Schachbrett zeichnen in GML

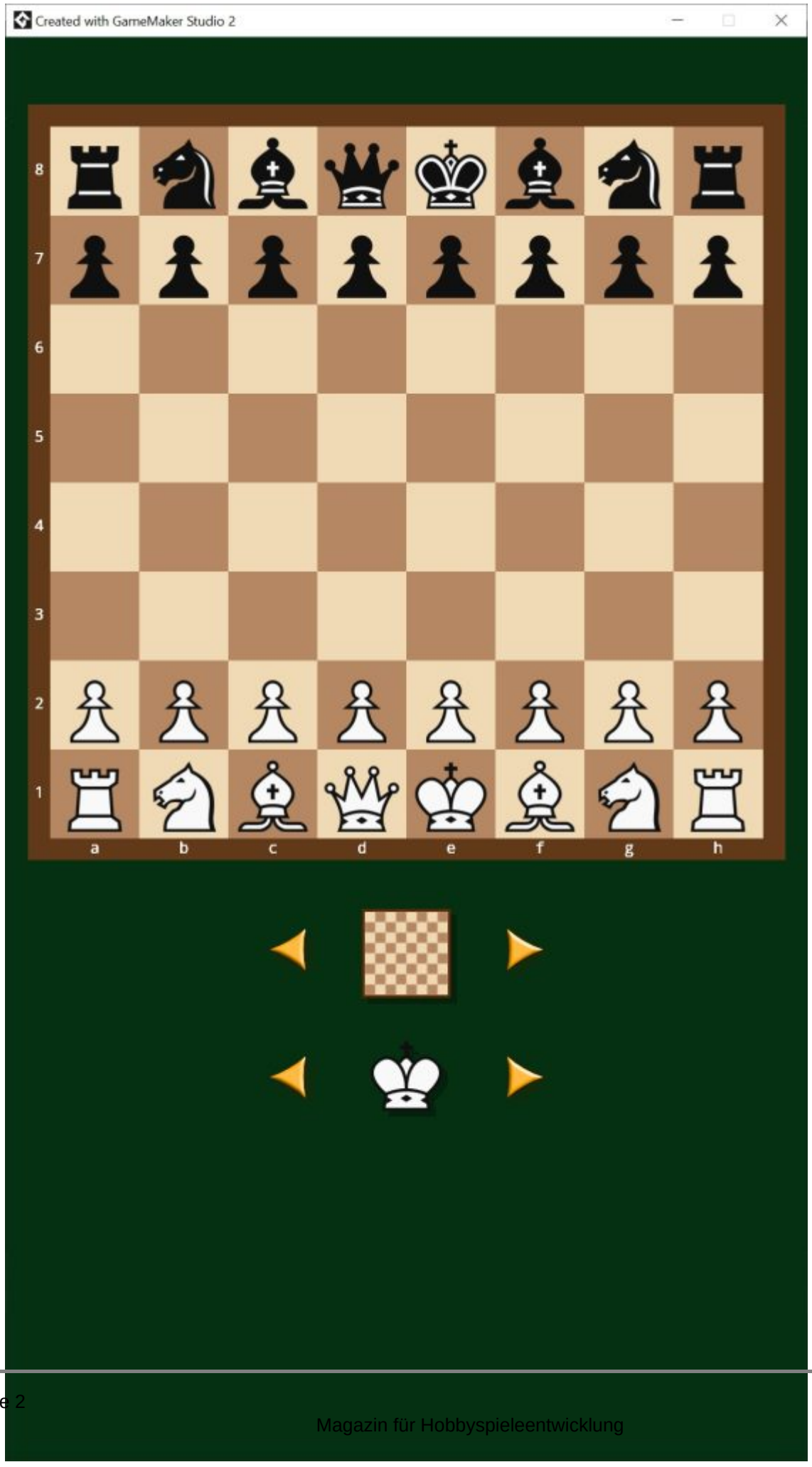
Description

Das wird ein etwas längeres Coding-Tutorial. Ich möchte einen Weg zeigen, wie man im GML ein Schachbrett mit Figuren zeichnet. Dabei sollen Brett und Figuren ausgetauscht werden können und das Brett wird auf zwei Seiten beschriftet. Außerdem gibt es die Möglichkeit, das Brett um 180° zu drehen.

Natürlich lassen sich auch andere Figuren einsetzen, etwa Dame. Die Logik dahinter ist die selbe. Die auftretenden Probleme lösen wir mit eindimensionalen Arrays. Das hat gleich mehrere Gründe:

1. In GML wurden 2D-Arrays intern auf 1D umgestellt.
2. 1D-Arrays sind i. d. R. schneller als 2D-Arrays.
3. Wenn wir auf GM-Eigene Befehle wie DS-List, DS-Map etc. verzichten, können wir den Code einfacher in eine andere Programmiersprache übersetzen, falls nötig.

