BGHiggins/UCDavis/May_2015

# Working with *Mathematica*'s ColorFunction

## Background

In these notes we discuss how to use *Mathematica*'s ColorFunction to embellish plots. As part of these notes we also provide a brief introduction to *Mathematica*'s use of colors

## Basic Color Functions:

### RGBColor

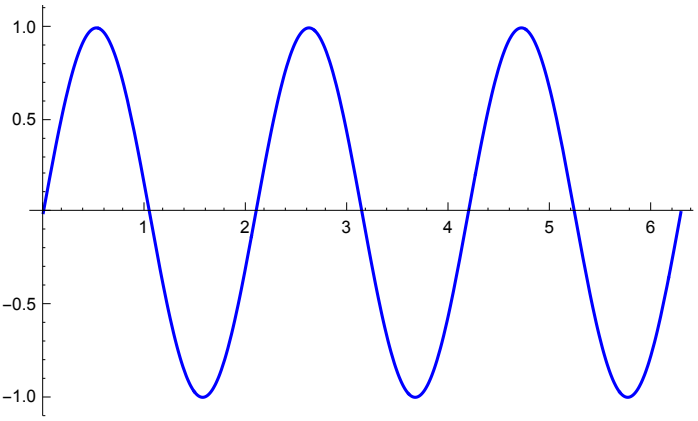
Here is a brief overview of the use of "colors" in plots and more generally the use of ColorData. At the simplest level we can specify a color for a plot. At the simplest level is to use a RGBColor function

```
RGBCOlor[red, green, blue]
```

where the specified colors {red, green, blue) take on values between 0 and 1. How is an example using Plot and specifying the plot will be displayed in Blue.

```
In[3]:= Plot[Sin[3 x], {x, 0, 2 π}, PlotStyle → RGBColor[0, 0, 1]]
```

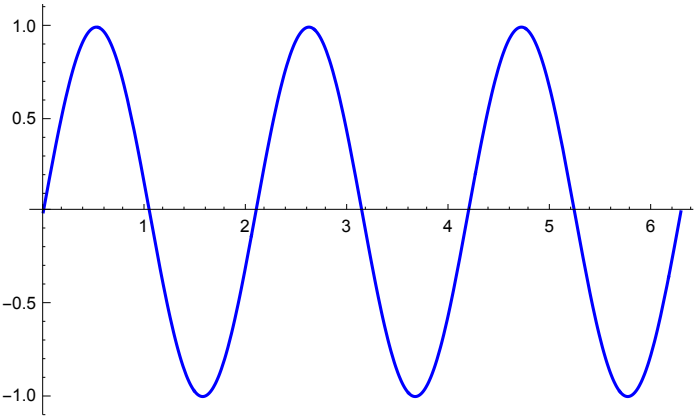
Out[3]=




Instead of specifying the function RGBColor you could simply use the function Blue

In[5]:=
```
Plot[Sin[3 x], {x, 0, 2 π}, PlotStyle → Blue]
```

Out[5]=

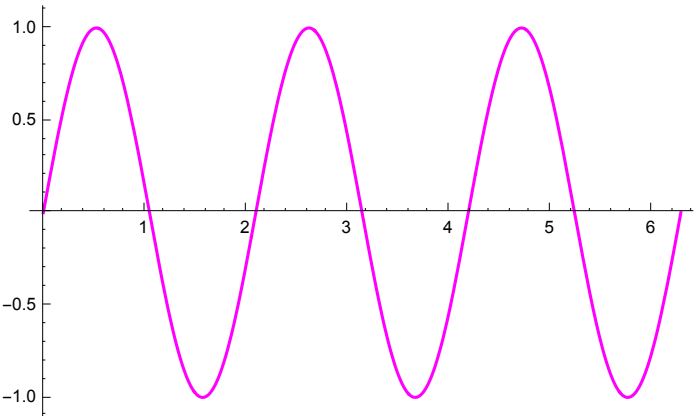



## CMYKColor

Instead of specifying RGBColor you can specify CMYKColor[*cyan, magenta,yellow,black*] or simply use the shortcuts, Cyan, Magenta, Yellow, etc

In[6]:=
```
Plot[Sin[3 x], {x, 0, 2 π}, PlotStyle → Magenta]
```

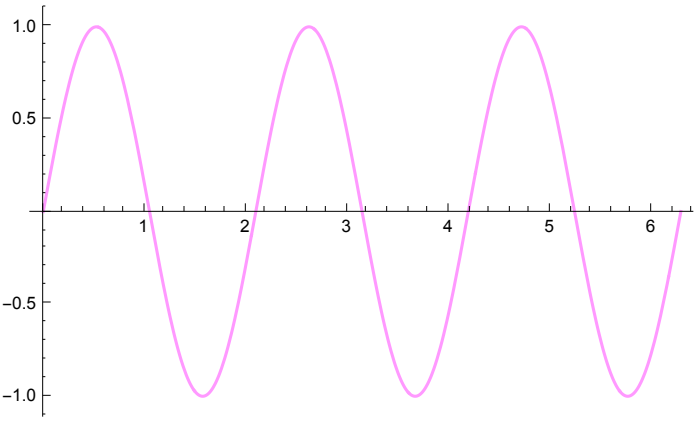
Out[6]=




Again each color can be specified with a value between 1 and 0

In[8]:=
```
Plot[Sin[3 x], {x, 0, 2 π}, PlotStyle → CMYKColor[0, 0.4, 0, 0]]
```

Out[8]=



# ColorData

Here is a simple example of

In[1]:=
```
? ColorData
```

ColorData["*scheme*"] gives a function that generates
colors in the named color scheme when applied to parameter values.
ColorData["*scheme*", "*property*"] gives the specified property of a color scheme.
ColorData["*collection*"] gives a list of color schemes in a named collection.
ColorData[] gives a list of named collections of color schemes. ≫

Here is a simple example that shows the use of ColorData using the scheme called "TemperatureMap". First let us see what ColorData produces when you specify a color scheme

In[9]:=
```
ColorData["TemperatureMap"]
```

Out[9]=
```
ColorDataFunction[ Name: TemperatureMap
                   Gradient: ]
```

You get what is called a ColorDataFunction. If you specify a parameter value between 0 and 1 you get a color from this ColorData function

In[11]:=
```
ColorData["TemperatureMap"][0.8]
```

Out[11]=

You can also specify the parameter value as follows

In[12]:=
```
ColorData["TemperatureMap", 0.8]
```

Out[12]=

The second argument could be one of the Properties of ColorData

In[13]:=
```
ColorData["Properties"]
```

Out[13]=
```
{AlternateNames, ColorFunction, ColorList, ColorRules,
 Image, Name, Panel, ParameterCount, Range, StandardName}
```

