Bitmap-Fonts im GameMaker

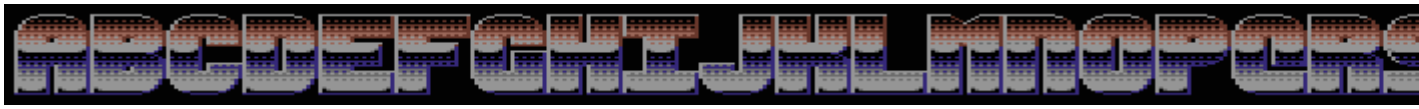
Description

Ist ist sehr einfach, in GameMaker eine Schriftart zu laden und damit zu arbeiten. Doch es ist nicht immer das, was man will. Klassische Bitmap-Fonts sehen stylischer aus, vor allem, wenn mehr Retro-Feeling aufkommen soll.

Was sind Bitmap-Fonts?

Normalerweise arbeiten wir mit gewöhnlichen Schriftarten, die das Betriebssystem geladen hat. Diese stellen wir auf unsere Bedürfnisse ein (Größe, Zeichen) und können sie mit *draw_set_font()* laden.

Bitmap-Fonts sind Grafiken, auf denen sich die Zeichen der Schrift befinden. Sie wurden vor allem in den 1980er und 1990er Jahren eingesetzt. Jeder kann somit eine eigene Schriftart pixeln und sie für sein Spiel nutzen. Für dieses Tutorial habe ich eine Schrift von einer C64-Seite genutzt.


C64 Bitmap Font

Dort kann man die benötigte Schrift auswählen. Es gibt aber zwei Einschränkungen:

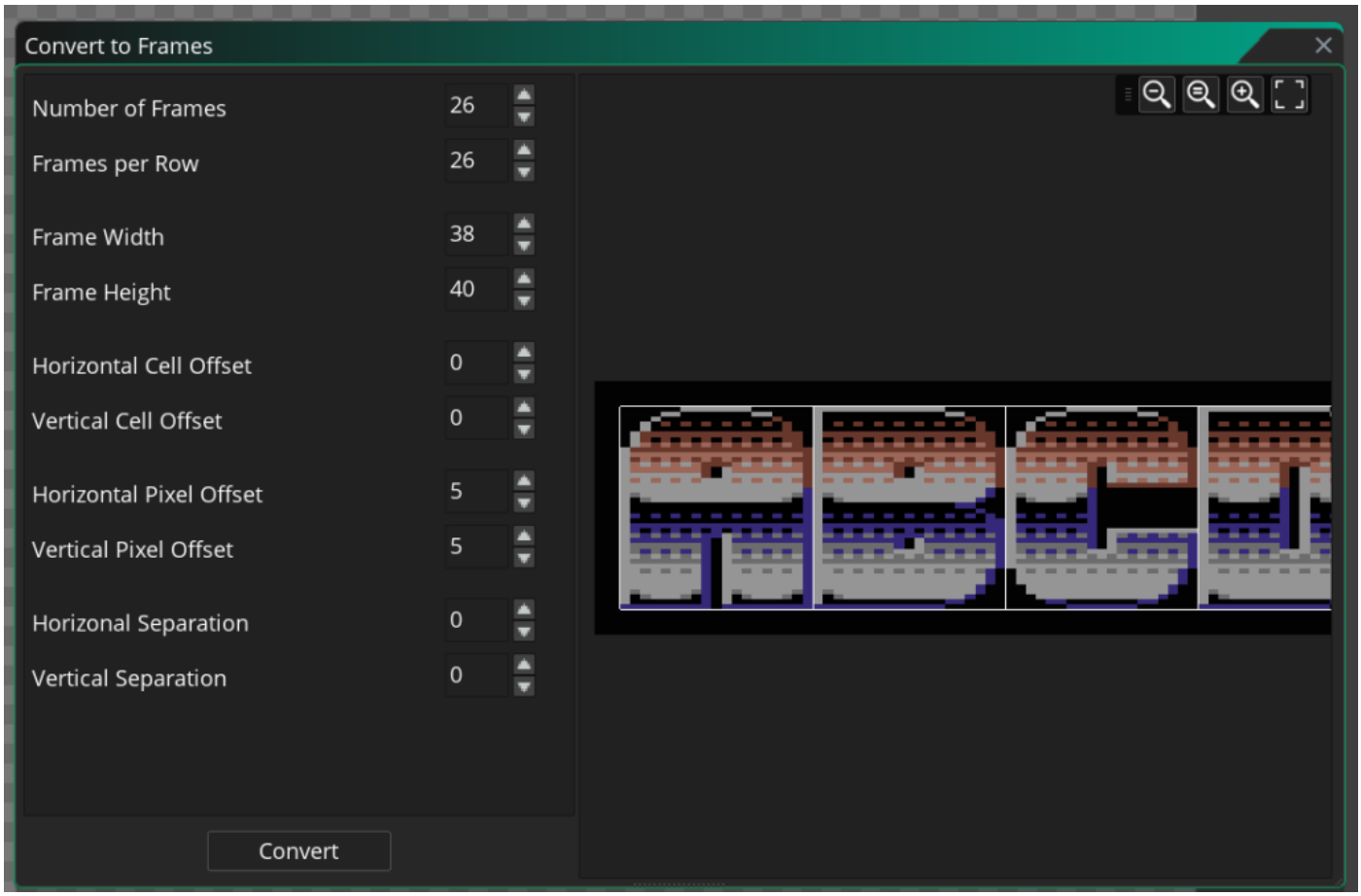
1. Man kann nur die Buchstaben a bis z wählen. Keine Sonderzeichen, Umlaute, Satzzeichen oder Zahlen.
2. Wir benötigen ein Bitmap-Bild mit einer Schrift mit fester Breite. Solche Schriften nennt man auch Mono-Spaced-Schriften.