Das Resultat soll so aussehen:


Created with GameMaker Studio 2
8
7
6
5
4
3
2
1
a
b
c
d
e
f
g
h

Schachbrett Setting

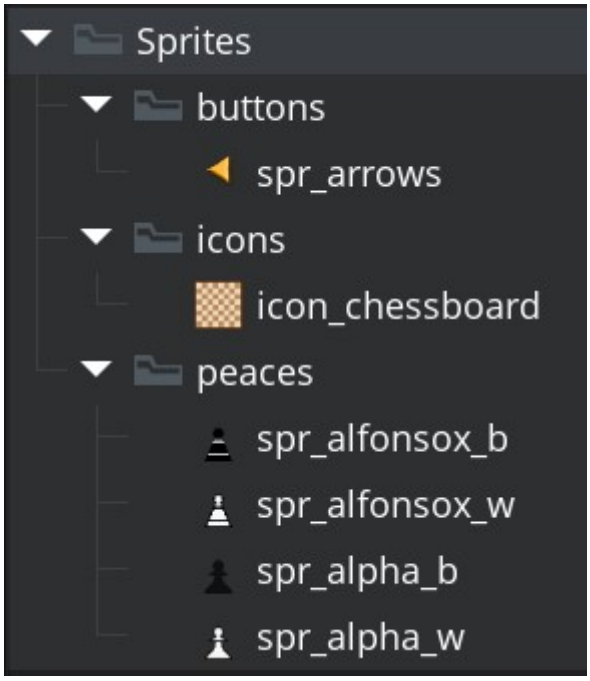
Ausgangszustand und Ressourcen

Wir erstellen ein neues Projekt. Den **Raum** habe ich *ChessboardSettings* genannt. Er hat die Größe von *1080×1920*, ist von der Ausrichtung her also eher für Mobile gedacht. Dies lässt sich aber beliebig anpassen.

Die Zahl der **Objekte** lässt sich auf eins reduzieren. Bei mir heißt es *obj_ChessboardSettings*.

Meine **Schrift** heißt *fnt_boart_letter*. Die brauchen wir, um das Brett zu beschriften. Hier verwende ich Regular in der Größe 16.

Außerdem brauchen wir noch sechs Sprites, die bei mir wie folgt benannt sind:

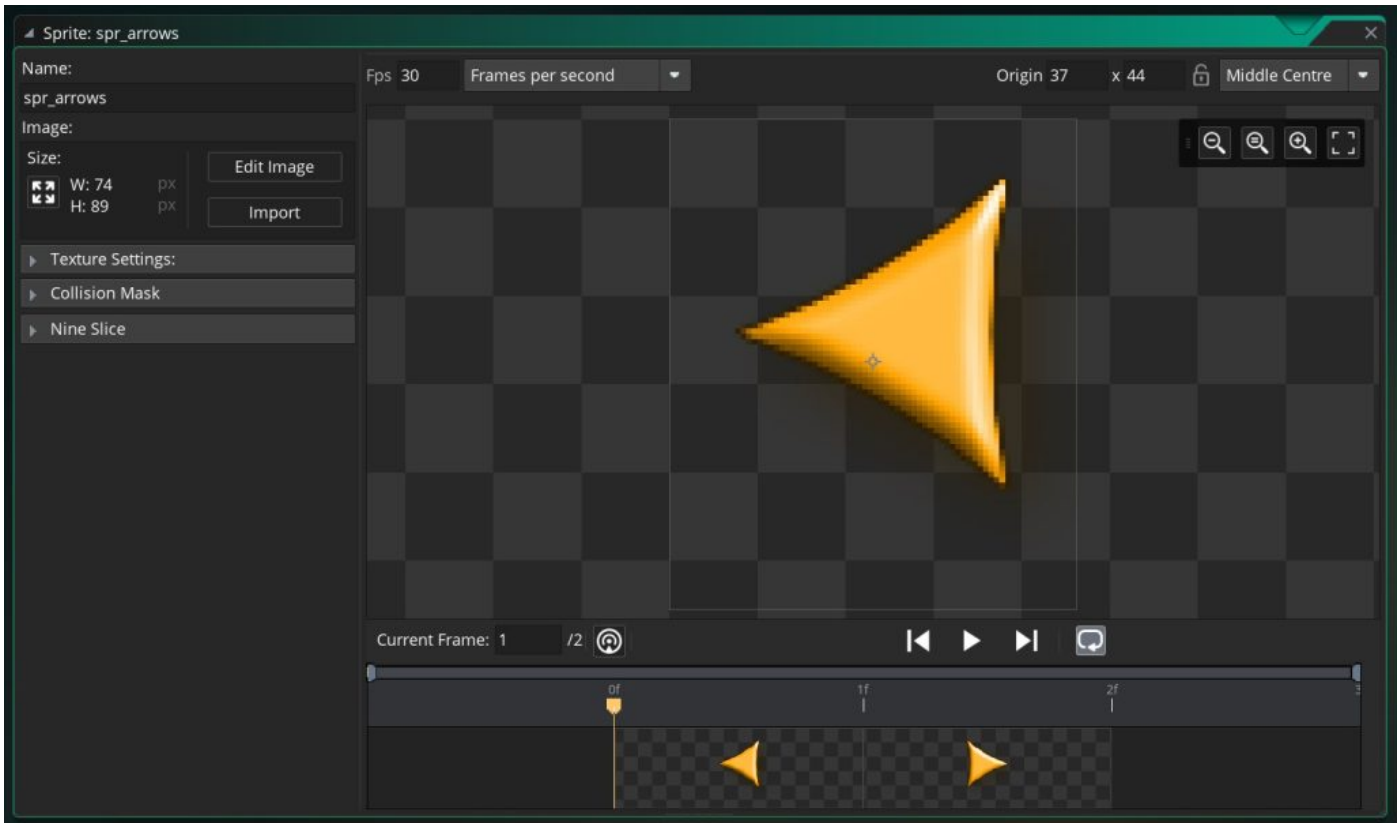


Schachbrett Sprites 1

Wie man sehen kann, gibt es nur das Icon für das Schachbrett. Die Farben zeichnen wir im Code.

