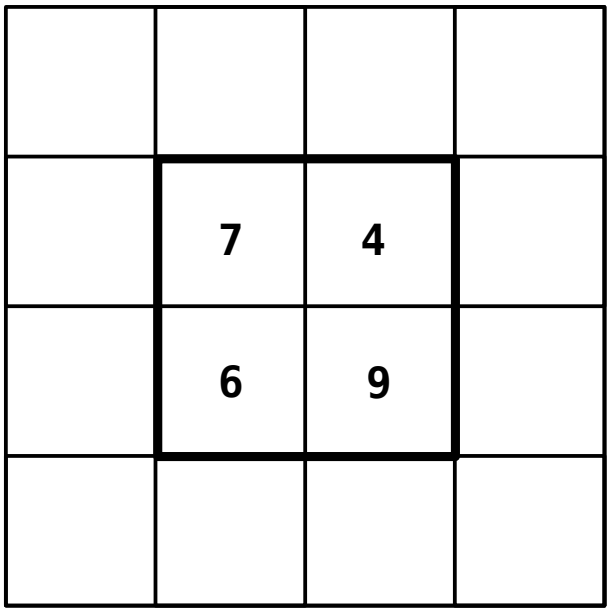
Error diffusion

1) Perform Floyd-Steinberg dithering in the below image for the 4 pixels inside the black frame. Round all errors to integer numbers.

Gray values range form 0-15.



error filter

	*	7/16
3/16	5/16	1/16

Error diffusion

1) Perform Floyd-Steinberg dithering for the below image but only for the 4 pixels inside the black frame.

$7 \Rightarrow 0$	$4+7\frac{7}{16}$
$6+7\frac{5}{16}$	$9+7\frac{1}{16}$

$7 \Rightarrow 0$	$4+3=7 \Rightarrow 0$
$6+\frac{7x5+7x3}{16}$	$9+\frac{7x1+7x5}{16}$

$7 \Rightarrow 0$	$7 \Rightarrow 0$
$10 \Rightarrow 15$	$9+\frac{7x1+7x5-7x5}{16}$

$7 \Rightarrow 0$	$7 \Rightarrow 0$
$10 \Rightarrow 15$	$9 \Rightarrow 15$

error filter

	*	7/16
3/16	5/16	1/16

Error diffusion

Im Archiv **`floydt_steinberg_templ2.tgz`** finden Sie den Stand der letzten Veranstaltung.

2) Bisher haben wir nur den Grauwert auf 0 oder 255 quantisiert. Wie sieht ein Farbbild aus, dessen RGB Werte man einzeln Quantisiert. Quantisieren Sie nun statt der Grauwerte die Farben, zuerst ohne Dithering.

3) Wie viele verschiedene Farben erhalten Sie bei einer Quantisierung von Rot, Grün und Blau in nur zwei Stufen? Nennen Sie alle Farben.

4) Wie lässt sich Floydt-Steinberg nun auf Farben übertragen. Implementieren Sie dies.

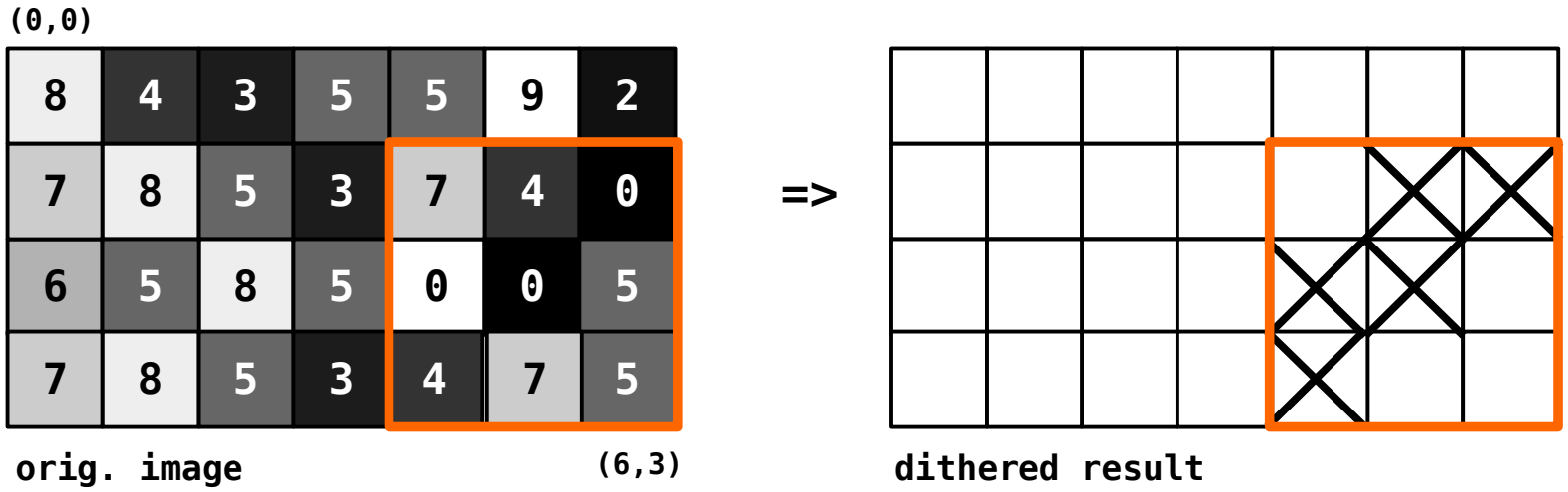
5) Bisher hatten wir nur zwei Abstufungen pro Kanal. Was ändert sich, wenn mehr als zwei Quantisierungsschritte verwendet werden?

