Scalar Field Visualization I

Goal: know what is a scalar field; what are the standard visualization techniques for 2D scalar fields, including color plots and iso-contouring.

What is a Scalar Field?

- An **approximation of certain scalar function in space *f(x,y,z).***

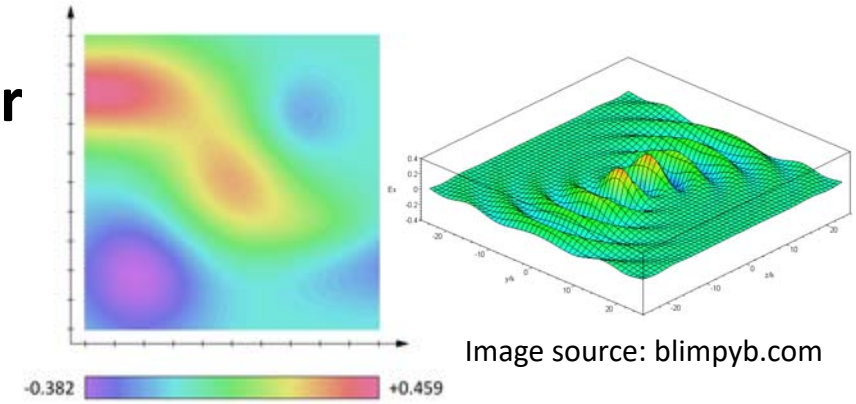
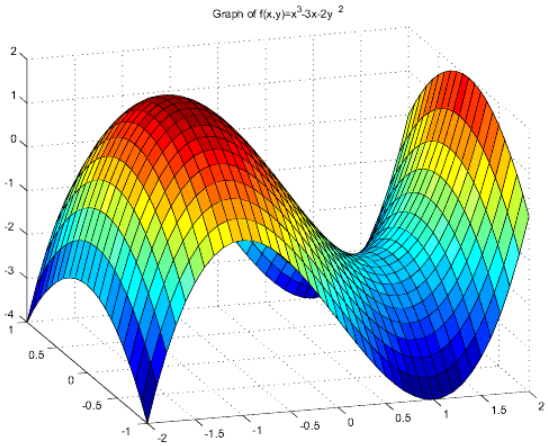
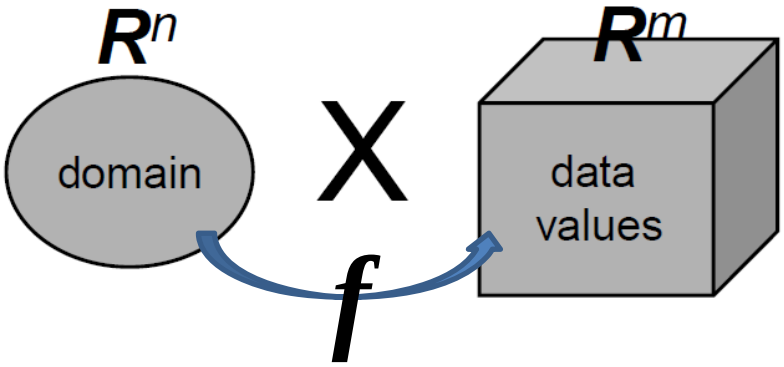
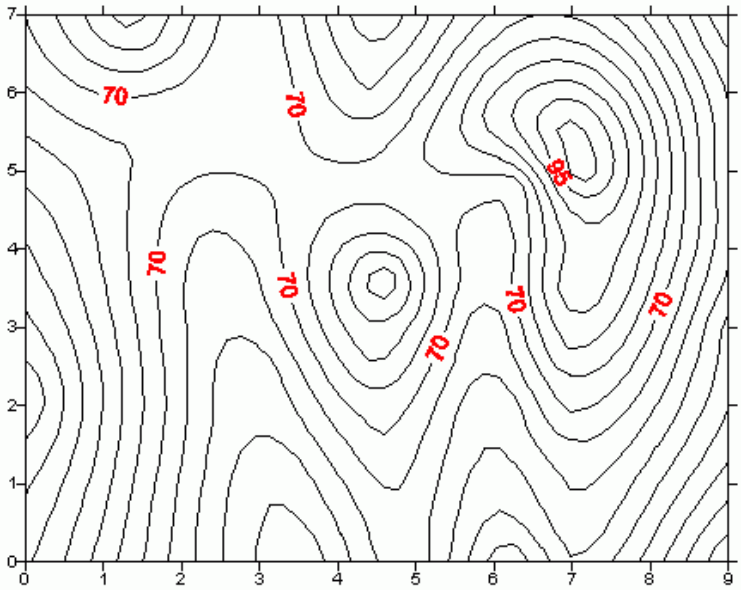


Image source: blimpyb.com







What is a Scalar Field?

- An approximation of certain scalar function in space *f(x,y,z).*

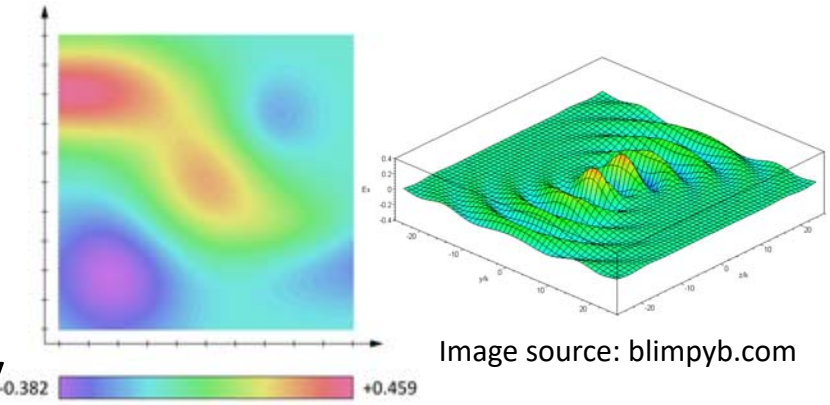


Image source: blimpyb.com

- **Representation**: Most of time, they come in as some scalar values defined on some sample points.

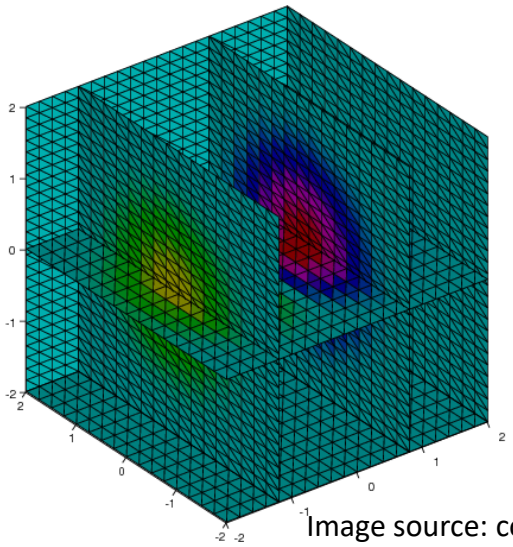
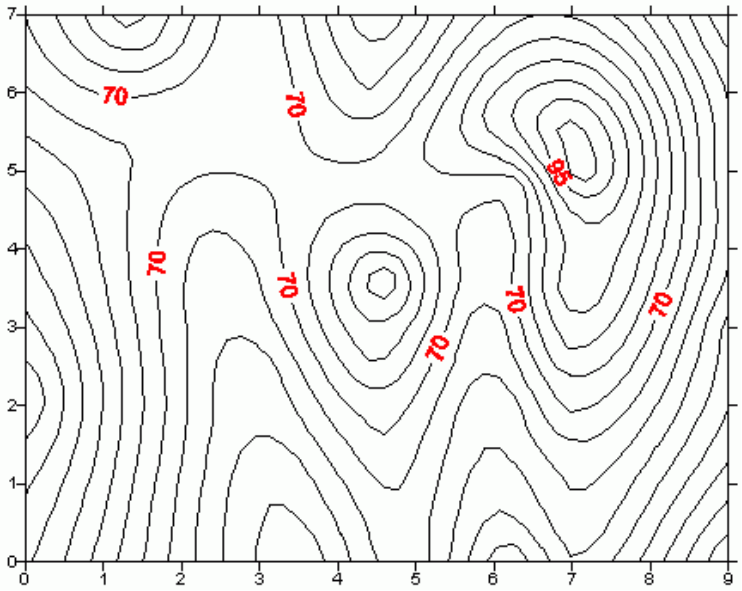


Image source: code google.com



What is a Scalar Field?

- An approximation of certain scalar function in space *f(x,y,z).*

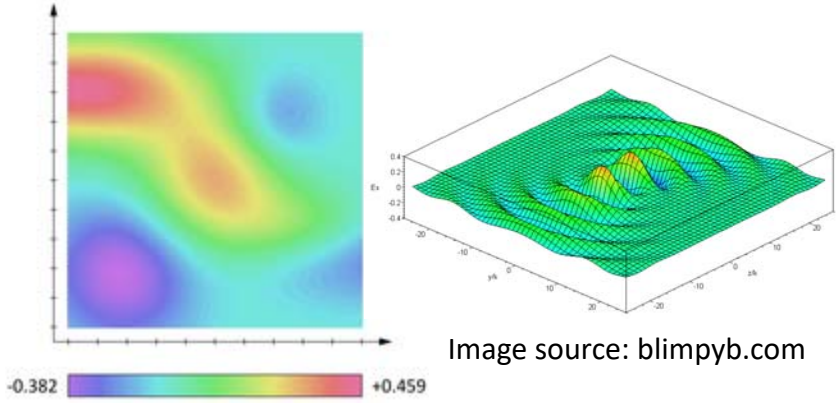
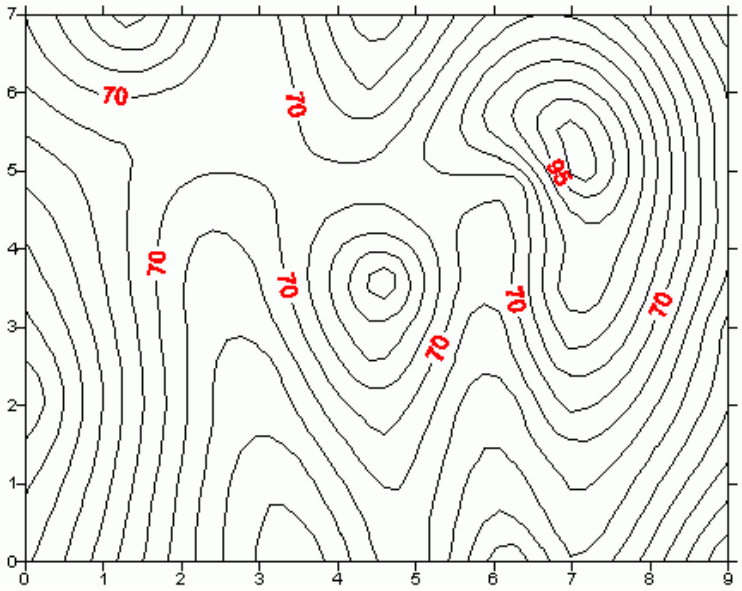


Image source: blimpyb.com

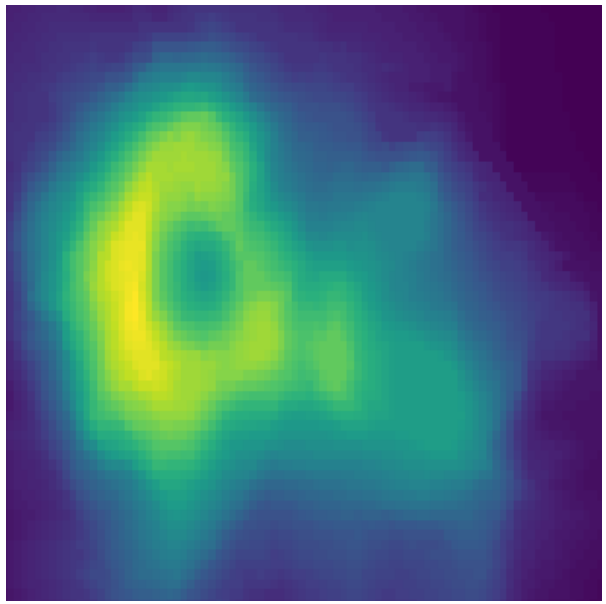
- Most of time, they come in as some scalar values defined on some sample points.

- **Visualization primitives:**
  - Geometry:
    - iso-contours (2D), iso-surfaces (3D),
  - Optical attributes:
    - colors, transparency (3D), 3D textures.