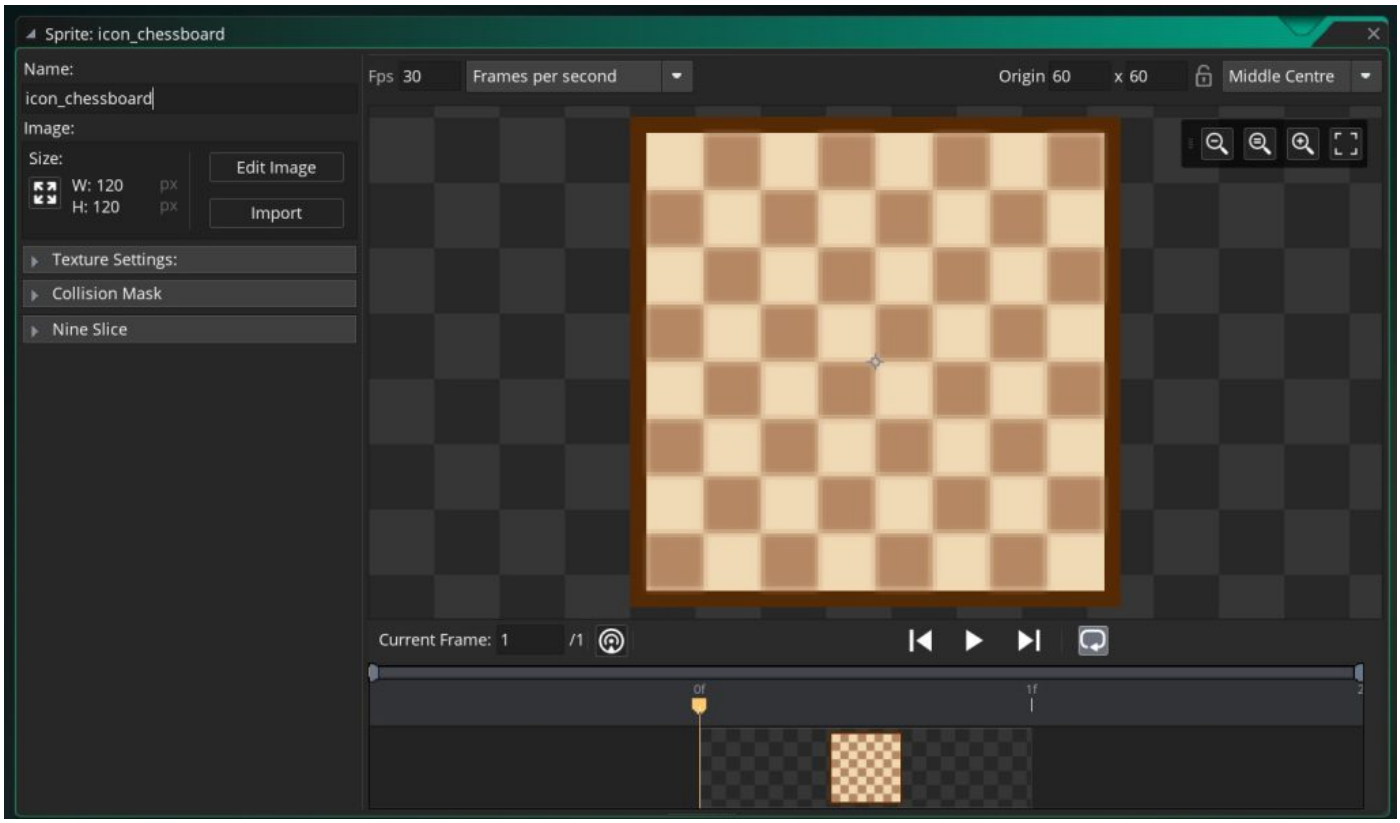
Infos zu den Sprites

Die Pfeile brauchen wir zum durchschalten. Hätte ich bei der Grafik keinen Lichteffekt verwendet, hätte sogar ein Sprite gereicht.