Ordered dithering

1) How many gray value does the 3x3 patter allow? **Answer:** 10 (0-9)

2) Dither the following gray image only within the orange area. Mark black pixels with a cross in the dithered result, leave white ones blank.

5	8	3
7	1	9
4	6	2



Ordered dithering

3) Implementieren Sie nun Ordered Dithering. Dafür können Sie die Funktion

```
int DitherOrdered(int x, int y, int v)
```

im Template verwenden.

4) Wie lässt sich der Ansatz auf Farben übertragen, wenn wir wieder davon ausgehen, dass pro Farbkanal nur zwei Quantisierungsstufen verwendet werden? Verändern Sie Ihr Programm entsprechend.

5) Wie funktioniert ordered Dithering, wenn z.B. 4 Graustufen verwendet werden?
Hinweis: Denken Sie daran, dass Ihr Fehler positiv und negativ werden kann. In der Dithering Matrix stehen aber nur positive Werte.

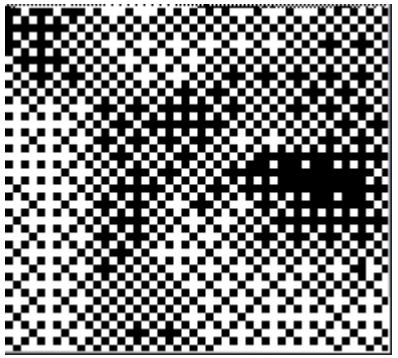






Ordered dithering

3) Implementieren Sie nun Ordered Dithering.



Lösung:

In der Doppelschleife:

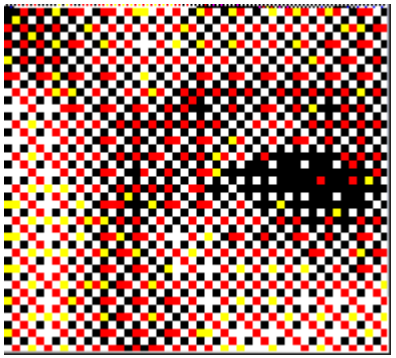
```
int dithered_gray = DitherOrdered(x, y, orig_gray);
SetGray(p_source_image, x, y, dithered_gray);
```

In DitherOrdered(int x, int y, int v):

```
if((v*NO_DOTS*NO_DOTS)/255 >
                dot_matrix[x % NO_DOTS][y % NO_DOTS]) return 255;
else return 0;
```

Ordered dithering

4) Wie lässt sich der Ansatz auf Farben übertragen, wenn wir wieder davon ausgehen, dass pro Farbkanal nur zwei Quantisierungsstufen verwendet werden?



Lösung:

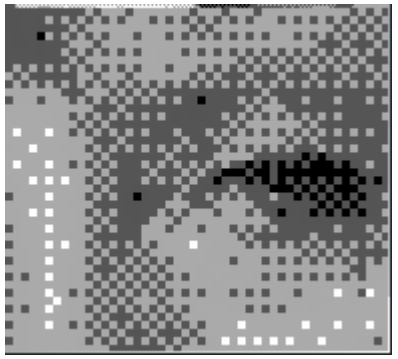
In der Doppelschleife:

```
int dithered_red =   DitherOrdered(x, y, orig_r);
int dithered_green = DitherOrdered(x, y, orig_g);
int dithered_blue =  DitherOrdered(x, y, orig_b);

SetColor(p_source_image, x, y, dithered_red,
                               dithered_green,
                               dithered_blue);
```

Ordered dithering

5) Wie funktioniert ordered Dithering, wenn z.B. 4 Graustufen verwendet werden?



Lösung:

Idee: Es wird immer so quantisiert, dass die letzte Quantisierungsstufe nicht mehr erreicht wird. Beispiel: QStufen bei 4 Shades: 0, 64, 128, 192. Nur der Fehler, also **`orig_gray - quantized_gray`** geht dann in das Ordered Dithering. “Gewinnt” der Fehler, dann wird nochmal eine Quantisierungsstufe hinzugerechnet. In der Doppelschleife:

```
int quantized_gray = ConvertShadeLowerLimit(orig_gray);
int dithered_error = DitherOrdered(x, y, orig_gray - quantized_gray);
SetGray(p_source_image, x, y, quantized_gray + dithered_error);
```

```
int ConvertShadeLowerLimit(int v)
{
  int size_interval = 256/(NO_SHADES-1);

  v = v / size_interval;
  v = v * size_interval; // gleicher Faktor zum Zurückskalieren!

  return v;
} // Convert ShadeLowerLimit
```