Generate 2D color plots

29, 31, 37, 41, 23, 43, 47, 19, 53, 17, 59, 61, 67, 71, 73, 13, 79, 83, 89, 11, 97, 101, 103, 107, 1
49, 179, 181, 121, 191, 193, 197, 199, 5, 211, 25, 169, 223, 227, 229, 233, 239, 241, 990, 251,
281, 283, 960, 209, 966, 85, 95, 65, 221, 930, 293, 115, 924, 55, 203, 145, 307, 247, 155, 311,
936, 3, 205, 9, 882, 337, 259, 978, 215, 27, 858, 954, 972, 918, 299, 347, 948, 912, 349, 910,
888, 301, 265, 367, 894, 876, 373, 361, 175, 341, 379, 864, 846, 852, 69, 87, 93, 295, 816, 828,
141, 756, 950, 822, 39, 159, 804, 880, 275, 397, 720, 770, 177, 33, 714, 1000, 183, 401, 920,
219, 371, 726, 768, 391, 762, 952, 171, 850, 99, 243, 744, 237, 325, 355, 738, 660, 207, 419,
630, 696, 820, 21, 261, 431, 830, 938, 684, 291, 433, 279, 303, 63, 700, 760, 868, 309, 988,
740, 339, 437, 654, 826, 147, 427, 449, 884, 333, 946, 415, 648, 624, 642, 728, 730, 636, 836,
369, 618, 650, 381, 451, 461, 742, 425, 588, 945, 463, 594, 748, 606, 670, 814, 351, 393, 874,
754, 782, 417, 620, 686, 616, 644, 658, 832, 576, 135, 423, 610, 469, 473, 546, 564, 682, 479,
646, 453, 481, 992, 928, 550, 676, 273, 475, 487, 574, 357, 455, 572, 598, 722, 231, 459, 528,
944, 976, 493, 530, 532, 848, 975, 405, 520, 608, 510, 516, 578, 752, 489, 656, 688, 904, 345,
592, 632, 664, 712, 844, 892, 908, 916, 932, 956, 964, 255, 435, 499, 506, 536, 568, 584, 585,
724, 734, 746, 758, 764, 766, 772, 778, 788, 794, 796, 802, 818, 838, 842, 862, 866, 878, 886,
128, 256, 422, 446, 452, 454, 458, 465, 466, 472, 478, 482, 488, 494, 495, 496, 500, 501, 502,
586, 614, 622, 626, 634, 662, 298, 302, 314, 326, 334, 346, 358, 362, 382, 386, 388, 394, 398,
262, 274, 278, 316, 332, 356, 503, 555, 194, 202, 206, 214, 218, 268, 284, 292, 376, 490, 492,
142, 146, 195, 212, 486, 615, 118, 122, 296, 368, 416, 519, 106, 188, 442, 448, 470, 511, 561,
82, 86, 148, 165, 304, 460, 480, 418, 434, 609, 74, 124, 338, 468, 537, 675, 116, 272, 352, 517,
374, 819, 92, 152, 462, 627, 885, 46, 400, 410, 444, 521, 136, 535, 539, 915, 76, 208, 438, 523,
545, 595, 364, 176, 370, 380, 104, 591, 777, 322, 414, 597, 603, 621, 52, 105, 286, 402, 408,
759, 861, 308, 551, 22, 196, 266, 310, 633, 420, 639, 366, 372, 547, 350, 559, 605, 903, 290,
987, 342, 324, 681, 200, 687, 260, 563, 56, 318, 230, 583, 699, 182, 360, 711, 28, 160, 569, 14,
625, 154, 220, 282, 577, 729, 957, 747, 635, 637, 100, 170, 783, 276, 753, 891, 80, 258, 623,
300, 789, 216, 805, 20, 130, 801, 10, 222, 599, 837, 228, 601, 807, 234, 813, 192, 110, 252,
174, 617, 649, 619, 144, 198, 873, 240, 725, 679, 879, 631, 909, 108, 138, 156, 671, 70, 96,
132, 647, 102, 951, 963, 999, 54, 180, 755, 48, 775, 653, 689, 721, 981, 36, 210, 659, 24, 126,
749, 677, 84, 120, 713, 683, 763, 737, 815, 691, 90, 731, 847, 42, 835, 701, 791, 833, 60, 709,
905, 743, 751, 799, 757, 761, 769, 931, 817, 955, 889, 773, 965, 787, 985, 917, 869, 995, 797,
839, 901, 899, 923, 853, 857, 859, 863, 949, 877, 881, 979, 883, 943, 887, 907, 911, 919, 989,



Goal: know how to *design proper transfer functions*, how to produce color plots

Do we need to construct/extract additional geometry?

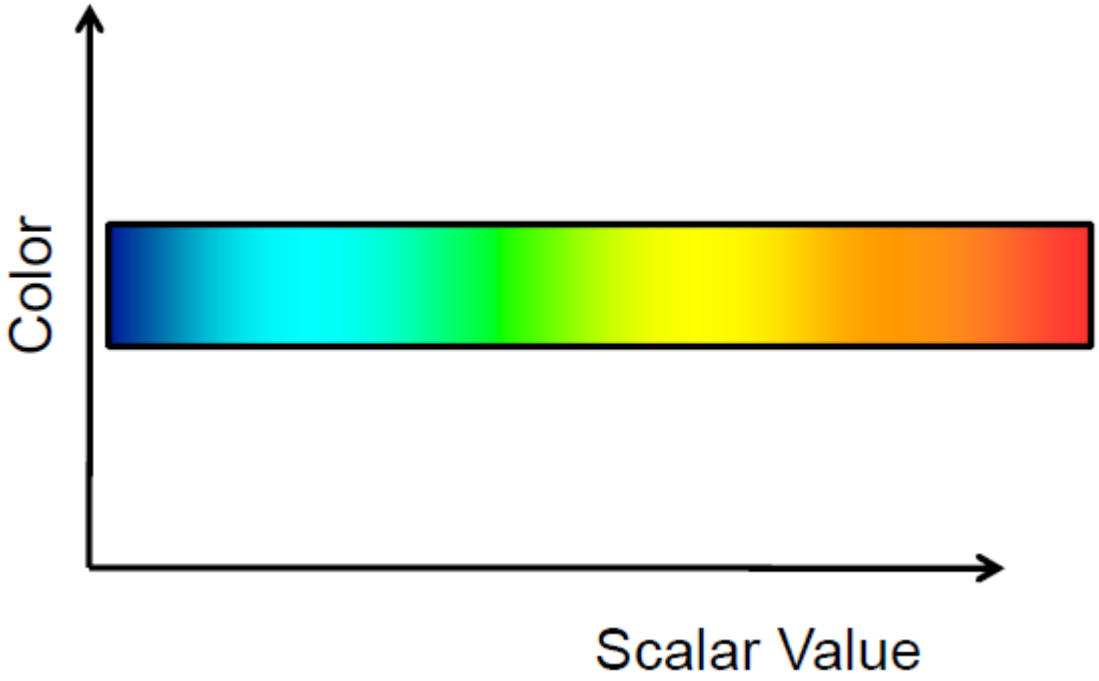
NO!

How to generate color plots?

Steps of generating 2D color plots

1. **Design color transfer function**
2. Color interpolation

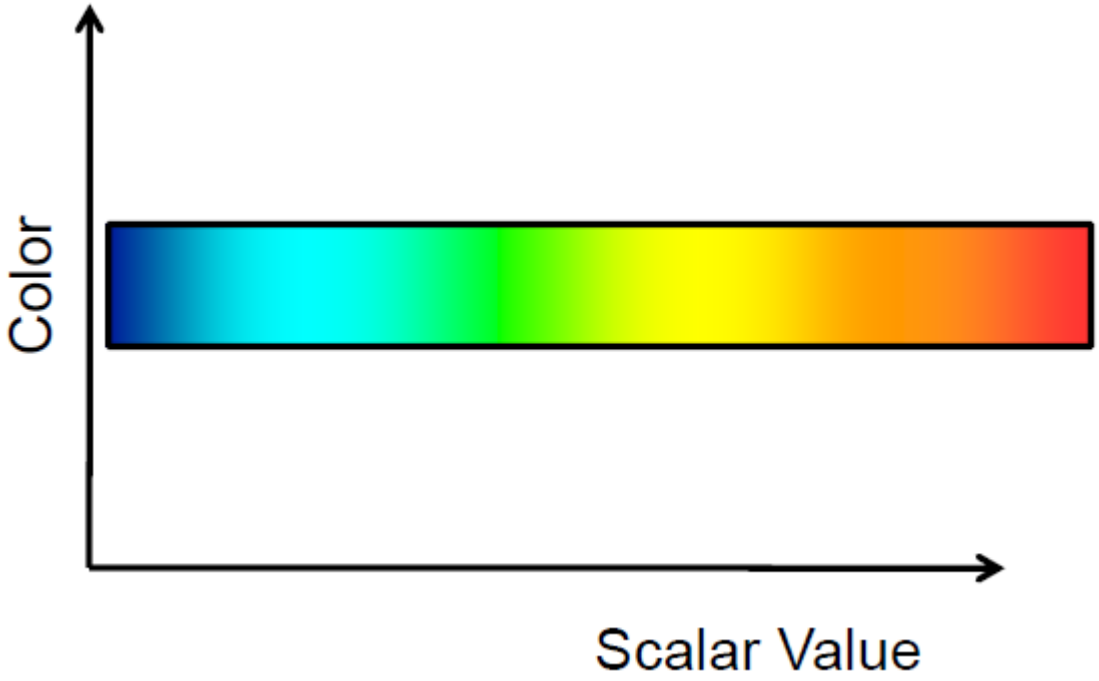
To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.



Scalar values [smin, smax] ->[0,1] -> Colors

Transfer function

To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.

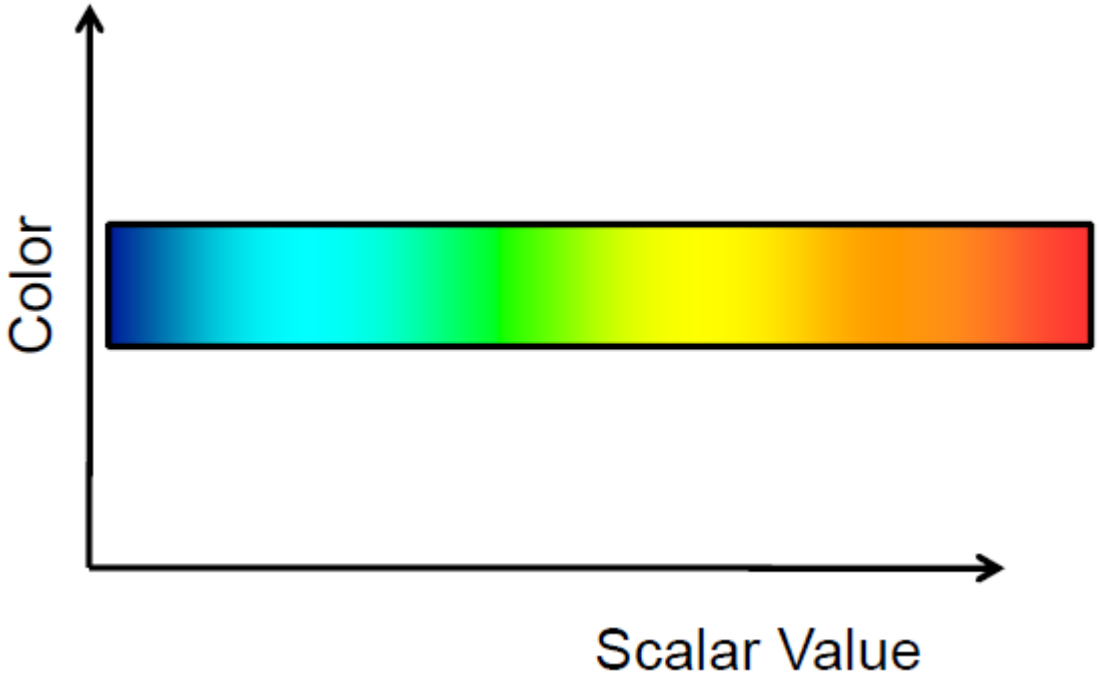


Scalar values ->[0,1]

$$t = \frac{S - S_{min}}{S_{max} - S_{min}}$$

normalization

To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.

