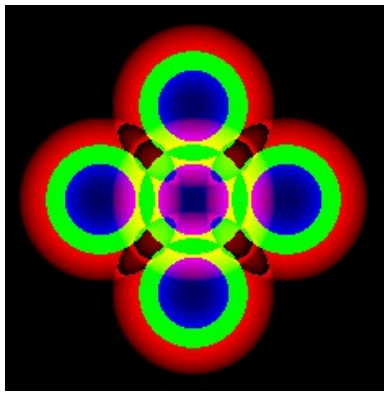
KALEIDOSCOPE

Creative graphical pattern generator application

Version: v0.1

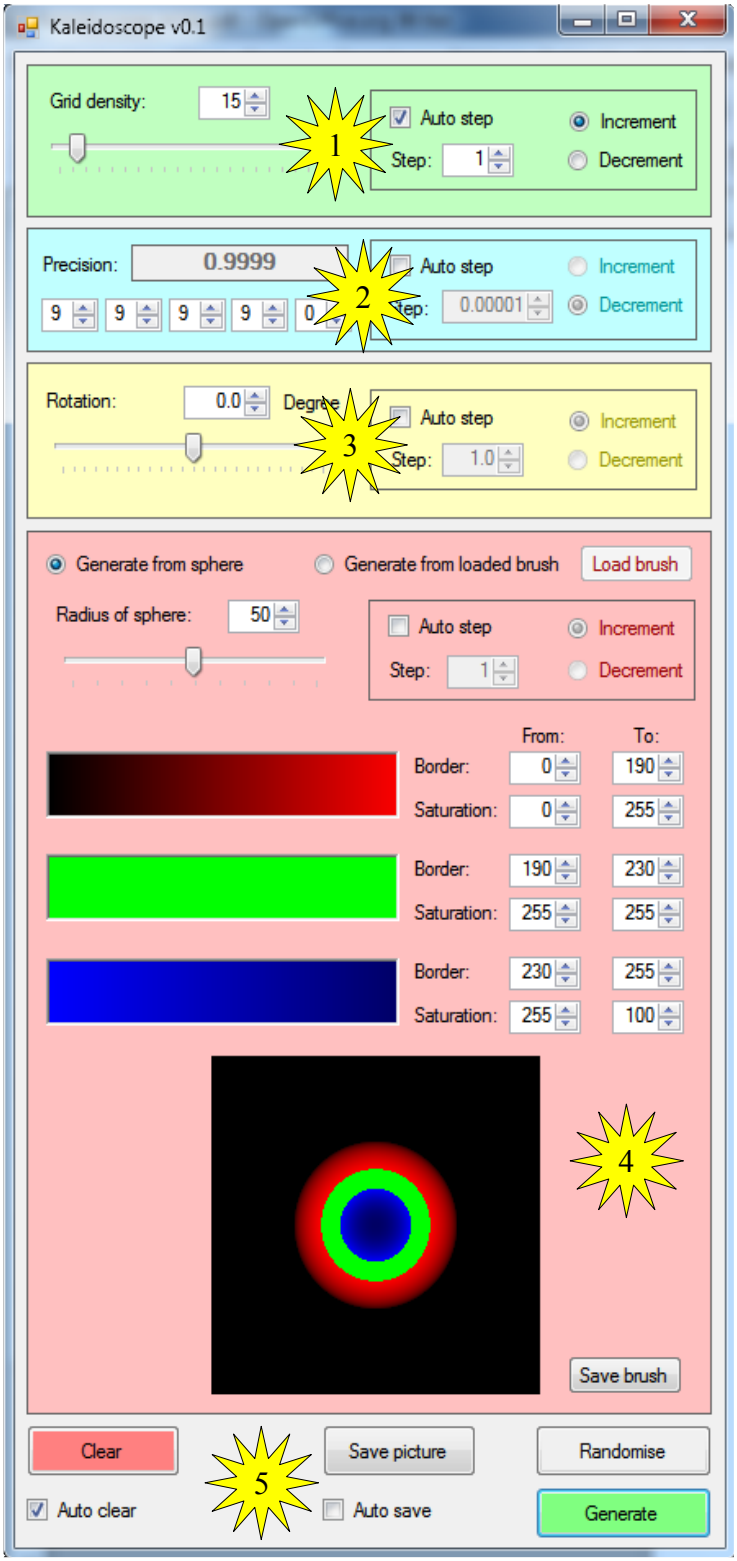


KALEIDOSCOPE is an easy to use pattern generator application. The user interface is uncomplicated. All the functions are easy to access. You don't need to search for settings in menus. The patterns are a result of calculations, but the look of the generated patterns is greatly depends on the actual settings, therefor the pattern generation process can only be partially controlled. We cannot really know what the pattern appearing will look like exactly, similarly to the pictures in a real kaleidoscope. This uncertainty and randomness with the almost infinite numbers of variations make using of Kaleidoscope application fun. It is possible to use the pictures created in other graphical applications as background pattern or wallpaper.

The application basically creates the patterns from tinted and shaded sphere surfaces, but it is possible to load optional bitmap images the application can use instead of the sphere surface to create the pattern. The application doesn't simply draw the sphere surfaces one on top of the other but sums up the colors of each overlapped pixels. This method does some kind of interference between the shapes and colors and creates spectacular geometric formations.

General use of the application

The graphical user interface consists of five main area or panel. These panels has different background colors for the better perspicuity. The stars with numbers means the sequence of setting steps.



1 On this panel you can set up that how many spheres the program places in a row or rather in a column. On the right side with the **Auto step** check box you can enable or disable that the program automatically increase or decrease the **Grid density** value with the **Step** value after the pattern generation process.

2 On this panel you can set up that the program how precisely calculates the location of the spheres. As much this value as less spheres the program will use for the pattern generation. But if the **Precision** is too little the program will place as many spheres on the picture that the calculation time is will be too long and the pattern will be too confused. On the right side with the **Auto step** check box you can enable or disable that the program automatically increase or decrease the **Precision** value with the **Step** value after the pattern generation process.