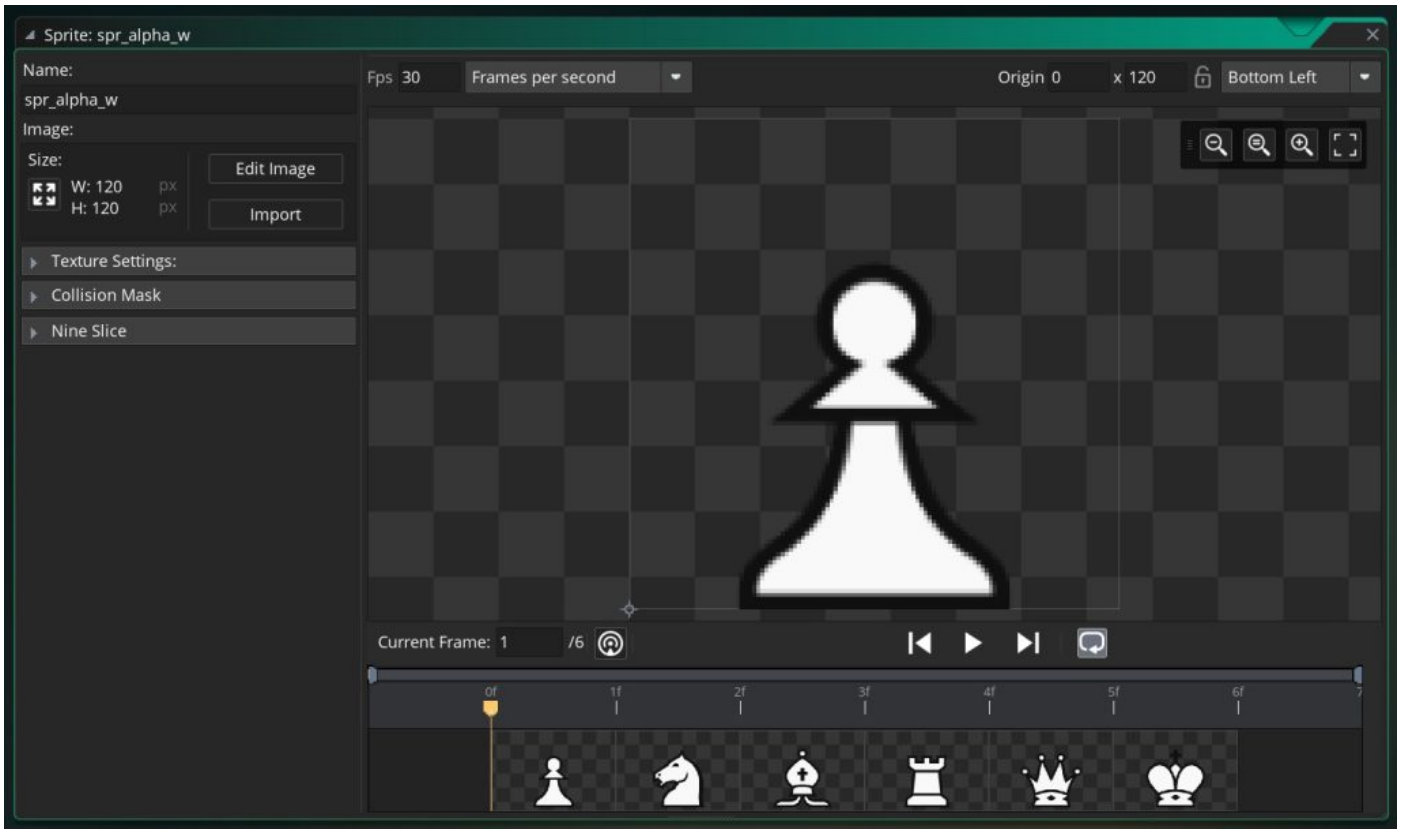
spr_arrows

Wir auf dem oberen Screenshot zu sehen ist, haben wir unter dem Brett noch ein Icon. Bei mir heißt es *icon_chessboard*:



icon_chessboard

Zum Schluss haben wir noch vier Sprites mit Schachfiguren. Je zwei gehören zu einem Set, etwa *spr_alpha_w* und *spr_alpha_b*:



spr_alpha_w

Wie man sehen kann, sind je sechs Figuren von einer Farbe in einem Sprite. Jede Figur hat die Maße 120×120 Pixel. Damit es gut aussieht, ist darauf zu achten, dass alle Figuren unten bündig sind und an den Seiten sowie oben etwas Platz haben, da unsere Felder ebenfalls 120×120 Pixel haben werden.

Der Code

Bevor ich die grundlegende Logik erkläre, zeige ich das **Create-Event**:

```
colorSet = 1;
piecesSet = 1;
boardStep = 120;
startX  = 60;
startY  = 1080;
flip  = false;

board_color = [
 1, 0, 1, 0, 1, 0, 1, 0,
 0, 1, 0, 1, 0, 1, 0, 1,
 1, 0, 1, 0, 1, 0, 1, 0,
 0, 1, 0, 1, 0, 1, 0, 1,
 1, 0, 1, 0, 1, 0, 1, 0,
 0, 1, 0, 1, 0, 1, 0, 1,
 1, 0, 1, 0, 1, 0, 1, 0,
 0, 1, 0, 1, 0, 1, 0, 1];

init_color = [
```

```
 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0, 0, 0, 0,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 1, 1, 1, 1, 1, 1, 1, 1,
 1, 1, 1, 1, 1, 1, 1, 1];

init_board = [
 3, 1, 2, 4, 5, 2, 1, 3,
 0, 0, 0, 0, 0, 0, 0, 0,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 6, 6, 6, 6, 6, 6, 6, 6,
 0, 0, 0, 0, 0, 0, 0, 0,
 3, 1, 2, 4, 5, 2, 1, 3];

Flip = [
 56, 57, 58, 59, 60, 61, 62, 63,
 48, 49, 50, 51, 52, 53, 54, 55,
 40, 41, 42, 43, 44, 45, 46, 47,
 32, 33, 34, 35, 36, 37, 38, 39,
 24, 25, 26, 27, 28, 29, 30, 31,
 16, 17, 18, 19, 20, 21, 22, 23,
  8,  9, 10, 11, 12, 13, 14, 15,
  0,  1,  2,  3,  4,  5,  6,  7];
```

Zuerst definieren wir sechs Variablen. *colorSet* und *piecesSet* sagen uns, welches Set verwendet werden soll. Diese schalten wir später um. *boardStep* gibt die Abstände bzw. die Größe der Felder vor. Hier sehen wir wieder unsere 120 Pixel. *startX* und *startY* sind unser Ausgangspunkt zum zeichnen. Die Variable *flip* sagt aus, ob das Brett gedreht werden soll.