Let us examine what "ColorList" does

In[14]:=
```
ColorData["TemperatureMap", "ColorList"]
```

Out[14]=
```
Missing[NotApplicable]
```

In this case there is no "ColorList" property for the color scheme "TemperatureMap"

Here is what the property "Image" yields for our color scheme

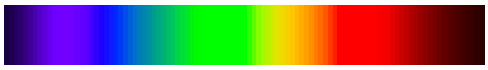
In[16]:=
```
ColorData["TemperatureMap", "Image"]
```

Out[16]=

Here is what the property "Image" yields for the color scheme "VisibleSpectrum":

In[17]:=
```
ColorData["VisibleSpectrum", "Image"]
```

Out[17]=



The property Range gives an idea of what type of parameter you need to pass to the selected color scheme. For example:

In[19]:=
```
{ColorData["TemperatureMap", "Range"], ColorData["VisibleSpectrum", "Range"]}
```

Out[19]=
```
{{0, 1}, {380, 750}}
```

Shown below are the various color schemes one can use with ColorData

In[47]:=
```
ColorData[]
```

Out[47]=
```
{Gradients, Indexed, Named, Physical}
```

In[48]:=
```
ColorData["Named"]
```

Out[48]=
```
{Atoms, Crayola, GeologicAges, HTML, Legacy, WebSafe}
```

*Mathematica* has a number of ColorData gradients. Here is a list of the “gradients”

In[50]:=
```
ColorData["Gradients"]
```

Out[50]=
```
{AlpineColors, Aquamarine, ArmyColors, AtlanticColors, AuroraColors,
 AvocadoColors, BeachColors, BlueGreenYellow, BrassTones, BrightBands,
 BrownCyanTones, CandyColors, CherryTones, CMYKColors, CoffeeTones, DarkBands,
 DarkRainbow, DarkTerrain, DeepSeaColors, FallColors, FruitPunchColors,
 FuchsiaTones, GrayTones, GrayYellowTones, GreenBrownTerrain, GreenPinkTones,
 IslandColors, LakeColors, LightTemperatureMap, LightTerrain, MintColors,
 NeonColors, Pastel, PearlColors, PigeonTones, PlumColors, Rainbow,
 RedBlueTones, RedGreenSplit, RoseColors, RustTones, SandyTerrain,
 SiennaTones, SolarColors, SouthwestColors, StarryNightColors, SunsetColors,
 TemperatureMap, ThermometerColors, ValentineTones, WatermelonColors}
```

In addition to Gradients there is also a list of “indexed” colors

```
ColorData["Indexed"]
```

Out[56]=
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41,
 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60,
 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,
 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113}
```

## Indexed Color Scheme

Let us consider the Indexed ColorData “46”

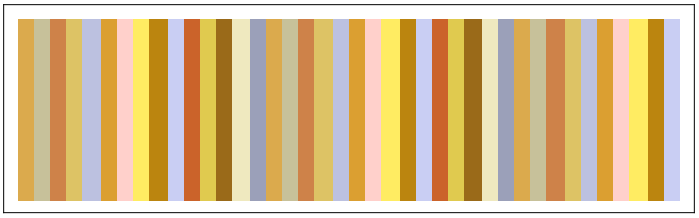
In[59]:=
```
ColorData[46]
```

Out[59]=
```
ColorDataFunction[ Index: 46  Colors: 15
                   Palette: ]
```

We see we have 15 colors in this indexed scheme, We can display the colors in this scheme using ArrayPlot:

In[60]:=
```
ArrayPlot[{Range[40]}, ColorFunction → ColorData[46],
 ColorFunctionScaling → False, AspectRatio → .3]
```

Out[60]=



## TemperatureMap Color Scheme

Let us consider a particular gradient scheme called “TemperatureMap”:

In[58]:=
```
ColorData["TemperatureMap"]
```

Out[58]=


By specifying a value as the second argument we get a color from the specified gradient scheme ( in this case “TemperatureMap”

In[52]:=
```
ColorData["TemperatureMap"][0.1]
```

Out[52]=

Here are the range of colors for values between 0 and 1 in steps of 0.1

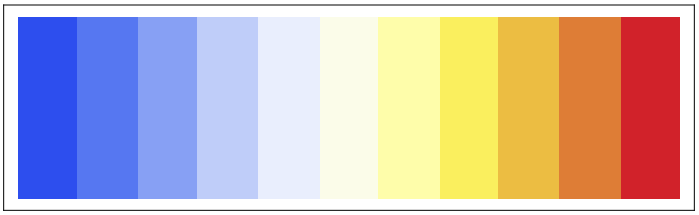
```
Table[ColorData["TemperatureMap"][i], {i, 0, 1, 0.01}]
```

Out[45]=

A nicer way to display the colors is using ArrayPlot ( here with intervals of 0.1 and not 0.01)

In[65]:=
```
ArrayPlot[{Range[0, 1, 0.1]},
 ColorFunction → ColorData["TemperatureMap"], AspectRatio → .3]
```

Out[65]=



If you specify a value outside the range you get the color associated with the lower or upper bound values. For example

In[54]:=
```
ColorData["TemperatureMap"][-4] == ColorData["TemperatureMap"][0]
```

Out[54]=
```
True
```

In[55]:=
```
ColorData["TemperatureMap"][4] == ColorData["TemperatureMap"][1]
```

Out[55]=
```
True
```

# Using the function ColorFunction in conjunction with a ColorScheme

With the function ColorFunction the user can specify how a color is going to be applied to the plot. For example suppose we a a Sinc function and we want to color the plot by specifying a color related to the height of the function. Let us select the  color scheme “NeonColors” Here is the “Image of this color scheme

In[20]:=
```
ColorData["NeonColors", "Image"]
```

Out[20]=

and here is the “range” for the parameter that can be passed to this color scheme

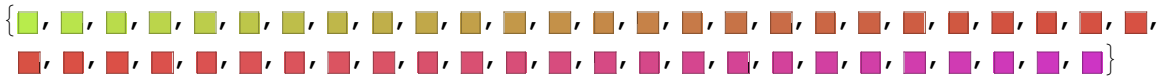
In[21]:=
```
ColorData["NeonColors", "Range"]
```

Out[21]=
```
{0, 1}
```

Thus as we vary the parameter as shown below we get different colors

In[15]:=
```
Table[ColorData["NeonColors"][i], {i, 0, 1, 0.02}]
```

Out[15]=



If we inspect the Help on the use of ColorFunction, we see that for the function Plot we can either pass values related to “x” or “y” values of the plot. The most general syntax for using color function is

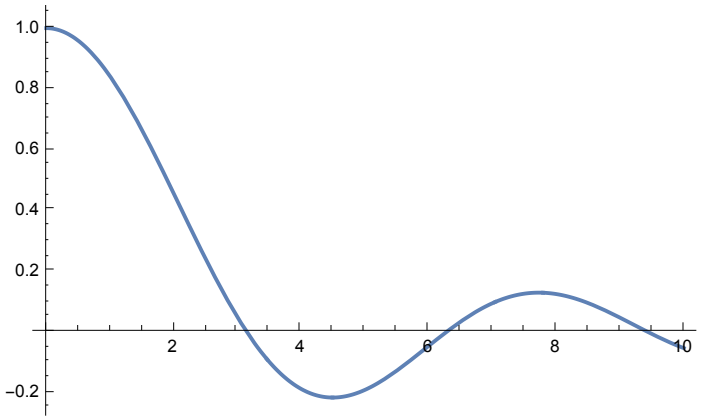
```
Specify the color scheme based on y values:
        ColorFunction→Function[{x,y},ColorData["NeonColors"][y]]