3 On this panel you can set up that how many degrees the program rotates the grid which was discussed above. This value has a mighty influence to the looks of the final pattern, so this allow of further variation possibilities. On the right side with the **Auto step** check box you can enable or disable that the program automatically increase or decrease the **Rotation** value with the **Step** value after the pattern generation process.

On this panel you can set up the properties of the spheres, such as the radius and the colors. Defining colors happens as follows. At every color component you can set up two borders (**From** and **To**) to that color. 0 value belongs to the side of the sphere and 255 value belongs to the center of the sphere. Every one of them define a circle on the sphere which size is depends on the values. The area between those two circle define a ring on the sphere. The program fills this ring with a color transition. The color transition has defined by **Saturation** values. The program makes a transition between the **From** and **To** values as you can see on the color bars. On the right side with the **Auto step** check box you can enable or disable that the program automatically increase or decrease the **Radius of sphere** value with the **Step** value after the pattern generation process.

If you check the **Generate from loaded brush** radio button you can load a **png** bitmap picture into the brush area. The program will use this picture for pattern generation instead of the sphere. This possibility extremely extends the ability of the software. The size of the loaded picture must be not more then 201x201 pixels.

With the controls on this area possible to start the pattern creation as well as set up and doing some other function. After clicking on the **Generate** button the program calculates a new pattern on the basis of the actual settings which was discussed above and displays the generated pattern.

Clicking on the **Randomize** button the program randomly sets up the main parameters for pattern generation. In this manner you can create spectacular patterns with one or two click. These patterns offer a start-up basis for the further experiments.

Clicking on the **Save picture** button you can save the actual pattern in **png** format.

