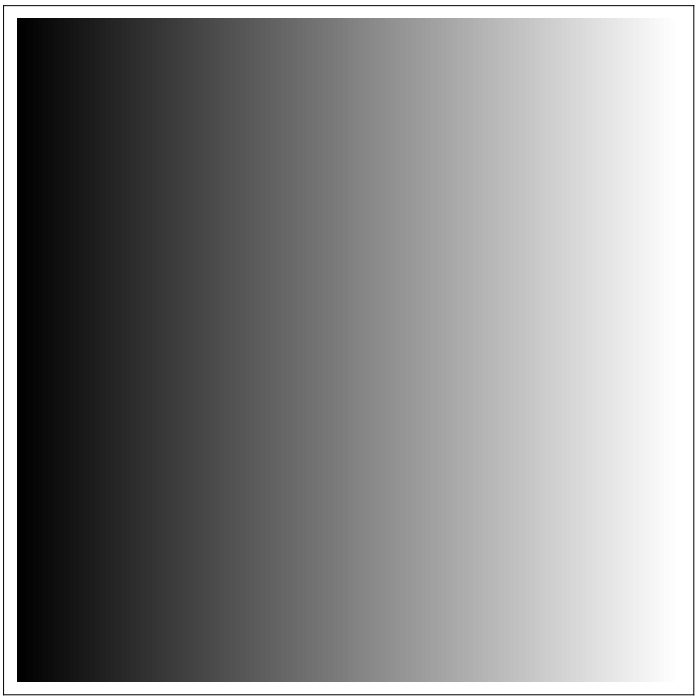
In[1]:=
```
SetDirectory@NotebookDirectory[];
QuantizeBits[x_, bits_, dither_: 0] :=
 Round[x * (Power[2, bits] - 1) + dither] / (Power[2, bits] - 1)
```

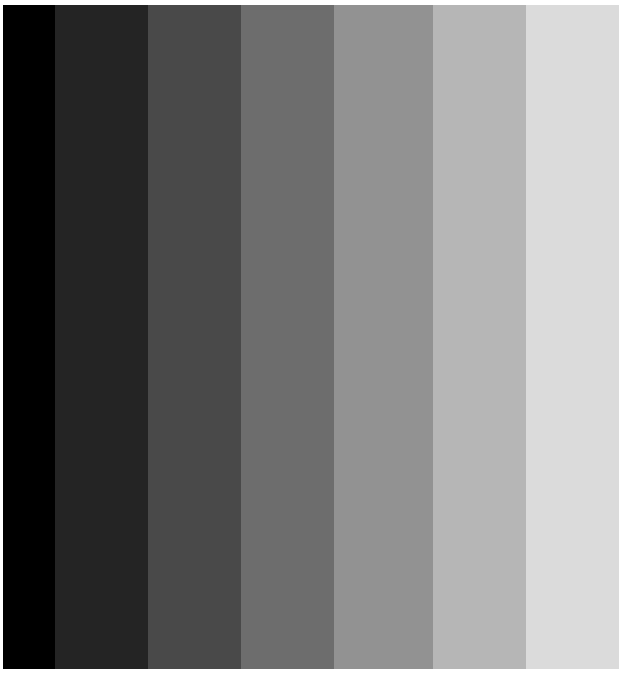
In[3]:=
```
SampleArrayWithWrap[arr_, x_, y_] :=
 Module[{dims = Dimensions[arr]}, arr[[Mod[x, dims[[1]], 1], Mod[y, dims[[2]], 1]]]]
origSignal = Table[x / 128, {y, 0, 127}, {x, 0, 127}];
ArrayPlot[origSignal, ColorFunction → GrayLevel]
```

Out[5]=



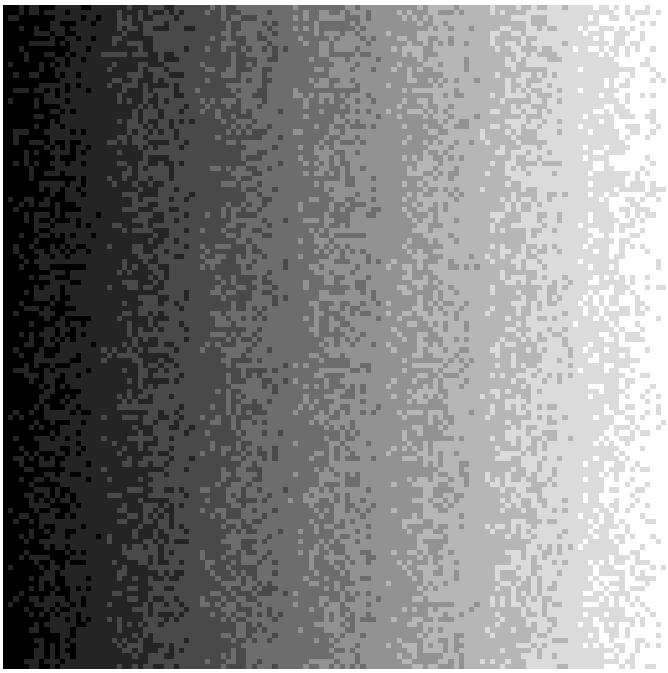
In[6]:=
```
ArrayPlot[QuantizeBits[#1, 3] & /@ origSignal, ColorFunction → GrayLevel]
```

Out[6]=



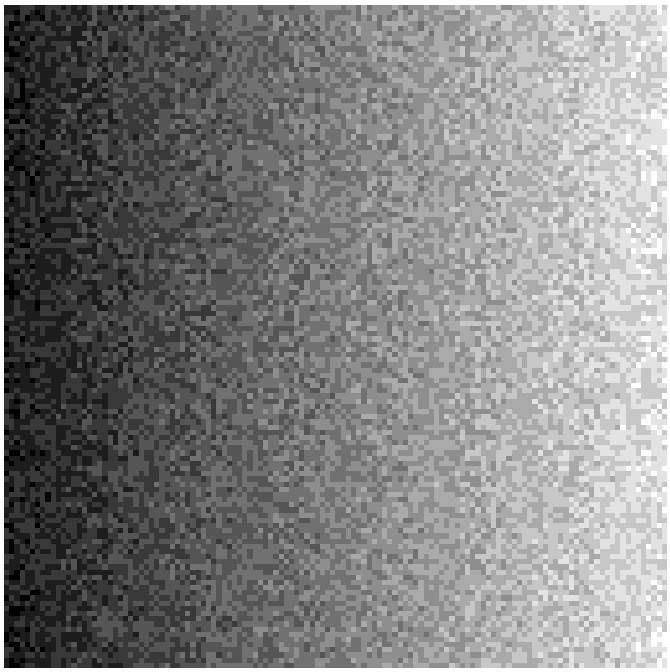
In[7]:=
```
ditheredWhite = Map[QuantizeBits[#1, 3, RandomReal[] - 0.5] &, origSignal, {2}];
ArrayPlot[ditheredWhite, ColorFunction → GrayLevel]
```

Out[8]=



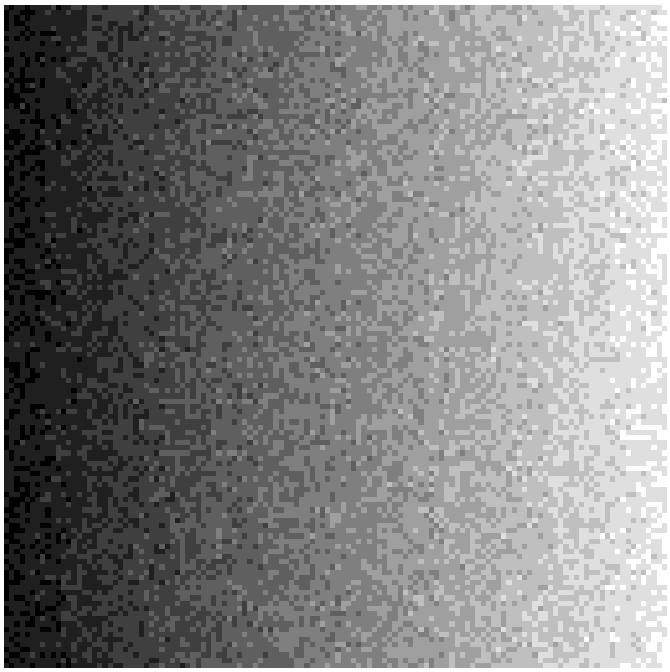
In[9]:=
```
ArrayPlot[Map[QuantizeBits[#1, 3, (RandomReal[] - 0.5) * 2] &, origSignal, {2}],
 ColorFunction → GrayLevel]
```

