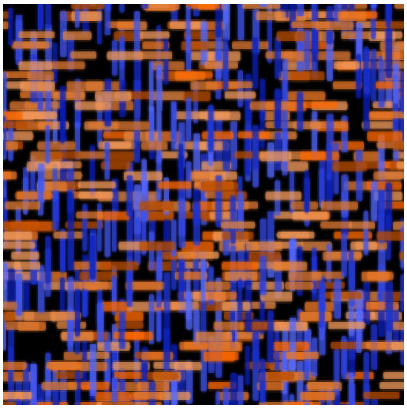
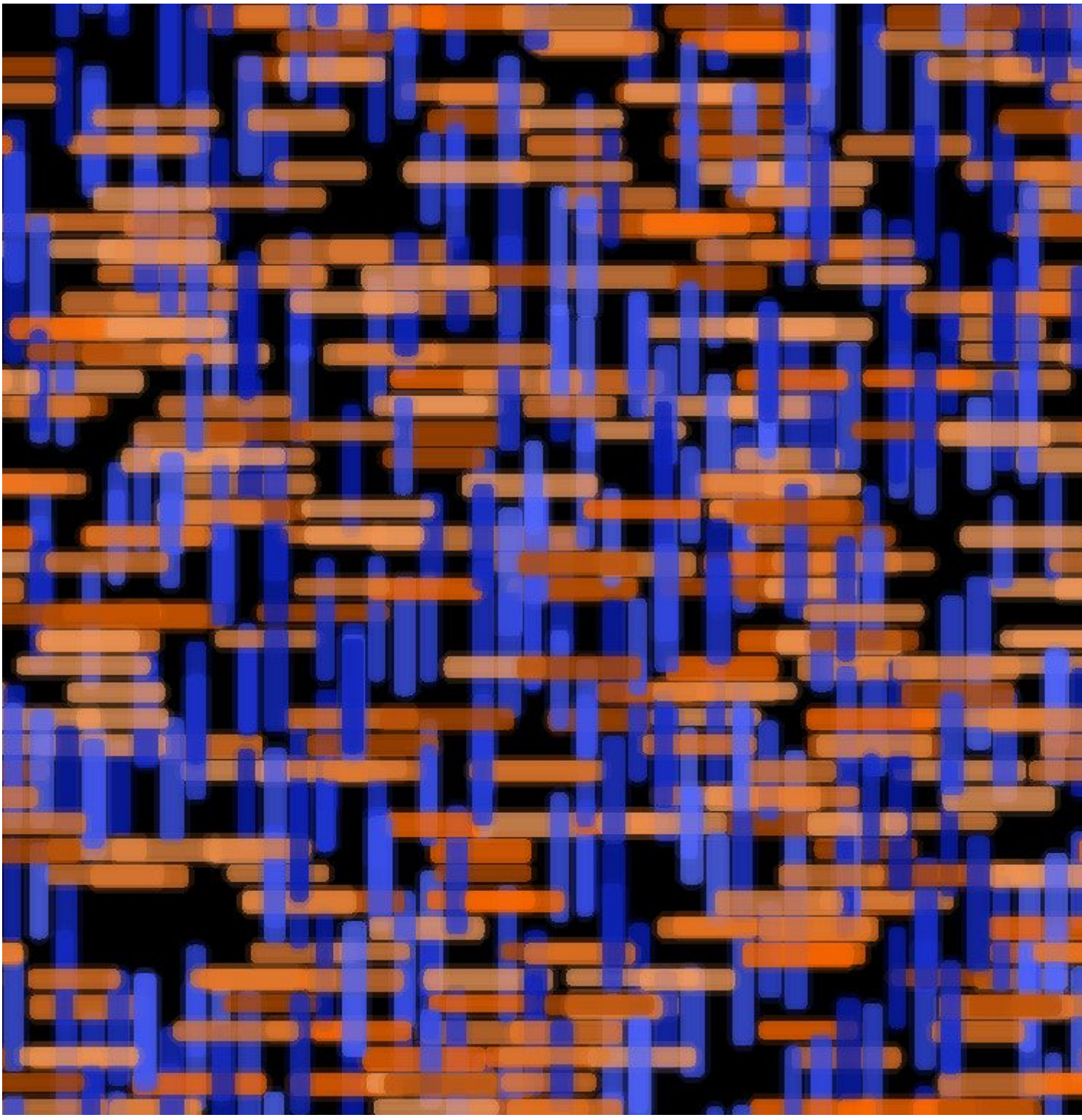