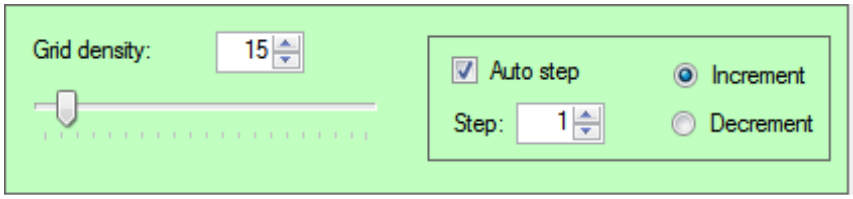
If the **Auto save** check box is in checked state the program will save all pattern automatically, in this way none of the pattern vanishing during the experiment.

If the **Auto clear** check box is in checked state (it is the basic position) the program automatically clears the view panel before a new pattern generation. Otherwise the program draws the patterns one to the other as a montage. In this manner you can create additional interesting visual effects.

With the **Clear** button you can clear the view panel whenever you wish.

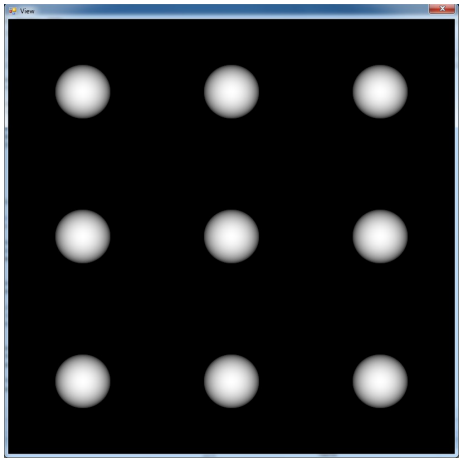
Detailed review of setting possibilities

Grid density

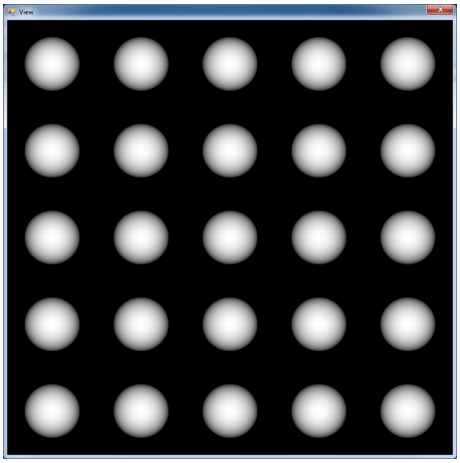


The program places the spheres to the intersection points of a square grid when generates a pattern. The **Grid density** value determines that how thick this grid will be. As less this value as infrequent the pattern will be, but in this case you need to set up larger spheres for that the interesting patterns comes into being by the overlapping. If you use less spheres you should set up higher **Grid density** value for the overlaps. That how many spheres will the program place on the intersections depends on the **Precision** value. It will be discussed later.