Out[9]=



In[10]:=
```
ArrayPlot[Map[QuantizeBits[#1, 3, (RandomReal[] + RandomReal[] - 0.5)] &,
  origSignal, {2}], ColorFunction → GrayLevel]
```

Out[10]=

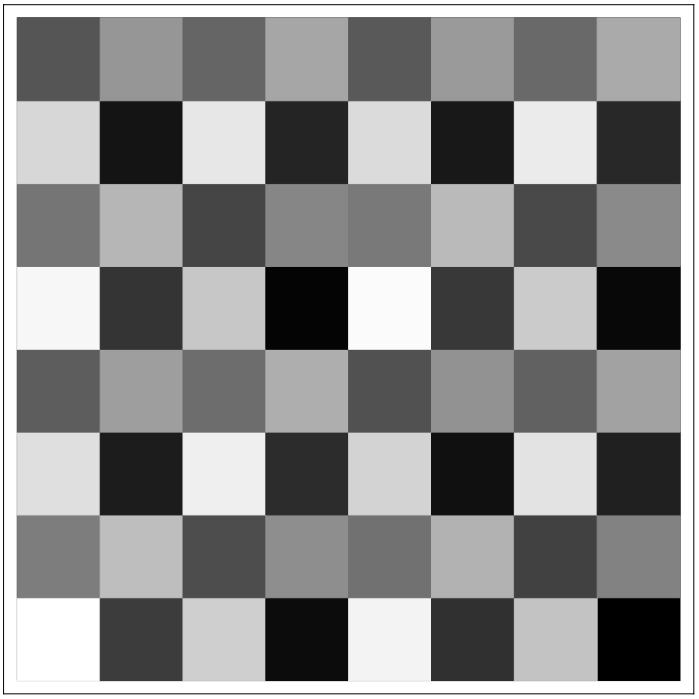


In[11]:=
```
Bayer[x_, y_, level_] :=
 Mod[Mod[BitShiftRight[x, level], 2] + 1 + 2 * Mod[BitShiftRight[y, level], 2], 4] +
  If[level == 0, 0, 4 * Bayer[x, y, level - 1]]
bayer3 = Table[Bayer[x, y, 2], {y, 0, 7}, {x, 0, 7}] / 63;
bayer6 = Table[Bayer[x, y, 5], {y, 0, 63}, {x, 0, 63}] / (64 * 64 - 1);
```

In[14]:=
```
ditheredBayer = MapIndexed[
   QuantizeBits[#1, 3, SampleArrayWithWrap[bayer3, #2[[1]], #2[[2]]] - 0.5] &,
   origSignal, {2}];
```

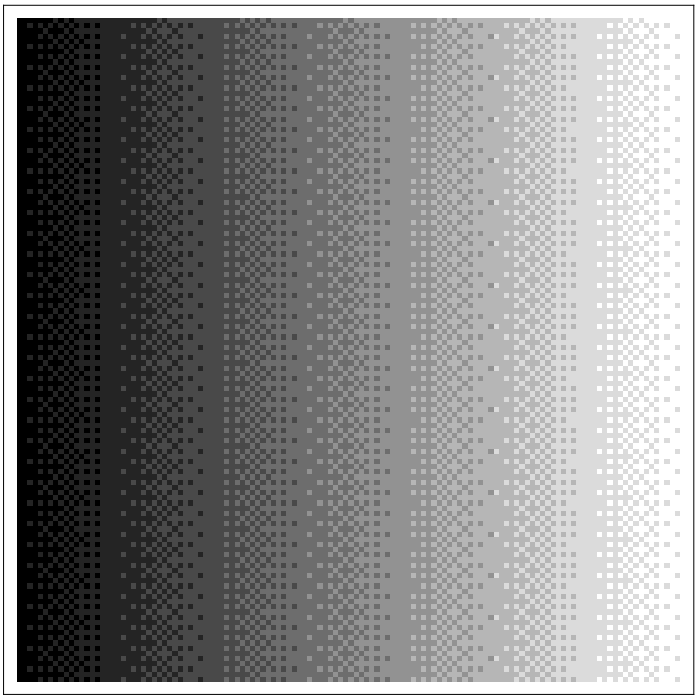
In[15]:=
```
ArrayPlot[bayer3, ColorFunction → GrayLevel]
```

Out[15]=



In[16]:=
```
ArrayPlot[ditheredBayer, ColorFunction → GrayLevel]
```

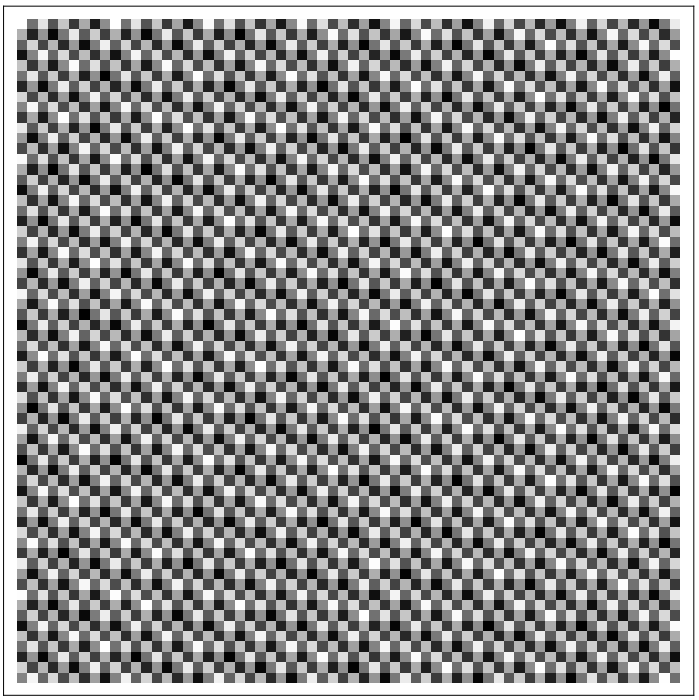
Out[16]=



In[17]:=
```
InterleavedGradientNoise[x_, y_] :=
 FractionalPart[52.9829189 * FractionalPart[0.06711056 * x + 0.00583715 * y]]
```

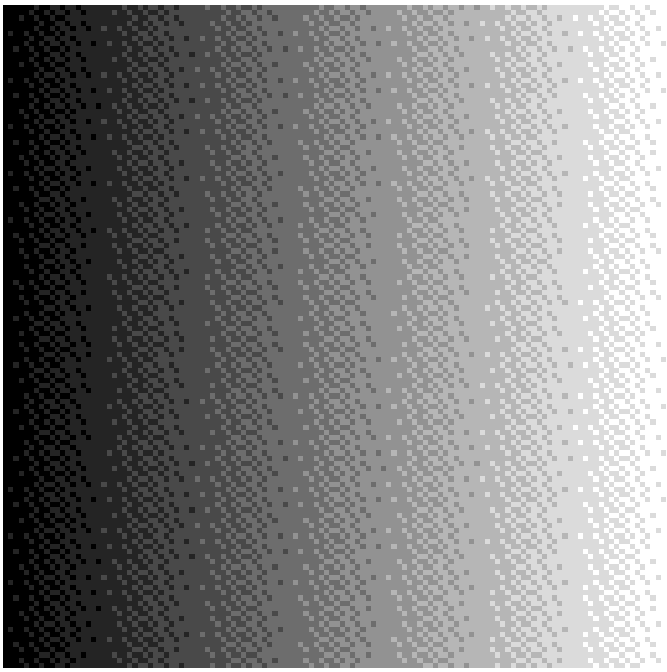
In[18]:=
```
ArrayPlot[Table[InterleavedGradientNoise[x, y], {y, 0, 63}, {x, 0, 63}]]
```