Auf anderen Seiten, etwa GitHub, findet man noch viele andere Schriften, oder man pixelt selbst. Bspw. wäre es auch denkbar, in Blender eine 3D-Schrift zu erstellen, mit der man arbeitet.

Wichtig: Die nachfolgende Funktion stellt ein Grundgerüst dar. Je nach Schriftart und zusätzlichen Zeichen muss nachgebessert werden.

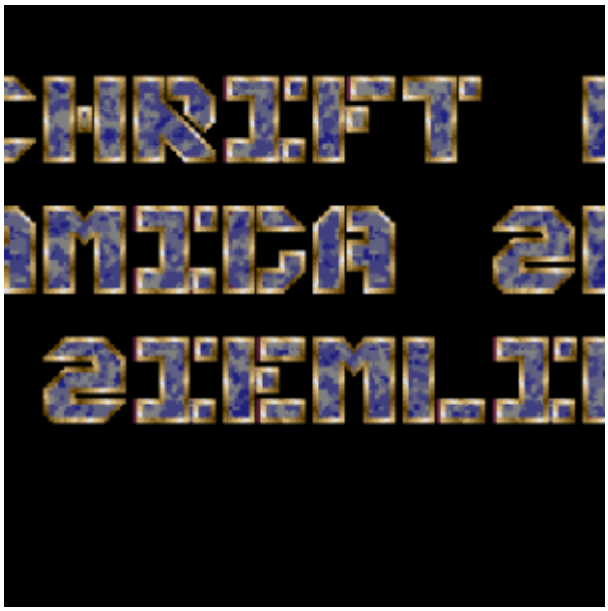
Schrift einladen

Egal wie wir es anstellen, am Ende brauchen wir eine Schrift, am besten als PNG oder GIF (siehe oben). Diese importieren wir als *Import Strip Image*. Dann müssen wir nur noch die richtigen Einstellungen finden.



Schrift importieren

Das funktioniert natürlich auch mit Schriften, deren Buchstaben nicht alle in einer Reihe sind.



Die Funktion

```
/// Zeichnet einen Text mit einem Bitmap-Font
/// Quelle: Bytegame.de
/// @param text
/// @param x
/// @param y
/// @param sprite
/// @param vertical distance
/// @param scale
/// @param color
/// @param alpha
/// Beispiel: draw_bitmap_text("Test", 64, 64, spr_font, sprite_get_height(spr
function draw_bitmap_text(text, xx, yy, bitmap, vDist, scale, col, alpha)
{
 var letPos = 0;
 var xStart = xx;

 for (var i=1; i<string_length(text)+1; i++)
 {
  var char = string_byte_at(text, i) - 97;
```

```
  if (char == -87)
  {
   // Zeilenumbruch
   xx = xStart;
   yy += sprite_get_height(bitmap)*scale+vDist;
   letPos = -1;
  } else if (char != -65) {
   // Wenn kein Leerzeichen...
   draw_sprite_ext(bitmap, char, xx+letPos*sprite_get_width(bitmap)*scale, yy,
  }

  letPos++;
 }
}
```

Die Funktion ist relativ kurz und wir können einige Parameter einstellen. So lässt sich die Schrift bspw. skalieren sowie Farbe und Alpha-Wert ändern. *letPos* ist dabei die Zeichenposition pro Zeile.

Die Funktion besteht aus einer for-Schleife, die drei Zustände bearbeitet:

1. Zeilenumbrüche
2. Leerzeichen
3. Normaler Text

Zentral ist die Variable *char*. Mit *string_byte_at(text, i) – 97* wird dem dem aktuellen Textzeichen ein Wert zugewiesen, der zwischen 0 und 25 liegen sollte (a bis z). Wenn der Wert -87 hat, ist es ein Zeilenumbruch, den wir im Text mit *\n* erzeugen. Wird dieser erkannt, setzen wir *xx* auf *xStart* zurück, yy wandert eine Zeile runter, *letPos* wird auf -1 gesetzt, da es nach der Addition ganz unten wieder auf 0 stehen muss.

Der Wert -65 entspricht einem Leerzeichen. Wenn das aktuelle Zeichen kein Leerzeichen ist, wird der Sprite mit dem Subimage *char* gezeichnet. Wenn es ein Leerzeichen ist, wird es übersprungen und der Abstand ergibt sich über *letPos*.

Eigentlich ziemlich simpel.

Mit weiteren Zeichen lässt es sich so (oder besser durch eine Switch) erweitern:

Bitmap Font mit weiteren Zeichen.

Date Created
17. Juni 2022
Author
sven