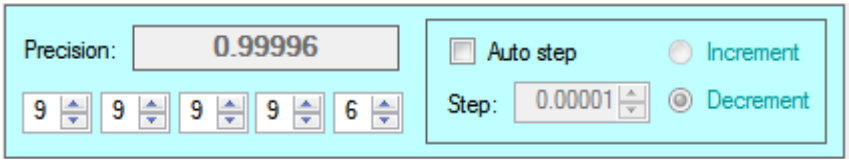
Example 1: Grid density = 3



Example 2: Grid density = 5

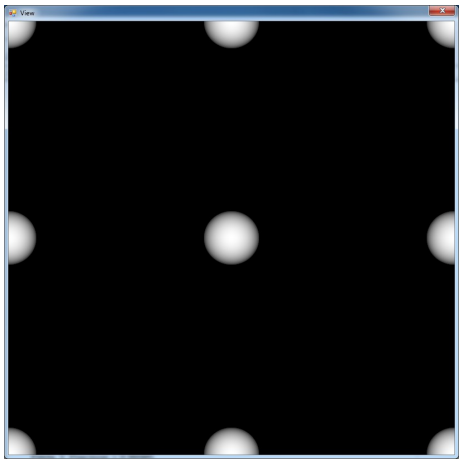


Precision

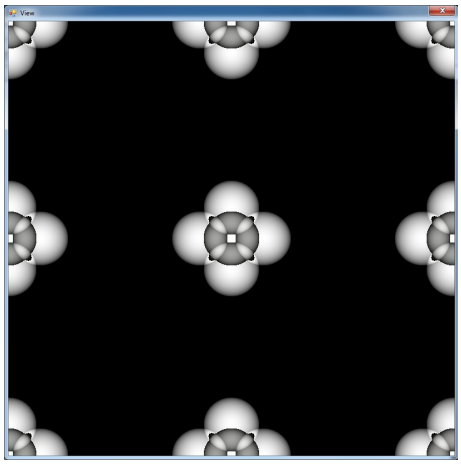


The **Precision** value determines that how many spheres will the program places to the intersection points. Depends on this value the program places the spheres closer or farther from the intersection point. As small this value as far surroundings of the intersection the program places the spheres. Hence the **Precision** value defines the complexity of the pattern. In that case if the **Precision** value is too small the pattern will contain too much tiny detail and will be too confused, in addition the calculation time will be too long. But if the **Precision** value is too large the number of spheres won't be enough for that the geometrical patterns and color combinations evolves by the multiple overlapping. The suitable **Precision** values for the desired appearance can find by experiments, but the appearance is depends on the **Grid density** and **Radius of sphere** value too.