Out[18]=



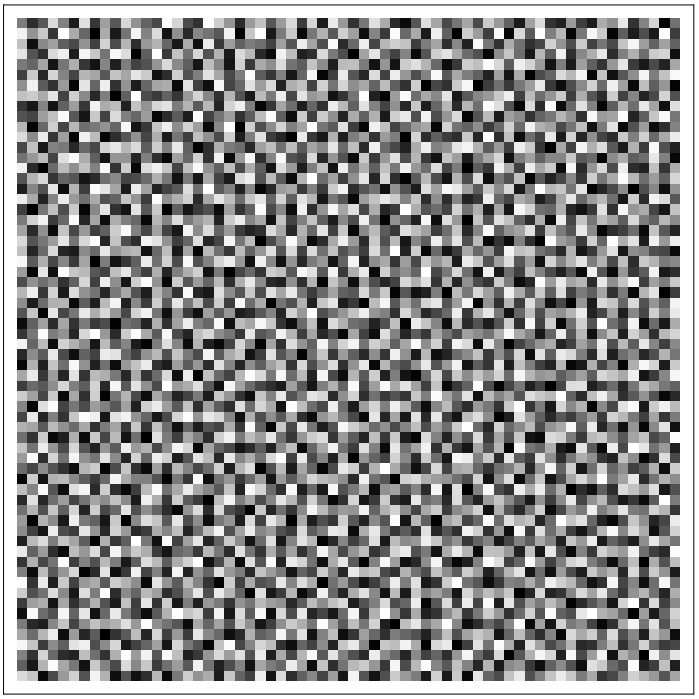
In[19]:=
```
ditheredInterleaved =
  MapIndexed[QuantizeBits[#1, 3, InterleavedGradientNoise[#2[[1]], #2[[2]]] - 0.5] &,
   origSignal, {2}];
ArrayPlot[ditheredInterleaved, ColorFunction → GrayLevel]
```

Out[20]=



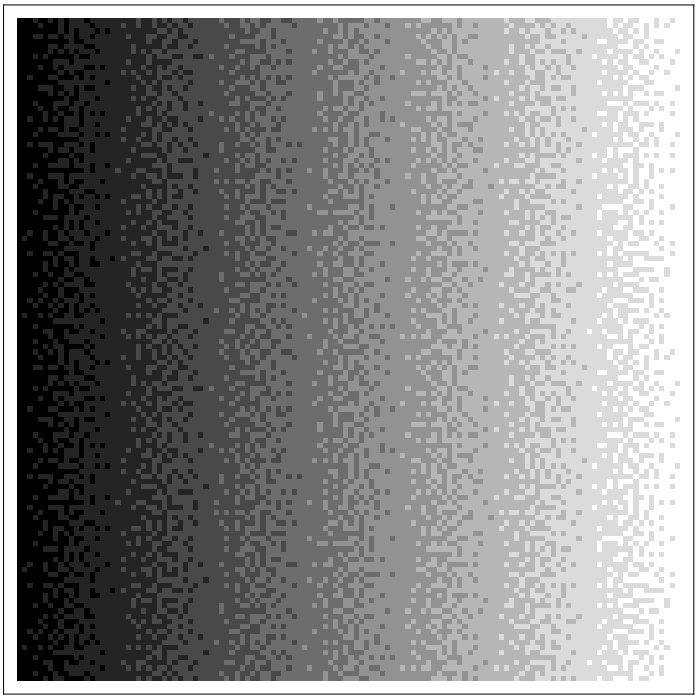
In[59]:=
```
(* Blue noise definition ommited for PDF *)

ArrayPlot[blueNoise64, ColorFunction → GrayLevel]
```

Out[59]=



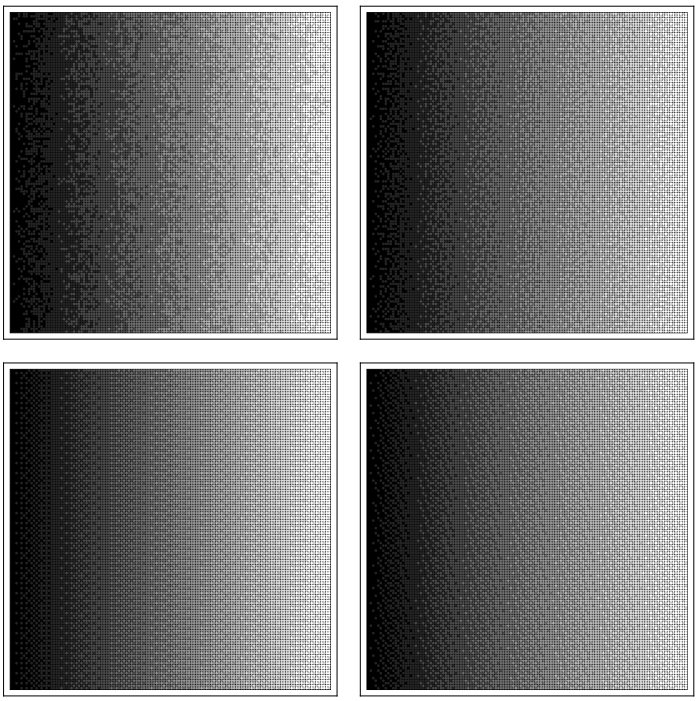
In[23]:=
```
ditheredBlue = MapIndexed[
   QuantizeBits[#1, 3, SampleArrayWithWrap[blueNoise64, #2[[1]], #2[[2]]] - 0.5] &,
   origSignal, {2}];
ArrayPlot[ditheredBlue, ColorFunction → GrayLevel]
```

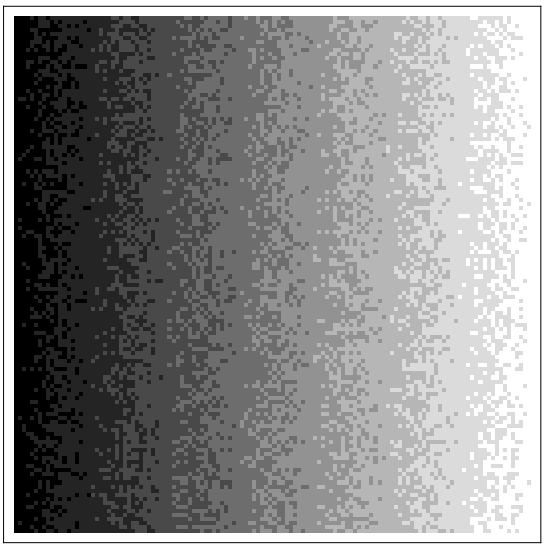
Out[24]=

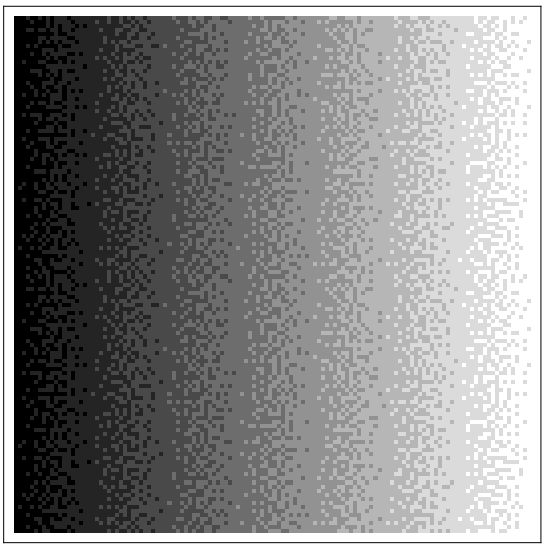


In[25]:=
```
GraphicsGrid[{{ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
    ArrayPlot[ditheredBlue, ColorFunction → GrayLevel]},
   {ArrayPlot[ditheredBayer, ColorFunction → GrayLevel],
    ArrayPlot[ditheredInterleaved, ColorFunction → GrayLevel]}}]
Export["2dditheringalltechniquescompared.gif",
   {ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
    ArrayPlot[ditheredBlue, ColorFunction → GrayLevel],
    ArrayPlot[ditheredBayer, ColorFunction → GrayLevel], ArrayPlot[
     ditheredInterleaved, ColorFunction → GrayLevel]}, "DisplayDurations" → 2];
```