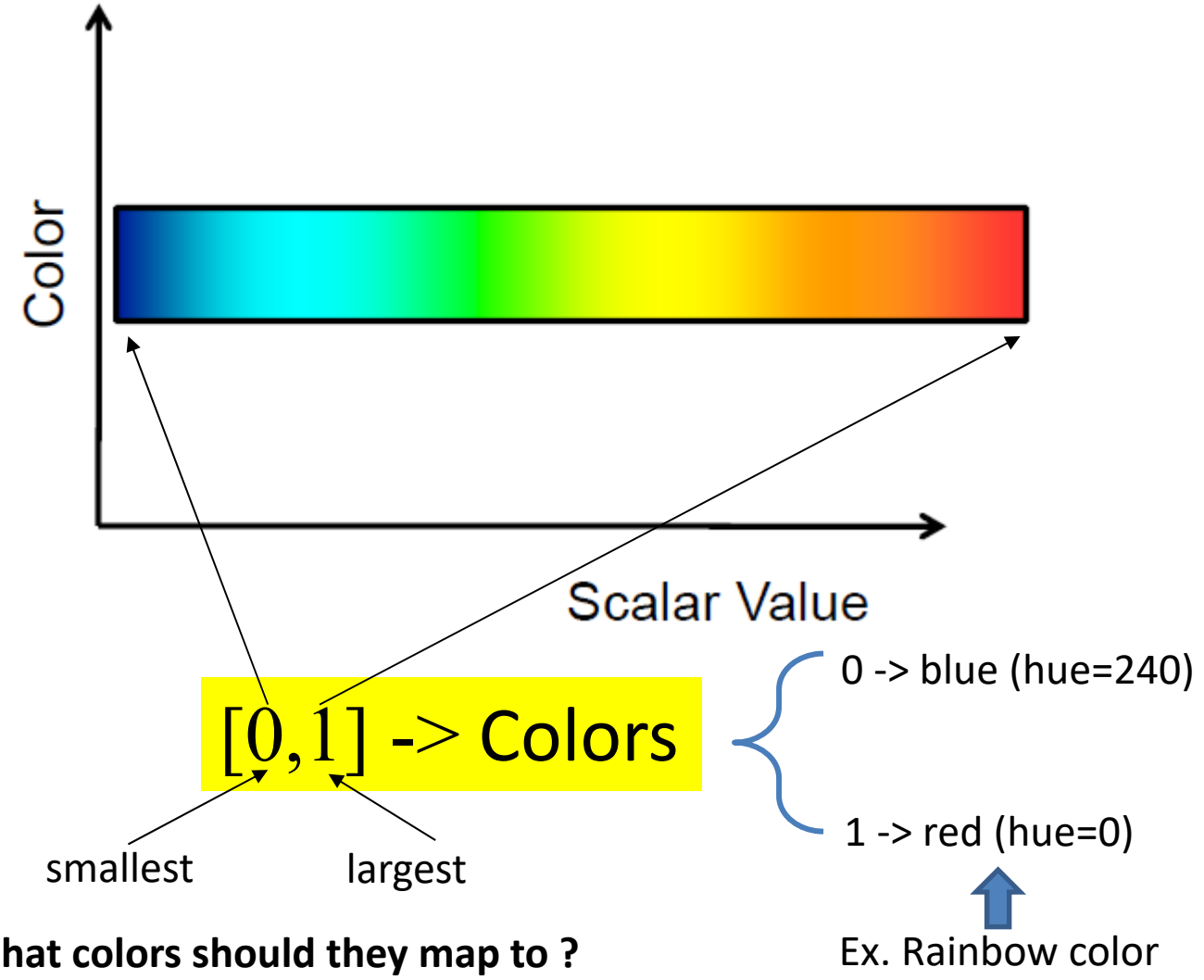


[0,1] -> Colors

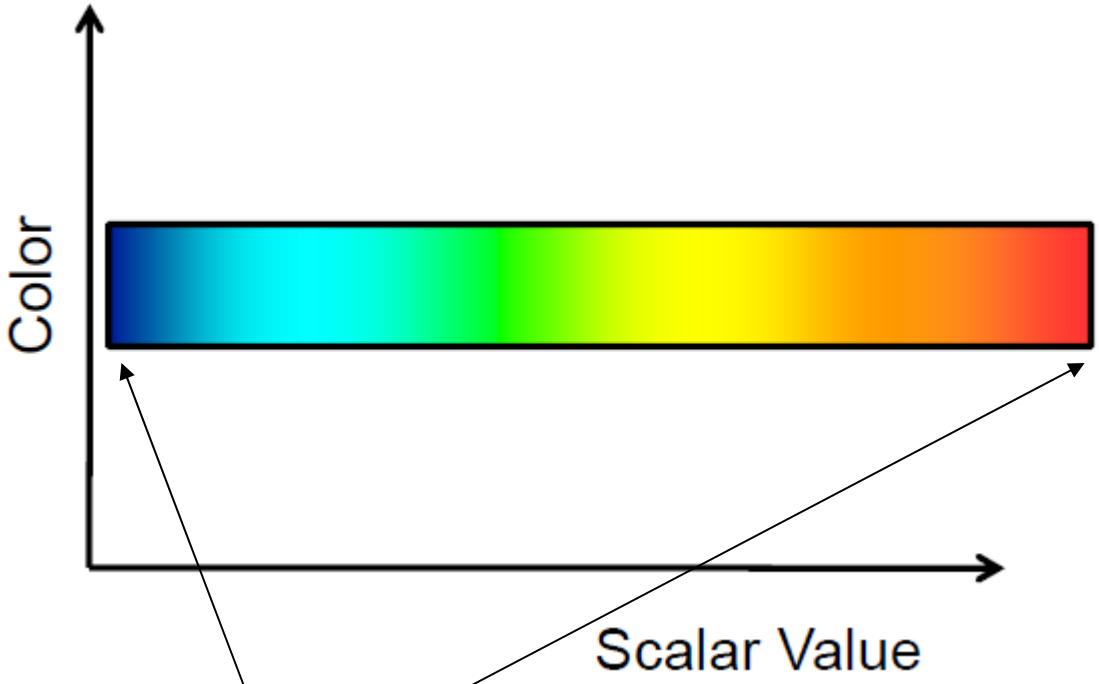
smallest largest

What colors should they map to ?

To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.



To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.



[0,1] -> Colors

0 -> blue (hue=**240, large**)

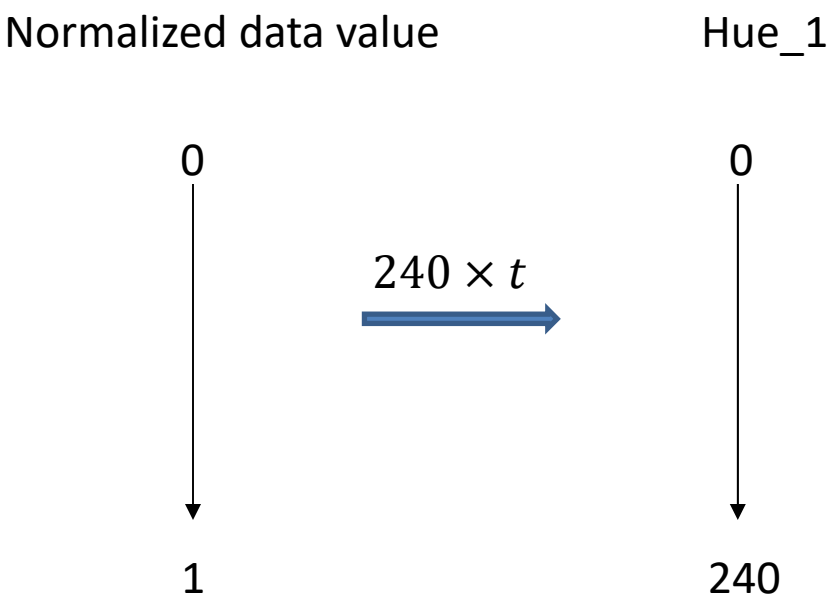
0<t<1 ???

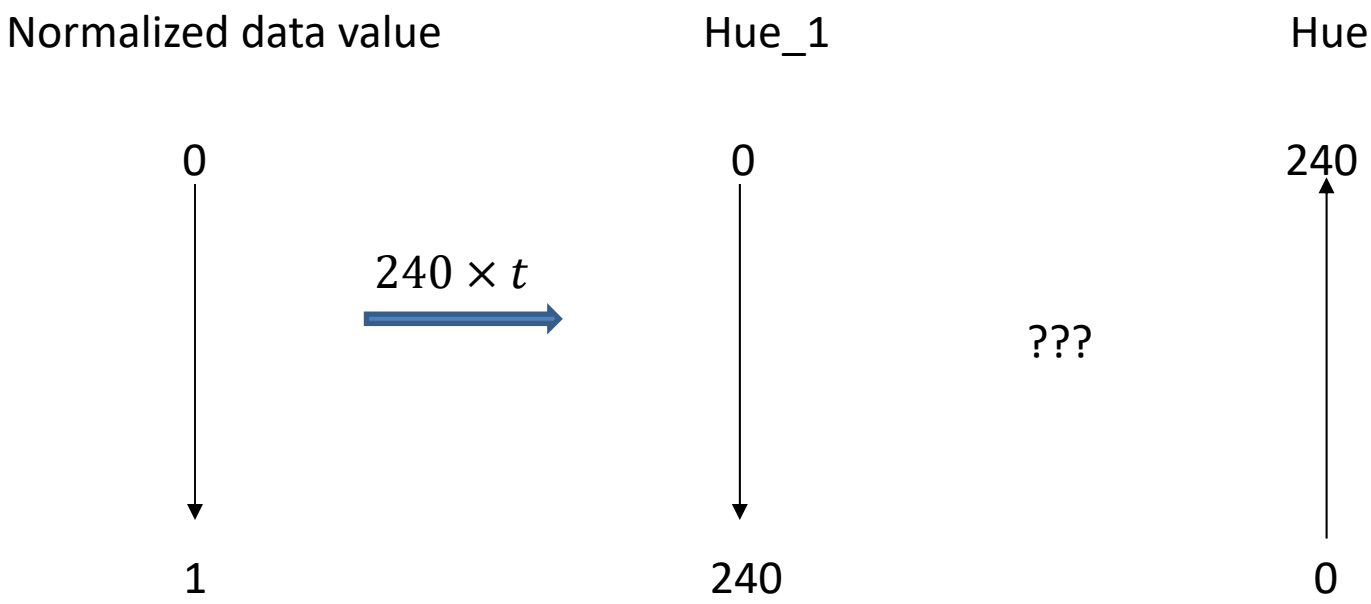
1 -> red (hue=**0, small**)

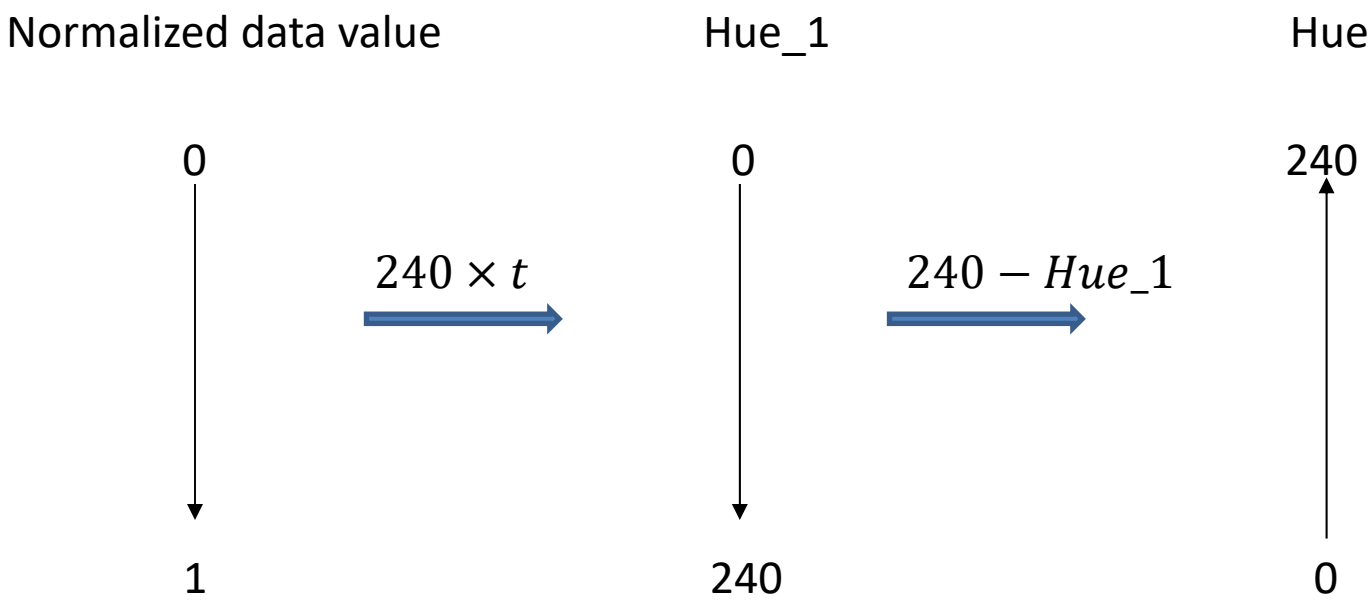
smallest largest

What colors should they map to ?