Schicker Hintergrundeffekt

Description

Für mein aktuelles Spiel habe ich einen kleinen Hintergrundeffekt erstellt. Da er einigen Leuten gut gefallen hat, zeige ich nun, wie man so einen Effekt programmiert.

Im Kern basiert die Logik auf dem Falling numbers Effekt. Vor allem Anfänger sollten sich erst diesen Effekt anschauen und die Logik verstehen, sonst kann es hier schwierig werden. Da der hier gezeigte Effekt etwas komplexer ist, solltet ihr euch sowohl den Text, als auch das Video dazu anschauen. Das Video habe ich unten eingebunden. Folgenden Effekt werden wir heute erstellen:



Der Effekt besteht lediglich aus einem Objekt in dem wir nur zwei Events brauchen. Außerdem brauchen wir natürlich noch einen Raum.

Grundlogik

Auf dem Screenshot sehen wir zwei Arten von Linien. Die einen liegen horizontal, die anderen vertikal. Auffällig ist, dass sie aussehen, als wären sie ineinander verwoben, es sind also nicht nur zwei Ebenen. Wenn man den Effekt in Bewegung sieht, wird man erkennen, dass die orangenen Linien, die eigentlich gerundete Rechtecke sind, sich von rechts nach links bewegen. Die blauen Rechtecke bewegen sich von unten nach oben.

Um das zu realisieren brauchen wir zwei Arrays, in denen wir die Grundwerte definieren. Für jedes Rechteck brauchen wir eine Breite, eine Höhe, eine Farbe, eine Geschwindigkeit sowie X und Y-Position. Theoretisch könnte man sogar den Alphawert und sogar die Form eines jeden Rechtecks (es könnten auch Kreise sein etc) definieren, aber dies könnt ihr nach Belieben erweitern.

Wenn die zwei Arrays vordefiniert wurden, geht es an das Zeichnen. Beim Zeichnen gehen wir die Liste aller Rechtecke in einer Schleife durch. Das Verweben der Rechtecke erreichen wir, indem wir zwischen den beiden Effekten im Draw hin und her schalten. Erst wird ein blaues Rechteck gezeichnet, dann ein orangenes, dann ein blaues und so weiter. Von der Logik her ist es also sehr simpel.

obj_background

Event Create

Da der Code etwas umfangreicher ist, schauen wir ihn uns einmal in kleineren Abschnitten an:

```
randomize();

////////////////////////////
//
// Lines up
//
////////////////////////////

var upLineMinThickness = 12;
var upLineMaxThickness = 18;

var upLineMinLength  = 64;
var upLineMaxLength  = 96;

// Number of Lines at x
var upLineCountX  = ceil(room_width / upLineMaxThickness);

// Number of Lines at y
var upLineCountY  = ceil(room_height / upLineMaxLength);

upLinesComplete   = upLineCountX * upLineCountY;

upLineGrid[0, 0]  = 0;

var upxCounter   = 0;
var upyCounter   = 0;
```

```
////////////////////////////
//
// Lines left
//
////////////////////////////

var leftLineMinThickness = 12;
var leftLineMaxThickness = 18;

var leftLineMinLength  = 64;
var leftLineMaxLength  = 96;

// Number of Lines at x
var leftLineCountX   = ceil(room_width / leftLineMaxLength);

// Number of Lines at y
var leftLineCountY   = ceil(room_height / leftLineMaxThickness);

leftLinesComplete   = leftLineCountX * leftLineCountY;

leftLineGrid[0, 0]   = 0;

var leftxCounter   = 0;
var leftyCounter   = 0;

////////////////////////////////////////
```