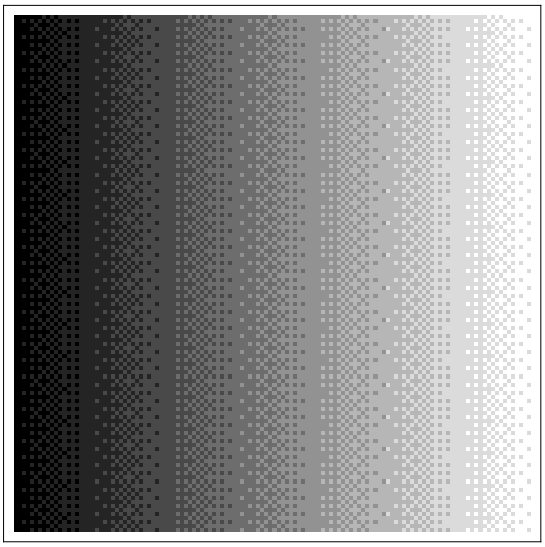
Out[25]=

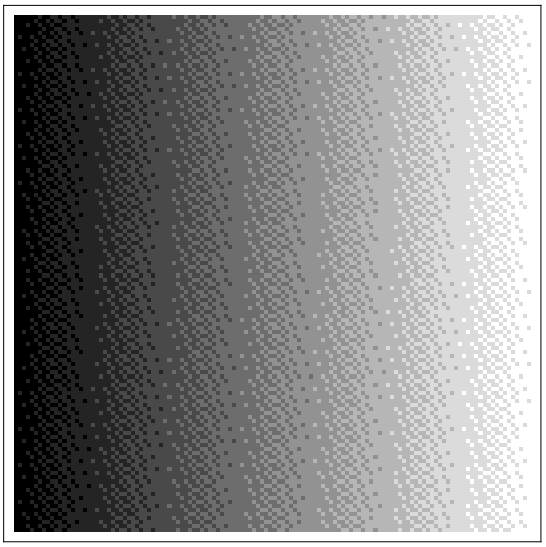




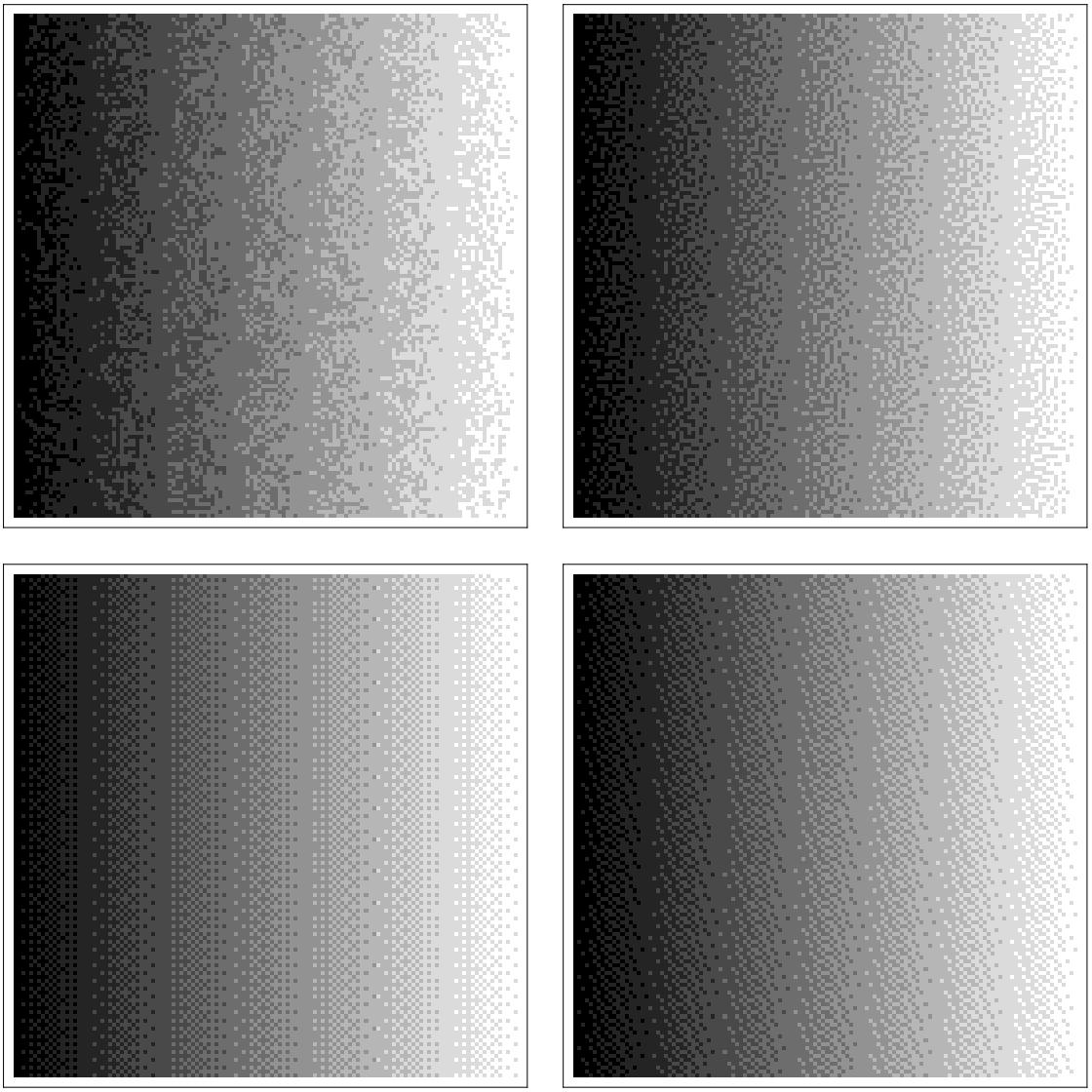


In[27]:=





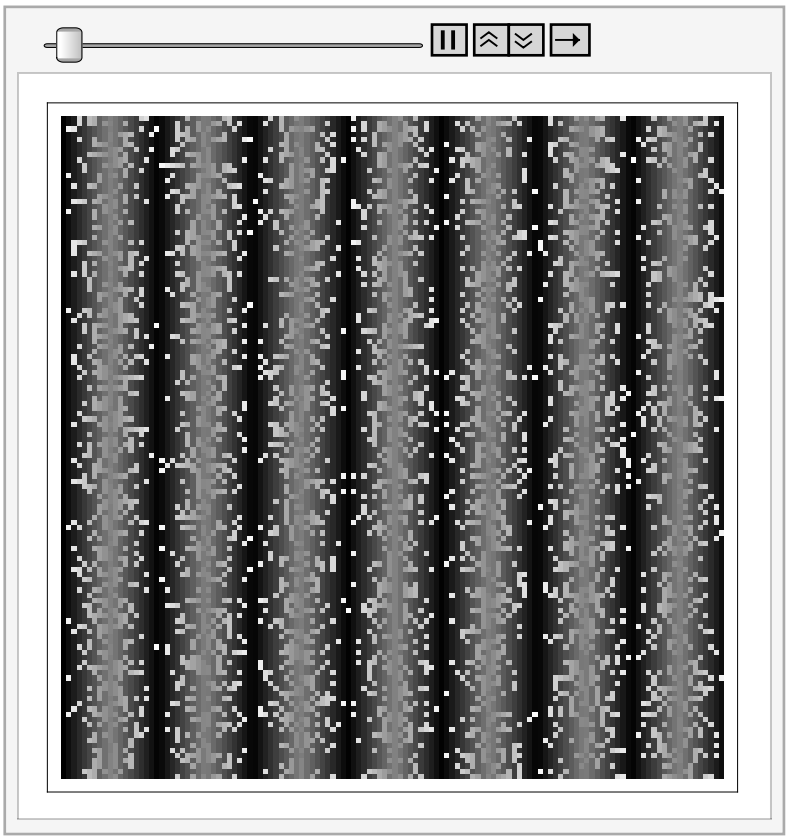
Out[27]=



In[28]:=
```
Export["2dditheringquandizecomparisongrid.png",
  GraphicsGrid[{{ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
     ArrayPlot[ditheredBlue, ColorFunction → GrayLevel]},
    {ArrayPlot[ditheredBayer, ColorFunction → GrayLevel],
     ArrayPlot[ditheredInterleaved, ColorFunction → GrayLevel]}}]];
```

In[29]:=
```
ListAnimate[{ArrayPlot[Abs[ditheredWhite - origSignal], ColorFunction → GrayLevel],
  ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
  ArrayPlot[origSignal, ColorFunction → GrayLevel]}]
```

Out[29]=

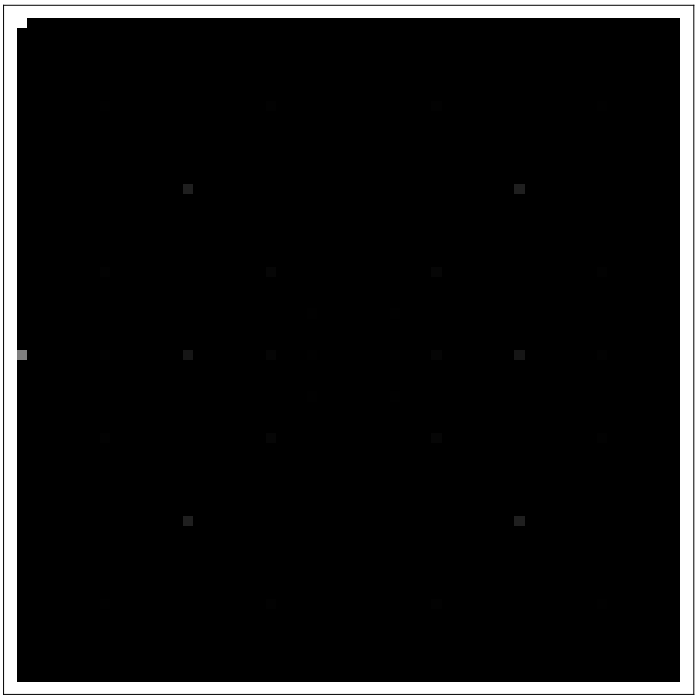


In[30]:=
```
Export["2dditheringwhitebands.gif",
  {ArrayPlot[Abs[ditheredWhite - origSignal], ColorFunction → GrayLevel],
   ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
   ArrayPlot[origSignal, ColorFunction → GrayLevel]}, "DisplayDurations" → 2];
```

In[31]:=
```
MyTwoDimPeriodogram[arr_] := ArrayPlot[
  Abs[RotateLeft[Fourier[arr], Dimensions[arr] / 2]], ColorFunction → GrayLevel]
```

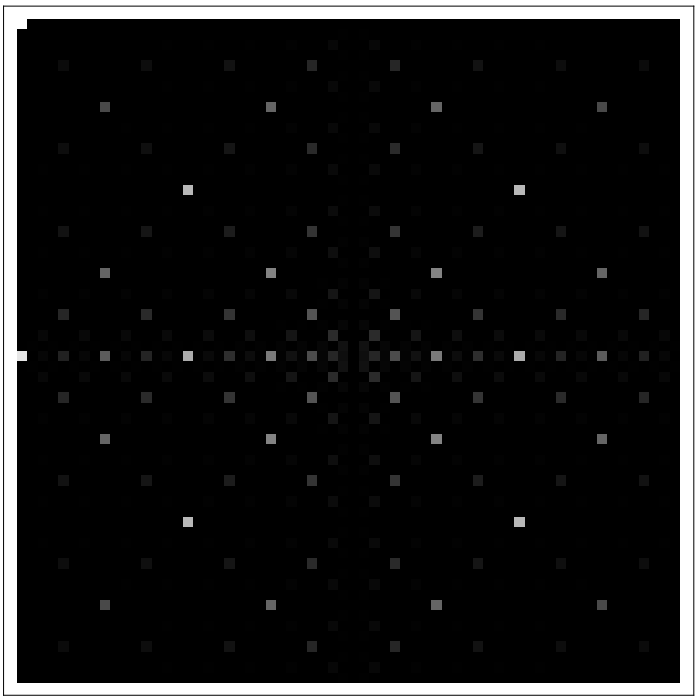
In[32]:=
```
MyTwoDimPeriodogram[bayer6 - 0.5]
```

Out[32]=



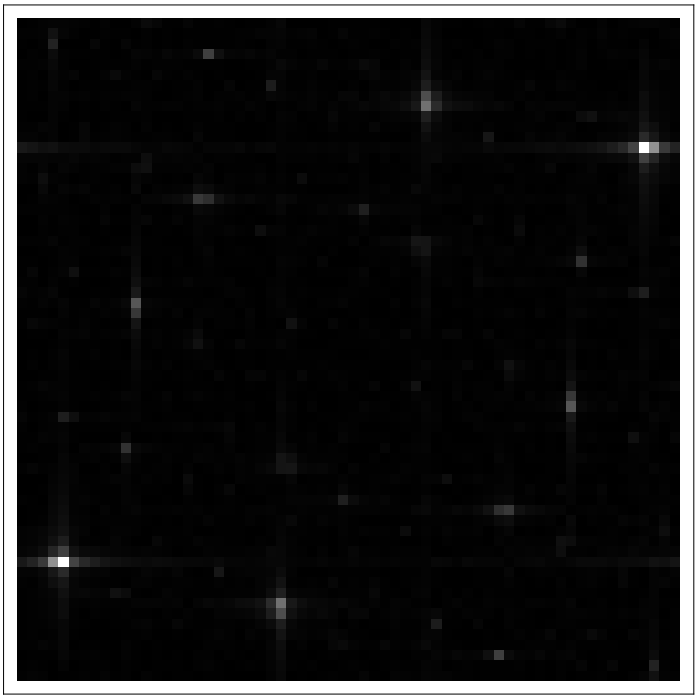
In[33]:=
```
ArrayPlot[Log[0.01 + Abs[RotateLeft[Fourier[bayer6 - 0.5], Dimensions[bayer6] / 2]]],
 ColorFunction → GrayLevel]
```

Out[33]=

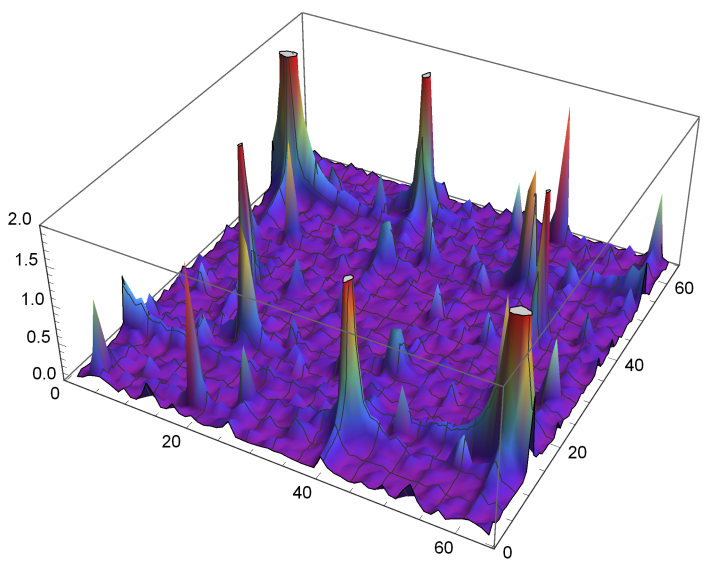


In[34]:=
```
MyTwoDimPeriodogram[
 Table[InterleavedGradientNoise[x, y], {y, 0, 63}, {x, 0, 63}] - 0.5]
ListPlot3D[Abs[RotateLeft[Fourier[
    Table[InterleavedGradientNoise[x, y], {y, 0, 63}, {x, 0, 63}] - 0.5], {32, 32}]],
 PlotRange → {0, 2}, ColorFunction → "Rainbow"]
```