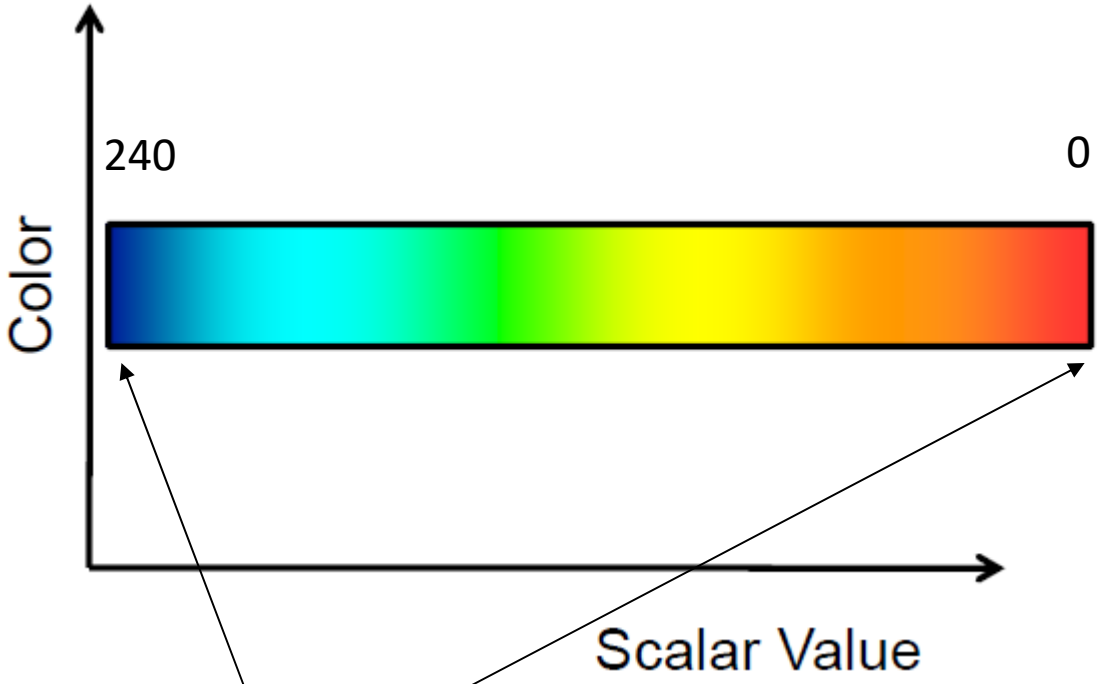
Ex. Rainbow color


Normalized data value
Hue_1
0
$240 \times t$
0
1
240


Normalized data value
Hue_1
Hue
0
1
$240 \times t$
0
240
???
240
0


Normalized data value
Hue_1
Hue
0
1
$240 \times t$
0
240
$240 - Hue\_1$
240
0

To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.



[0,1] -> Colors

0 -> blue (hue=240)

t -> $hue = 240. - 240.\,t$

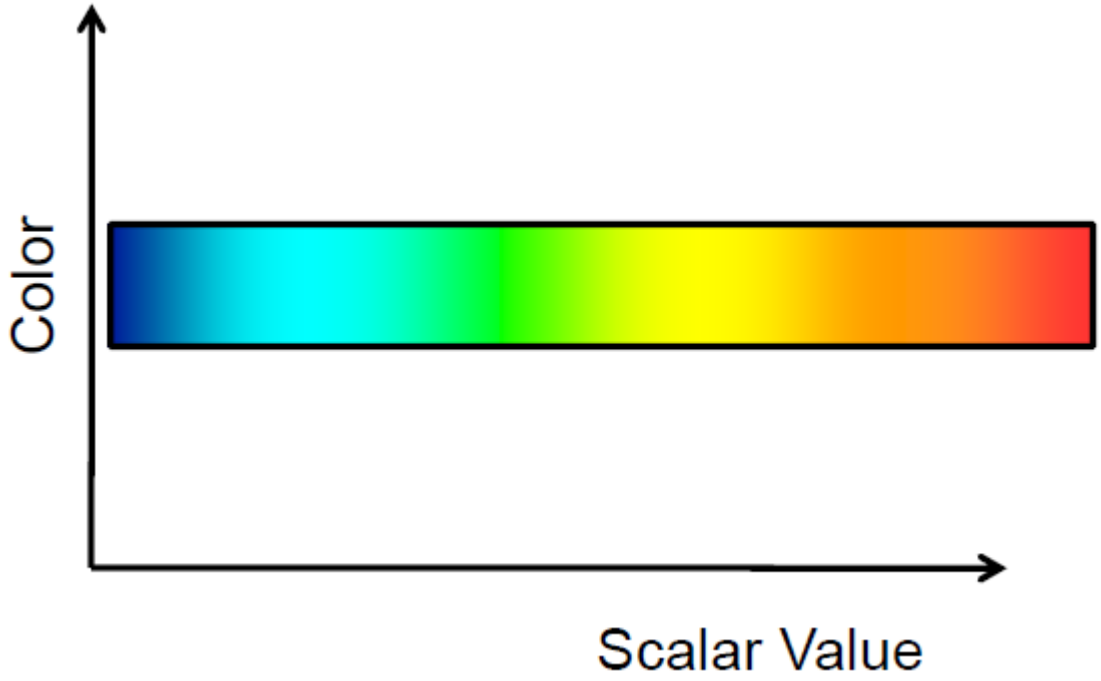
1 -> red (hue=0)

smallest largest

What colors should they map to ?

Ex. Rainbow color

To create a color plot, we need to define a proper ***Transfer Function*** to set Color as a function of Scalar Value.
The following shows a simple transfer function.



Scalar values ->[0,1] -> Colors

$$Hue = 240. - 240. \frac{S - S_{min}}{S_{max} - S_{min}}$$

Verify low and high!

How to achieve the following color scale?

Scalar values ->$[0,1]$

$$t = \frac{S - S_{min}}{S_{max} - S_{min}}$$

normalization

How to achieve the following color scale?

Scalar values ->$[0,1]$

0 -> white

1 -> full saturated red
(hue=0)

How to achieve the following color scale?

Scalar values ->$[0,1]$

0 -> white

Saturation=0

What is between 0 and 1?

1 -> full saturated red (hue=0)

Saturation=1

How to achieve the following color scale?

Scalar values ->$[0,1]$

0 -> white

1 -> full saturated red
(hue=0)

Saturation=0

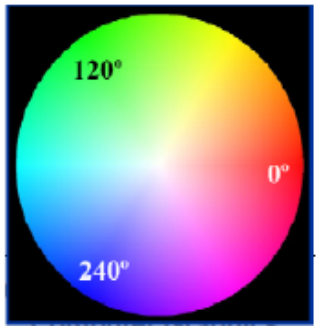
Saturation = t, where t is in [0, 1]

Saturation=1

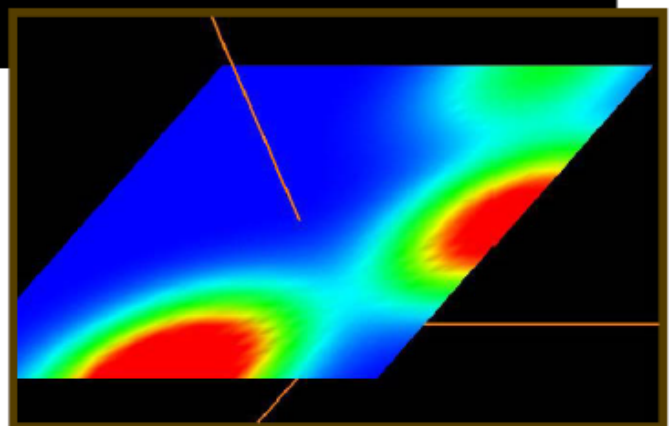
Hue = 0
Value = 1

Use the Proper **Transfer Function** Color Scale to Represent a Range of Scalar Values

Rainbow Scale



$$Hue = 240. - 240. \frac{S - S_{min}}{S_{max} - S_{min}}$$



Gray Scale

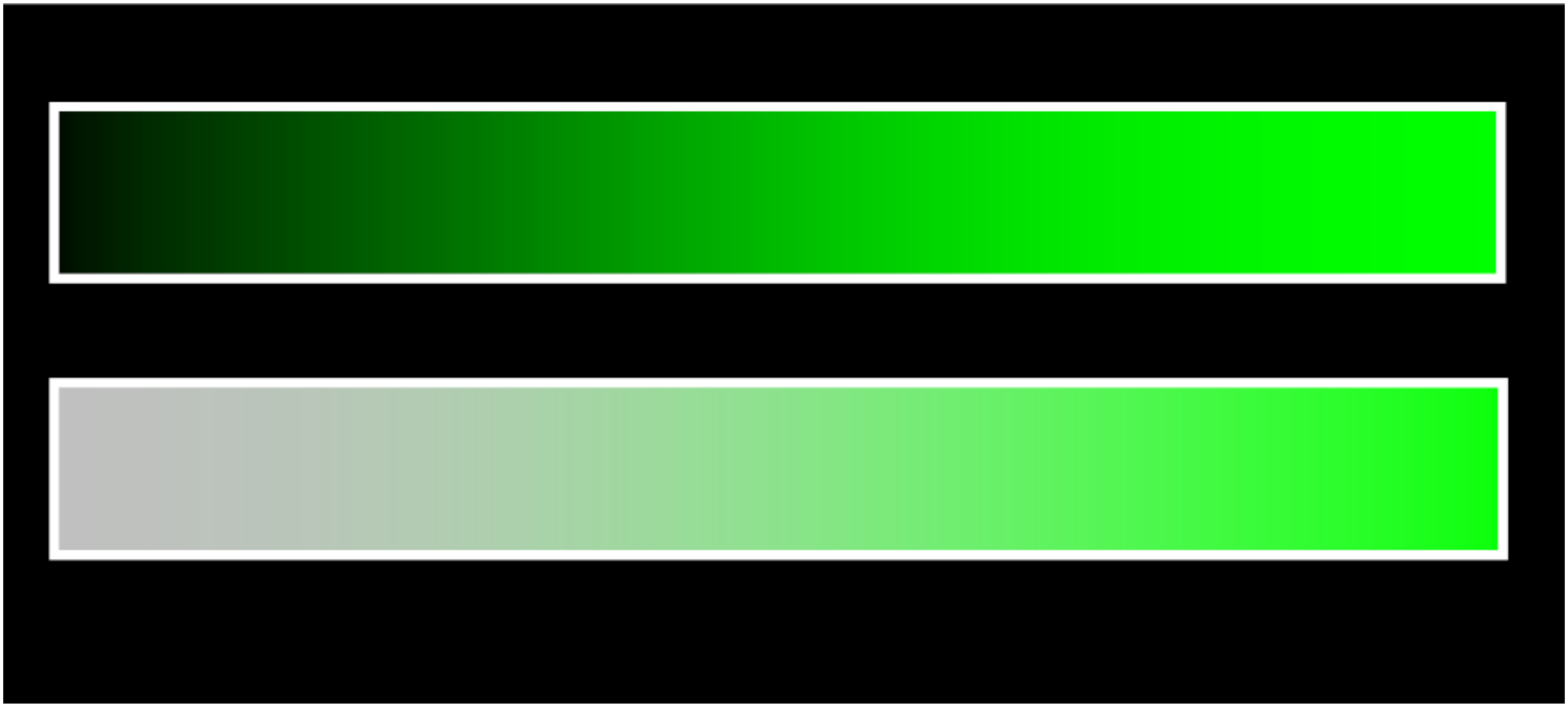
How to generate gray scale color scheme?