Specify the color scheme based on x-values:
        ColorFunction→Function[{x,y},ColorData["NeonColors"][x]]
```

## Apply ColorFunction to Plot

Suppose we want to apply our color scheme to the sync function

In[24]:=
```
Plot[Sinc[x], {x, 0, 10}, PlotStyle → Thick]
```

Out[24]=




The first thing to note is that the range of y values for the plot are approximately -0.2<y<1, while the range for the x values is 0≤x≤10. But the range for the parameter that can be passed to NeonColors scheme is {0,1}. This is handled conveniently in *Mathematica* by using the function

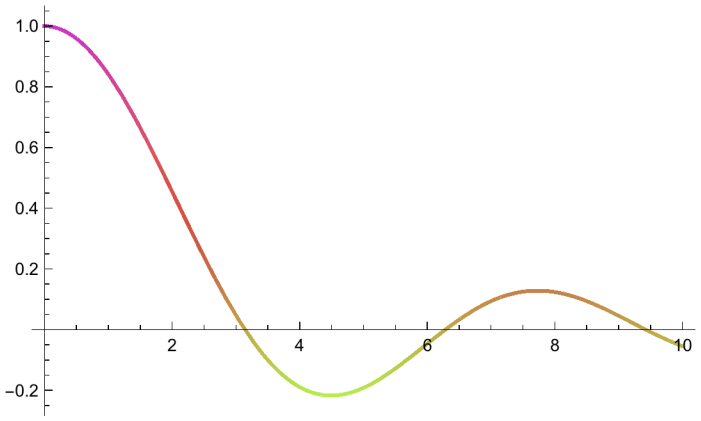
In[25]:=
```
? ColorFunctionScaling
```

ColorFunctionScaling is an option for graphics functions that specifies whether
arguments supplied to a color function should be scaled to lie between 0 and 1. ≫

For many Plot functions the default value is “True”. Let us apply our color scheme “NeonColors” to the sync plot using ColorFunction. Here is the code

In[26]:=
```
Plot[Sinc[x], {x, 0, 10}, PlotStyle → Thick,
 ColorFunction → Function[{x, y}, ColorData["NeonColors"][y]], PlotRange → All]
```

Out[26]=




Because ColorFunctionScaling→True, the minimum value is colored by

In[27]:=
```
ColorData["NeonColors"][0]
```

Out[27]=

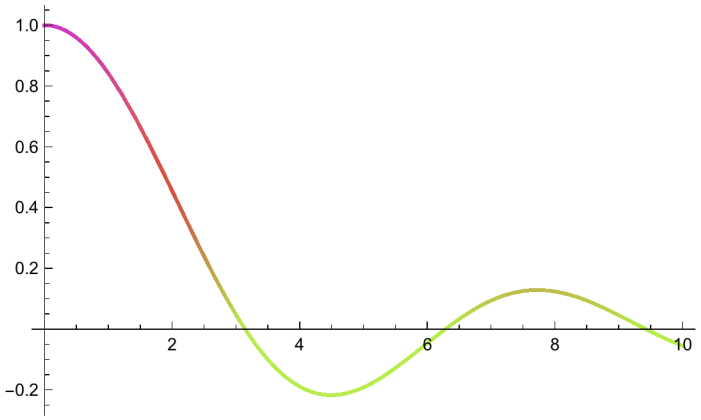
Note further that the ColorFunction option is specified as a pure function:

In[32]:=
```
? Function
```

Function[*body*] or *body* & is a pure function. The formal parameters are # (or #1), #2, etc.
Function[*x*, *body*] is a pure function with a single formal parameter *x*.
Function[{$x_1$, $x_2$, ...}, *body*] is a pure function with a list of formal parameters. >>

In[28]:=
```
Plot[Sinc[x], {x, 0, 10}, PlotStyle → Thick,
 ColorFunction → Function[{x, y}, ColorData["NeonColors"][y]],
 PlotRange → All, ColorFunctionScaling → False]
```

Out[28]=




Because we have set ColorFunctionScaling→False, *all* negative values of y are colored with

In[31]:=
```
? Function
```

Function[*body*] or *body* & is a pure function. The formal parameters are # (or #1), #2, etc.
Function[*x*, *body*] is a pure function with a single formal parameter *x*.