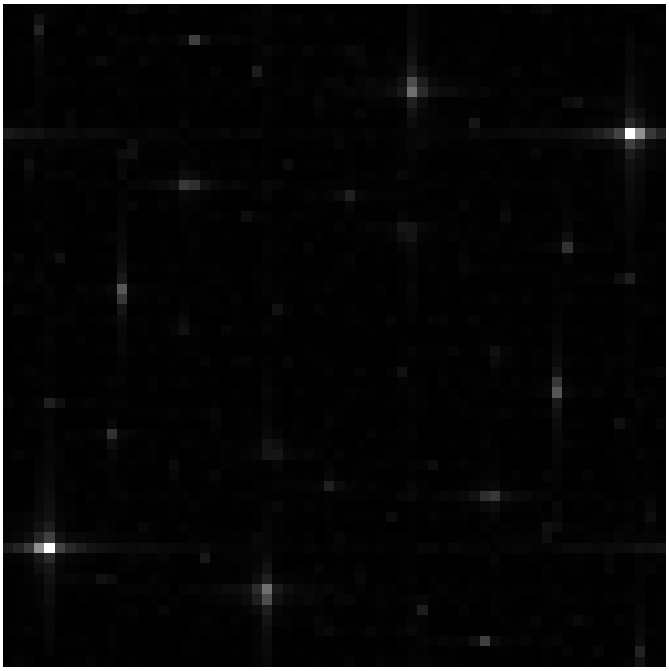
Out[34]=



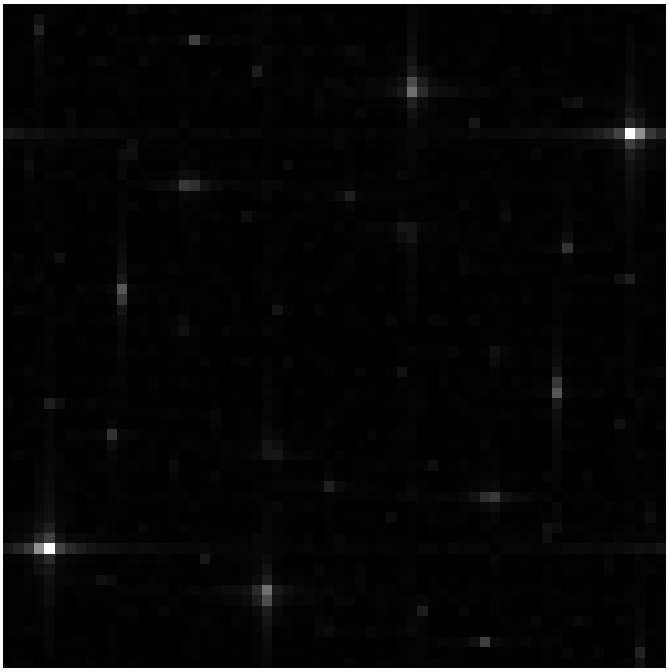
Out[35]=



In[36]:=

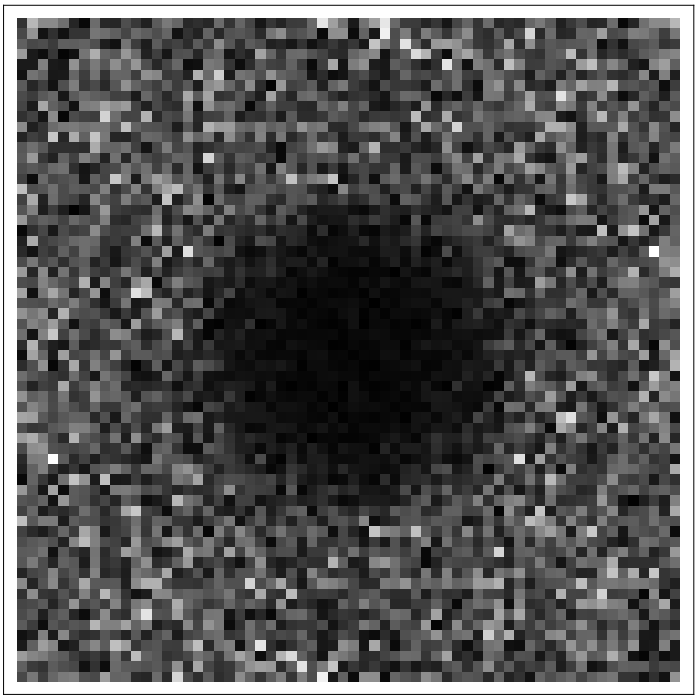


Out[36]=



In[37]:=
```
MyTwoDimPeriodogram[blueNoise64 - 0.5]
```

Out[37]=

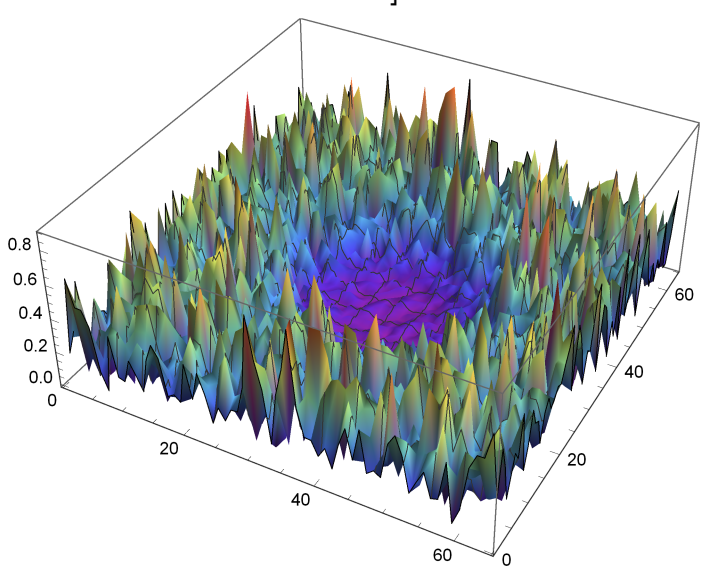


In[38]:=

In[39]:=

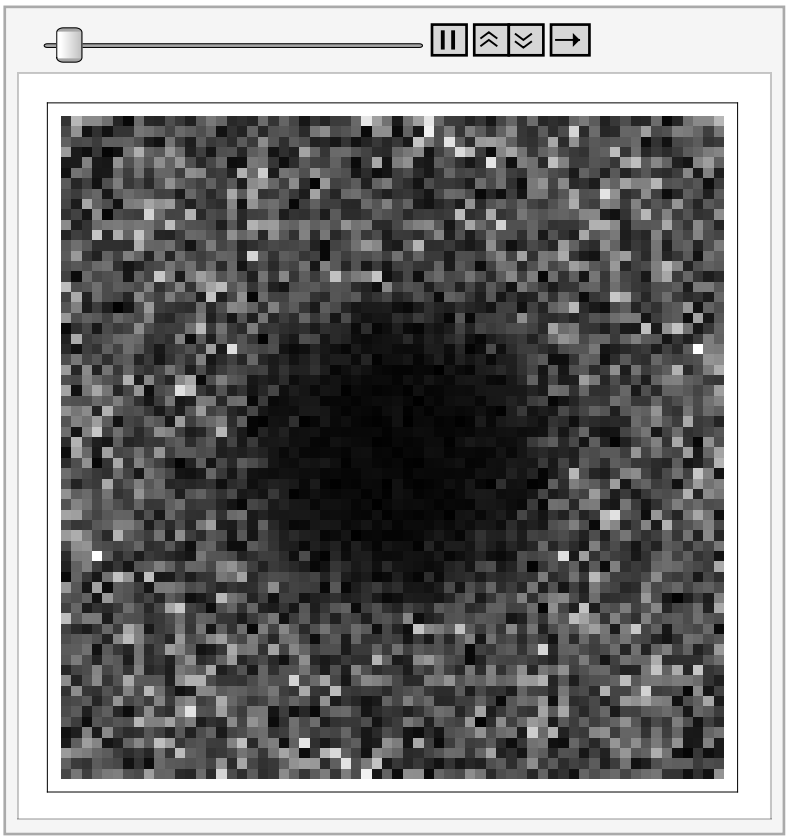
In[40]:=
```
ListPlot3D[Abs[RotateLeft[Fourier[blueNoise64 - 0.5], Dimensions[bayer6] / 2]],
 ColorFunction → "Rainbow"]
```

Out[40]=



In[41]:=
```
ListAnimate[{MyTwoDimPeriodogram[blueNoise64 - 0.5],
  MyTwoDimPeriodogram[Table[RandomReal[], {x, 0, 63}, {y, 0, 63}] - 0.5]}]
```

Out[41]=

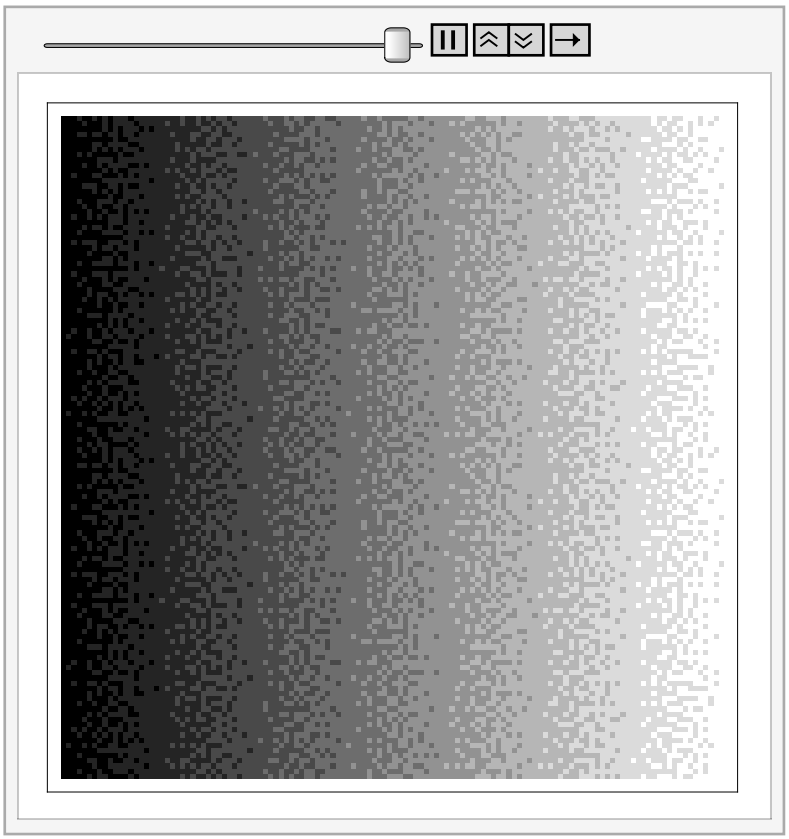


In[42]:=
```
