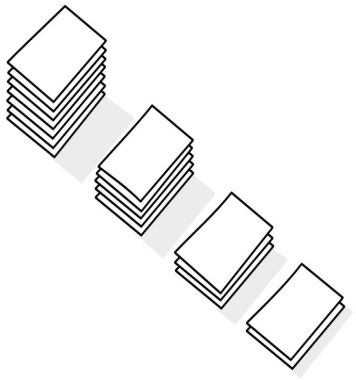

Mehrfache Sortierung von Arrays

Description

Häufig steht man bei Spielen vor dem Problem, dass man Tabellen (meistens Arrays) nach mehreren Kriterien sortieren möchte. Dieses Tutorial zeigt anhand einer Sport-Tabelle, wie man das einfach realisiert.

Die Tabelle

In einem der letzten Tutorials ging es um einen Spielplangenerator. Diesen Gedanken möchte ich aufgreifen, um das Tutorial zu erklären. Angenommen, wir haben ein Spiel, in dem in einem Ligamodus mehrere Spieler gegeneinander spielen. Nach jedem Spieltag wollen wir eine Tabelle sehen. Diese sieht zufällig wie eine Fußballtabelle aus.

#	Mannschaft	Sp.	S	U	N	Tore	Diff.	P
1.	1. FC Hoch und weit	23	14	7	2	66:22	44	4
2.	FC Blutgrätsche	23	16	1	6	47:28	19	4
3.	Borussia Maulkorb	23	13	5	5	35:27	8	4
4.	SC Dunkelwald	23	12	6	5	46:29	17	4
5.	Eintracht Zwietracht	23	12	6	5	35:32	3	4
6.	Ajax Dauerstramm	23	11	6	6	45:33	12	3
7.	Dynamo Tresen	23	10	5	8	38:34	4	3
8.	FC Saufhemden	23	9	7	7	42:31	11	3
9.	Glashoch Rangers	23	9	4	10	40:37	3	3
10.	AS Koma	23	8	6	9	38:39	-1	3
11.	Hangover 96	23	9	3	11	35:43	-8	3
12.	Inter Heiland	23	6	9	8	32:36	-4	2
13.	VFL Vollsuff	23	8	3	12	28:39	-11	2
14.	SC Heiß Haufen	23	5	5	13	28:44	-16	2
15.	Feiern 04 Leberbluten	23	4	7	12	28:40	-12	1
16.	AS Pirin	23	5	4	14	29:42	-13	1
17.	FC Biercelona	23	4	7	12	18:42	-24	1
18.	Arminia Bierzelt	23	6	1	16	24:56	-32	1

Sortierte Tabelle

Wir haben Platzierung, Name, Spiele, Siege, Unentschieden, Niederlagen, Tore, Tordifferenz und die Punkte. Sortiert wurde die Tabelle wie folgt:

- Alphabetisch
- nach Punkten
- nach der Tordifferenz
- Siege
- Anzahl Spiele

Das ist ein bisschen komplexer, aber ich denke, dass viele von euch mehrere solcher Kriterien suchen. Bevor es los geht, brauchen wir aber ein Array, welches sortiert werden kann.

Array und Structs

Bevor wir Daten verarbeiten, müssen wir uns darüber Gedanken machen, wie wir Daten erfassen. Um GameMaker Studio gibt es dazu sehr viele verschiedene Methoden, die alle ihre Vor- und Nachteile haben. Ich bin generell ein großer Fan von Arrays, weil die in nahezu jeder Sprache funktionieren und ich dadurch die Option behalte, Code einfach von einer Sprache in die andere zu konvertieren (etwa in PHP oder JavaScript).

Nun gibt es aber auch die Möglichkeit von **Structs**. Ein Struct ist eine Variable, die eine Sammlung von anderen Variablen enthält. Die Variablen, die ein Struct enthält, können verschiedene Datentypen haben, und diese Variablen können nach der anfänglichen Struct-Deklaration gelesen und beschrieben werden.

