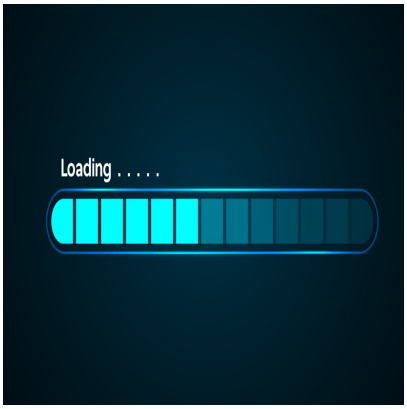

Loading-Sequenz in GameMaker

Description

Normalerweise wird bei einem GameMaker-Projekt alles beim Start geladen. Doch manchmal hat man externe Dateien, etwa 3D-Objekte und deren Texturen. Diese müssen extra geladen werden. Statt einfach nur „Loading…" zu schreiben, kann man auch seinen eigenen Ladebalken und Prozentangabe programmieren.

Die Theorie

Zunächst brauchen wir eine Liste aller Dateien, die geladen werden sollen. Hierfür eignet sich ein Array . Dann müssen wir diese Liste schrittweise abarbeiten. Hierfür nutzen wir in GameMaker einen Alarm. Der Rest ist simple Mathematik, zumal wie den Fortschritt nicht an der Dateigröße, sondern der Anzahl der Dateien angeben.

Hier mein Projekt zur Veranschaulichung:

Der Code

Der Code stammt 1:1 aus einem aktuellen Projekt, an dem ich arbeite.

Create-Event

```
// Initialisiere eine Liste aller zu ladenden 3D-Objekte
global.models_to_load = [
    „3D\\Cube\\cube.obj",
    „3D\\Cube03\\cube03.obj",
    „3D\\Cube04\\cube04.obj",
    „3D\\Diamant\\diamant.obj",
    „3D\\Starship01\\starship-01.obj",
    „3D\\Starship02\\starship-02.obj",
    „3D\\Torus-Low\\torus_low.obj",
    „3D\\Plane\\plane.obj",
    „3D\\SciFi-City\\ground.obj",
```

```
    „3D\\SciFi-City\\scificity.obj",
    „3D\\SciFi-City\\skybox.obj",
    „3D\\Bytegame\\bytegame3d.obj",
    „3D\\Dodekaeder\\dodecahedron.obj",
    „3D\\Monitor\\monitor.obj",
    „3D\\Floppy-Disk\\floppy_disk.obj"
];

// Index der aktuellen Ladung
global.current_load_index = 0;

// Flag, um den Ladevorgang zu überprüfen
global.loading_complete = false;
```

Zunächst definieren wir die Objekte in unserem Array. Theoretisch könnte man auch die Ordner durchsuchen und alle *.obj*-Dateien einladen, aber da ich bei Tests nicht immer alle brauche, fand ich diese Version (für mein Projekt) einfacher. Dann haben wir noch zwei Variablen: Wir geben an, wie viele Objekte bereits geladen wurden und ob alle Dateien geladen wurden. Nun brauchen wir noch den Alarm.

Außerdem müssen wir noch Variablen für den Ladebalken definieren:

```
// Setze die Position und Größe des Fortschrittsbalkens
bar_width = 256; // Breite des Balkens
bar_height = 40; // Höhe des Balkens
bar_x = room_width/2-bar_width/2; // X-Position des Balkens
bar_y = room_height-distanceBelow/2-20; // Y-Position des Balkens
```

Alarm-Event

```
// Schrittweises Laden der 3D-Modelle
if (!global.loading_complete) {
    if (global.current_load_index < array_length(global.models_to_load)) {
        var model_path = global.models_to_load[global.current_load_index];
        var model = DotobjModelLoadFile(model_path);

        // Speichere das geladene Modell in der entsprechenden globalen Variab
        switch (global.current_load_index) {
            case 0: global.Cube1 = model; break;
            case 1: global.cube2 = model; break;
            case 2: global.cube3 = model; break;
            case 3: global.diamond = model; break;
            case 4: global.starship = model; break;
            case 5: global.starship2 = model; break;
            case 6: global.torus1 = model; break;
            case 7: global.plane = model; break;
            case 8: global.ground = model; break;
            case 9: global.scificity = model; break;
            case 10: global.skybox = model; break;
            case 11: global.bytegame = model; break;
            case 12: global.dodecahedron = model; break;
            case 13: global.monitor = model; break;
            case 14: global.floppy_disk = model; break;
```

```
        }

        global.current_load_index++;
        // Setze den Alarm 0 neu, um das nächste Modell zu laden
        alarm[0] = 5;
    } else {
        // Alle 3D-Modelle wurden geladen
        global.loading_complete = true;
    }
}

// Überprüfe, ob alle 3D-Objekte geladen wurden
if (global.loading_complete) {
    canStart = true;
    canClick = true;
}
```

Da meine OBJ-Dateien in verschiedenen GM-Objekten und Räumen genutzt werden, sind es globale Variablen. Somit ist die manuelle Liste aus dem Create-Event durchaus sinnvoll.

Im Alarm-Event gehen wir das Array durch und laden schrittweise die Objekte. So lange nicht alle Objekte geladen sind, ruft sich der Alarm laufend selbst auf. Am Ende, wenn es komplett ist, kann man irgendetwas machen. In meinem Beispiel wird im Draw-Event aus dem Ladebalken ein Start-Button.

Draw-Event

```
draw_text(display_get_gui_width()/2, distanceAbove/2, „Loading");

// Berechne den Fortschritt in Prozent
var progress_percent = (global.current_load_index / array_length(global.models

// Berechne die Breite des gefüllten Teils des Balkens
var filled_width = (progress_percent / 100) * bar_width;

// Zeichne den leeren Balken
draw_set_color(#A1A1A1);
draw_rectangle(bar_x, bar_y, bar_x + bar_width, bar_y + bar_height, false);

// Zeichne den gefüllten Teil des Balkens
draw_set_color(#436424);
draw_rectangle(bar_x, bar_y, bar_x + filled_width, bar_y + bar_height, false);

// Zeichne den Text mit dem aktuellen Fortschritt
draw_set_halign(fa_center);
draw_set_valign(fa_middle);
draw_set_color(c_white);
draw_set_font(f_pixel_12);
draw_text(bar_x + (bar_width / 2), bar_y + (bar_height / 2), string(progress_p
```

Das ist nur der Teil mit der Überschrift und dem Ladebalken inklusive Prozentangabe. Am Anfang haben wir die Überschrift. Dann berechnen wir den Fortschritt.

Der Ladebalken soll sich ja füllen. Somit haben wir die 100%-Breite und die Füllung, welche davon

abhängt, was bereits geladen wurde. Anschließend müssen wir den Ladebalken entsprechend zeichnen.

Natürlich kann der Code noch weiter vereinfacht werden. Du musst es auch an dein Projekt anpassen, aber ich denke, dass es durch diese kleine Vorlage kein Problem sein sollte.

Weiterführende Links

Timing in GameMaker-Projekten
Mehrfache Sortierung von Arrays
Texte im Kreis “tippen”
Shader-Programmierung 1: Grundlagen und Sprites
Shader-Effekt: Tunnel mit Maussteuerung

Date Created
23. Februar 2024
Author
sven