Function[{$x_1$, $x_2$, ...}, *body*] is a pure function with a list of formal parameters. >>

In[29]:=
```
ColorData["NeonColors"][0]
```

Out[29]=

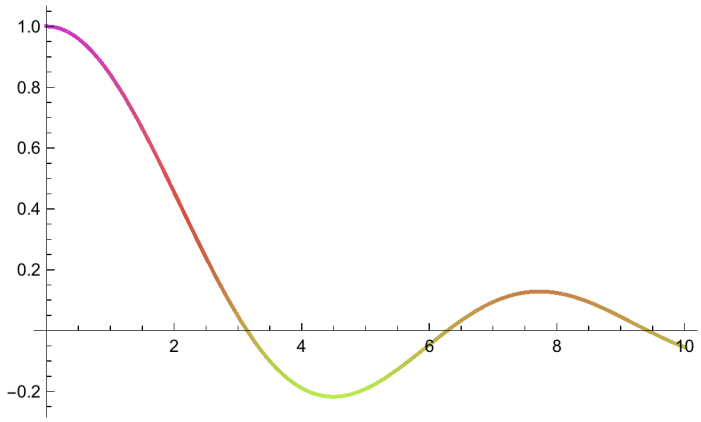
We can also use the short-hand notation for specifying a pure function, which is some times more convenient. Using this format we have to specify which “slot value” must be passed to the pure function :

#1 denotes x values
#2 denotes y values

Here is the code using a pure function. Note the use of the parentheses and the “&”.

In[30]:=
```
Plot[Sinc[x], {x, 0, 10}, PlotStyle → Thick,
 ColorFunction → (ColorData["NeonColors"][#2] &), PlotRange → All]
```

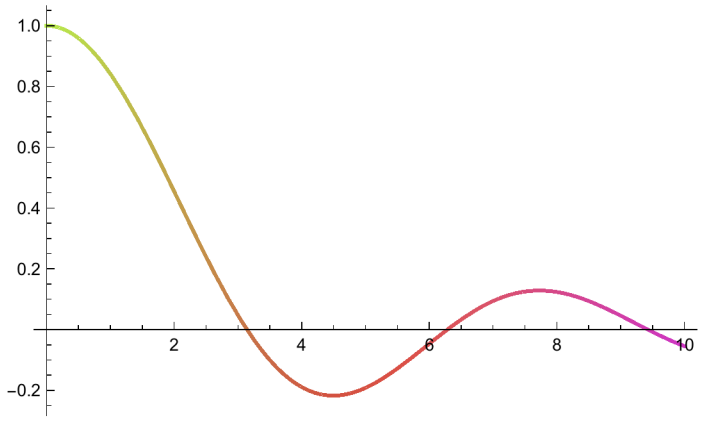
Out[30]=




Let us now color the plot using the x-values

In[33]:=
```
Plot[Sinc[x], {x, 0, 10}, PlotStyle → Thick,
 ColorFunction → (ColorData["NeonColors"][#1] &)]
```

Out[33]=




In[15]:=
```
Table[ColorData["NeonColors"][i], {i, 0, 1, 0.02}]
```

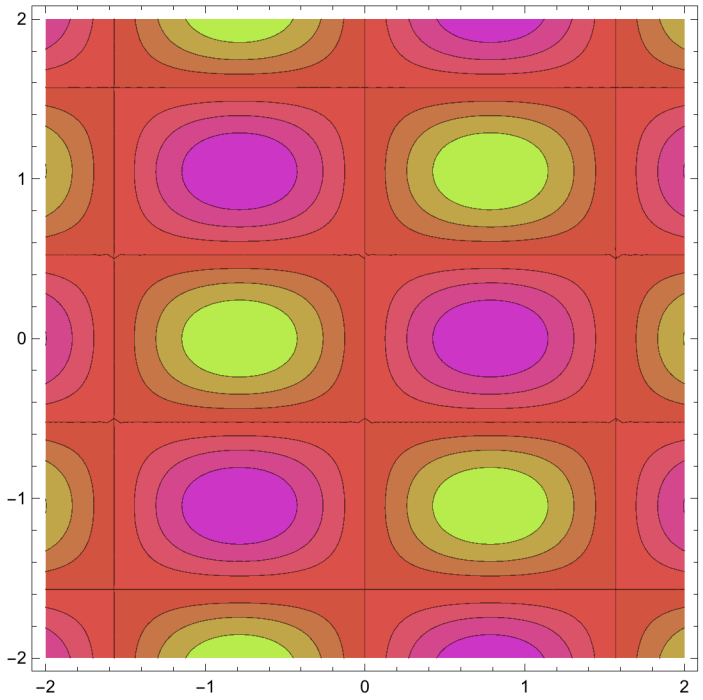
Out[15]=

## Applying ColorFunction to ContourPLot

In this case the parameter that is passed to the pure function is the contour value. Thus there is no need to specify which slot value is to be used, when the short-hand version of a pure function is used.

In[12]:=
```
ContourPlot[2 Sin[2 x] Cos[3 y], {x, -2, 2}, {y, -2, 2},
 ColorFunction → (ColorData["NeonColors"][#] &), ColorFunctionScaling → True]
```

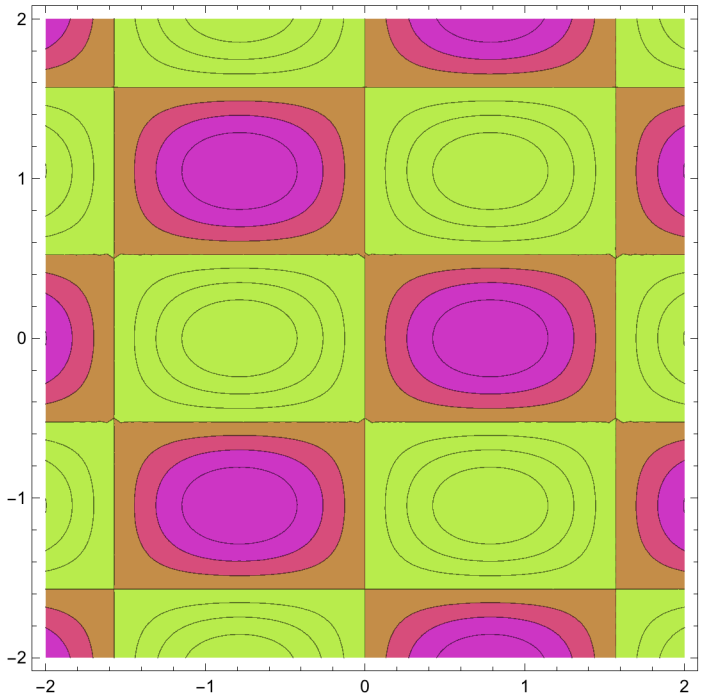
Out[12]=




Here is the same plot but now setting ColorFunctionScaling→False, and using the long version for specifying a pure function:

In[13]:=
```
ContourPlot[2 Sin[2 x] Cos[3 y], {x, -2, 2}, {y, -2, 2},
 ColorFunction → Function[{z}, ColorData["NeonColors"][z]],
 ColorFunctionScaling → False]
```

Out[13]=

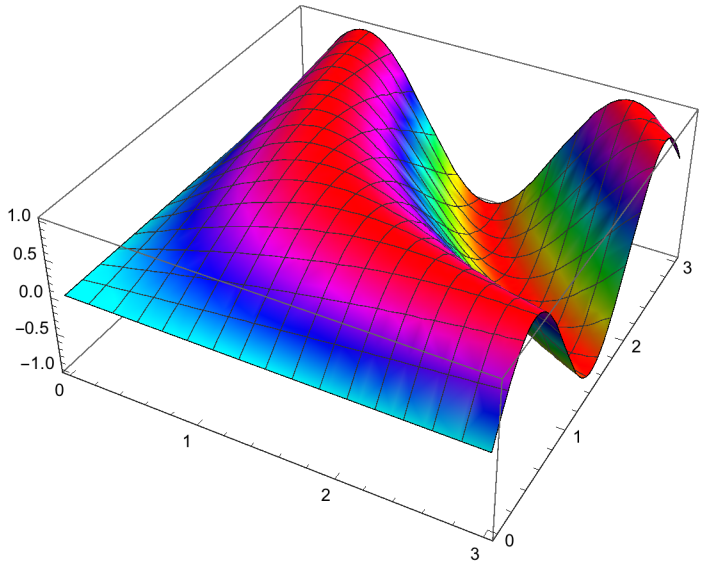


## Applying ColorFunction to Plot3D: Example 1

In Plot3D the color function takes on 3 possible values: x,y,z. Here is how we use ColorFunction with the colorFunction called Hue. Note in this case we are passing the z value to the ColorFunction, and the Long hand version of the pure function is used

In[32]:=
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}, ColorFunction → Function[{x, y, z}, Hue[z]]]
```