Man kann Structs und Arrays kombinieren, was im Code große Vorteile bringt. Um das zu demonstrieren, zeige ich gleich das **Create-Event**:

```
arr_clubs[0] = { name: „Ajax Dauerstramm",  league: 1, wins: 11, draws: 6, def
arr_clubs[1] = { name: „Hangover 96",   league: 1, wins: 9, draws: 3, defeat:
arr_clubs[2] = { name: „Arminia Bierzelt",  league: 1, wins: 6, draws: 1, defe
arr_clubs[3] = { name: „SC Dunkelwald",   league: 1, wins: 12, draws: 6, defea
arr_clubs[4] = { name: „Eintracht Zwietracht", league: 1, wins: 12, draws: 6,
arr_clubs[5] = { name: „1. FC Hoch und weit", league: 1, wins: 14, draws: 7, d
arr_clubs[6] = { name: „Dynamo Tresen",   league: 1, wins: 10, draws: 5, defea
arr_clubs[7] = { name: „FC Saufhemden",   league: 1, wins: 9, draws: 7, defeat
arr_clubs[8] = { name: „Glashoch Rangers",  league: 1, wins: 9, draws: 4, defe
arr_clubs[9] = { name: „AS Koma",    league: 1, wins: 8, draws: 6, defeat: 9,
arr_clubs[10] = { name: „FC Blutgrätsche",  league: 1, wins: 16, draws: 1, def
arr_clubs[11] = { name: „Inter Heiland",  league: 1, wins: 6, draws: 9, defeat
arr_clubs[12] = { name: „VFL Vollsuff",   league: 1, wins: 8, draws: 3, defeat
arr_clubs[13] = { name: „SC Heiß Haufen",  league: 1, wins: 5, draws: 5, defea
arr_clubs[14] = { name: „Feiern 04 Leberbluten",league: 1, wins: 4, draws: 7,
arr_clubs[15] = { name: „AS Pirin",    league: 1, wins: 5, draws: 4, defeat: 1
arr_clubs[16] = { name: „FC Biercelona",  league: 1, wins: 4, draws: 7, defeat
arr_clubs[17] = { name: „Borussia Maulkorb", league: 1, wins: 13, draws: 5, de

arr_table = give_club_table(arr_clubs, 1);
```

Wir haben das Array *arr_clubs* angelegt. Hier befinden sich die 18 Vereine (0 bis 17). Doch anstatt wie in einem 2D-Array mit Nummern zu arbeiten, enthält jedes Array weitere Variablen. In diesem Fall:

- league (die Liga, falls es mehrere Ligen geben sollte)
- wins
- draws
- defeat
- go (die Tore)
- goa (die Gegentore)
- player

Im Beispiel wird es drei menschliche Spieler geben, die in der Tabelle mit ihren eigenen Farben

angezeigt werden.

Wenn man sich das anschaut, ist der Vorteil von Structs offensichtlich: Statt bspw. nach *arr_table[i][2]* zu suchen, suchen wir *arr_table[i].wins*. So lässt sich der Code leichter schreiben und lesen.

Wichtig ist noch folgendes: Das Array *arr_clubs* wird nicht sortiert. Hier verändern wir lediglich die Werte. Sortiert wird das Array *arr_table*. Dafür nutzen wir die Funktion *give_club_table*, welche wiederum auf drei kleine Funktionen zugreift.