Nun zu den großen Arrays. Wenn man sie so schreibt, wie ich, erkennt man gleich, dass es jeweils 64 Werte sind, die wie ein Schachbrett angeordnet wurden (8×8). Die Anordnung dient nur der Übersicht, man hätte es auch jeweils in eine Zeile schreiben können.

board_color

Das Array *board_color[]* gibt für jedes Feld an, ob es hell (0) oder dunkel (1) sein soll. Hier erkennt man schon eines der Vorteile des Arrays: Man kann hier, falls gewünscht, auch andere Farbfolgen definieren. Es muss also nicht bei einem zweifarbigen Muster bleiben.

init_color

Das sagt aus, wo die Figuren zu welcher Farbe stehen. Weiß unten (1), Schwarz oben (0) und in der Mitte stehen keine Figuren (6).

init_board

Durch *init_color[]* wissen wir zwar, wo die Farben sind, aber nicht, welche Figur wo steht. Dafür haben wir *init_board[]*. 0 = Bauer, 1 = Springer, 2 = Läufer, 3 = Turm, 4 = Dame und 5 = König. 6 ist wieder ein neutrales Feld. Jetzt versteht man eher, warum bei *init_color* die 6 ebenfalls neutral ist.

Flip

Flip[] sieht interessant aus, aber auf die Logik dahinter gehe ich im Draw-Event ein.

Key Up-F-Event

Hier definieren wir die Drehung des Bretts.

```
flip = !flip;
```

Draw-Event

Auf den ersten Blick erschlägt einen der Code, aber ich werde ihn bei der Erklärung, wie immer, zerstückeln:

```
var col0, col1, colBor, colLet, peaceWhite, peaceBlack;
var alpha = .3;

draw_set_font(fnt_boart_letter);

// Field color
switch colorSet
{
 case 1:
  // Set 1
  col0 = #F0D9B5; //light
  col1 = #B58863; //dark
  colBor = #623A19;
  colLet = #FAFAFA;
 break;
 case 2:
  // Set 2
  col0 = #FFFFDD; //light
  col1 = #86A666; //dark
  colBor = #03550A;
  colLet = #DCFAE6;
 break;
 case 3:
  // Set 3
  col0 = #F1F6B3; //light
  col1 = #58945D; //dark
  colBor = #005000;
  colLet = #DCFFDC;
 break;
 case 4:
```

```
  // Set 4
  col0 = #E7DCF1; //light
  col1 = #967BB1; //dark
  colBor = #50321E;
  colLet = #FFFFFF;
 break;
 case 5:
  // Set 5
  col0 = #9F90B0; //light
  col1 = #7D4A8D; //dark
  colBor = #400481;
  colLet = #DEDEDE;
 break;
}

// Peaces
switch piecesSet
{
 case 1:
  // Classic
  peaceWhite = spr_alpha_w;
  peaceBlack = spr_alpha_b;
  break;
 case 2:
  // Alpha
  peaceWhite = spr_alfonsox_w;
  peaceBlack = spr_alfonsox_b;
  break;
}

// Border
draw_set_color(colBor);
draw_rectangle(startX-30, startY+30, room_width-30, 90, 0);

// Draw Board
var j=0;

if (flip)
{
 var let=["h", „g", „f", „e", „d", „c", „b", „a"];
 var column=0, line=7;
} else {
 var let=["a", „b", „c", „d", „e", „f", „g", „h"];
 var column=0, line=0;
}

draw_set_halign(fa_center);
draw_set_valign(fa_middle);


for (var i=0; i<64; ++i)
{
 var stepsColumn = boardStep * column;
 var stepsLine = boardStep * line;
 var xx   = startX + stepsColumn;
```

```
var yy   = startY - stepsLine;

if (flip)
{
 j = 63-Flip[i];

 switch (board_color[j])
 {
  case 0:
   draw_set_color(col0);
   break;
  case 1:
   draw_set_color(col1);
   break;
 }

 draw_rectangle(xx, yy+1, xx+boardStep, yy-boardStep, 0);

 draw_set_color(colLet);

 // Lines
 if (line >= 0)
  draw_text(startX-15, startY-60-(line)*120, string(8-line));

 // Peaces
 if (init_color[j] == 0) { draw_sprite(peaceWhite, init_board[j], xx, yy-8);
 if (init_color[j] == 1) { draw_sprite(peaceBlack, init_board[j], xx, yy-8);

 if (j %8 == 0 && j != 63) { line--; }

} else {

 switch (board_color[i])
 {
  case 0:
   draw_set_color(col0);
   break;
  case 1:
   draw_set_color(col1);
   break;
 }

 draw_rectangle(xx, yy+1, xx+boardStep, yy-boardStep, 0);

 draw_set_color(colLet);

 // Lines
 draw_text(startX-15, startY-60-line*120, string(line+1));

 // Peaces
 if (init_color[i] == 0) { draw_sprite(peaceWhite, init_board[i], xx, yy-8);
 if (init_color[i] == 1) { draw_sprite(peaceBlack, init_board[i], xx, yy-8);

 if ((i + 1) %8 == 0 && i != 63) { line++; }
}
```

```
 // Rows
 draw_text(startX+60+column*120, startY+15, string(let[column]));

 if (column < 7)
  column++;
 else
  column = 0;
}

// Settings / Buttons / Icons
// Board
draw_sprite_ext(icon_chessboard, 0, room_width/2+8, 1236+8, 1, 1, 0, #000000,
draw_sprite_ext(icon_chessboard, 0, room_width/2, 1236, 1, 1, 0, #FFFFFF, 1);
draw_sprite(spr_arrows, 0, room_width/2-160, 1236);
draw_sprite(spr_arrows, 1, room_width/2+160, 1236);

// Peaces-Sprite
draw_sprite_ext(peaceWhite, 5, room_width/2-60+8, 1448+8, 1, 1, 0, #000000, al
draw_sprite_ext(peaceWhite, 5, room_width/2-60, 1448, 1, 1, 0, #FFFFFF, 1);
draw_sprite(spr_arrows, 0, room_width/2-160, 1408);
draw_sprite(spr_arrows, 1, room_width/2+160, 1408);
```

Das sind 162 Zeilen GML-Spaß. ? Geübtere Leser werden zwei Dinge erkannt haben:

1. Wir nutzen für das Brett 5 Farb-Sets.
2. Die Farben werden in Hex-Werten angegeben, was in GML mittlerweile möglich ist. Das erlöst uns vom lästigen *make_color_rgb()*-Befehl.