Am Anfang definieren wir nur die Basiswerte für den Effekt. Mit den Variablen *upLineMinThickness* und *upLineMaxThickness* definieren wir den Bereich der Dicke. Mit *upLineMinLength* und *upLineMaxLength* die Länge. Gleiches gilt auch für die Rechtecke, die nach links scrollen. Mit *upLineCountX* und *upLineCountY* berechnen wir, wie viele Rechtecke wir brauchen. Der Bedarf hängt von der maximalen Breite bzw. Höhe und den Raummaßen ab. In den Zeilen 16 und 19 wird dies berechnet.

```
// Number of Lines at x
var upLineCountX  = ceil(room_width / upLineMaxThickness);

// Number of Lines at y
var upLineCountY  = ceil(room_height / upLineMaxLength);
```

Dies sind die entsprechenden Zeilen als Auszug. Wir haben eine Raumbreite von 1024 Pixeln und eine maximale Breite von 18 Pixeln. Das ergibt 56,888 Rechtecke. Da wir mit *ceil* arbeiten, wird die Zahl auf 57 aufgerundet. Unsere Raumhöhe beträgt 768 Pixel, unsere maximale Länge sind 96 Pixel. Das ergibt genau 8 Rechtecke. Mit *upLinesComplete* berechnen wir nun die gesamte Anzahl der Rechtecke. 57 * 8 = 456 Rechtecke die nach oben scrollen.

Das Gleiche machen wir auch mit den Rechtecken die sich nach links bewegen. Nun müssen wir die Arrays nur noch mit zwei Schleifen füllen.

```
linesComplete = leftLinesComplete + upLinesComplete;

var upCounter = 0;
var leftCounter = 0;
```

```
// create up lines
for (var i=0; i<upLinesComplete; i++)
{
 var upLineLength = floor(random_range(upLineMinLength, upLineMaxLength));
 var upLineThickness = floor(random_range(upLineMinThickness, upLineMaxThickne

 // Color
 var upcolorNum  = floor(random_range(1, 7));
 var upcolor   = c_white;

 switch (upcolorNum)
 {
  case 1:
   upcolor  = make_colour_rgb(84, 104, 255);
   break;
  case 2:
   upcolor  = make_colour_rgb(65, 85, 239);
   break;
  case 3:
   upcolor  = make_colour_rgb(50, 71, 227);
   break;
  case 4:
   upcolor  = make_colour_rgb(34, 55, 215);
   break;
  case 5:
   upcolor  = make_colour_rgb(22, 43, 199);
   break;
  case 6:
   upcolor  = make_colour_rgb(11, 31, 181);
   break;
 }

 var upLineSpeed  = random_range(2, 9); // Speed

 upLineGrid[upCounter, 0] = upLineThickness;
 upLineGrid[upCounter, 1] = upLineLength;
 upLineGrid[upCounter, 2] = upcolor;
 upLineGrid[upCounter, 3] = upxCounter * upLineMaxThickness; // x-position
 upLineGrid[upCounter, 4] = upyCounter * upLineMaxLength;  // y-position
 upLineGrid[upCounter, 5] = upLineSpeed;
 upLineGrid[upCounter, 6] = upLineSpeed;      // shift / move

 upCounter++;
 upxCounter++;
 upyCounter++;

 if (upxCounter >= upLineCountX) {upxCounter = 0;}
 if (upyCounter >= upLineCountY) {upyCounter = 0;}
}

// create left lines
for (var i=0; i<leftLinesComplete; i++)
{
 var leftLineLength  = floor(random_range(leftLineMinLength, leftLineMaxLength
 var leftLineThickness = floor(random_range(leftLineMinThickness, leftLineMaxT
```

```
 // Color
 var leftcolorNum  = floor(random_range(1, 7));
 var leftcolor   = c_white;

 switch (leftcolorNum)
 {
  case 1:
   leftcolor  = make_colour_rgb(242, 154, 93);
   break;
  case 2:
   leftcolor  = make_colour_rgb(227, 133, 67);
   break;
  case 3:
   leftcolor  = make_colour_rgb(220, 121, 52);
   break;
  case 4:
   leftcolor  = make_colour_rgb(235, 121, 41);
   break;
  case 5:
   leftcolor  = make_colour_rgb(184, 77, 2);
   break;
  case 6:
   leftcolor  = make_colour_rgb(255, 105, 0);
   break;
 }

 var leftLineSpeed  = random_range(2, 9); // Speed

 leftLineGrid[leftCounter, 0] = leftLineLength;
 leftLineGrid[leftCounter, 1] = leftLineThickness;
 leftLineGrid[leftCounter, 2] = leftcolor;
 leftLineGrid[leftCounter, 3] = leftxCounter * leftLineMaxLength;  // x-positi
 leftLineGrid[leftCounter, 4] = leftyCounter * leftLineMaxThickness; // y-posi
 leftLineGrid[leftCounter, 5] = leftLineSpeed;
 leftLineGrid[leftCounter, 6] = leftLineSpeed;      // shift / move

 leftCounter++;
 leftxCounter++;
 leftyCounter++;

 if (leftxCounter >= leftLineCountX) {leftxCounter = 0;}
 if (leftyCounter >= leftLineCountY) {leftyCounter = 0;}
}
```