Out[32]=




## TemperatureMap Color Scheme

Let us consider a particular gradient scheme called “TemperatureMap” which we will use in Plot3D

In[58]:=
```
ColorData["TemperatureMap"]
```

Out[58]=


By specifying a value as the second argument we get a color from the specified gradient scheme ( in this case “TemperatureMap”

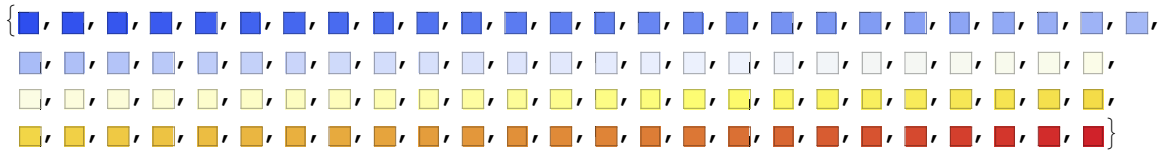
In[52]:=
```
ColorData["TemperatureMap"][0.1]
```

Out[52]=

Here are the range of colors for values between 0 and 1 in steps of 0.1

```
Table[ColorData["TemperatureMap"][i], {i, 0, 1, 0.01}]
```

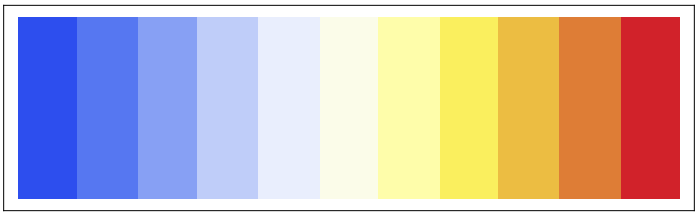
Out[45]=



A nicer way to display the colors is using ArrayPlot ( here with intervals of 0.1 and not 0.01)

In[65]:=
```
ArrayPlot[{Range[0, 1, 0.1]},
 ColorFunction → ColorData["TemperatureMap"], AspectRatio → .3]
```

Out[65]=



If you specify a value outside the range you get the color associated with the lower or upper bound values. For example

In[54]:=
```
ColorData["TemperatureMap"][-4] == ColorData["TemperatureMap"][0]
```

Out[54]=
```
True
```

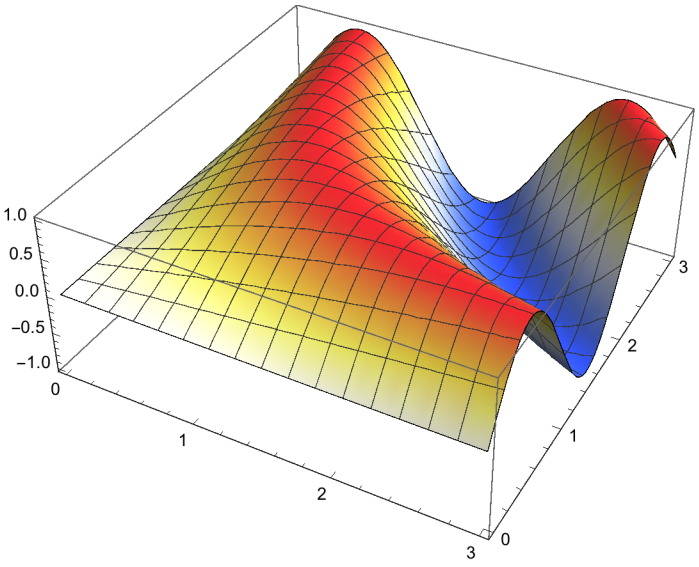
In[55]:=
```
ColorData["TemperatureMap"][4] == ColorData["TemperatureMap"][1]
```

Out[55]=
```
True
```

Here is the same plot but using the TemperatureMap color scheme instead of a Hue color scheme and as before we pass the z value to the color scheme. In this case we use the short-hand version for creating the Pure function ( see the previous section using “NeonColors” scheme:

In[36]:=
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}, ColorFunction → (ColorData["TemperatureMap"][#3] &)]
```

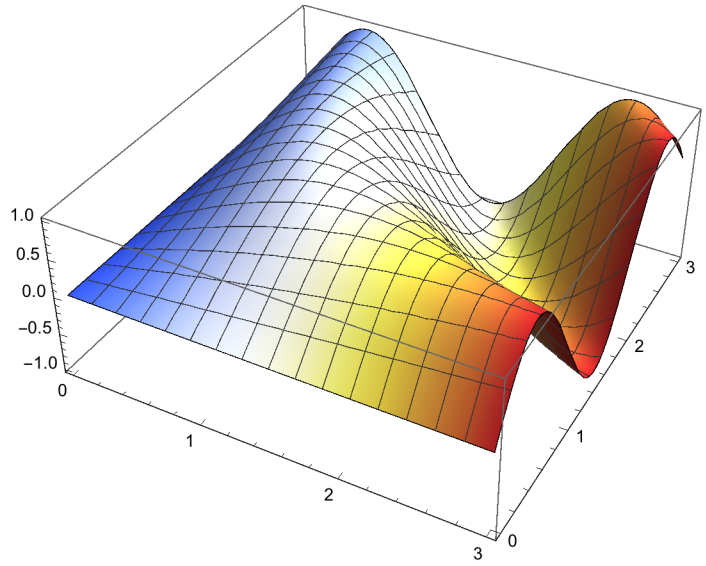
Out[36]=




Here is the same plot but using the TemperatureMap color scheme but passing the x value to the color scheme. Note ColorFunctionScaling is by default True.

In[37]:=
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3}, ColorFunction → (ColorData["TemperatureMap"][#1] &)]
```

