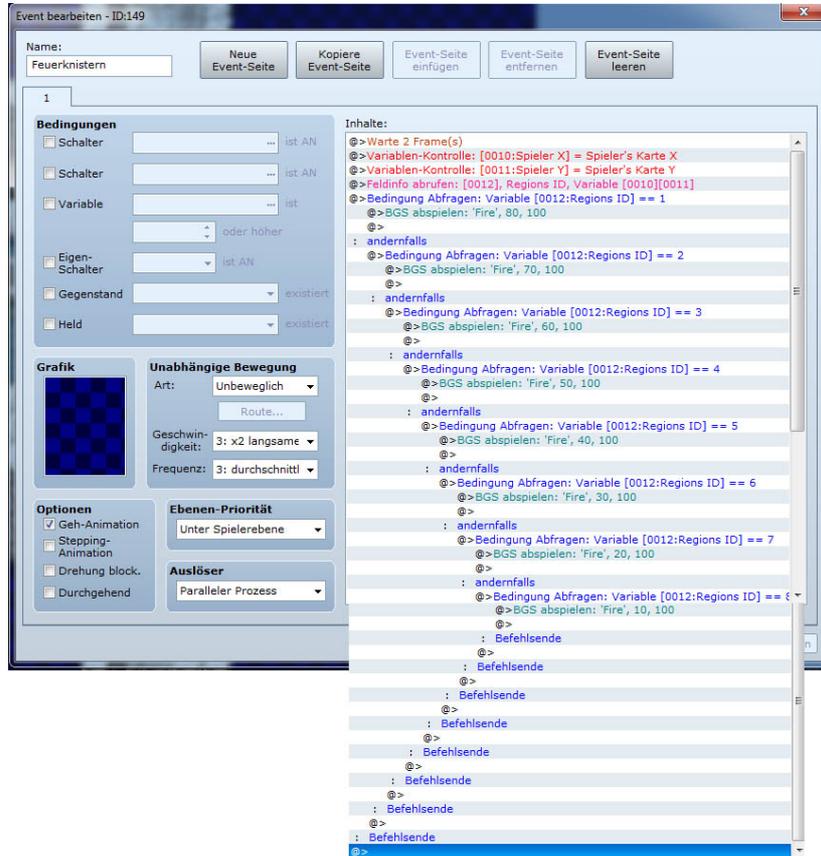
BGS „Fire“ abspielen, Lautstärke 50 %

Sollte die Variable nicht 4 sein, kommen wir zur nächsten Abfrage:

etc.

Ich denke, ihr habt das System durchschaut. Dies macht ihr nun bis Regions-ID = 8 und damit wäre das Event auch schon fertig.

Aussehen müsste es wie bei mir:



Speichert noch und das wars. Nun habt ihr auch das passende Hintergrundgeräusch für euer Rätsel.

Damit wären wir am Ende und wünsche ich euch viel Spaß beim nachbauen!

Demo-Tutorial

Das komplette Tutorial könnt ihr als *RPG Maker VX Ace* Projekt hier laden:

<http://gamealchemists.com/?wpdmdl=183>