Texte im Kreis „tippen”

Description

Wie man einen Text im Kreis bewegt, hatten wir bereits. Nun nehmen wir diesen Effekt und erweitern ihn ein wenig, indem wir ihn „tippen” und somit die Möglichkeit erhalten, wesentlich längere Texte anzuzeigen.

Die Vorteile

Es ist eigentlich offensichtlich: Wenn ein Text getippt wird und die alten Buchstaben verschwinden, können wir wesentlich mehr Text in einem Kreis anzeigen. Im vorherigen Tutorial haben wir im Beispiel nur 16 Zeichen angezeigt. Es wären natürlich mehr möglich gewesen, aber dann wären die Zeichen sehr eng zusammen gewesen oder wir hätten den Radius erweitern müssen.

https://www.bytegame.de/wp-content/uploads/2023/08/Textscroller-getippt-YT.mp4

Im nachfolgenden Beispiel verwenden wir 20 Zeichen. Sobald ein neues Zeichen (ab 20) hinzukommt, wird das erste Zeichen bzw. das 20. wenn man rückwärts zählt, gelöscht.

Da zeigt sich auch ein weiterer Vorteil: Dadurch, dass wir Zeichen für Zeichen tippen, baut sich der Kreis langsam auf. Das sieht hübscher aus, als den Text sofort komplett anzuzeigen, auch wenn wir vielleicht gar keinen langen Text zeigen wollen.

GML-Neuerungen

Zwischen den beiden Tutorials gab es natürlich wieder Updates von GameMaker Studio. U. a. gab es dabei auch Änderungen an der Sprache GML, denen ich hier Rechnung tragen möchte. Kurz und gut: der Befehl `room_speed`, den ich eigentlich ziemlich oft verwende, existiert nicht mehr. Stattdessen definieren wir nun die Spielgeschwindigkeit mit `game_set_speed(60, gamespeed_fps)` und fragen diese per `game_get_speed(gamespeed_fps)` ab.

Außerdem meckert GameMaker nun bei der Namensgebung. Lokale Variablen sollen alle mit einem

Unterstrich geschrieben werden. Bspw. `_i` in for-Schleifen. Ich versuche mich in Zukunft daran zu halten. Schriftarten sollen jedoch alle mit `fnt_` benannt werden und nicht, wie ich es im Beispiel mache, mit `f_`.

Keine Sorge, der Code funktioniert auch noch, wenn ihr es anders macht, aber zur Bevormundung werden solche Namen und Variablen blau unterstrichen.

Das Prinzip

Doch wie funktioniert das nun mit dem tippenden Text? In der vorherigen Version war das noch einfach, weil wir den Text in der Variable `txt` hatten und im Draw-Event per Schleife die Position jedes Zeichens neu gesetzt haben. Das ist hier nun ähnlich, aber wir schreiben die Variable in ein Array um. Hier können wir jedes Zeichen bequem „anfassen”.

Der Ablauf ist somit wie folgt: Im Create-Event definieren wir wie gehabt unsere Ausgangsparameter und den Text. Allerdings definieren wir, wie viele Zeichen angezeigt werden dürfen und schreiben die Zeichen unseres Textes in das Array `txt_array`.

Im Draw-Event wird das nun gezeichnet, wobei die Schleife lediglich zwischen `last_letter_index` und `max_letters_in_circle` liegt. Das bedeutet, dass sich eine Variable laufend verändern muss, damit die nächsten Zeichen kommen. Dies ist last_letter_index, welches wir im Alarm[0]-Event hochzählen.

Wer mitgedacht hat, dem werden bei der verkürzten Erklärung ein paar Ungereimtheiten aufgefallen sein, aber die schauen wir uns im Code an.

Create-Event

```
game_set_speed(60, gamespeed_fps);
xx = room_width/2;
yy = room_height/2;
x = xx;
y = yy;

txt = „ www.Bytegame.de     Wir haben sehr viel Spass mit scrollenden Texten.
rotation = 45;
rotation_speed = 0.05;
radius = 40;

// Initialisiere einen Counter für den letzten Buchstaben
last_letter_index = 0;

// Initialisiere das Array für den Text
txt_array = [];

// Füge jeden Buchstaben des Textes dem Array hinzu
for (var _i = 0; _i < string_length(txt); _i++)
{
 txt_array[_i] = string_char_at(txt, _i);
```

```
}

// Setze die maximale Anzahl an Buchstaben, die im Kreis angezeigt werden soll
max_letters_in_circle = 20;

col = #FFFFFF; // Textfarbe

alarm[0] = 1;
```

Das war ziemlich einfach. Wir definieren unsere Parameter, initialisieren ein paar Variablen und definieren den Text. Außerdem fügen wir dem Array `txt_array` alle Buchstaben nacheinander zu. Damit können wir dann später arbeiten. Am Ende rufen wir den Alarm auf.

