INI Dateien in GMS2

Description

INI Dateien sind nützliche Textdateien, in denen sich Einstellungen und zahlreiche Informationen speichern lassen. Dieses Tutorial zeigt, wie man mit INI-Dateien im GameMaker Studio 2 umgeht.

Was sind INI Dateien?

Manchmal ist es nützlich, Informationen, die das Spiel braucht, extern zu speichern. Der häufigste Anwendungsfall sind die Settings. Wenn der Spieler Einstellungen wie Bildschirmauflösung, Lautstärke, Steuerung und andere Dinge vornimmt, dann hätte es jeder Spieler gern, wenn das Spiel sich diese Einstellungen merken würde.

Im GameMaker gibt es mehrere Methoden um Dateien zu speichern. Für Einstellungen und Profildaten sind INI-Dateien perfekt geeignet. Dabei handelt es sich um Textdateien, die einem bestimmten Aufbau folgen. Eine Einschränkung im GameMaker liegt in der maximalen Größe der Datei: sie darf nicht größer sein als 64kb. Das klingt nach wenig, ist aber für die erwähnten Einstellungen völlig ausreichend. Das Speichern von Leveldaten (Positionen von Objekten etc.) sollte man aber lieber nicht mit INIs durchführen.

Ein weiteres Problem kann bei sensiblen Daten wie Highscorewerten auftreten. Die GameMaker Hilfe gibt zwar explizit die Speicherung von lokalen Highscores als Anwendungsfall an, aber eine normale INI ist unverschlüsselt. Jeder Spieler kann sie mit einem Texteditor öffnen und beliebig die Werte manipulieren. Bei Einstellungen ist das durchaus so gewollt. Wenn der Spieler versehentlich eine Auflösung einstellt, die nicht funktioniert, dann sollte er die Möglichkeit erhalten, von Außen diese Einstellung zu ändern.

Dies bedeutet: Wenn man sensible Daten in eine INI speichern möchte, dann sollte man sie verschlüsseln. Eine Verschlüsselung wird vom GameMaker nicht mitgeliefert. Diese muss man entweder selbst programmieren oder sich entsprechende Skripte im GM-Marktplatz besorgen. Geht man diesen Weg, dann sollten die Settings dennoch unverschlüsselt gespeichert werden. Wir reden also davon, mehrere INI-Dateien anzulegen.

Unterstützt das Spiel mehrere lokale Spielerprofile, dann kann man für jedes Profil eine eigene INI anlegen.

Aufbau einer INI

Eine INI ist immer in Sektionen (section) und Schlüsselwerten (key) unterteilt. Eine INI kann beispielsweise so aussehen:

```
[Settings]
, also das Wort in den eckigen Klammern, ist die Sektion. Theoretisch könnte m


Die Werte Lang, Fullscreen und Theme
 sind die Schlüsselwerte. Die Informationen nach dem Gleichheitszeichen werden
```

INIs erstellen

```
Man kann direkt mit GameMaker INI Dateien erstellen. Das ist extrem praktisch.


Wer INI Dateien manuell erstellt, um etwa Sprachdateien zu erzeugen, sollte di


Jetzt erzeugen wir mal eine INI und schauen uns den Code an.


// Prüft, ob der Ordner Settings existiert. Wenn nicht, wird er erstellt.
if !directory_exists(working_directory + „\Settings\\")
{
    directory_create(working_directory + „\Settings")
}

// INI wird eingelesen. Wenn sie nicht existiert, wird sie erstellt und beschr
if !file_exists("Settings\\" + global.iniFile)
{
    file = file_text_open_write("Settings\\" + string(global.iniFile));   // D
    file_text_write_string(file,"[Settings]");                             //
    file_text_writeln(file);                                            // Lee
    file_text_write_string(file,"Lang="+string(global.activeLanguage)); // Spr
    file_text_writeln(file);                                            // Lee
    file_text_write_real(file,"Fullscreen="+string(global.isFullscreen)); // F
    file_text_writeln(file);                                            // Lee
    file_text_write_real(file,"Theme="+string(global.activeTheme));   // Theme
    file_text_writeln(file);                                            // Lee
    file_text_close(file);
}
```

Wie man sieht, arbeite ich mit globalen Variablen. Das hat sich bei mir auch b

Zunächst schauen wir, ob das Verzeichnis für die INI existiert. Es ist der Ord
Settings.

