

Old School Textscroller

Description

Das Thema Textscroller hatten wir schon vor einigen Jahren. Doch es wäre schön, wenn sich, wie in den 1980er Jahren, die einzelnen Buchstaben auf und ab bewegen würden.

Das Ziel sehen wir bereits auf dem Titelbild. Ein längerer Text scrollt horizontal über den Bildschirm. Alle Buchstaben bewegen sich dabei sanft auf und ab. Um es optisch etwas ansprechender zu machen, gibt es noch einen Farbverlauf auf den Buchstaben. Hierbei nutzen wir die neue Möglichkeit in GMS, direkt Hexwerte als Farben zu verwenden.

Theorie

Das Problem ist, dass wir in GM einen Text angeben, etwas „Hallo Welt!" und diesen mit *draw_text(x, y, text)* anzeigen. Das heißt, dass sich alle Bewegungen auf den ganzen Text auswirken. Damit wir jeden einzelnen Buchstaben ansteuern, müssen wir den Text zunächst zerhacken und die einzelnen Zeichen in ein Array packen.

Anschließend müssen wir den Text mit dem jeweiligen Abstand pro Zeichen scrollen lassen. Das vertikale Wackeln der Buchstaben ist dabei das kleinste Problem.

Raum

Ich habe einen Raum mit der Größe 1280×768 erstellt. Als Hintergrundfarbe verwende ich #494F5F. Anschließend wird ein Objekt (bei mir *o_effect*) erstellt und im Raum platziert.

Schrift

Am besten verwendet man eine sog. „Monospaced" Schriftart. Hier sind die Zeichen alle gleich groß, was das Lesen vereinfacht. Die Abstände zwischen den Zeichen bestimmen wir mit der Variable *fontWidth* im Code. Für die Demonstration verwende ich die Schrift **Digiface**. Sie ist nicht nur

monospaced sondern auch schön Fett, wodurch der Farbverlauf gut zur Geltung kommt. Die Größe ist 72. Wichtig: Charset sollte 32-255 enthalten. In den meisten solcher Schriften fehlt mindestens das ß, also schreibt bspw. Spass statt Spaß. Es sollte auch darauf geachtet werden, ob sich Umlaute in der Schrift befinden.

Objekt o_effect

Create-Event

```
text  = „                ...kleiner old school textscroller von Bytegame.de! V
fontWidth = 76;
letters  = string_length(text)+1;
xx   = [];
yy   = 0;
char  = [];
wiggle  = 24;
counter  = 0;

for (n = 0; n < letters; n++)
{
  char[n] = string_char_at(text, n);
  xx[n]  = n * fontWidth;
}
```

Zu beginn definieren wir viele Variablen. Darunter den eigentliche Text. Mit den Leerzeichen beginnen wir, damit der Text später am rechten Bildrand startet. Wie viele wir brauchen, hängt von der Schrift und der nächsten Variable, *fontWidth* ab.

Dann haben wir die Variable *letters*, mit der wir angeben, wie viele Zeichen sich im Text befinden. Das machen wir über die Funktion **string_length()**. +1 brauchen wir, damit wir das letzte Zeichen nicht abschneiden.

xx wird ein Array für die x-Koordinate jedes einzelnen Zeichens. *yy* wird die Höhe, die aufgrund des Wackelns stets unterschiedlich sein wird.

char ist das oben angesprochene Array für die einzelnen Zeichen.

wiggle ist unsere Distanz für den Wackeleffekt. Hier kann man ruhig ein wenig mit der Zahl spielen.

counter ist selbstredend, die genaue Funktionsweise sehen wir gleich im Draw-Event.

Nun kommt noch eine kleine Schleife. Dabei schreiben wir den Text in das Array *char[]*. In der nächsten Zeile setzen wir die x-Position der einzelnen Zeichen in das Array *xx[n]*.

Draw-Event

```
draw_set_font(f_text);

for (n = 0; n < letters; n++)
{
```

```
    yy = room_height/2 + wiggle * sin(n + counter / (pi*3));
 draw_text_color(xx[n], yy, char[n],#FCF8EC, #FCF8EC, #9CA292, #9CA292, 1);
    xx[n] -= 8;

    if (xx[n] < -fontWidth)
 {
      xx[n] = letters * fontWidth;
 }
}

counter++;
```

Das ist wirklich nicht viel Code. In der ersten Zeile definieren wir die Schriftart *f_text*, dann kommt eine Schleife und am Ende zählen wir den *counter* hoch.

Die Schleife zählt, wohlgemerkt pro Step, durch jedes einzelne Zeichen (*n* < *letters*).

Zunächst berechnen wir die y-Koordinate für das jeweilige Zeichen. Wesentlich ist der Teil *wiggle * sin(n + counter / (pi*3)*. Hier fließt unser Wackler ein, der mit der Sinus-Funktion multipliziert wird. Der Counter wird dabei durch *pi*3* dividiert. Pi ist die Kreiszahl und in GMS eine Konstante. Man kann sie auch mit 2 multiplizieren, dann wirkt es aber nicht ganz so sanft. Man hätte auch statt *pi*3* einfach *9.42* nehmen können. Ich verwende aber lieber die Konstante, wenn wir sie schon von GM zur Verfügung gestellt bekommen. *room_height/2* am Anfang der Berechnung sagt lediglich nur aus, wo der Scroller starten soll. In dem Fall in der vertikalen Mitte.

In der nächsten Zeile zeichnen wir bereits das Zeichen *char[n]*. Im Beispiel habe ich *draw_text_color* verwendet, aber ihr könnt auch *draw_text* nehmen, wenn ihr es einfarbig darstellen möchtet.

xx[n]– sorgt dafür, dass es überhaupt scrollt. Um das besser verstehen zu können, schauen wir kurz auf das **Create-Event** zurück. In der Schleife bestimmen wir die x-Koordinate für jedes Zeichen. Die Zeile lautet *xx[n] = n * fontWidth;*. Bei n = 0 ist es 0. Bei n = 1 ist es 76 usw. Das bedeutet: *xx[0]* beginnt bei 0, *xx[1]* bei 76, *xx[2]* bei 152. Da wir von rechts nach links scrollen, muss die x-Koordinate für jedes Zeichen verringert werden (*xx[n]–*). Wenn ihr es schneller haben wollt, dann testet mal *xx[n] -= 8*.

Danach fragen wir mit *if (xx[n] < -fontWidth)* ab, ob die x-Position kleiner ist als der negative Wert von *fontWidth*. Im Beispiel wäre das -76. Warum? Weil wir dadurch sicher sein können, dass das Zeichen links außerhalb vom Bild ist und wir es nicht mehr sehen.

Mit *xx[n] = letters * fontWidth;* schieben wir die x-Koordinate ganz nach links.

Das war es schon mit der Schleife. Am Ende zählen wir noch den *counter* hoch, den wir ja für die Sinus-Funktion brauchen.

Variationen

Natürlich könnte man andere Farben nehmen, dem Text einen Schatten verpassen, oder ihn rückwärts laufen lassen. Über Alarme wäre es auch leicht möglich, die Textgeschwindigkeit zwischendurch zu ändern. Etwa am Anfang schnell, damit er die ganze Breite füllt, dann langsamer und ab einem bestimmten Punkt immer schneller, bis er nicht mehr lesbar ist. Der Kreativität sind hier quasi keine

Grenzen gesetzt, also tobt euch aus!

Date Created
11. März 2022
Author
sven