Die Hilfsfunktionen

Wie man aus dem Array sehen kann, speichern wir nicht die Anzahl der gespielten Spiele, Punkte und die Tordifferenz. Das berechnen wir, bevor die Tabelle sortiert wird. Da es jeweils nur eine Zeile ist, könnte man das auch in die Hauptfunktion legen, aber ich fand es so eleganter. Hier sind die drei selbsterklärenden Funktionen:

```
// Anzahl der Spiele in der laufenden Saison
function club_all_games(club)
{
 return club.wins + club.draws + club.defeat;
}

// Anzahl der Punkte in der laufenden Saison
function club_points(club)
{
 return club.wins * 3 + club.draws;
}

// Anzahl der Punkte in der laufenden Saison
function club_goal_difference(club)
{
 return club.go - club.goa;
}
```

Array sortieren

Nun geht es an die Hauptfunktion. Zunächst der Code:

```
function give_club_table(arr, league)
{
 var arr_table = [];

 for (var i=0; i<array_length(arr); i++)
 {
  if (arr[i].league == league)
  {
   arr_table[i] = arr[i];
   arr_table[i].score = club_points(arr_table[i]);
   arr_table[i].games = club_all_games(arr_table[i]);
   arr_table[i].diff = club_goal_difference(arr_table[i]);
  }
```

```
}

// Sortiere Array
for (i=0; i<array_length(arr_table); ++i)
{
 for (var j=i+1; j<array_length(arr_table); ++j)
 {
  // Alphabetisch
  if (string_lower(arr_table[i].name) > string_lower(arr_table[j].name))
  {
   var a = arr_table[i];
   arr_table[i] = arr_table[j];
   arr_table[j] = a;
  }

  // Punkte
  if (arr_table[i].score < arr_table[j].score)
  {
      var a = arr_table[i];
      arr_table[i] = arr_table[j];
      arr_table[j] = a;
  }

  // Tordifferenz
  if (arr_table[i].score == arr_table[j].score)
  {
   if (arr_table[i].diff < arr_table[j].diff)
   {
       var a = arr_table[i];
       arr_table[i] = arr_table[j];
       arr_table[j] = a;
   }
  }

  // Siege
  if (arr_table[i].diff == arr_table[j].diff)
  {
   if (arr_table[i].wins < arr_table[j].wins)
   {
       var a = arr_table[i];
       arr_table[i] = arr_table[j];
       arr_table[j] = a;
   }
  }

  // Spiele
  if (arr_table[i].wins == arr_table[j].wins)
  {
   if (arr_table[i].games < arr_table[j].games)
   {
       var a = arr_table[i];
       arr_table[i] = arr_table[j];
       arr_table[j] = a;
   }
  }
```

```
  }
 }

 return arr_table;
}
```

Wir übergeben zwei Argumente: Array und Liga. Zuerst deklarieren wir die lokale Variable *arr_table*. Daraufhin gehen wir das übergebene Array *arr* in einer Schleife durch. Wenn der Verein der jeweiligen Liga entspricht, übergeben wir die Zeile an das lokale Array. Anschließend berechnen wir Punkte, Spiele und Differenz mit den Hilfsfunktionen.

Sobald das geschafft ist, haben wir das lokale Array vorbereitet und können es nun sortieren. Dafür benötigen wir zwei Schleifen. In der ersten Schleife greifen wir uns einen Verein. Dann gehen wir in der zweiten Schleife alle Vereine durch, die darauf folgen.

Zuerst sortieren wir das Array **alphabetisch**. In der Praxis ist das vor allem vor dem ersten Spieltag wichtig, da wir keine anderen Werte haben bzw. alles andere auf null steht. Mit **string_lower** wandeln wir intern die Vereinsnamen in Kleinbuchstaben um und Vergleichen dann ihre „größe“. Bspw. ist „b“ größer „a“, „c“ größer „b“ etc. Wenn aus den beiden Schleifen *i* größer ist als *j*, tauschen die Vereine die Plätze. Hierfür wird die Hilfsvariable *a* benutzt.

Nachdem das getan ist, geht es an die **Punkte**. Im Kern funktioniert das ebenso, nur verschieben wir hier *i* nur, wenn es kleiner ist als *j*.