Mit *!file_exists(“Settings\\” + global.iniFile)*
schauen wir, ob bereits eine INI existiert. Der Dateiname ist bei mir immer ei
global.iniFile. Meistens nennt man sie *settings.ini*
. Wenn die Datei noch nicht existiert, wird sie mit den Standardwerten erzeugt

file = file_text_open_write()
erzeugt die Datei. Genau genommen erzeugt der Code nur dann eine Datei, wenn

Kleiner Hinweis: Früher musste man die Datei vorher kopieren, bevor man sie be

Mit *file_text_write_string(file,”[Settings]”);*
legen wir die Sektion an. In unserem Fall heißt sie *[Settings]*.

file_text_writeln(file)
legt eine neue Zeile an. Anschließend schreiben wir mit jedem
file_text_write_string bzw. *file_text_write_real*
eine neue Zeile in die INI. Der Vorgang ließe sich mit weiteren Einstellungen

Am ende wird die Datei mit *file_text_close(file)*
geschlossen und damit auch gespeichert.

Im Prinzip ist es also ganz einfach.

INI laden

```
if (file_exists("Settings\\" + global.iniFile))
{
 ini_open("Settings\\" + global.iniFile);           // INI Datei wird geöffnet
 global.activeLanguage = ini_read_string("Settings", „Lang", „de"); // Sprache
 global.isFullscreen = ini_read_real("Settings", „Fullscreen", 1); // Fullscre
 global.activeTheme = ini_read_real("Settings", „Theme", 1);   // Theme
 ini_close();               // INI Schließen
}
```

Das Laden ist ebenso einfach. Erst schauen wir, ob die INI existiert. Dann wir
zwingend

zwischen String und Real unterscheiden.

Schauen wir uns exemplarisch *ini_read_real(“Settings”, „Fullscreen”, 1)* an. Das erste Argument zielt auf die Sektion *“Settings”*. Das zweite Argument schaut nach dem Key *“Fullscreen”*. Sollte der nicht existieren, wird ein Default-Wert benutzt. In diesem Fall *1*. Beim laden der INI ist somit ganz wichtig, dass wir String und Real untersch

INI speichern

```
if file_exists("Settings\\" + global.iniFile)
{
 ini_open("Settings\\" + global.iniFile);        // INI Datei wird geöffnet
 ini_write_string("Settings","Lang",string(global.activeLanguage));  // Sprach
 ini_write_string("Settings","Fullscreen",string(global.isFullscreen)); // Ful
 ini_write_string("Settings","Theme",string(global.activeTheme));  // Theme
 ini_close();                // INI Schließen
}
```

Ich persönlich verwende immer *ini_write_string*. Wie oben beschrieben, ist die Unterscheidung nur beim Lesen wichtig. GMS set*ini_write_string* oder *ini_write_real* benutzt. Der Unterschied ist nur, dass GMS statt einer *3* ein *3.00000000* schreibt, wenn man Real benutzt.

Das Prinzip sollte aber klar sein. Wir schauen wieder, ob die Datei existiert*ini_write_string* geben wir erneut die Sektion, den Schlüsselwert und zuletzt die Variable an.

Praktische Anwendung

Am besten setzt man die Codes in drei Skripte. **scr_ini_create** beispielsweise ruft man beim Spielstart auf, gefolgt von **scr_ini_load**. Es hat sich bewährt, globale Variablen in ein extra Skript zu speichern, so

- scr_global_variables();
- scr_ini_create();
- scr_ini_load();

Wir holen uns erst die ganzen Variablen. Dann wird geschaut, ob die INI existiert und ggf. angelegt. Falls sie existiert, wird die INI ausgelesen und die Einstellungen geladen. Das bedeutet zwar, dass die Variablen doppelt bzw. dreifach beim Spielstart angepackt werden, aber zeitlich macht das keinen spürbaren Unterschied und ist die sicherste Methode.

Bei **scr_ini_save** gibt es unterschiedliche Ansätze. Manche speichern die Werte erst, wenn das Spiel beendet wird. Das kann nützlich sein, falls das Spiel aufgrund unbedachter Einstellungen abstürzt. Auf der anderen Seite kann ein Spiel auch unabhängig von den Einstellungen abstürzen und dann wurden womöglich wichtige Einstellungen nicht gespeichert.

Beispiel: Das Spiel hat eine komplexe Steuerung. Im Level stürzt das Spiel wegen eines Bugs, oder einfach nur, weil Windows einen schlechten Tag hat, ab. Dann kann der Spieler die ganzen Einstellungen neu vornehmen. Die meisten Spieler treten das Spiel genau in diesem Moment in den Papierkorb. Deshalb: Am besten speichert man das Spiel bei jeder Änderung der Einstellungen ab. Sollte es Probleme geben, kann der Spieler die INI immer noch löschen oder von Hand anpassen. Das ist auf jeden Fall der kleinere Schaden.

Wie das Ganze in der Praxis aussieht, zeige ich noch in einem kurzen Video:

Date Created
6. Juni 2018
Author
sven