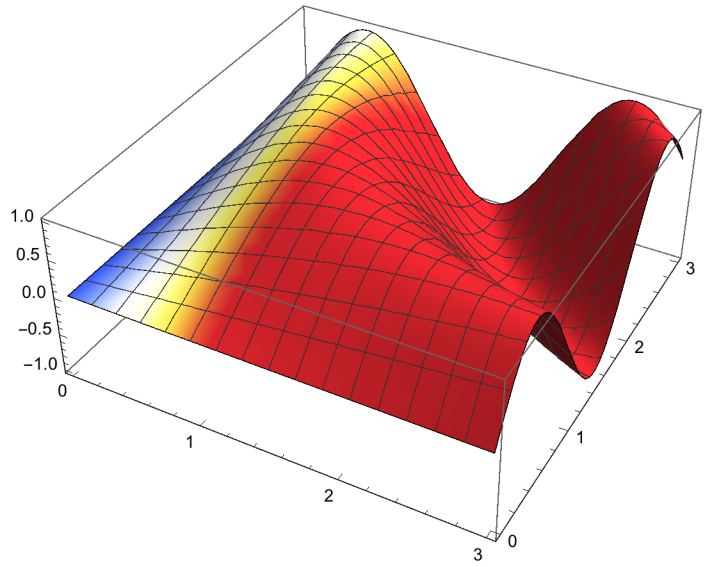
Out[37]=




When we set ColorFunctionScaling->False and pass the x-values to the color scheme. For all values of x>1 we have a Red plot

In[41]:=
```
Plot3D[Sin[x y], {x, 0, 3}, {y, 0, 3},
 ColorFunction → (ColorData["TemperatureMap"][#1] &), ColorFunctionScaling → False]
```

Out[41]=

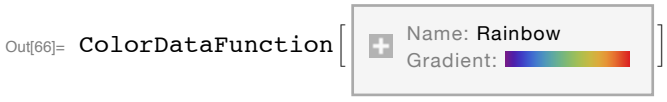



## Specifying a ColorFunction for a Plot3D: Example 2

Let us work with the following ColorData

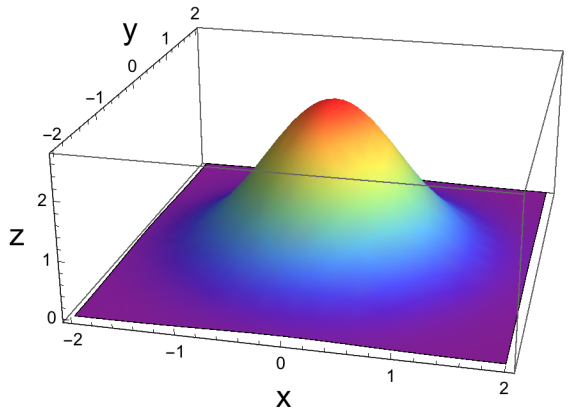
In[66]:=
```
ColorData["Rainbow"]
```

Out[66]=



In[77]:=
```
Plot3D[ Exp[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2},
 ColorFunction → (ColorData["Rainbow"][#3] &), PlotRange → All,
 Mesh → None, AxesLabel → {Style["x", 16], Style["y", 16], Style["z", 16]}]
```

Out[77]=




Note in this case we are passing the z value to the ColorFunction, which ranges from Exp[1]

In[72]:=
```
{Exp[1.], Exp[1 - 8.]}
```

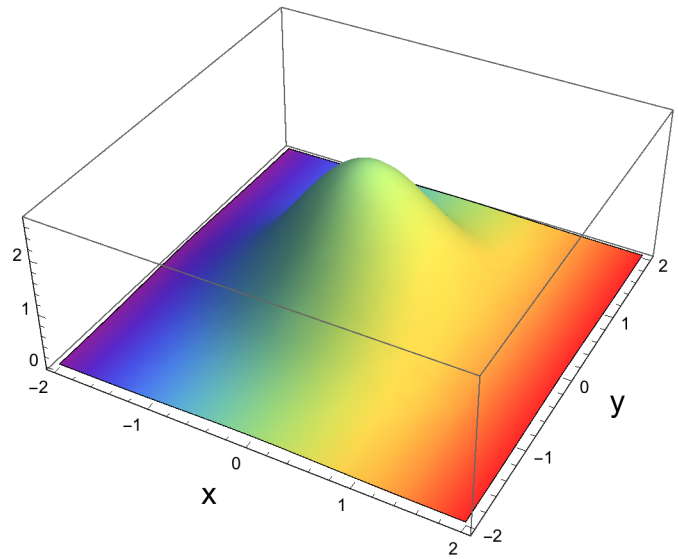
Out[72]=
```
{2.71828, 0.000911882}
```

By default, the option ColorFunctionScaling→True is used so that the z values passed to the ColorFunction are scaled from 0 to 1. Thus we get the full range of colors in the plot with no repeats.

Let us now pass the x value (slot #1) to the pure function. As the plot below shows now the x values are colored with the specified ColorData. Again ColorFunctionScaling→True is implemented. That is to say the x- values passed to ColorData are scaled from 0 to 1 so that there are no repeat values of the colors in the plot.

In[78]:=
```
Plot3D[ Exp[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2},
 ColorFunction → (ColorData["Rainbow"][#1] &), PlotRange → All,
 Mesh → None, AxesLabel → {Style["x", 16], Style["y", 16], Style["z", 16]}]
```

Out[78]=

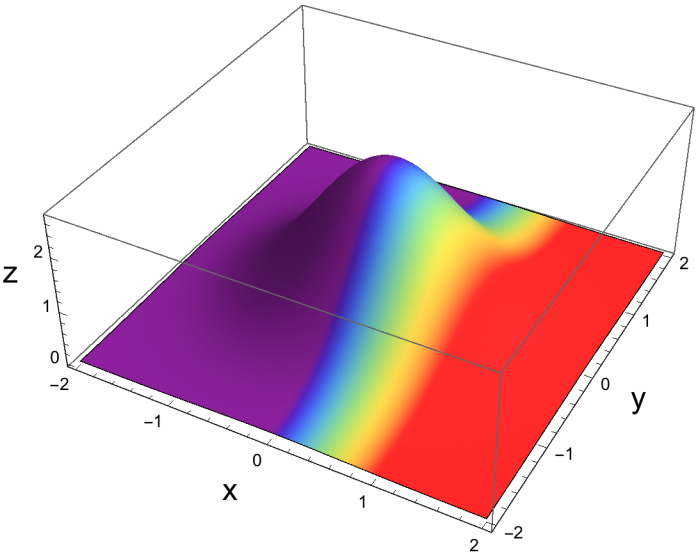



If we specify ColorFunctionScaling→False, then the ColorData is applied for x values between 0 and 1,

but for x values outside this range the upper ( 1<x<2) and lower (-2<x<0) bound colors from the Color-Data Function are used.