Gray Scale

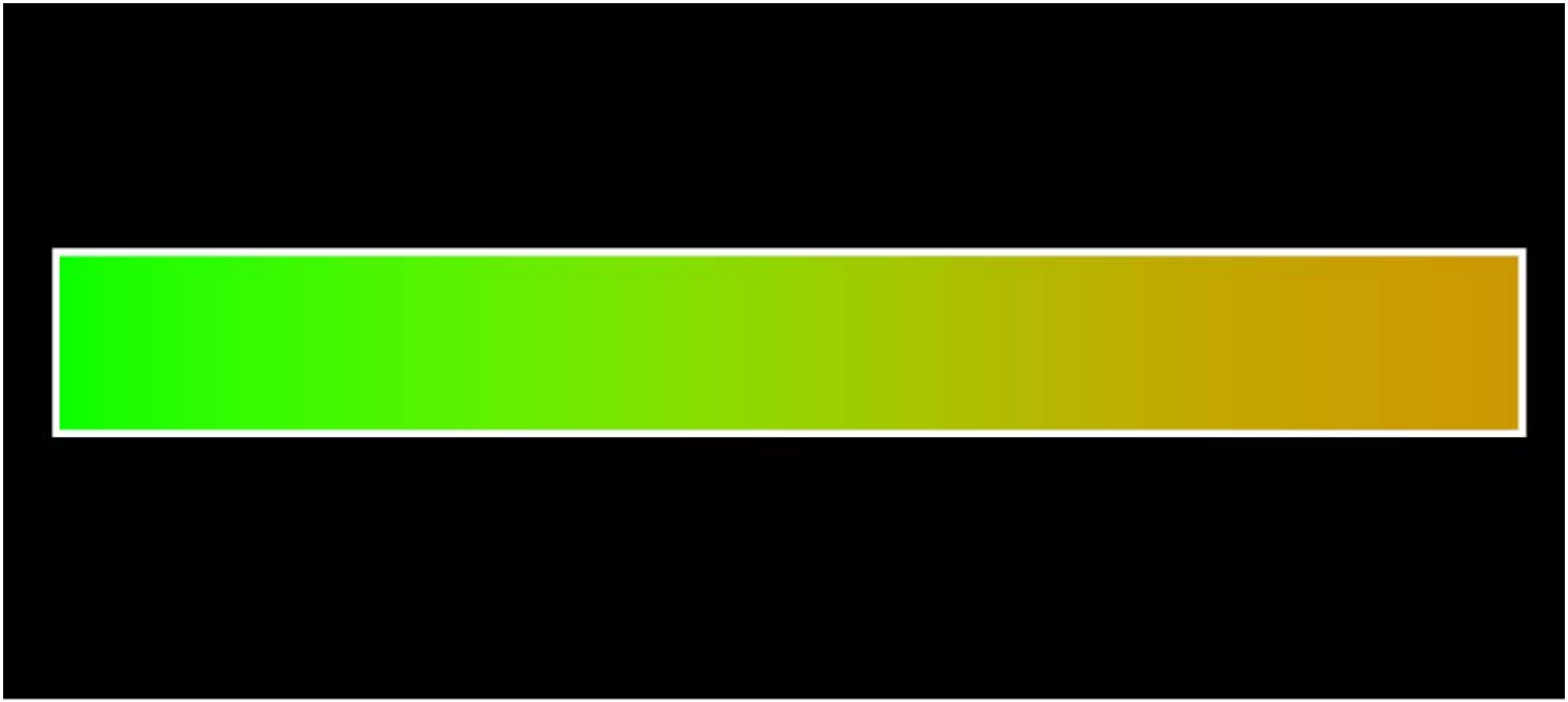
How to generate gray scale color scheme?

Set r=g=b= the same value (say, the normalized data value between 0 and 1)

Intensity and Saturation Color Scales

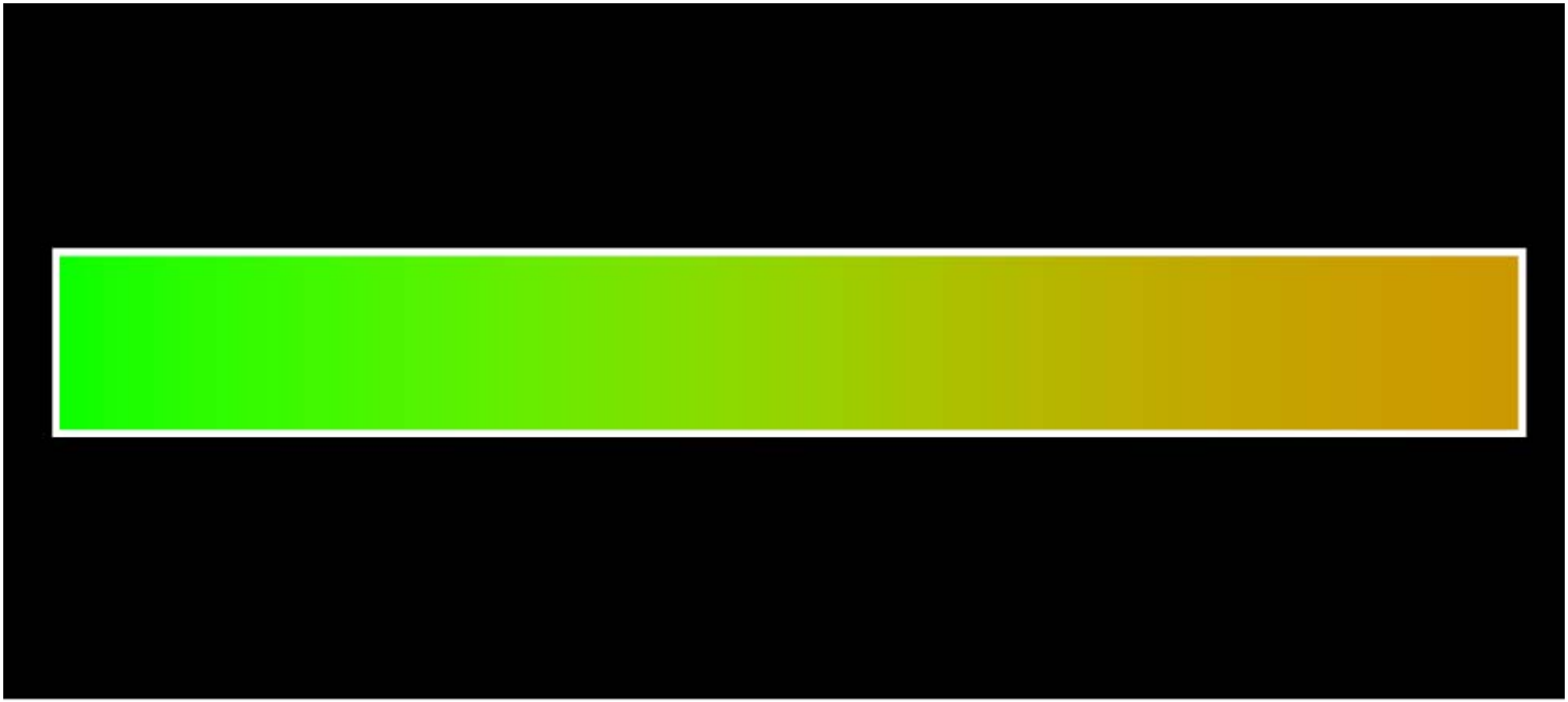


Two-Color Interpolation



How to achieve the above color scheme?

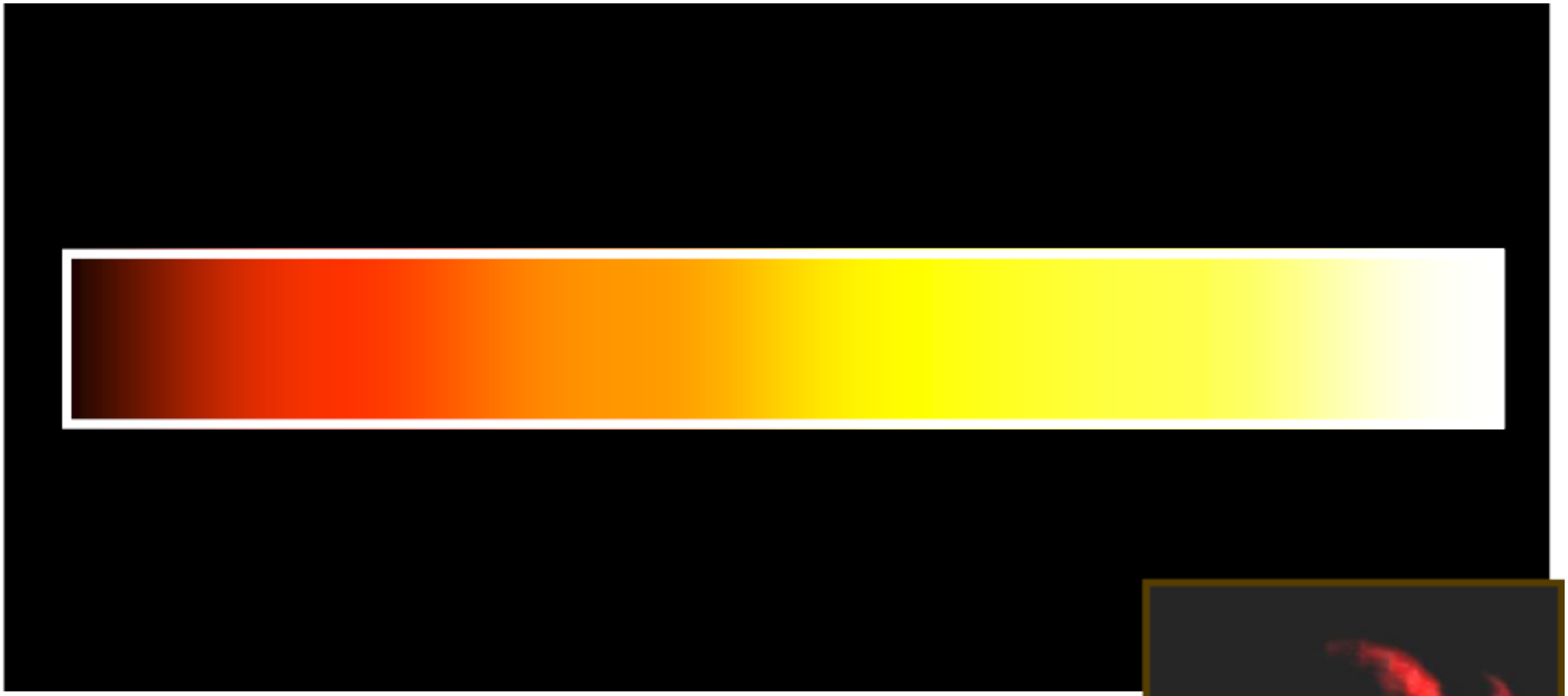
Two-Color Interpolation



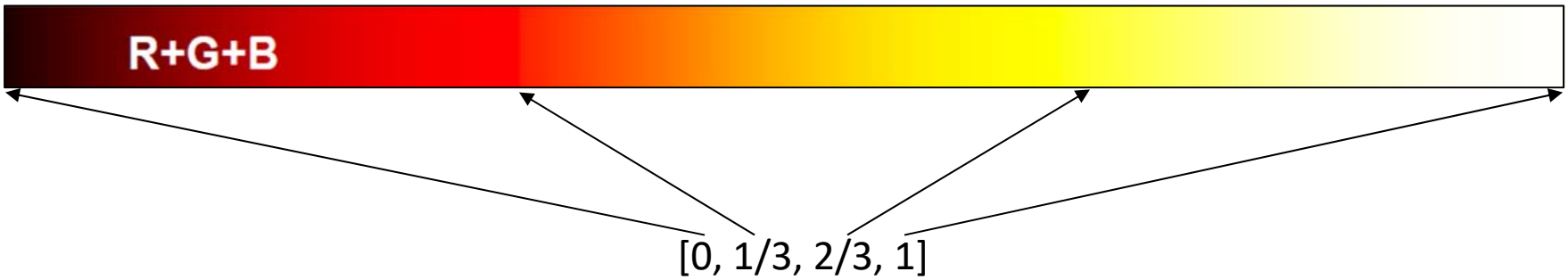
How to achieve the above color scheme?

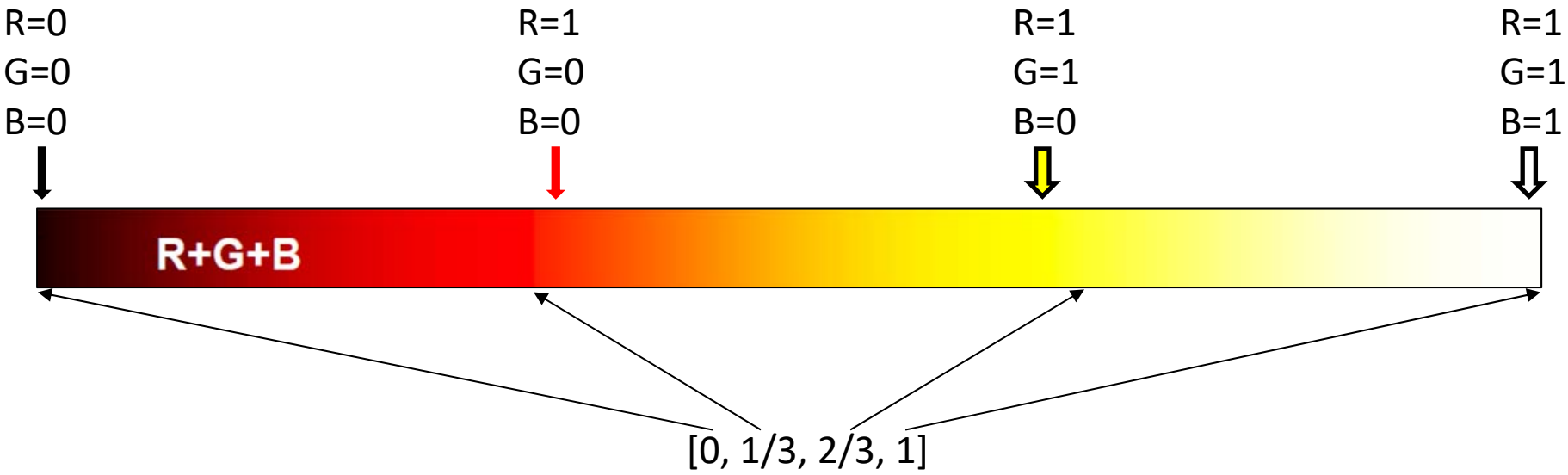
You may try a simple linear interpolation of the three color-channels of the two colors.

