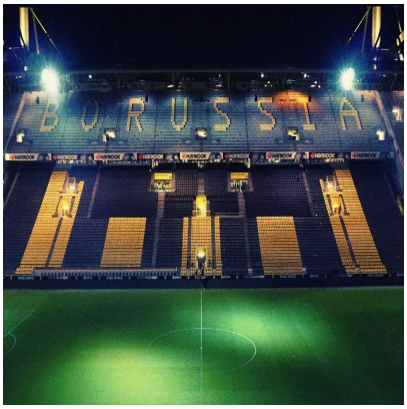

Spielplangenerator für den Ligamodus

Description

Spielpläne sind das Fundament von Turnieren. Egal ob Mannschaften oder einzelne Spieler, ob Fußball oder Schach: Ohne einen Spielplan lässt sich kein vernünftiges Turnier organisieren. Bei KO-Turnieren ist – die richtige Anzahl Spieler vorausgesetzt – die Sache noch relativ einfach. Doch bei einer Liga wird es schon etwas komplizierter. Mit der hier vorgestellten Funktion kann man das Problem einfach lösen.

Die Lage der Liga

Eine Liga zu organisieren ist nicht ganz einfach. Wenn es wenige Teilnehmer sind, lässt sich das noch händisch regeln, doch sobald es sich um mehrere Teams oder Einzelspieler handelt, wird es komplex. Z. B. brauchen wir die Anzahl der Spiele und Runden. Nehmen wir einmal die Bundesliga als Beispiel, in der 18 Mannschaften spielen. Jede Mannschaft spielt gegen die 17 anderen je zweimal, als Hin- und Rückrunde.

Anzahl Spieltage = (Anzahl Teams – 1) * 2 = 34 Spieltage
Anzahl Spiele = Anzahl Teams * Anzahl Teams – Teams = 306

Die zweite Formel lässt sich als x^2-x ausdrücken.

Das ist alles noch einfach. Doch bei einem Spielplan muss noch berücksichtigt werden, dass:

- jedes Team gegen das andere Team in Hin- und Rückrunde je einmal spielt
- das Heim- und Auswärtsspiele möglichst gleichmäßig verteilt werden*
- das Hin- und Rückspiele an den „gleichen“ Spieltagen stattfinden, bspw. 1. und 18. Spieltag die gleichen Teams gegeneinander aber mit verdrehtem Heimrecht antreten
- das bei einer ungeraden Anzahl Teams immer ein Team spielfrei hat

*: Bei Spielen wie Schach geht es i. d. R. darum, die Farben gleichmäßig zu verteilen

Es gibt somit einige Probleme zu lösen.

Blaupausen

Diese Probleme sind nicht neu und wurden in vielen Programmiersprachen auf teils sehr unterschiedliche Art gelöst. Ich wurde auf folgendes PHP-Skript aufmerksam gemacht, welches recht gut funktioniert: https://www.klamm.de/forum/threads/php-spielplan-script.279919/#post-4683872

Meine Aufgabe bestand darin, dies in eine GML-Version umzuwandeln. Das Beispiel dient auch recht gut dazu, zu sehen, wie man PHP-Code in GML übersetzt, wobei ich noch ein paar kleine Veränderungen vorgenommen hatte. So kann man der Funktion nun sagen, ob man nur Hin-, Rückrunde oder beides angezeigt haben will. Theoretisch kann man das Ganze um eine weitere Funktion erweitern, welche nur einen bestimmten Spieltag zurückgibt.

Die Theorie des Spielplangenerators

Zunächst brauchen wir die Teams in einem Array. In meinem Testprojekt sieht das so aus:

```
var teams = ["1. Hoch und weit",
   "FC Blutgrätsche",
   "Borussia Maulkorb",
   "FCKW frei",
   "SC Dunkelwald",
   "Eintracht Zwietracht"];
```

Anschließend wird die Funktion, auf die ich gleich eingehe, wie folgt aufgerufen:

```
plan = game_schedule_generator(teams, true, true);
```

Die Variable *plan* enthält den Text, welcher vom Generator erzeugt wird. Diesen kann man anschließend auf dem Bildschirm anzeigen. Alternativ funktioniert auch

```
clipboard_set_text(plan);
```

Das kopiert den Plan in den Zwischenspeicher und man kann ihn z. B. in einen Texteditor einfügen. Das ist für Tests sehr nützlich.

Aber nun zum eigentlichen Generator. Die Funktion **game_schedule_generator** erwartet drei Argumente. Die Teams als Array, ob die Hinrunde angezeigt werden soll und ob die Rückrunde angezeigt werden soll.

Als erstes wird überprüft, ob die Anzahl der Teams gerade oder ungerade ist. Wenn sie ungerade ist, fügen wir ein weiteres Team hinzu, welches „[FREI]“ heißt. Später haben wir die Möglichkeit, entweder diese Spiele nicht anzuzeigen oder zu vermerken, dass eine bestimmte Mannschaft an dem Spieltag frei hat.