Export["2dwhitenoisevsbluenoise.gif",
  {MyTwoDimPeriodogram[blueNoise64 - 0.5], MyTwoDimPeriodogram[
    Table[RandomReal[], {x, 0, 63}, {y, 0, 63}] - 0.5]}, "DisplayDurations" → 2];
```

In[43]:=

```
ListAnimate[{ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
  ArrayPlot[ditheredBlue, ColorFunction → GrayLevel]}]
```

Out[43]=

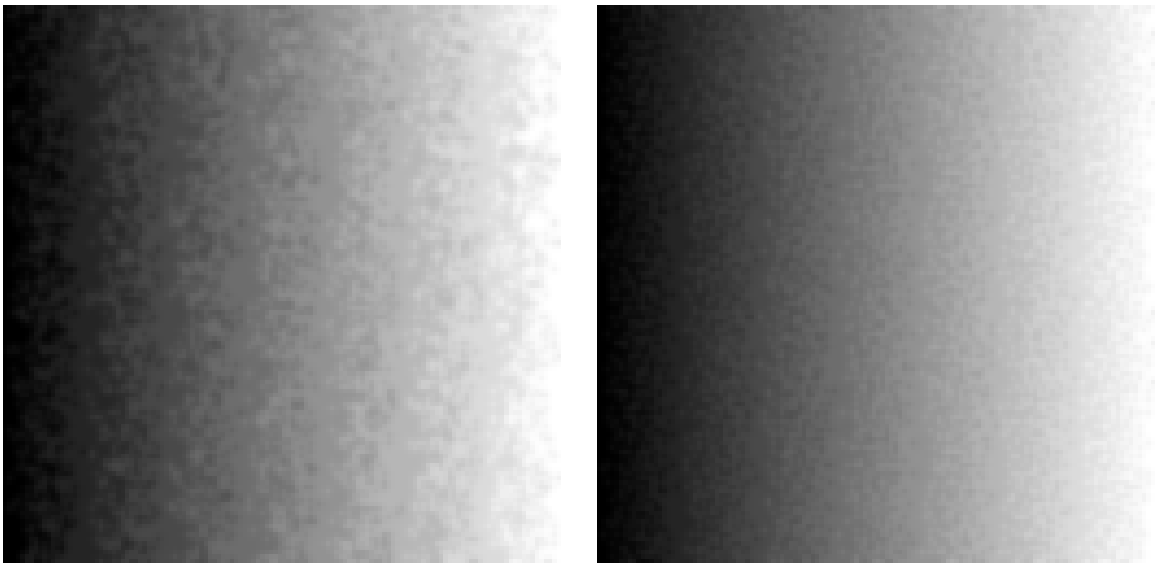


In[44]:=

```
Export["2dwhitenoisevsbluenoise.gif",
  {ArrayPlot[ditheredWhite, ColorFunction → GrayLevel],
   ArrayPlot[ditheredBlue, ColorFunction → GrayLevel]}, "DisplayDurations" → 2];
```

In[45]:=
```
GraphicsRow[
 {GaussianFilter[Image[ditheredWhite], 2], GaussianFilter[Image[ditheredBlue], 2]}]
```