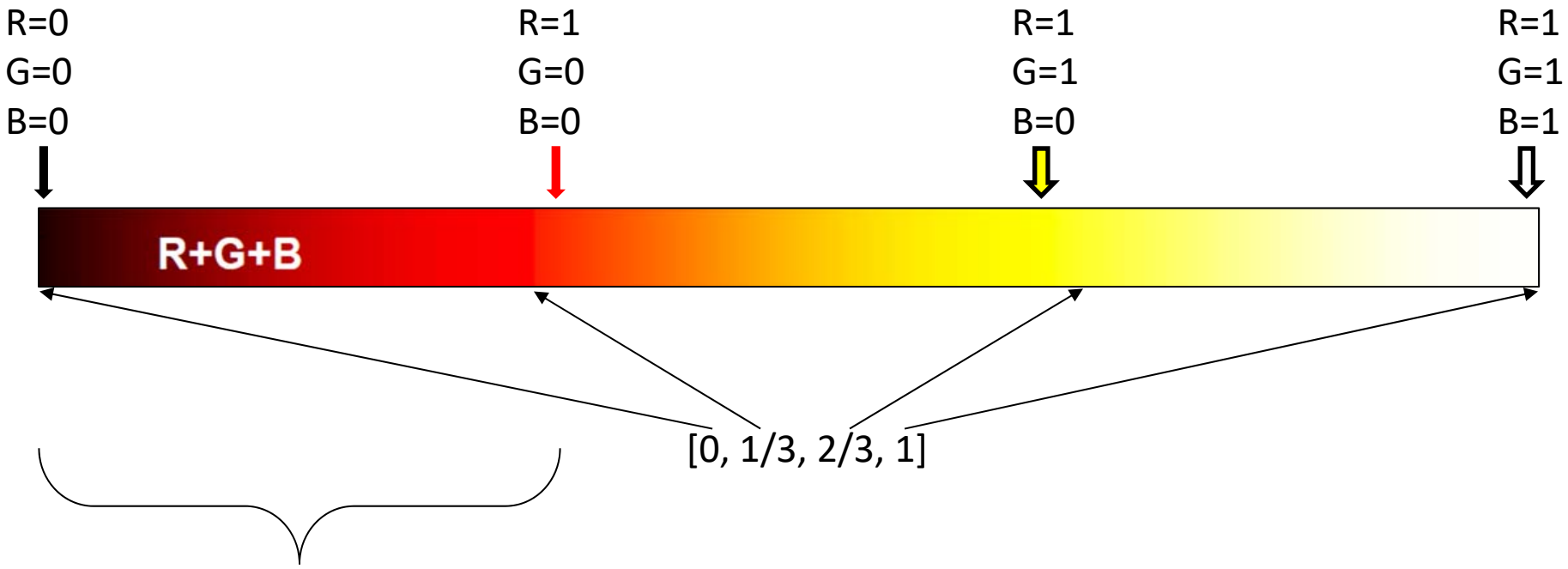
Heated Object Color Scale



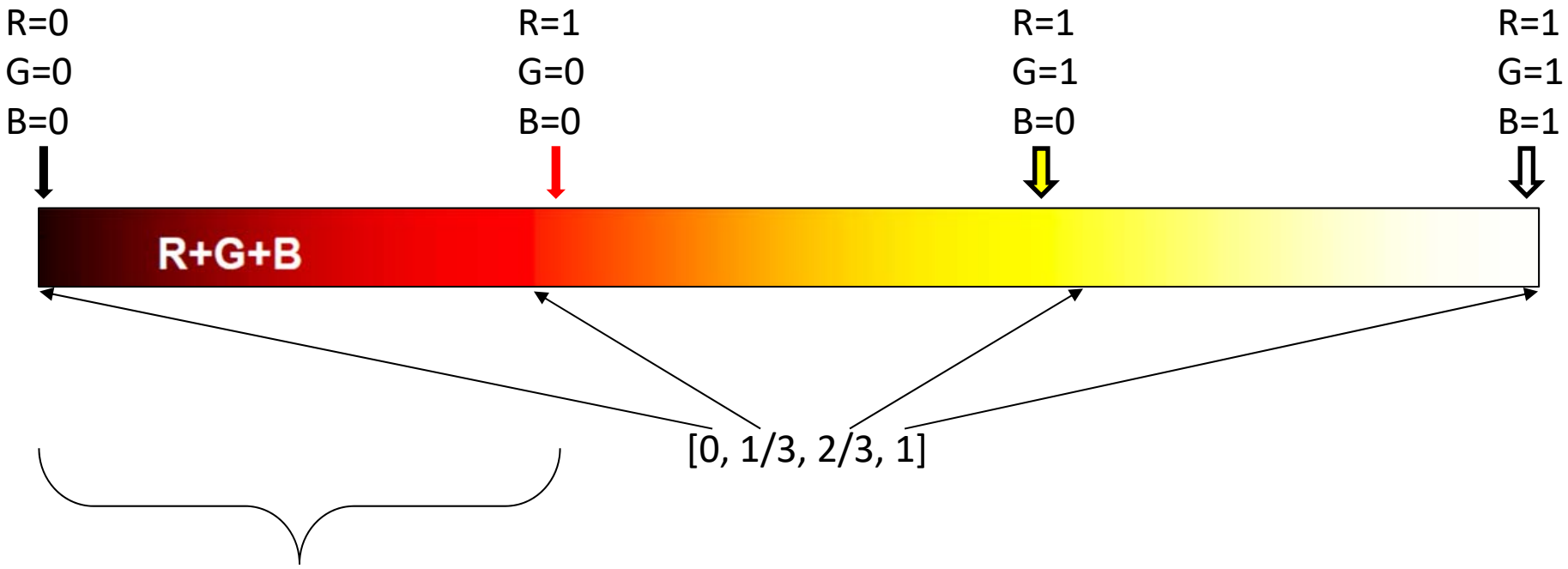
Implementation: add one color component at a time


R+G+B
[0, 1/3, 2/3, 1]


R=0
G=0
B=0
R=1
G=0
B=0
R=1
G=1
B=0
R=1
G=1
B=1
R+G+B
[0, 1/3, 2/3, 1]



For s in [0, 1/3], fixed G=B=0, map s to **R** from [0, 1/3] to [0, 1], that is, *R=3.0*s*



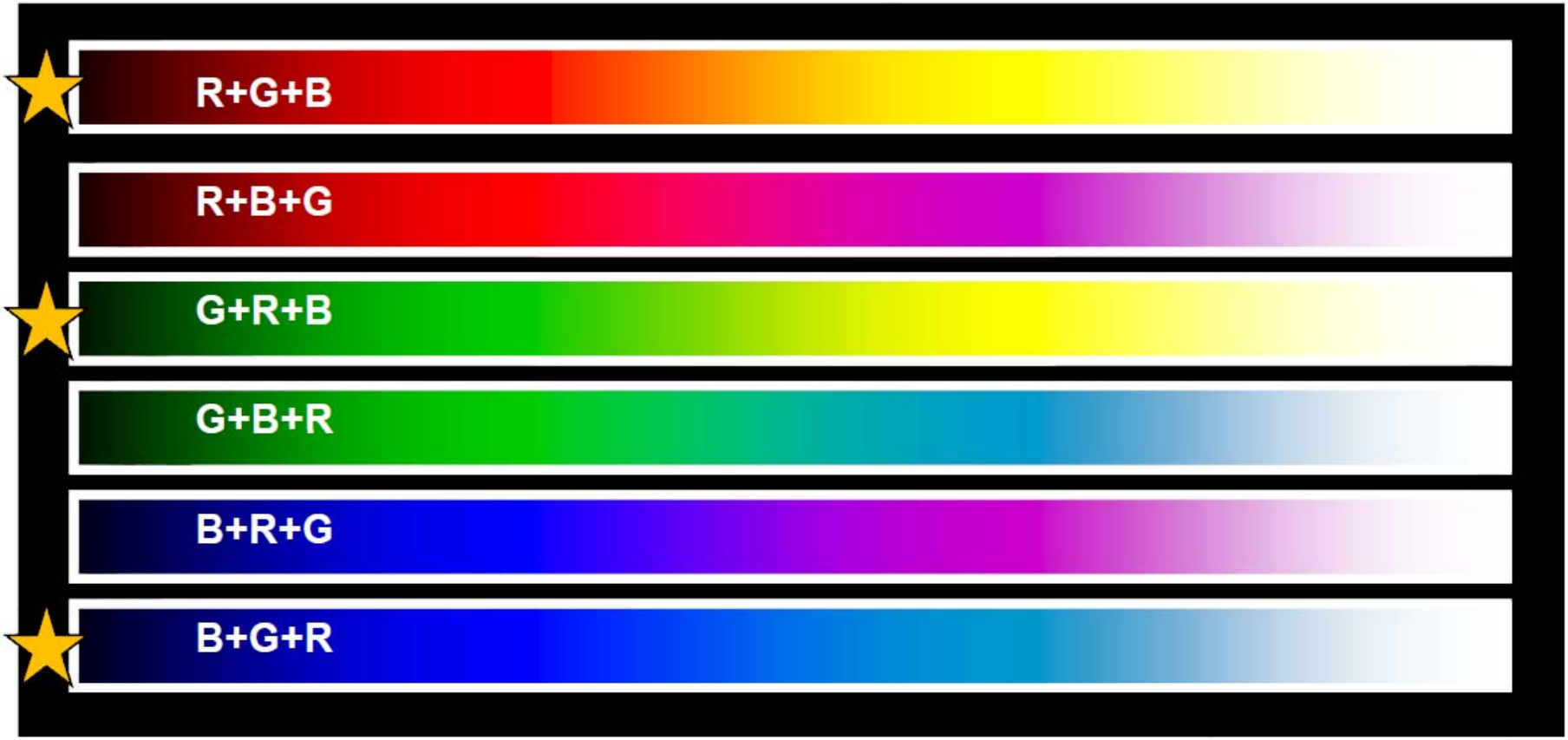
For s in [0, 1/3], fixed G=B=0, map s to **R** from [0, 1/3] to [0, 1], that is, *R=3.0*s*

For s in (1/3, 2/3], fixed R=1, B=0, map s to **G** from (1/3, 2/3] to [0, 1], that is, G=?

For s in (2/3, 1], fixed R=G=1, map s to **B** from (2/3, 1] to [0, 1], that is, B=?

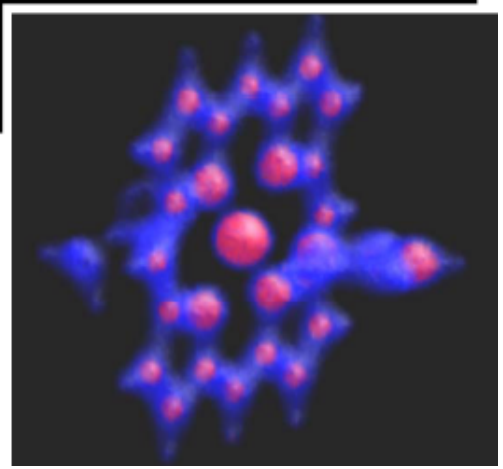
Figuring out how to determine G and B is part of your **assignment 2**!