Die Variable *schedule* wird der Text für den Spielplan, die Variable *rr* ist die Rückrunde. Wir erzeugen also gleichzeitig Hin- und Rückrunde, indem wir die Teams der Hinrunde beim entsprechenden

Spieltag für die Rückrunde vertauschen.

Anschließend starten wir eine große Schleife für die Spieltage (*rounds*), geben die Nummer des Spieltags aus und berechnen die Paarungen. Am Ende fügen wir *rr* dem String *schedule* hinzu und fertig ist der Lack.

Paarungen berechnen

Das ist die Kernaufgabe des Generators. Die Logik ist relativ einfach. Man muss sich lediglich eine Liste der Mannschaften vorstellen. Dabei lost man das erste Team gegen das Letzte, das zweite gegen das Vorletzte usw. aus. Das Problem mit einer ungeraden Anzahl Teams haben wir mit dem freien Team gelöst, weshalb das immer aufgehen wird. Im unten stehenden Beispiel werden dann Mannschaften, die frei haben, nicht angezeigt.

Nun kommt der eigentliche Kniff. Am Ende der Runde verschieben wir die Liste. Wir merken uns das letzte Team und gehen dann von hinten nach vorne durch und versetzen jedes Team um eine Stelle nach vorne. Am Ende setzen wir das letzte Team an den Anfang bzw. an die zweite Position.
Das erste Team bleibt immer an derselben Stelle der Liste.

Hinrunde

Spieltag 1:
Eintracht Zwietracht - 1. Hoch und weit
FC Blutgrätsche - SC Dunkelwald
Borussia Maulkorb - FCKW frei

Spieltag 2:
SC Dunkelwald - 1. Hoch und weit
FCKW frei - Eintracht Zwietracht
Borussia Maulkorb - FC Blutgrätsche

Spieltag 3:
1. Hoch und weit - FCKW frei
SC Dunkelwald - Borussia Maulkorb
Eintracht Zwietracht - FC Blutgrätsche

Spieltag 4:
Borussia Maulkorb - 1. Hoch und weit
FC Blutgrätsche - FCKW frei
Eintracht Zwietracht - SC Dunkelwald

Spieltag 5:
1. Hoch und weit - FC Blutgrätsche
Borussia Maulkorb - Eintracht Zwietracht
FCKW frei - SC Dunkelwald

Rückrunde

Spieltag 6:
1. Hoch und weit - Eintracht Zwietracht
SC Dunkelwald - FC Blutgrätsche
FCKW frei - Borussia Maulkorb

Spieltag 7:
1. Hoch und weit - SC Dunkelwald
Eintracht Zwietracht - FCKW frei
FC Blutgrätsche - Borussia Maulkorb

Spieltag 8:
FCKW frei - 1. Hoch und weit
Borussia Maulkorb - SC Dunkelwald
FC Blutgrätsche - Eintracht Zwietracht

Spieltag 9:
1. Hoch und weit - Borussia Maulkorb
FCKW frei - FC Blutgrätsche
SC Dunkelwald - Eintracht Zwietracht

Ausschnitt des erzeugten Spielplans

Die Funktion

Nachdem alles erklärt wurde, folgt nun der Code:

```
/**
 * Generates a game schedule for a given set of teams.
 * @param teams   An array of teams.
 * @param h       A boolean indicating whether to generate the home games
 * @param a       A boolean indicating whether to generate the away games
 * @return schedule  A string containing the generated schedule
 */
function game_schedule_generator(teams, h, a)
{
 // Überprüfen, ob die Anzahl der Teams ungerade ist
    if(array_length(teams) % 2 != 0)
 {
        array_push(teams, „[FREI]");
    }

    var schedule = „";
 var rr = „";
    var rounds = array_length(teams) - 1;   // Anzahl der Runden
    var gamesPerRound = array_length(teams) / 2; // Anzahl der Spiele pro Rund
 var paar = 0;

 if (h) { schedule += „Hinrunde\n"; }
 if (a) { rr += „\nRückrunde\n"; }

    for(var i=0; i<rounds; i++)
 {
        if (h) { schedule += „\nSpieltag „ + string(i + 1) + „: \n"; }
        if (a) { rr += „\nSpieltag „ + string(rounds + i + 1) + „: \n"; }

  for(var j=0; j<gamesPerRound; j++)
  {
   paar++;
   // Auswärts und Heim vertauschen
   if (i%2==0) && (paar%rounds!=1)
   {
          var homeTeam = teams[j];
          var awayTeam = teams[array_length(teams) -1 -j];
   } else {
    var awayTeam = teams[j];
          var homeTeam = teams[array_length(teams) -1 -j];
   }

   if (homeTeam != „[FREI]") && (awayTeam != „[FREI]")
   {
    if (h) { schedule += homeTeam + „ - „ + awayTeam + „\n"; }
          if (a) { rr += awayTeam + „ - „ + homeTeam + „\n"; }
   }
```

```
        }

  var last = teams[array_length(teams)-1];
  for(var k=array_length(teams)-1; k>1; k--)
  {
   teams[k] = teams[k-1];
  }
  teams[1] = last;
    }

 schedule += rr;
    return schedule;
}
```

Das war es auch schon wieder.

Date Created
10. März 2023
Author
sven