In[54]:=
```
Plot3D[ Exp[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2},
 ColorFunction → (ColorData["Rainbow"][#1] &), PlotRange → All,
 Mesh → None, AxesLabel → {Style["x", 16], Style["y", 16], Style["z", 16]},
 ColorFunctionScaling → False, PlotPoints → 100]
```

Out[54]=




Note by increasing the number of PlotPoints, one gets smother degradations of the ColorData

## Using an Indexed ColorData for Plot3D: Example 3

In this example we will use  an indexed ColorData. Let us look at the ColorData with index 54, which has 21 colors:

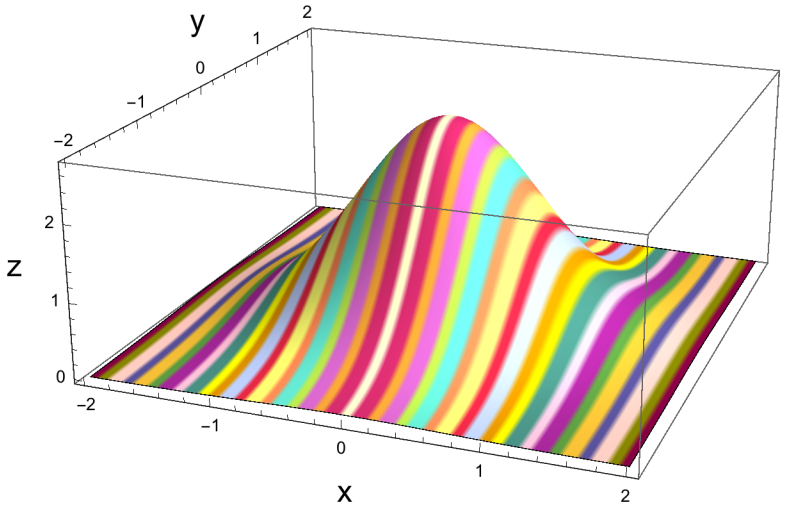
In[99]:=
```
Table[ColorData[54][i], {i, 1, 21}]
```

Out[99]=

In the example below we pass the x value to the Colordata[54]. For all x values between 0 and 2 we scale the values so that we get 20 values. Here is the result. Note it is important to increase the number of PlotPoints so that we get uniform strips for a given x value,

In[107]:=
```
Plot3D[ Exp[1 - x^2 - y^2], {x, -2, 2}, {y, -2, 2},
 ColorFunction → (ColorData[54][Abs[Round[10 #1]]] &), PlotRange → All,
 Mesh → None, AxesLabel → {Style["x", 16], Style["y", 16], Style["z", 16]},
 ColorFunctionScaling → False, PlotPoints → 100]
```

Out[107]=




Note that the colors are repeated

## Using the Blend Color Scheme for ArrayPlot

Let us now examine the function Blend which blends two or more colors

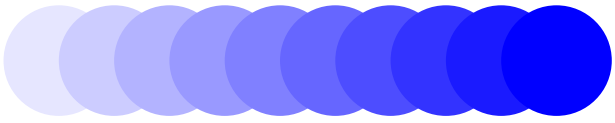
In[55]:=
```
? Blend
```

Blend[{$col_1$, $col_2$}, $x$] gives a color obtained by blending a fraction $1 - x$ of color $col_1$ and $x$ of color $col_2$.
Blend[{$col_1$, $col_2$, $col_3$, ...}, $x$] linearly interpolates between colors $col_i$ as $x$ varies from 0 to 1.
Blend[{{$x_1$, $col_1$}, {$x_2$, $col_2$}, ...}, $x$] interpolates to give $col_i$ when $x = x_i$.
Blend[{$col_1$, $col_2$, ...}, {$u_1$, $u_2$, ...}] blends all the $col_i$, using fraction $u_i$ of color $col_i$.
Blend[{$image_1$, $image_2$, ...}, ...] blends pixel values of 2D or 3D images $image_i$. ≫

Here is an example that blends Blue and White for coloring a disk

In[114]:=
```
Graphics[Table[{Blend[{White, Blue}, x], Disk[{10 x, 0}]}, {x, 0, 1, 1 / 10}]]
```

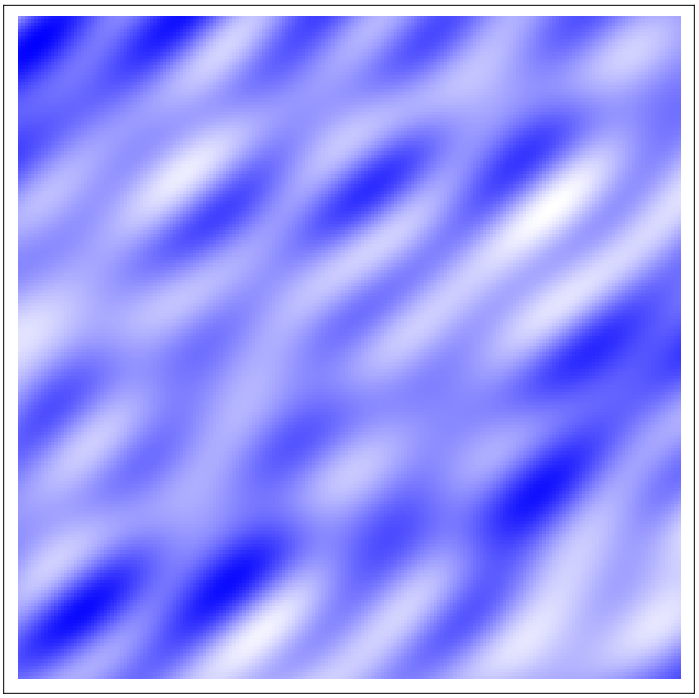
Out[114]=



In this next plot we use ArrayPlot and use Blend for our ColorFunction. In this case the color function is applied to each cell of the array. The value of each cell is sent to the ColorFunction. In the example below. The cell values may be greater than 1, so the default ColorFunctionScaling->True is used.

In[115]:=
```
ArrayPlot[Table[Evaluate[Sum[Sin[RandomReal[5, 2].{x, y}], {10}]],
  {x, 0, 10, .05}, {y, 0, 10, .05}], ColorFunction → (Blend[{White, Blue}, #] &)]
```

Out[115]=



# Specifying a ColorFunction for a SphericalPlot3D: Example 1

Our final Example is applying a color function to a spherical 3D shape using SphericalPlot3D

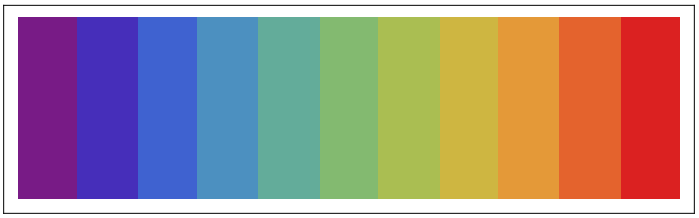
In[59]:=
```
? SphericalPlot3D
```

SphericalPlot3D[$r$, $\theta$, $\phi$] generates a 3D plot
with a spherical radius $r$ as a function of spherical coordinates $\theta$ and $\phi$.
SphericalPlot3D[$r$, {$\theta$, $\theta_{min}$, $\theta_{max}$}, {$\phi$, $\phi_{min}$, $\phi_{max}$}] generates a 3D spherical
plot over the specified ranges of spherical coordinates.
SphericalPlot3D[{$r_1$, $r_2$, ...}, {$\theta$, $\theta_{min}$, $\theta_{max}$}, {$\phi$, $\phi_{min}$, $\phi_{max}$}] generates a 3D spherical plot with multiple surfaces. ≫