Add-One-Component at a time -- an extension from the heated object color scale



Blue-White-Red Color Scale

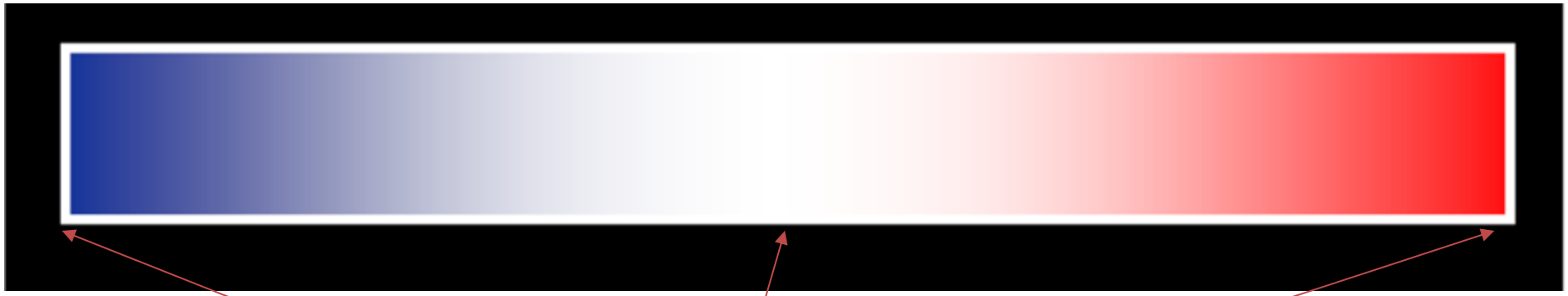
Diverging color scheme!



Blue-White-Red Color Scale

How to achieve it?

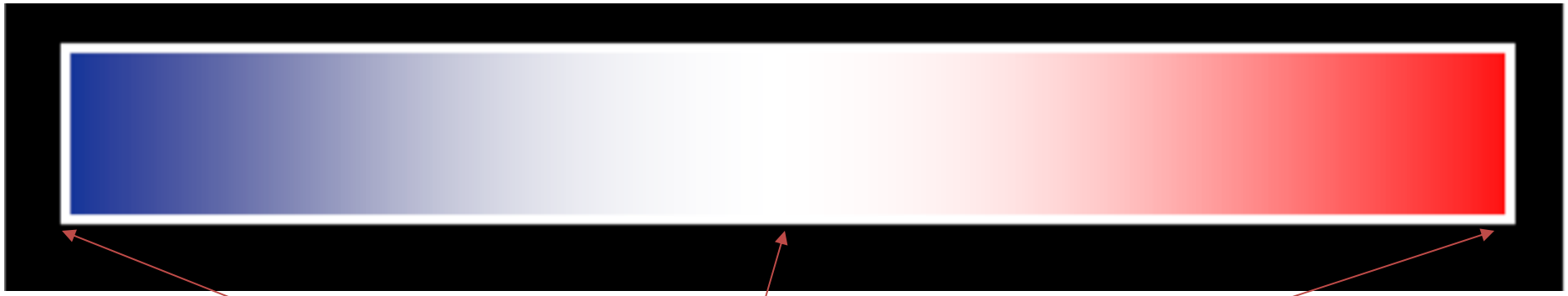
Blue-White-Red Color Scale



How to achieve it?

Scalar value -> [0, ½, 1]

Blue-White-Red Color Scale



How to achieve it?

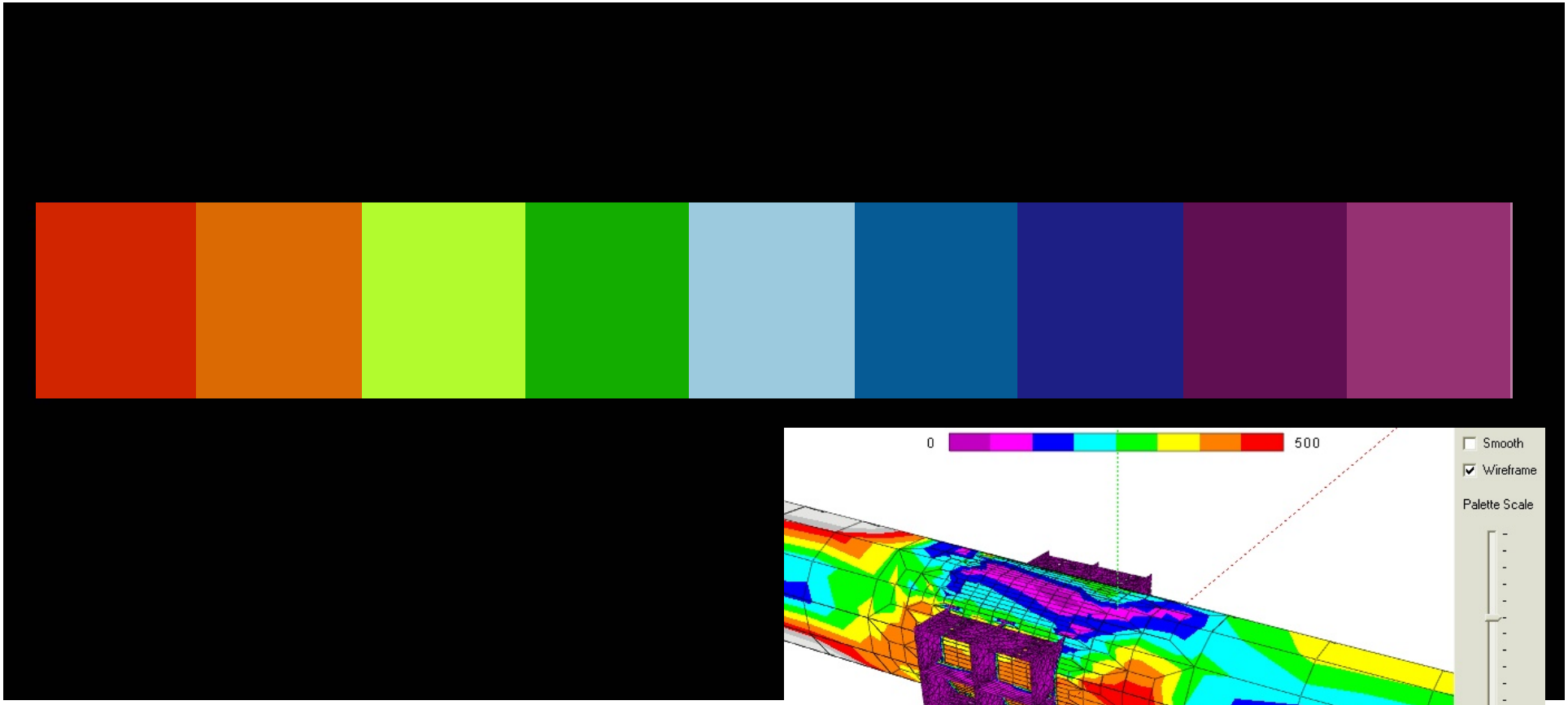
Scalar value -> [0, ½, 1]

For [0, ½], saturation **reduces** from 1 to 0, hue=240 (fixed)

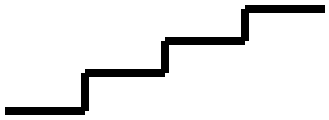
For (½, 1], saturation **increases** from 0 to 1, hue=0 (fixed)

Figuring out how to update the saturation is part of your **assignment 2**!

(Discrete) Color Scale Contour

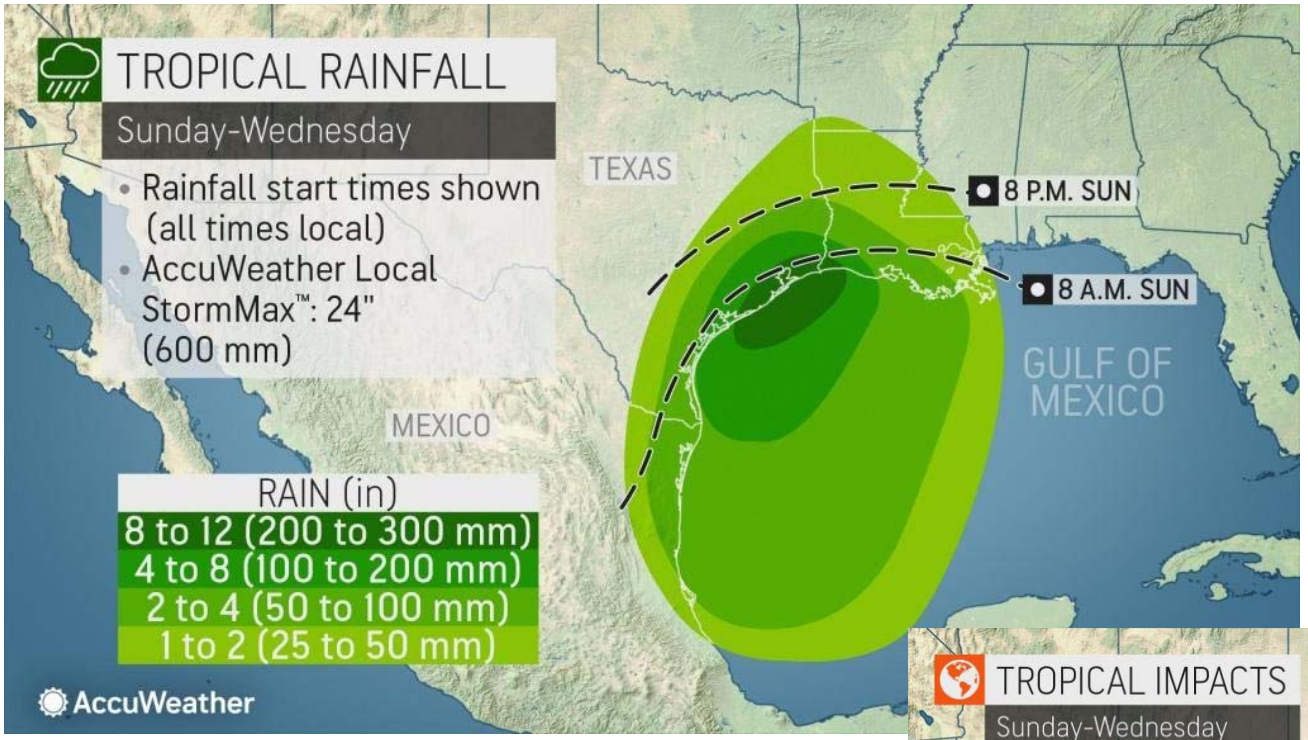


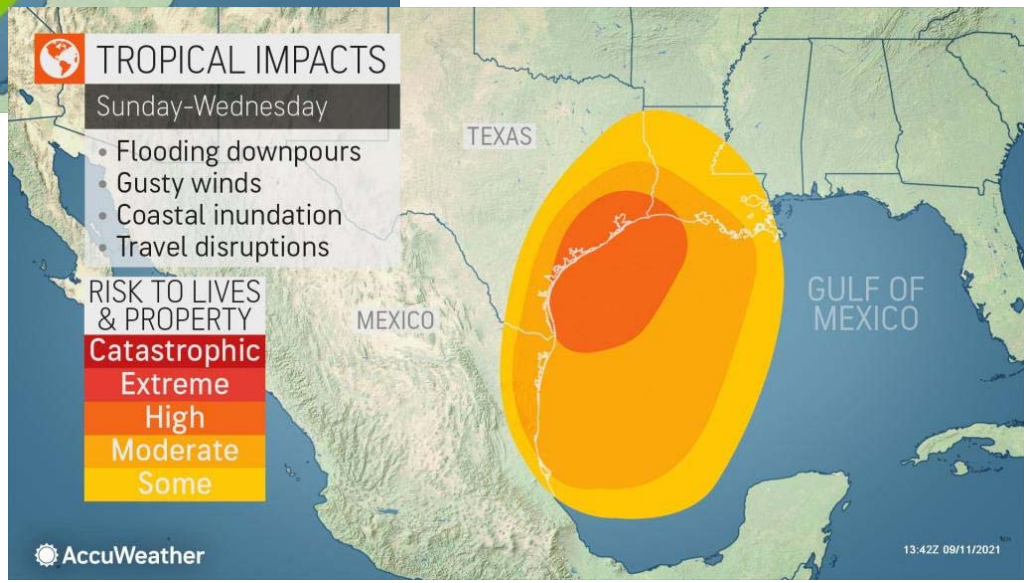
Qualitative/discrete color scheme



Source: http://glscene.sourceforge.net

Sequential color scales can also be discretized to help read the configuration of the scalar fields!





from AccuWeather.com

Use VTK color look-up table (discrete!)