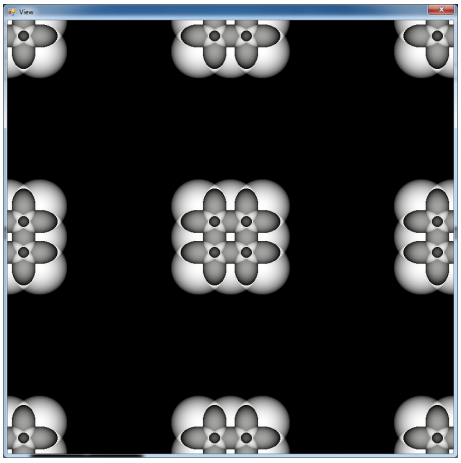
Example 1: Precision = 0.99999



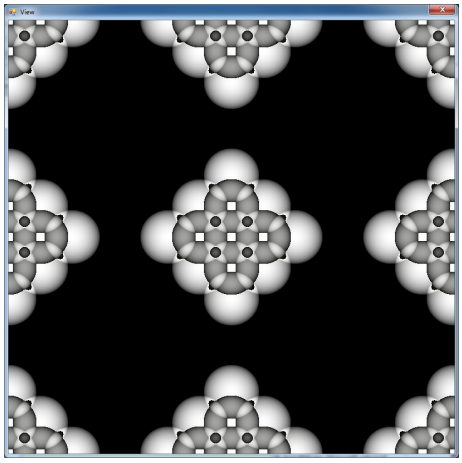
Example 2: Precision = 0.99993



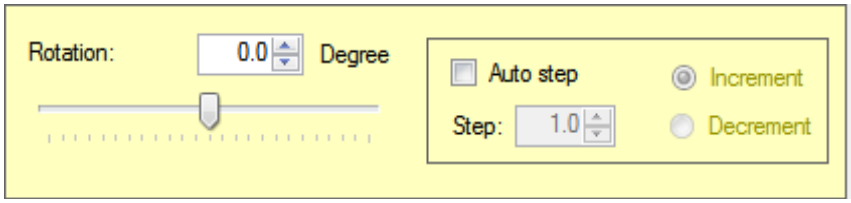
Example 3: Precision = 0.99985



Example 4: Precision = 0.99973

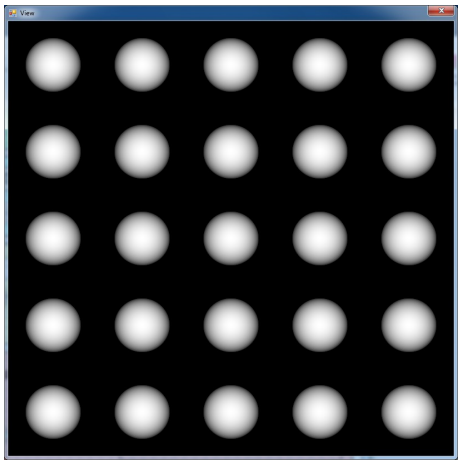


Rotation

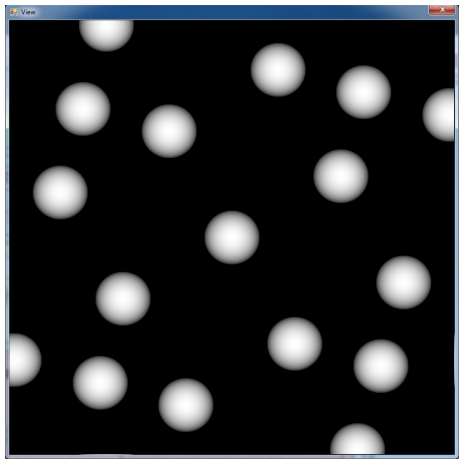


You can rotate the previously discussed grid which to the program places the spheres. This possibility with otherwise unchanged other settings brings on significantly differing pattern. Since the program use a square grid therefor the 90 and 180 degrees rotation produces pattern which practically equal with the original. However a rotation with 45 degrees results in a pattern which similar to the original but the details will be different.