Ab der **Tordifferenz** läuft es etwas anders, weil hier erst sichergestellt werden muss, dass die Punktzahl gleich ist. So geht das stufenweise weiter. Ist die Tordifferenz ebenfalls identisch, geht es um die **Siege**. Man könnte hier natürlich auch nach der Anzahl erzielter Tore sortieren. Und wenn das ebenfalls gleich ist, sortieren wir nach den **Spielen**, wobei die Mannschaft höher steht, die mehr Spiele gespielt hat.

Noch einmal zum Verständnis: Wenn die Punkte unterschiedlich sind, spielt der Rest keine Rolle mehr.

Das Prinzip lässt sich beliebig erweitern, etwa mit der Anzahl der Karten, Platzverweise, Fans und was auch immer.

Draw-Event

Die sortierte Tabelle soll angezeigt werden. Hier ein Beispiel:

```
draw_set_color(#FFFFFF);
draw_set_font(fnt_table);
draw_set_alpha(1);

var startX = 20;
var startY = 20;

// Header
draw_text(startX+20, startY+16, „#");
draw_text(startX+60, startY+16, „Mannschaft");
draw_text(startX+380, startY+16, „Sp.");
```

```
draw_text(startX+420, startY+16, „S");
draw_text(startX+460, startY+16, „U");
draw_text(startX+500, startY+16, „N");
draw_text(startX+540, startY+16, „Tore");
draw_text(startX+630, startY+16, „Diff.");
draw_text(startX+690, startY+16, „Pkt.");

for (var i=0; i<array_length(arr_table); i++)
{
 switch arr_table[i].player
 {
  case 0: draw_set_color(#FFFFFF);
  break;
  case 1: draw_set_color(#E7FE3B); // Spieler 1
  break;
  case 2: draw_set_color(#3BD6FE); // Spieler 2
  break;
  case 3: draw_set_color(#5EFE3B); // Spieler 3
  break;
 }

 // Array
 draw_text(startX+20, startY+50+30*i,  string(i+1) + „.");
 draw_text(startX+60, startY+50+30*i,  string(arr_table[i].name)); // Name
 draw_text(startX+380, startY+50+30*i, string(arr_table[i].games)); // Gespiel
 draw_text(startX+420, startY+50+30*i, string(arr_table[i].wins)); // Siege
 draw_text(startX+460, startY+50+30*i, string(arr_table[i].draws)); // Unentsc
 draw_text(startX+500, startY+50+30*i, string(arr_table[i].defeat)); // Nieder
 draw_text(startX+540, startY+50+30*i, string(arr_table[i].go) + „:" + string(
 draw_text(startX+630, startY+50+30*i, string(arr_table[i].diff)); // Tordiffe
 draw_text(startX+690, startY+50+30*i, string(arr_table[i].score)); // Punkte
}
```

Mit *startx* und *starty* kann man die Tabelle beliebig verschieben. In der Switch unterscheiden wir die Farben für die drei Spieler. Selbstredend ist, dass wir erst die Kopfzeile der Tabelle zeichnen und dann in der Schleife die einzelnen Vereine darstellen.

Mögliche Variationen

In solchen Spielen gibt es häufig die Möglichkeit, dass verschiedene Arten von Tabellen angezeigt werden. So gibt es eine sehr ausführliche Tabelle und vielleicht eine verkürzte Fassung, die lediglich Mannschaft, Spiele, Differenz und Punkte anzeigt.

Dafür kann man entweder die vorhandene Funktion anpassen oder eine neue schreiben. Dann wird auch der Vorteil der Hilfsfunktionen deutlich: Man muss später nicht in jeder Hauptfunktion die Berechnungen ändern, sollte dies erforderlich sein.

Beispiele für weitere Anwendungen gibt es genug. Man kann das auch bei einem Array für die besten Torschützen machen, Vereine mit Titeln, eine Highscore-Liste, die mehr Werte hat als nur die Punktzahl u. v. m.

Viel Spaß dabei!

Date Created
31. März 2023
Author
sven