HusisTwitchSkripte – Dokumentation

Um sich mit dem IRC-Chat von Twitch zu verbinden, musst du zu erst einmal die Funktion
twitch_start(username, oauth, channelname)
aufrufen.

username	- dein Benutzername auf Twitch
oauth	- der Authentifikations-Key, den du beispielsweise auf der Seite http://twitchapps/tmi bekommst.
channelname	- Kanal mit dem du dich verbinden möchtest

Ob du dich anmelden konntest, kannst du später in der Debugging-Console in GameMaker sehen.

Um Nachrichten zu empfangen musst du die Funktion
twitch_networking()
in das Event: *Asynchronous>Networking* packen.

Die Funktion speichert alle ankommende Nachrichten in einen Stream. Stelle dir das so vor:
Du hast einen leeren Platz. Kommt eine Nachricht, legt Sie sich in Form einer Karte auf diesen Platz. Egal wie groß der Stapel wird, wir wollen die zuerst empfangene Nachricht auslesen. Dazu muss einfach die Unterste Karte herausgenommen werden und die anderen rutschen durch die Schwerkraft einen nach.
Ich empfehle es, die Nachrichten im Step-Event abzurufen.
Schaue zu erst nach, ob eine Karte vorhanden ist:

```
if(twitch_stream_available()>0)
```

Wenn sie vorhanden ist, kannst du diese Karte auslesen:

```
var message = twitch_stream_read();
```

die Karte wird in Form einer ds_map zurückgegeben. Hier sind folgende Werte abgespeichert:

- color/colour
- displayname
- emotes
- subscriber
- turbo
- sender
- msg

um die folgende Werte zu erhalten, kannst du es dir relativ einfach machen:

```
message[? „wert“]
```

um den Absender und die Nachricht in Variablen abzuspeichern, sähe es dann so aus:

```
var sender = message[? „sender“];
var msg    = message[? „msg“];
```

Anzeigen kann man es dann beispielsweise so:

```
show_message(sender+“: “+msg);
```

Um eine Nachricht abzusenden musst du nur folgende Funktion benutzen:
twitch_send_message(msg)
Um „Hi“ zu schreiben sähe das so aus:

```
twitch_send_message(„Hi“);
```

Jetzt gibt es noch die Möglichkeit die Zuschauernamen(einfache Zuschauer, Moderatoren und Administratoren).
Hierfür ist es wichtig, dass du im Event *Asynchronous>Http* die Funktion
twitch_http();
aufrufst.
Jetzt ist es dir zu jeder Zeit möglich, über die Funktion
twitch_request_info();
Die Werte abfragst bzw. erneuerst.
Zusätzlich gibt es noch die Möglichkeit über twitch_set_autoreload(true) die Aktualisierung zu automatisieren. Dieser Mechanismus muss aber erst mit der Funktion twitch_request_info() in Gang gesetzt werden. Dadurch wird aber sehr rasch aktualisiert, wodurch unnötig Internet gebraucht wird.

Die Namen werden in Listen (ds_list) abgespeichert, die du mit der ds_map, die du über die Funktion
twitch_get_info()
bekommst.
Hierbei gibt es folgende Möglichkeiten:

viewer	- alle Zuschauer, einschließlich Moderatoren und Administratoren
mods	- Moderatoren
admins	- Administaroren
connected	- alle Zuschauer, die nach dem vorletzten Aktualisieren dazu kamen
disconnected	- alle Zuschauer, die nach dem vorletzten Aktualisieren gegangen sind

Anschließend ist es möglich sich vom Chat zu trennen und den Arbeitsspeicher wieder zu leeren.
Hierzu musst du einfach die Funktion:
twitch_end();
aufrufen.
Um sich nocheinmal neu zu verbinden ist es nötig wieder **twitch_start** zu benutzen.

Immer wieder gerne gesehen ist es auch, wenn ein Computer/Bot den Chat und evtl. sogar den Stream interaktiv gestaltet. So kann man beispielsweise mit !join blue als Mitspieler (und sogar mit der Farbe blau) im Stream erscheinen!
Dies zu bewerkstelligen ist ganz einfach. Dazu habe ich extra eine Funktion geschrieben – parse_command(cmd) - , die diese Befehle zerstückelt und in ein Array steckt.
Wenn eine Nachricht eingegangen ist erstellt man eine Variable bspw. command und gibt ihr den Wert, den diese Funktion ausgibt. Den Parameter *cmd*, den man angeben muss, ist einfach die empfangene Nachricht.
Besitzt command den Wert -1, so wurde kein Befehl gesendet. Ansonsten enthält die Variable ein Array.
Auf der 1. Position (command[0]) ist der Befehl ansich, also in dem oberen Beispiel jetzt „join“. Über array_length_1d(command) kann man jetzt noch schauen, wie viele Parameter in der Nachricht stecken. Sind dort 2, wie in diesem Fall, so kann man auch noch die nächste Position auslesen (command[1]) mit dem Wert „blue“.
Eine Musterlösung:

```
var command = parse_command(msg); //msg ist die empfangene Nachricht
if(command != -1){ //Wenn ein Befehl empfangen wurde
  if(command[0] == “join“){ //Wenn es der Befehl „join“ ist
```

```
   var colour = “white“; //Voreingestellte Farbe, falls keine Farbe mitgegeben wurde
   if(array_length_1d(command)>=2) //wenn ein min. 2 Parameter übergeben wurden
    colour = command[1];

   twitch_send_message(“Bot: user joined with colour '“+colour); //Chatte, dass der
Benutzer mit der angegebenen Farbe gejoint ist
  }
}
```