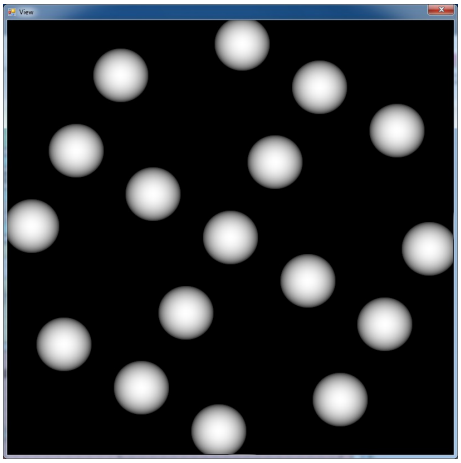
Example 1: Rotation = 0



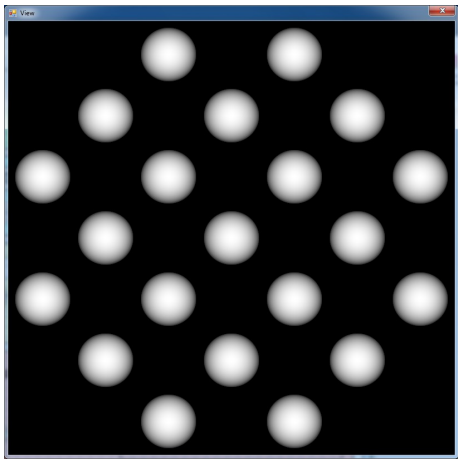
Example 2: Rotation = 15



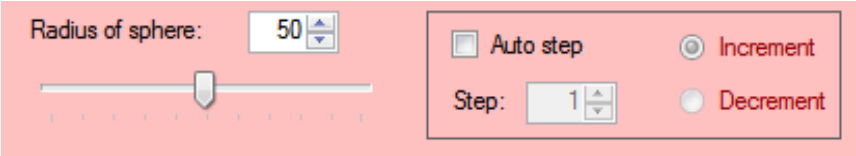
Example 3: Rotation = 30



Example 4: Rotation = 45

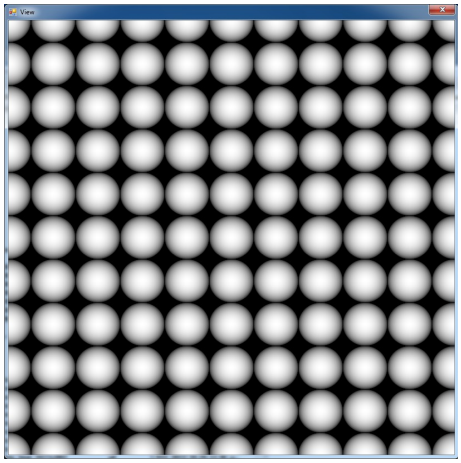


Radius of sphere

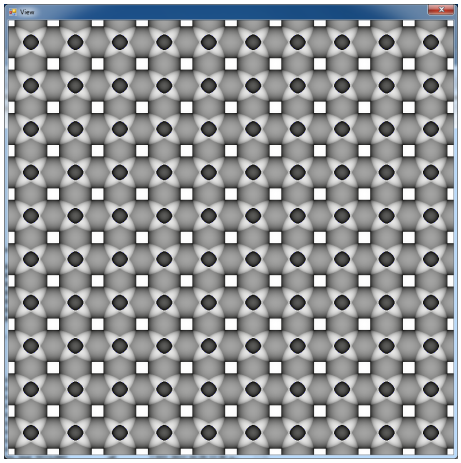


You can set up the size of the sphere in pixels with this value. This value has a significant influence to the appearance of the pattern because with the **Grid density** and **Precision** values it defines the size of the overlapped areas

Example 1: Radius of sphere = 40



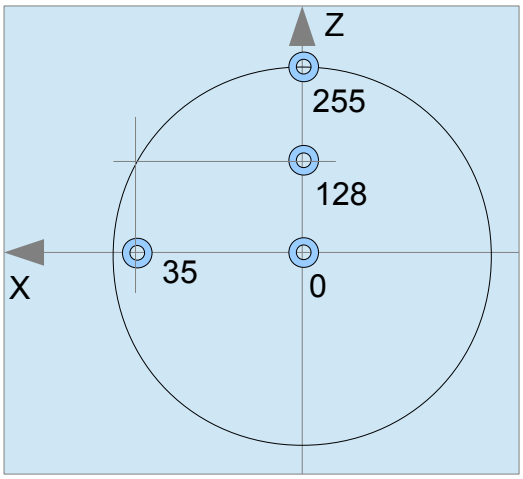
Example 2: Radius of sphere = 70

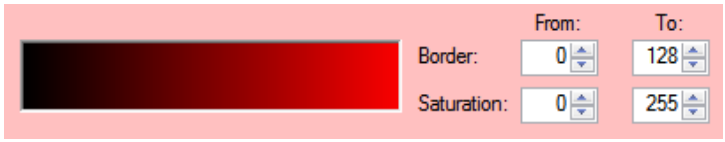


Color border

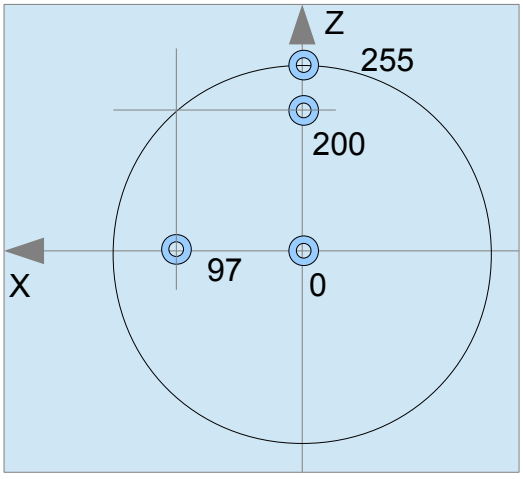
With the **Color border** value you can set up on how big area of the sphere the color component (in this example the red) expands. The 0 (zero) value denotes the side of the sphere and the 255 value denotes the center of the sphere. The division isn't linear but it proportional with the Z coordinate (or height) of the sphere surface. On the undermentioned picture you can see that if you set up 128 as **Color border** value it will means 35 on the X axis which only the 15% radius of the sphere. The preview picture helps to set up the desired value.