The ColorData function in this case can take 6 arguments, {x,y,z,r,$\theta$,$\phi$}. We will also use the Rainbow color scheme

In[3]:=
```
ArrayPlot[{Range[0, 1, 0.1]}, ColorFunction → ColorData["Rainbow"], AspectRatio → .3]
```

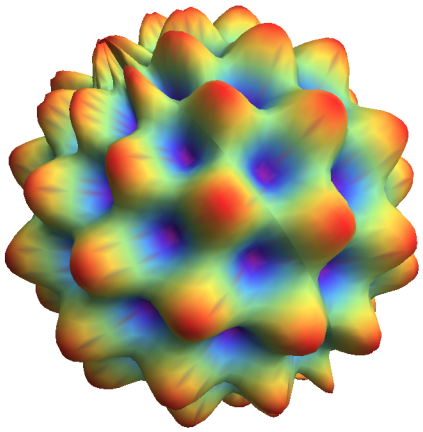
Out[3]=



In the first example we pass the $\phi$ value to the ColorFunction

In[2]:=
```
SphericalPlot3D[1 + Sin[8 ϕ] Cos[8 θ] / 5, {θ, 0, π},
 {ϕ, 0, 2 π}, ColorFunction → (ColorData["Rainbow"][#6] &),
 Mesh → None, PlotPoints → 25, Boxed → False, Axes → False]
```

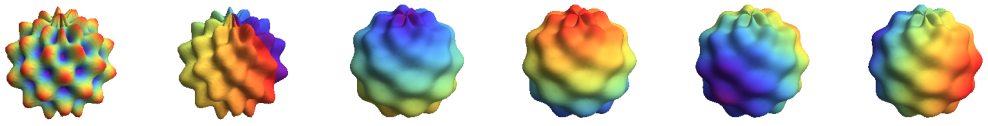
Out[2]=



Here is a overview that shows the application of the ColorFunction to all 6 variables

In[10]:=
```
Row[{SphericalPlot3D[1 + Sin[8 ϕ] Cos[8 θ] / 5, {θ, 0, π}, {ϕ, 0, 2 π},
   ColorFunction → (ColorData["Rainbow"][#6] &), Mesh → None, PlotPoints → 25,
   Boxed → False, Axes → False], SphericalPlot3D[1 + Sin[8 ϕ] Cos[8 θ] / 5,
   {θ, 0, π}, {ϕ, 0, 2 π}, ColorFunction → (ColorData["Rainbow"][#5] &),
   Mesh → None, PlotPoints → 25, Boxed → False, Axes → False],
  SphericalPlot3D[1 + Sin[5 ϕ] Sin[10 θ] / 10, {θ, 0, π}, {ϕ, 0, 2 π},
   ColorFunction → (ColorData["Rainbow"][#4] &), Mesh → None,
   PlotPoints → 25, Boxed → False, Axes → False],
  SphericalPlot3D[1 + Sin[5 ϕ] Sin[10 θ] / 10, {θ, 0, π}, {ϕ, 0, 2 π},
   ColorFunction → (ColorData["Rainbow"][#3] &), Mesh → None,
   PlotPoints → 25, Boxed → False, Axes → False],
  SphericalPlot3D[1 + Sin[5 ϕ] Sin[10 θ] / 10, {θ, 0, π}, {ϕ, 0, 2 π},
   ColorFunction → (ColorData["Rainbow"][#2] &), Mesh → None,
   PlotPoints → 25, Boxed → False, Axes → False],
  SphericalPlot3D[1 + Sin[5 ϕ] Sin[10 θ] / 10, {θ, 0, π}, {ϕ, 0, 2 π},
   ColorFunction → (ColorData["Rainbow"][#1] &), Mesh → None,
   PlotPoints → 25, Boxed → False, Axes → False]}]
```

Out[10]=

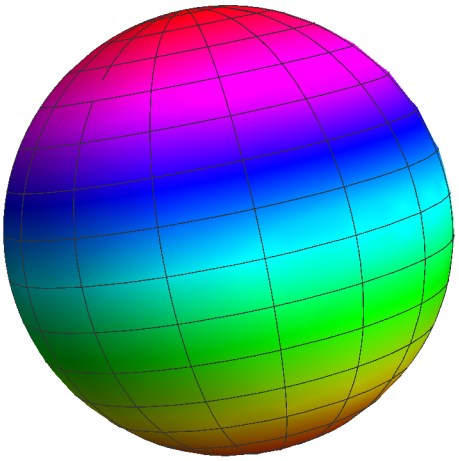


# Other Examples

In this example we use the color scheme Hue to color the surface of a sphere based on the value of the variable z

In[1]:=
```
SphericalPlot3D[1, {θ, 0, Pi}, {ϕ, 0, 2 Pi},
 ColorFunction → Function[{x, y, z, θ, ϕ, r}, Hue[z]], Axes → None, Boxed → False]
```

Out[1]=



# Apply a Texture to the Surface of a Sphere

Let us first generate an image based on the value of spherical coordinates using spherical harmonics

In[2]:=
```
myImag = Image[DensityPlot[Re[SphericalHarmonicY[6, 5, θ, ϕ]], {ϕ, 0, 2 π}, {θ, 0, π},
    AspectRatio → Automatic, ColorFunction → "DarkRainbow", Frame → False,
    ImagePadding → None, PerformanceGoal → "Quality", PlotPoints → 55,
    PlotRange → All, PlotRangePadding → None], ImageResolution → 144]
```

Out[2]=

We then use the TextureCoordinateFunction to apply the image to the surface of the sphere

In[3]:=
```
? TextureCoordinateFunction
```

TextureCoordinateFunction is an option to Plot3D and
similar functions that specifies a function that computes texture coordinates. ≫

If you apply it to SphericalPlot3D it can take up to 6 coordinates. In the example below, we apply it to $\phi$ (#5) and $\theta$(#4) coordinates

In[4]:=
```
SphericalPlot3D[1 , {θ, 0, π}, {ϕ, 0, 2 π},
 Mesh → None, TextureCoordinateFunction → ({#5, 1 − #4} &),
 PlotStyle → Directive[Specularity[White, 10], Texture[myImag]],
 Axes → False, RotationAction → "Clip", Boxed → False]
```

Out[4]=