Variablen und Font

```
var col0, col1, colBor, colLet, peaceWhite, peaceBlack;
var alpha = .3;

draw_set_font(fnt_boart_letter);
```

Wir fangen mit lokalen Variablen an und definieren noch unsere Schrift für das Brett. Auf die Variablen in der ersten Zeile gehe ich gleich noch ein. Die Variable *alpha* brauchen wir später für den Schatten der Icons.

Brettfarben

```
// Field color
switch colorSet
{
 case 1:
  // Set 1
  col0 = #F0D9B5; //light
  col1 = #B58863; //dark
  colBor = #623A19;
  colLet = #FAFAFA;
 break;
```

```
 case 2:
  // Set 2
  col0 = #FFFFDD; //light
  col1 = #86A666; //dark
  colBor = #03550A;
  colLet = #DCFAE6;
 break;
 case 3:
  // Set 3
  col0 = #F1F6B3; //light
  col1 = #58945D; //dark
  colBor = #005000;
  colLet = #DCFFDC;
 break;
 case 4:
  // Set 4
  col0 = #E7DCF1; //light
  col1 = #967BB1; //dark
  colBor = #50321E;
  colLet = #FFFFFF;
 break;
 case 5:
  // Set 5
  col0 = #9F90B0; //light
  col1 = #7D4A8D; //dark
  colBor = #400481;
  colLet = #DEDEDE;
 break;
}
```

Hier sind die fünf Sets definiert. Jedes Set verfügt über vier Farben. Eine helle (*col0*), eine dunkle (*col1*), sowie Rahmen- und Schriftfarbe (*colBor*, *colLet*). Zur Umschaltung nutzen wir eine Switch.

Figuren

```
// Peaces
switch piecesSet
{
 case 1:
  // Classic
  peaceWhite = spr_alpha_w;
  peaceBlack = spr_alpha_b;
  break;
 case 2:
  // Alpha
  peaceWhite = spr_alfonsox_w;
  peaceBlack = spr_alfonsox_b;
  break;
}
```

Bei den Figuren gehen wir so vor wie bei den Felder, nur dass wir hier die Sprites und nicht Farben definieren. Das Prinzip der Switch bleibt gleich.

Rahmen und Brett

```
// Border
draw_set_color(colBor);
draw_rectangle(startX-30, startY+30, room_width-30, 90, 0);

// Draw Board
var j=0;

if (flip)
{
 var let=["h", „g", „f", „e", „d", „c", „b", „a"];
 var column=0, line=7;
} else {
 var let=["a", „b", „c", „d", „e", „f", „g", „h"];
 var column=0, line=0;
}

draw_set_halign(fa_center);
draw_set_valign(fa_middle);
```

Zunächst schalten wir auf die Farbe des Rahmens um und zeichnen daraufhin ein Rechteck, welches zu allen Seiten 30 Pixel größer ist, als das eigentliche Brett. Dann geht es langsam an das eigentliche Brett. Je nachdem, ob das Brett gedreht wurde oder nicht, werden die Buchstaben, Reihen und Spalten definiert. Zum Ende dieses Blocks definieren wir noch horizontale und vertikale Ausrichtung der Texte. Dann geht es an die große Schleife:

```
for (var i=0; i<64; ++i)
{
 var stepsColumn = boardStep * column;
 var stepsLine = boardStep * line;
 var xx   = startX + stepsColumn;
 var yy   = startY - stepsLine;

 if (flip)
 {
  j = 63-Flip[i];

  switch (board_color[j])
  {
   case 0:
    draw_set_color(col0);
    break;
   case 1:
    draw_set_color(col1);
    break;
  }

  draw_rectangle(xx, yy+1, xx+boardStep, yy-boardStep, 0);

  draw_set_color(colLet);
```

```
  // Lines
  if (line >= 0)
   draw_text(startX-15, startY-60-(line)*120, string(8-line));

  // Peaces
  if (init_color[j] == 0) { draw_sprite(peaceWhite, init_board[j], xx, yy-8);
  if (init_color[j] == 1) { draw_sprite(peaceBlack, init_board[j], xx, yy-8);

  if (j %8 == 0 && j != 63) { line--; }

 } else {

  switch (board_color[i])
  {
   case 0:
    draw_set_color(col0);
    break;
   case 1:
    draw_set_color(col1);
    break;
  }

  draw_rectangle(xx, yy+1, xx+boardStep, yy-boardStep, 0);

  draw_set_color(colLet);

  // Lines
  draw_text(startX-15, startY-60-line*120, string(line+1));

  // Peaces
  if (init_color[i] == 0) { draw_sprite(peaceWhite, init_board[i], xx, yy-8);
  if (init_color[i] == 1) { draw_sprite(peaceBlack, init_board[i], xx, yy-8);

  if ((i + 1) %8 == 0 && i != 63) { line++; }
 }

 // Rows
 draw_text(startX+60+column*120, startY+15, string(let[column]));

 if (column < 7)
  column++;
 else
  column = 0;
}
```

Nun geht es ans Eingemachte!