Alarm-Event

```
// Füge 1 zum Counter hinzu
last_letter_index++;

// Setze den Alarm erneut, um den nächsten Buchstaben in einer bestimmten Zeit
alarm[0] = round(game_get_speed(gamespeed_fps)/3.05);
```

Das war noch einfacher. Wir zählen die Variable `last_letter_index` hoch. Außerdem starten wir den Alarm neu. Die Zahl sieht etwas komisch aus, aber es sorgt dafür, dass die neuen Buchstaben bei dieser Textlänge und Geschwindigkeit immer in der oberen Hälfte, bevorzugt um die 12 Uhr Position, erscheint. Damit ist es einfacher zu lesen.

Draw-Event

```
// Setze die Schriftart
draw_set_font(f_pixel_8);

// Berechne die Rotation des gesamten Textes
rotation += rotation_speed;

for (var _i = clamp(last_letter_index - max_letters_in_circle, 0, last_letter_
{
 // Stelle sicher, dass der Index im gültigen Bereich bleibt
 var _index = _i % array_length(txt_array);
 if (_index < 0) _index += array_length(txt_array);

 // Berechne die Position des Objekts entlang des Kreises
 var _angle = 360 / max_letters_in_circle * (_index - rotation);
 var _xx = xx + radius * cos(degtorad(_angle));
 var _yy = yy + radius * sin(degtorad(_angle));

 // Zeichne den Text an der berechneten Position und im berechneten Winkel
 draw_text_ext_transformed_color(_xx, _yy, txt_array[_index], 24, _angle / (pi
}
```

Das Meiste ist aus dem vorherigen Tutorial bekannt. Wir definieren unsere Schrift, lassen den Text im Kreis drehen und haben eine for-Schleife. Sogar die Winkelfunktion hat sich kaum verändert. Die

großen Unterschiede sind folgende:

Zuvor haben wir die Schleife so gestartet:

```
for (i = 0; i <= string_length(txt); i++)
```

Und jetzt so:

```
for (var _i = clamp(last_letter_index - max_letters_in_circle, 0, last_letter_
```

Clamp

Wie man sehen kann, ist `i` bzw. `_i` nicht wirklich unser Problem. Doch wir starten nicht mehr konsequent bei 0, sondern sagen, dass `_i` das hier ist: `clamp(last_letter_index - max_letters_in_circle, 0, last_letter_index)`.

Der Befehl `clamp`, den es in vielen Programmiersprachen gibt, erwartet drei Parameter: den eigentlichen Wert, Min und Max. Wir geben also eine Zahl vor und sagen, dass der Ausgabewert zwischen Min und Max liegen muss bzw. diese Werte nicht Unter- oder Überschreiten darf. Wenn der Wert sich dazwischen befindet, ist alles in Butter. Liegt er darunter, gilt Min, liegt er darüber, gilt Max.

Der Wert, den wir clamp schicken, ist `last_letter_index - max_letters_in_circle`. Also das letzte Zeichen (wir beginnen bei 0 und zählen laufend hoch) abzüglich `max_letters_in_circle`, was im Beispiel 20 entspricht.

Wenn wir bei 0 starten und 20 abziehen, liegt der Wert bei -20, `clamp` gibt somit 0 zurück. Das tut es, bis Min überschritten wurde.

Der Rest sieht so aus, dass die Schleife anschließend von dem Wert, was clamp zurückgibt, bis `last_letter_index` schrittweise durchgeht.

Korrektur

Jetzt haben wir noch diese beiden Zeilen:

```
var _index = _i % array_length(txt_array);
if (_index < 0) _index += array_length(txt_array);
```

Wir müssen ja die richtigen Buchstaben bzw. Zeichen anzeigen. Dafür haben wir die Variable `_index`, welche wir in `draw_text_ext_transformed_color()` (letzte Zeile der Schleife) verwenden. Im Alarm zählen wir aber endlos nach oben. Generell ist es dann so, dass _i eine Zahl sein kann, die wesentlich höher ist als die Länge des Textes.

Bei `_i % array_length(txt_array)` passiert folgendes: Statt `%` könnte man in GML auch `mod` schreiben. Das führt dazu, dass wir den Rest einer Division erhalten, also die Nachkommastelle.

Da `_i` immer weiter steigt, wäre der `_index` irgendwann in einem ungültigen Bereich. Durch `mod` geben wir aber nur den Restwert zurück, was folgendes Zeigt:

Im Beispiel hat unser Text 158 Zeichen (Leerzeichen nicht vergessen!). Bis 157 (wir beginnen ja bei 0) ist `_index` identisch mit `_i`. Sobald `_i` 158 ist, ist `_index` gleich 0, bei 159 ist es 1 usw.

Die Zeile `if (_index < 0) _index += array_length(txt_array)` korrigiert dann noch den Fall, dass `_index` kleiner ist als 0.

Weiterführende Links

Old School Textscroller
Bitmap-Fonts im GameMaker
Textscroller: Wellen und einzelne Farben
Texteingabe in GMS
Shader-Tutorial Teil 1

Date Created
11. August 2023
Author
sven