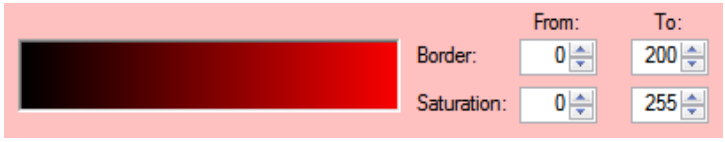
Example 1: Border = From 0 to 128

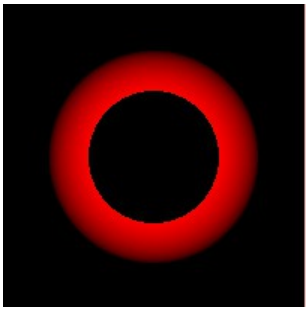




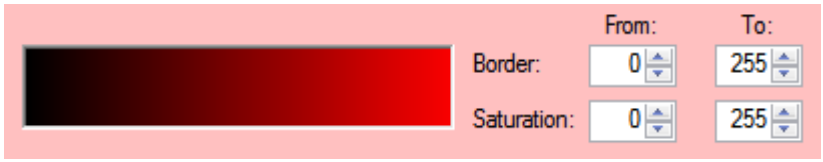
Example 2: Border = From 0 to 200





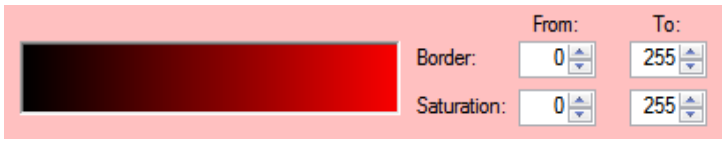


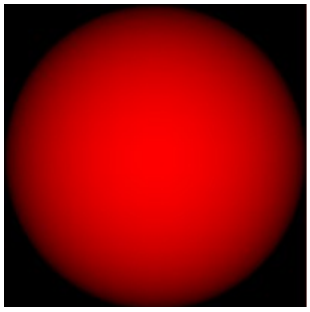
Saturation range



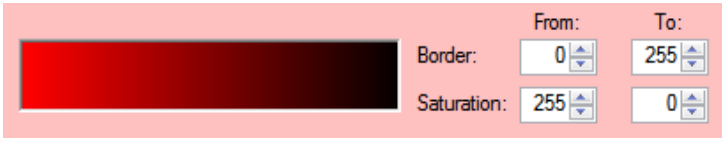
With the **Saturation range** value you can set up the range of the color component (in this example the red) which the program uses when fill the sphere. The 0 (zero) value denotes the black color and the 255 value denotes the maximal saturation of the color component.

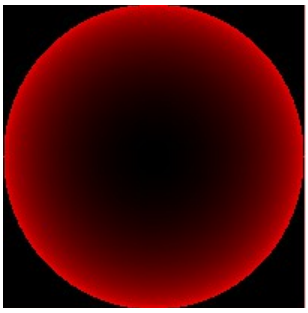
Example 1: Saturation = From 0 to 255



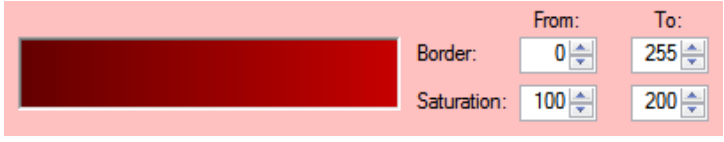


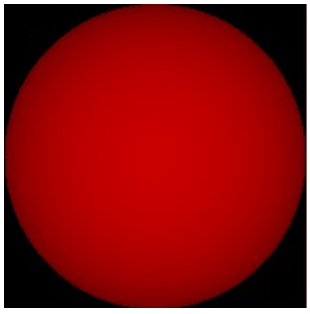
Example 2: Saturation = From 255 to 0





Example 3: Saturation = From 100 to 200





Requirements apply to software environment

Kaleidoscope requires Microsoft (c) Windows XP SP2 or recent operating system.

You have installed Microsoft (c) .NET Framework 4 (Download) to run Kaleidoscope.