Wir sehen eine große for-Schleife, die wir 64mal durchlaufen. Wenn man sich die Schleife genauer anschaut, erkennt man, dass sie sich gedanklich in vier Abschnitte unterteilen lässt. Dies wollen wir tun:

1. Wir definieren ein paar Variablen
2. Brett zeichnen, wenn es gedreht ist
3. Brett zeichnen, wenn es nicht gedreht ist
4. Irgendwas mit Zeilen und Spalten

Es wird klar, dass die Bereiche 2 und 3 nahezu identisch sind. Das macht es einfacher, die Schleife zu verstehen.

Die lokalen Variablen *stepsColumn* und *stepsLine* sind nur Zwischenberechnungen. Das ginge auch kompakter, aber wir wollen es verstehen, also steht es hier ausführlicher. Wir wir sehen, werden die Variablen *coloumn* und *line* rauf bzw. runter gezählt. Multipliziert mit *boardStep* (120) ergibt das eine relative Position auf dem Brett.

Bei *xx* und *yy* ziehen wir noch *startX* und *startY* hinzu, woraus eine absolute Position wird. Dann geht es auch schon ans Zeichnen.

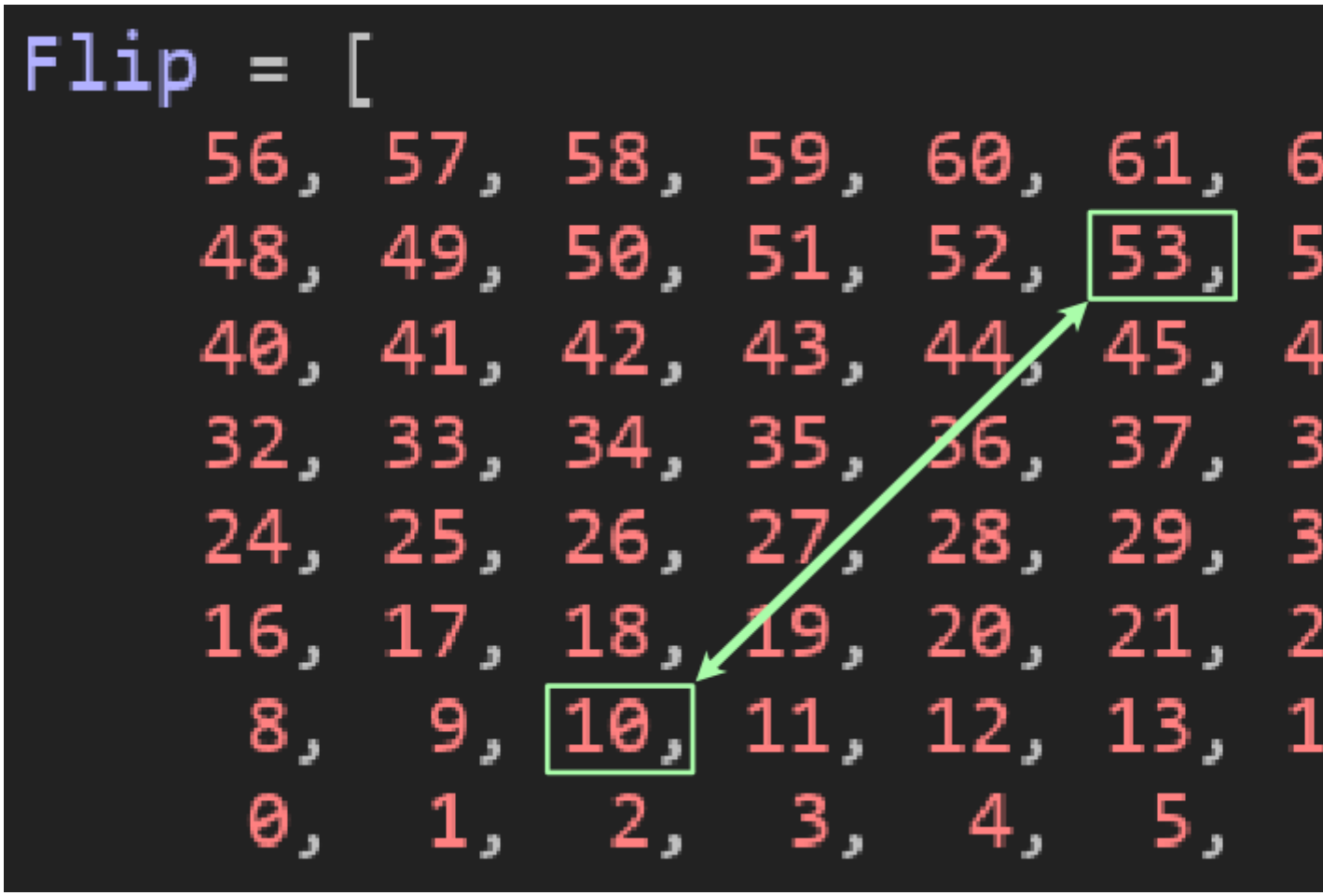
Flip

Nun kommen wir zum mysteriösen *Flip*. Das Array aus dem Create-Event brauchen wir nur einmal:

```
j = 63-Flip[i];
```

Unsere for-Schleife läuft von 0 bis 63. An der Variable *i* orientiert sich der Part ohne Flip. Für die Brettdrehung haben wir *j*. Damit wir beim gedrehten Brett alles richtig zeichnen, brauchen wir das Hilfs-Array Flip[] und ziehen die entsprechende Zahl von 63 ab.