Create and initialize a (color) lookup table

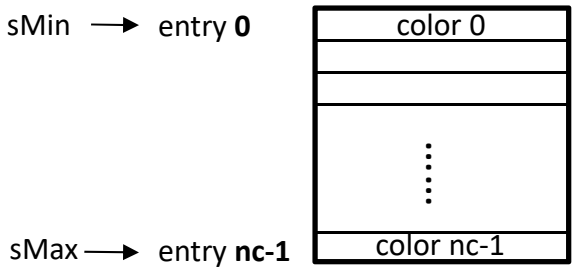
```
lut = vtk.vtkLookupTable() # Initialize the vtk lookup table
nc = 256 #the size of the table - increase this number to obtain
more precise color map
lut.SetNumberOfTableValues(nc)
lut.Build()
```

Generate the individual entries of the lookup table (**rainbow** color is shown below)

```
sMin = 0.
sMax = 1.

hsv = [0.0,1.0,1.0]

for i in range(0, nc):
      s = float(i) / nc #uniformlly interpolate the scalar values
      hsv[0] = 240. - 240. *(s-sMin)/(sMax-sMin) #rainbow color calculation
      rgba = hsvRgb(hsv) # convert hsv to rgb (function provided!)
      rgba.append(1.0)   # set alpha (or opacity) channel
      lut.SetTableValue(i, *rgba)
```

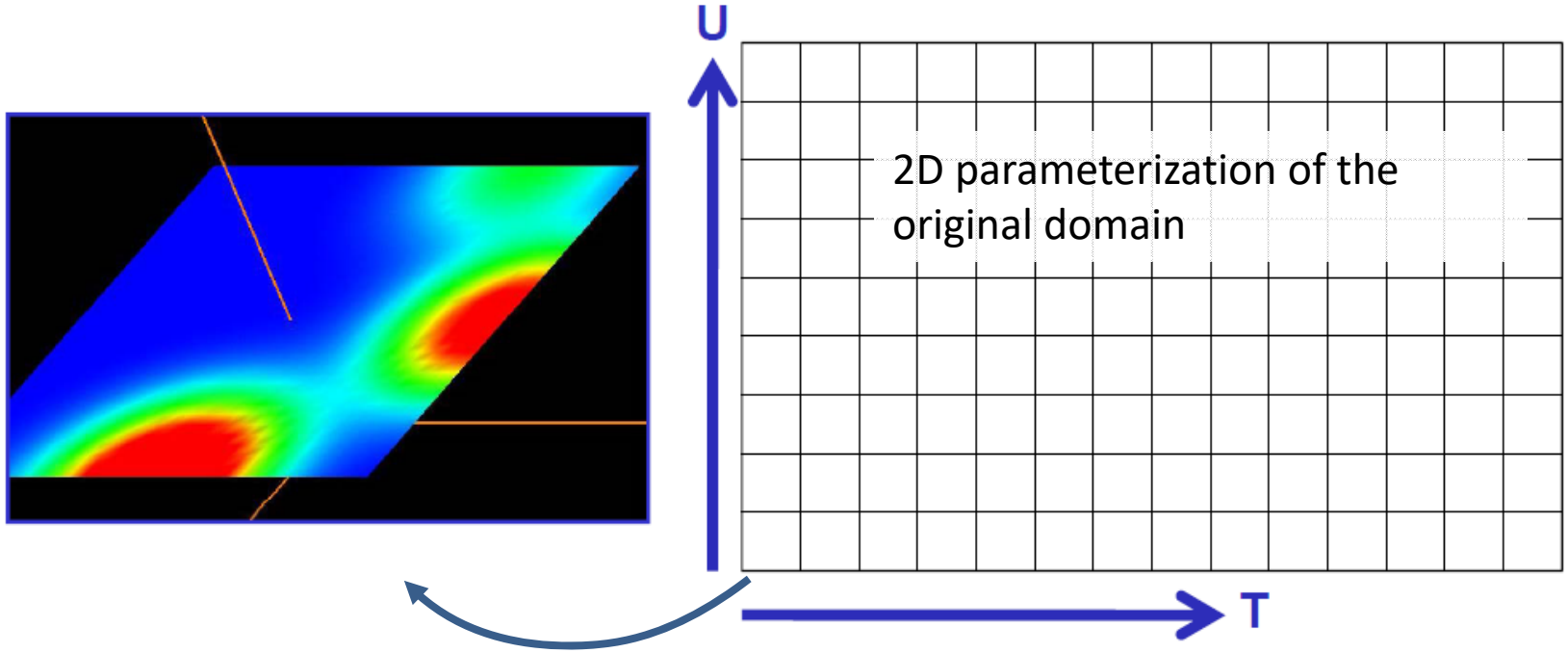


Generate 2D color plots

1. Color transfer function
2. Color interpolation

2D Interpolated Color Plots

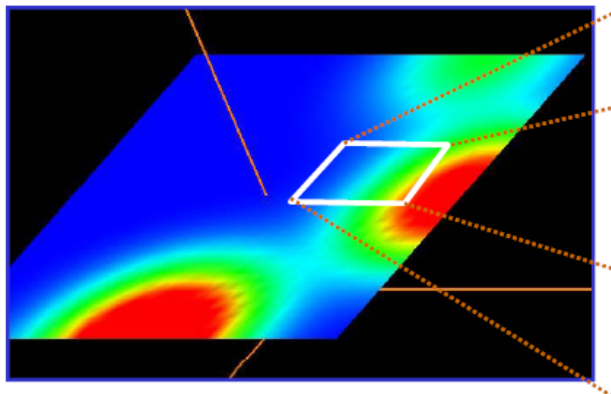
- How can we turn the **discrete** samples into a **continuous** color plot?
- Here's the **input**: we have a *2D grid* of data points. At each node, we have an X, Y, Z, and a scalar value S. We know S_{min}, S_{max}, and a **Transfer Function**.

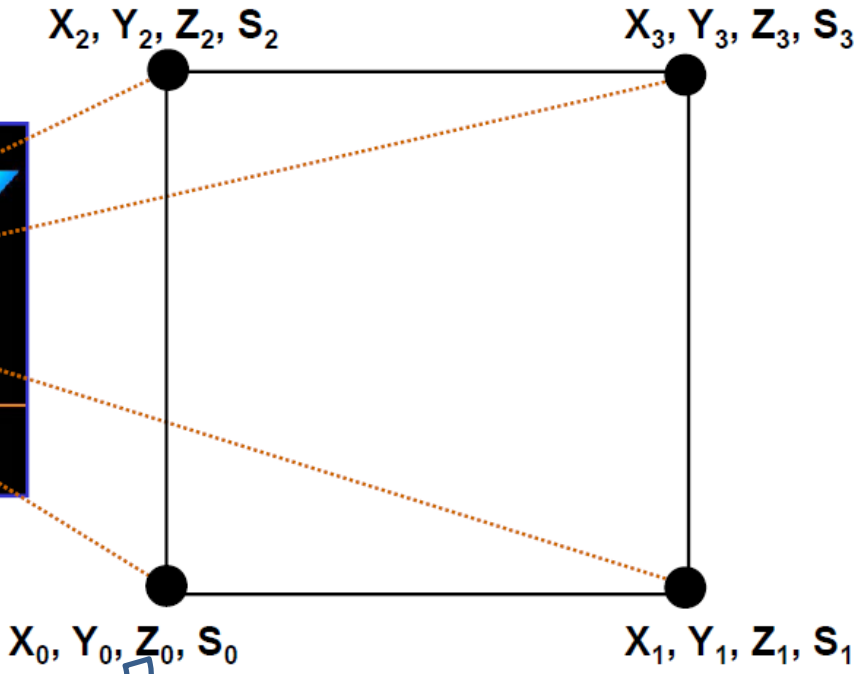


Even though this is a 2D technique, we keep around the X, Y, and Z coordinates so that the grid doesn't have to lie in any particular plane.

2D Interpolated Color Plots

- Let us look at one **square (or quad)** of the mesh at a time





For each scalar value at a vertex

float arrays `hsv[3], rgb[3]`

$\text{hsv}[0] = 240. - 240. \frac{S - S_{min}}{S_{max} - S_{min}};$

`HsvRgb (hsv, rgb);`

Convert hsv color to rgb color as monitor uses additive color model!

Use VTK color look-up table with a mapper

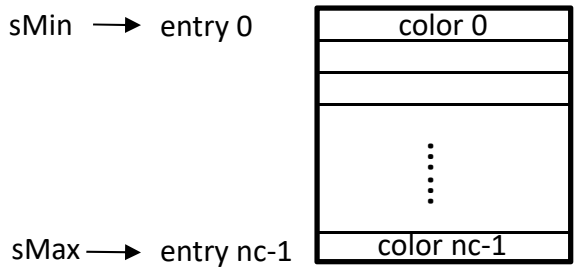
Get scalar value range

```
min_scalar,max_scalar =
vtk_geometry.GetOutput().GetPointData().GetArray("s").GetRange()
```

Create a vtkPolyDataMapper object and connect it to the geometry of the data domain (i.e., the **grid**). Assume a look up table, `lut` , has been built.

```
vtk_poly_mapper = vtk.vtkPolyDataMapper()
vtk_poly_mapper.SetInputData(vtk_geometry.GetOutput())

vtk_poly_mapper.SetScalarModeToUsePointData()
vtk_poly_mapper.SetLookupTable(lut)
vtk_poly_mapper.SetScalarRange(min_scalar, max_scalar) #Specify range in terms
of scalar minimum and maximum (smin,smax). These values are used to map scalars
into lookup table
vtk_poly_mapper.SelectColorArray(scalar_field) #set the name of scalar field
```



2D Interpolated Color Plots

- **What happen underneath is the calling of OpenGL drawing functions**

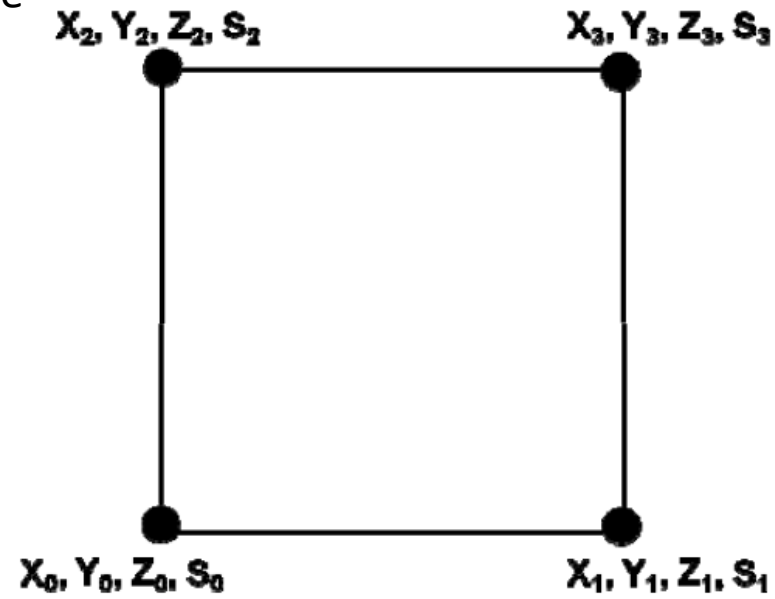
Loop through the individual quads to perform the following for each quad

```
// compute color at V0
glColor3fv (rgb0);
glVertex3f (x0, y0, z0);

// compute color at V1
glColor3fv (rgb1);
glVertex3f (x1, y1, z1);

// compute color at V3
glColor3fv (rgb3);
glVertex3f (x3, y3, z3);

// compute color at V2
glColor3fv (rgb2);
glVertex3f (x2, y2, z2);
```



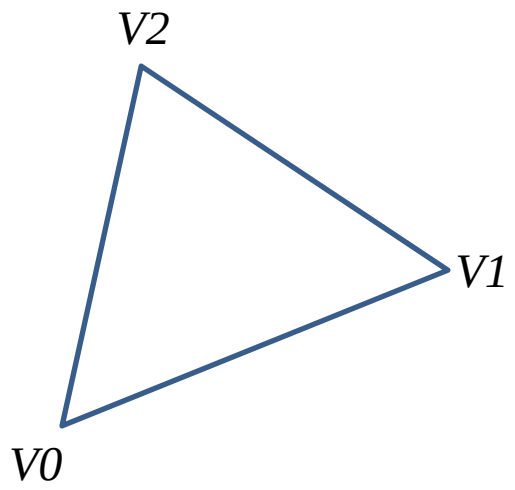
If the grid is unstructured like a triangle mesh, then…

Loop through the individual triangles to perform the following for each triangle

```
// compute color at V0
glColor3fv (rgb0);
glVertex3f (x0, y0, z0);

// compute color at V1
glColor3fv (rgb1);
glVertex3f (x1, y1, z1);

// compute color at V3
glColor3fv (rgb3);
glVertex3f (x3, y3, z3);
```



OpenGL mechanism

What are the benefits of using color plots?

What are the benefits of using color plots?

Full spatial coverage
Provide overview of the data
No need to extract additional information – direct method
Easy to implement

How to produce effective color plots?

How to produce effective color plots?

<u>Choose proper color scales and design transfer functions!!!</u>

scalar data

Linear mapping

$$\boldsymbol{y} = a\boldsymbol{x} + b$$

one or multiple dimensions of the selected color space (e.g., hue, or saturation, or value of the HSV space)

Color interpolation between samples

Using VTK or OpenGL

Limitations of color plots?