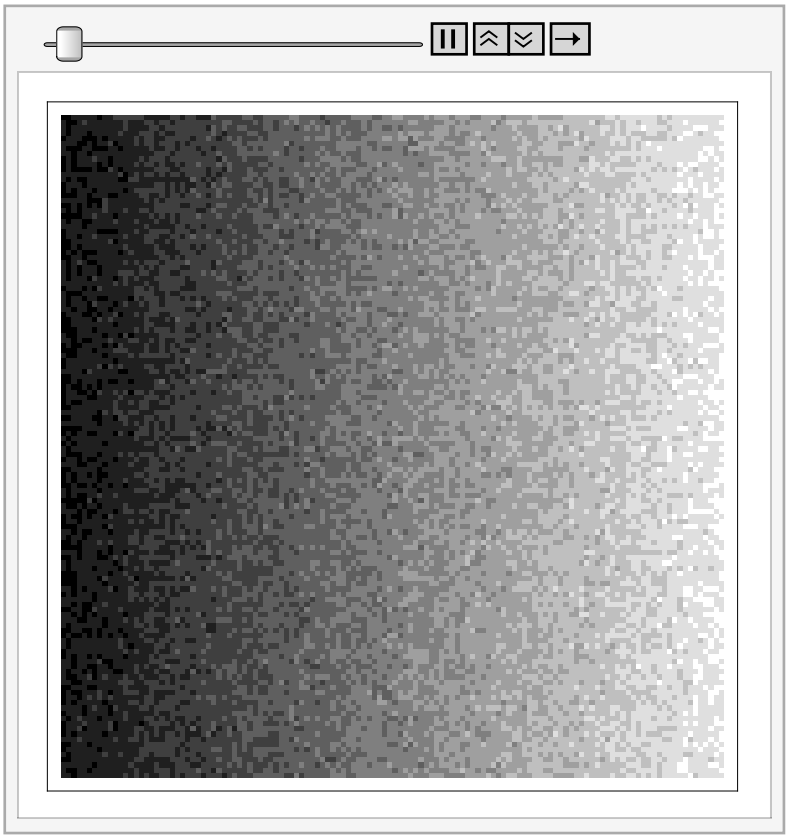
Out[45]=



In[46]:=

In[60]:=
```
remaptri[v_] := (orig = FractionalPart[v + 0.5] * 2 - 1;
  If[Abs[orig] == 0, -1, orig / Sqrt[Abs[orig]]] - Sign[orig] + 0.5)

ListAnimate[{ArrayPlot[
   Map[QuantizeBits[#1, 3, (RandomReal[] + RandomReal[] - 0.5)] &, origSignal, {2}],
   ColorFunction → GrayLevel], ArrayPlot[MapIndexed[QuantizeBits[#1, 3,
      remaptri[SampleArrayWithWrap[blueNoise64, #2[[1]], #2[[2]]]]] &,
    origSignal, {2}], ColorFunction → GrayLevel]}]
```

Out[61]=



In[49]:=
```
Export["2dwhitenoisevsbluenoisetriangle.gif",
  {ArrayPlot[Map[QuantizeBits[#1, 3, (RandomReal[] + RandomReal[] - 0.5)] &,
     origSignal, {2}], ColorFunction → GrayLevel],
   ArrayPlot[MapIndexed[QuantizeBits[#1, 3, remaptri[
        SampleArrayWithWrap[blueNoise64, #2[[1]], #2[[2]]]]] &, origSignal, {2}],
    ColorFunction → GrayLevel]}, "DisplayDurations" → 2];
```

In[50]:=

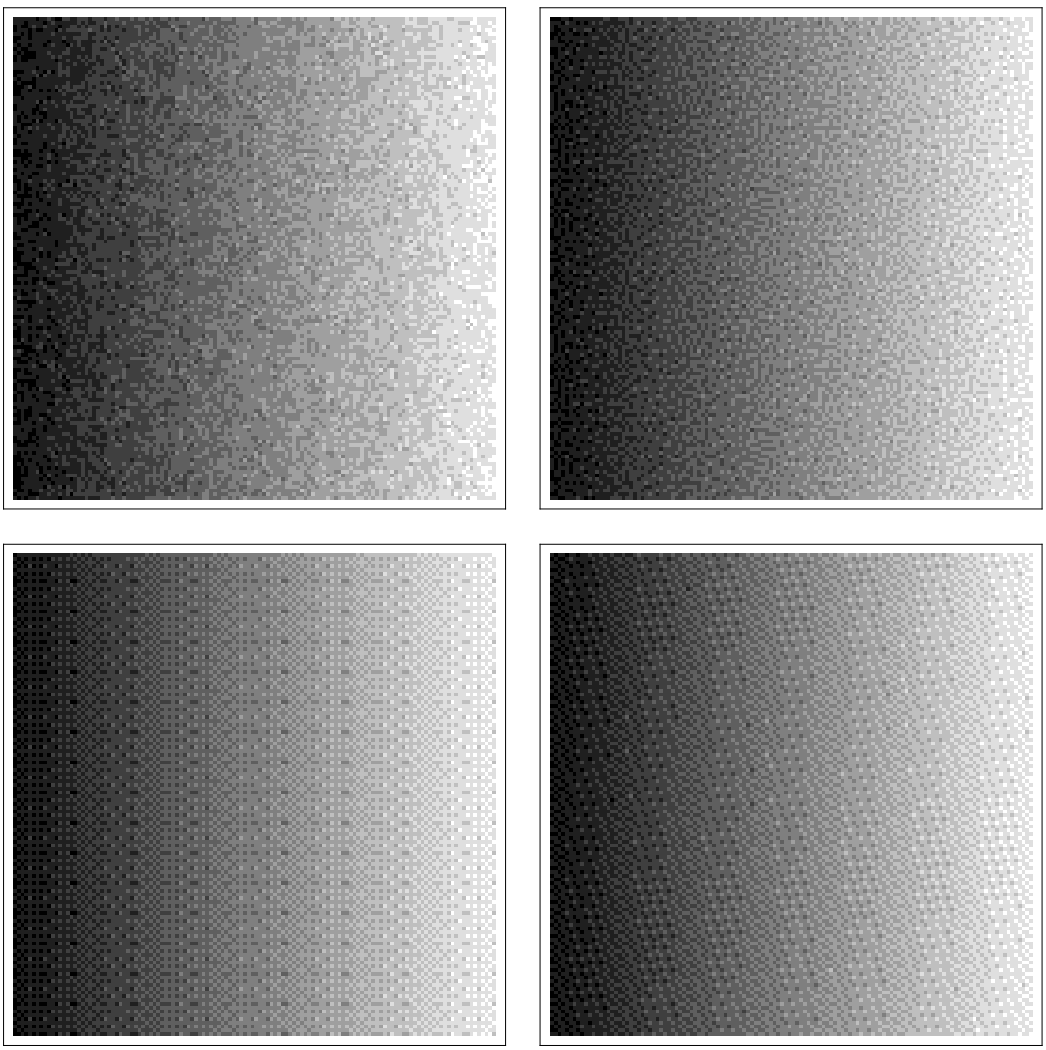
In[51]:=
```
ditheredInterleavedRemap = MapIndexed[
   QuantizeBits[#1, 3, remaptri[InterleavedGradientNoise[#2[[1]], #2[[2]]]]] &,
   origSignal, {2}];
```

In[52]:=
```
ditheredBayerRemap = MapIndexed[
   QuantizeBits[#1, 3, remaptri[SampleArrayWithWrap[bayer3, #2[[1]], #2[[2]]]]] &,
   origSignal, {2}];
ditheredWhiteRemap = Map[QuantizeBits[#1, 3, remaptri[RandomReal[]]] &,
   origSignal, {2}];
```

In[54]:=
```
ditheredBlueRemap = MapIndexed[QuantizeBits[#1, 3, remaptri[
      SampleArrayWithWrap[blueNoise64, #2[[1]], #2[[2]]]]] &, origSignal, {2}];
```

In[55]:=
```
GraphicsGrid[{{ArrayPlot[ditheredWhiteRemap, ColorFunction → GrayLevel],
   ArrayPlot[ditheredBlueRemap, ColorFunction → GrayLevel]},
  {ArrayPlot[ditheredBayerRemap, ColorFunction → GrayLevel],
   ArrayPlot[ditheredInterleavedRemap, ColorFunction → GrayLevel]}}]
```

Out[55]=



In[56]:=
```
Export["2dditheringallfourcomparedtriangular.gif",
  {ArrayPlot[ditheredWhiteRemap, ColorFunction → GrayLevel],
   ArrayPlot[ditheredBlueRemap, ColorFunction → GrayLevel],
   ArrayPlot[ditheredBayerRemap, ColorFunction → GrayLevel], ArrayPlot[
    ditheredInterleavedRemap, ColorFunction → GrayLevel]}, "DisplayDurations" → 2];
```