Dazu ein Beispiel: *i* ist bei 10. Da es bei 0 beginnt, brauchen wir den 11. Wert in *Flip[]*. Das ist die 50. *j* ist somit 63-50, also 13. Wenn man sich das Array *Flip[]* anschaut und die Zahlen 10 und 53 sucht, versteht man, dass die 180°-Drehung einer horizontalen und vertikalen Spiegelung entspricht.



Flip Array

Der Rest ist in Abschnitt 2 und 3 identisch, nur dass wir einmal *j* und einmal *i* verwenden.

Brettfarbe

```
switch (board_color[j])
{
 case 0:
  draw_set_color(col0);
  break;
 case 1:
  draw_set_color(col1);
  break;
}
```

Im Array *board_color[]* schauen wir nach, ob das Feld 0 (hell) oder 1 (dunkel) ist. Danach wird das Quadrat gezeichnet:

```
draw_rectangle(xx, yy+1, xx+boardStep, yy-boardStep, 0);
```

Beschriftung

Nun kommt die Beschriftung:

```
draw_set_color(colLet);

// Lines
if (line >= 0)
 draw_text(startX-15, startY-60-(line)*120, string(8-line));
```

Wir nutzen die Farbe für die Beschriftung der Reihen und zeichnen den Text (1 bis 8) an der entsprechenden Stelle.

Figuren

Jetzt kommen unsere Figuren:

```
// Peaces
if (init_color[j] == 0) { draw_sprite(peaceWhite, init_board[j], xx, yy-8); }
if (init_color[j] == 1) { draw_sprite(peaceBlack, init_board[j], xx, yy-8); }
```

Hier nutzen wir gleich zwei Arrays. *init_color[]* sagt uns die Farbe (schwarz/weiß) und *init_board[]* die Art der Figur.

Nächste Zeile, bitte

Die letzte Zeile im Block ist:

```
if (j %8 == 0 && j != 63) { line--; }
```

bzw.

```
if ((i + 1) %8 == 0 && i != 63) { line++; }
```

im Block ohne Flip. Doch was tun wir hier?

Wir haben ein eindimensionales Array, wollen aber nach 8 Feldern in die nächste Zeile (*yy*) umschalten. Also müssen wir prüfen, ob *j* bzw. *i* durch 8 teilbar ist und wir nicht schon am Ende der Schleife sind. Wenn ja, gehen wir eine Zeile hoch bzw. runter.

Zeilen und Spalten

Jetzt sind wir im letzten Abschnitt:

```
// Rows
draw_text(startX+60+column*120, startY+15, string(let[column]));

if (column < 7)
 column++;
else
 column = 0;
```

Letztlich machen wir hier nichts anderes, als die Buchstaben (*let[]*) zu schreiben. Dabei zählen wir brav die Spalten (*column*) durch und setzen sie auf 0, sobald wir bei 7 sind.

Buttons und Icons

```
// Settings / Buttons / Icons
// Board
draw_sprite_ext(icon_chessboard, 0, room_width/2+8, 1236+8, 1, 1, 0, #000000,
draw_sprite_ext(icon_chessboard, 0, room_width/2, 1236, 1, 1, 0, #FFFFFF, 1);
draw_sprite(spr_arrows, 0, room_width/2-160, 1236);
draw_sprite(spr_arrows, 1, room_width/2+160, 1236);

// Peaces-Sprite
draw_sprite_ext(peaceWhite, 5, room_width/2-60+8, 1448+8, 1, 1, 0, #000000, al
draw_sprite_ext(peaceWhite, 5, room_width/2-60, 1448, 1, 1, 0, #FFFFFF, 1);
draw_sprite(spr_arrows, 0, room_width/2-160, 1408);
draw_sprite(spr_arrows, 1, room_width/2+160, 1408);
```

Der letzte Block im Draw-Event ist selbstredend. Hier zeichnen wir das Brett-Icon, ein Symbol für das Figuren-Set und die zugehörigen Pfeile.

Global Left Released-Event

Am Ende geht es noch um die Funktionalität. Die Pfeile werden mit der Maus bedient. Im entsprechenden Event fragen wir den Klick mit *point_in_rectangle()* ab und schalten die Variablen *colorSet* und *piecesSet* rauf oder runter.

```
// Board left
if (point_in_rectangle(mouse_x, mouse_y, room_width/2-197, 1236-45, room_width
{
 if (colorSet > 1)
  colorSet--;
 else
  colorSet = 5;
}

// Board right
if (point_in_rectangle(mouse_x, mouse_y, room_width/2+123, 1236-45, room_width
{
 if (colorSet < 5)
  colorSet++;
 else
  colorSet = 1;
}

// Peaces left
if (point_in_rectangle(mouse_x, mouse_y, room_width/2-197, 1408-45, room_width
{
 if (piecesSet > 1)
  piecesSet--;
 else
```

```
  piecesSet = 2;
}

// Peaces right
if (point_in_rectangle(mouse_x, mouse_y, room_width/2+123, 1408-45, room_width
{
 if (piecesSet < 2)
  piecesSet++;
 else
  piecesSet = 1;
}
```

Das war es auch schon wieder!

Date Created
1. April 2022
Author
sven