Das sieht komplizierter aus als es ist. Oben zählen wir noch kurz zusammen, wie viele Linien es insgesamt sind. Dann kommen auch schon die beiden Schleifen. In der jeweiligen Schleife wird erst einmal mit den min und max Werten die Länge und Breite per Zufall definiert. Dann legen wir eine von sechs verschiedenen Farben fest. Hier ist es sinnvoll Abstufungen einer Farbe zu nehmen, es ist aber auch möglich, beliebige Farben zu wählen.
Nachdem eine der Farben gewählt wurde, wird das Array erstellt. Die letzten Zeilen befassen sich mit unseren Countern. Wir haben drei davon, Einer ist für die X-Koordinate, einer für die Y-Koordinate und eine für die Liste zuständig.

Event Draw

Nun werden die Rechtecke in einer großen Schleife gezeichnet. Hierfür haben wir wieder unsere beiden Counter und die Variable „effect”. Diese Variable brauchen wir um zwischen den beiden Effekten umzuschalten.

```
var effect  = 0; // switch the 2 effects
var upCounter = 0;
var leftCounter = 0;

for (var i=0; i<linesComplete; i++)
{
 if (effect = 0) && (upCounter < upLinesComplete)
 {
  // draw up-Lines
  upLineGrid[upCounter, 6] -= upLineGrid[upCounter, 5];
  var yPosAbove = upLineGrid[upCounter, 4]+upLineGrid[upCounter, 6];
  var yPosBelow = upLineGrid[upCounter, 4]+upLineGrid[upCounter, 1]+upLineGrid

  // Draw lines
  draw_set_alpha(0.6);
  draw_set_color(upLineGrid[upCounter, 2]);
  draw_roundrect(upLineGrid[upCounter, 3], yPosAbove, upLineGrid[upCounter, 3]

  // Draw blur
  draw_set_alpha(0.15);
  var distance = 4;
  draw_roundrect(upLineGrid[upCounter, 3] - distance, yPosAbove, upLineGrid[up
  draw_roundrect(upLineGrid[upCounter, 3] + distance, yPosAbove, upLineGrid[up
  draw_roundrect(upLineGrid[upCounter, 3], yPosAbove - distance, upLineGrid[up
  draw_roundrect(upLineGrid[upCounter, 3], yPosAbove + distance, upLineGrid[up

  // Move the line down
  if (yPosBelow <= 0)
  {
   upLineGrid[upCounter, 4] = room_height;
   upLineGrid[upCounter, 6] = 0;
  }

  if (upCounter <= upLinesComplete) {upCounter++;}

 } else if (leftCounter < leftLinesComplete) {
```

```
  // left-Lines
  leftLineGrid[leftCounter, 6] -= leftLineGrid[leftCounter, 5];

  var xPosFront = leftLineGrid[leftCounter, 3] + leftLineGrid[leftCounter, 6];
  var xPosBehind = leftLineGrid[leftCounter, 3] + leftLineGrid[leftCounter, 0]

  // Draw lines
  draw_set_alpha(0.6);
  draw_set_color(leftLineGrid[leftCounter, 2]);
  draw_roundrect(xPosFront, leftLineGrid[leftCounter, 4], xPosBehind, leftLine

  // Draw blur
  draw_set_alpha(0.15);
  var distance = 4;
  draw_roundrect(xPosFront - distance, leftLineGrid[leftCounter, 4], xPosBehin
  draw_roundrect(xPosFront + distance, leftLineGrid[leftCounter, 4], xPosBehin
  draw_roundrect(xPosFront, leftLineGrid[leftCounter, 4] - distance, xPosBehin
  draw_roundrect(xPosFront, leftLineGrid[leftCounter, 4] + distance, xPosBehin

  // Move the line left
  if (xPosBehind <= 0)
  {
   leftLineGrid[leftCounter, 3] = room_width;
   leftLineGrid[leftCounter, 6] = 0;
  }

  if (leftCounter < leftLinesComplete) {leftCounter++;}
 }

 // switch the 2 effects
 if (effect = 0)
 {
  effect = 1;
 } else {
  effect = 0;
 }
}
```

Hier berechnen wir die Y-Position des Rechtecks. Die Variablen 6 und 5 sind die Geschwindigkeit, was in dem Fall nichts anderes ist als die horizontale Verschiebung in Pixeln. 4 und 1 betreffen die Länge des Rechtecks. Das haben wir ja im Create-Event vorher definiert.

```
upLineGrid[upCounter, 6] -= upLineGrid[upCounter, 5];
var yPosAbove = upLineGrid[upCounter, 4]+upLineGrid[upCounter, 6];
var yPosBelow = upLineGrid[upCounter, 4]+upLineGrid[upCounter, 1]+upLineGrid[u
```

Hier zeichnen wir das Rechteck.

```
draw_set_alpha(0.6);
draw_set_color(upLineGrid[upCounter, 2]);
draw_roundrect(upLineGrid[upCounter, 3], yPosAbove, upLineGrid[upCounter, 3]+u
```

Nachfolgend kommt noch ein wesentlicher Teil des Codes. Wenn das Rechteck oben außerhalb des Bildschirms ist, wird es nach unten verschoben. *upLineGrid[upCounter, 4]* ist die neue Position, *upLineGrid[upCounter, 6]*

die Verschiebung, die wieder bei 0 beginnt. Der Code für die Linien nach links ist vom Prinzip her gleich, weshalb ich ihn nicht mehr erklären muss.

```
if (yPosBelow <= 0)
{
 upLineGrid[upCounter, 4] = room_height;
 upLineGrid[upCounter, 6] = 0;
}
```

Am Ende der Schleife wird zwischen den beiden Effekten hin und her geschaltet:

```
// switch the 2 effects
if (effect = 0)
{
 effect = 1;
} else {
 effect = 0;
}
```

Noch ein paar Worte zum Blur-Effekt. Dabei handelt es sich lediglich um eine Kopie des Rechtecks, nur wird dies in alle Richtungen um 4 Pixel verschoben. Mehr ist das nicht.

Schicker Hintergrundeffekt

1 Datei(en) 16.22 KB
Download

Date Created
22. März 2017
Author
sven