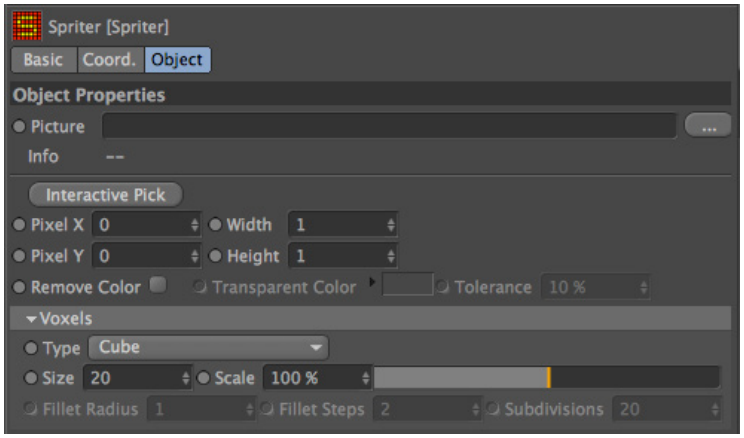
Spriter

Spriter is a generator object that will create a grid of cubic/spherical objects based on the pixel information of a bitmap. So, from this:

you get this:

The parameters are quite simple and powerful. Once you create a new **Spriter** object you get a simple, empty object, not much different from a **Null**. For anything to be produced, it is necessary to load a bitmap.

Load a bitmap by clicking the **(...)** button, on the right of the **Picture** field. Once a valid image is selected, the information about its width and height is displayed in the **Info** field, below the **Picture** field.



The **Pixel X** and **Pixel Y** fields allow you to define the top left coordinates of the rectangle from which the section of the bitmap will be picked.
The **Width** and **Height** fields allow you to define the dimensions of the rectangle from which the section of the bitmap will be picked.
However, picking an area using numerical values is not very intuitive. So, these numerical fields should be used mainly for fine adjustment of the picking area. To pick the area in a easier, more intuitive way, use the **Interactive Pick** button (it is only available when a valid bitmap is loaded but in the above image it appears already active).

After clicking the Interactive Pick button, a window with the bitmap is shown, like this:

Simply drag inside the bitmap to define a rectangular area from which the sprite will be sampled (in this screenshot there is already a rectangle set). After that, click **OK** to accept your selection or **Cancel** if you don't want to use that selection.

After clicking **OK**, you can adjust the **Pixel X**, **Pixel Y**, **Width** and **Height** parameters to fine tune the area from which the sprite will be sampled.

The **Interactive Pick** window can also be used to pick the color that will be used as the **Transparent Color** (see below). To pick the **Transparent Color** just press the **Shift** key and click on top of the color you want to use as "transparent".

Turning ON the **Remove Color** option will allow the use of a color that will be ignored. This means that all pixels sampled from the bitmap that have this color will be ignored and not produce any geometry.

The **Transparent Color** (when available, when the **Remove Color** option is ON) can be chosen by clicking the thumbnail or by using the **Interactive Pick** window. When clicking the thumbnail, the Cinema 4D color picker will appear.

The **Tolerance** option defines how close the sampled color from the bitmap must be to the **Transparent Color** to be considered "transparent" (ignored). A **Tolerance** of 0% means that the color sampled from the bitmap must be EXACTLY the same as the color set as **Transparent Color** to be considered "transparent" (ignored).

The **Voxels** panel (closed by default) is where you define the geometry that will represent the pixels from the sampled bitmap.

The voxels are exactly that: the single geometric elements that are generated to represent the pixels of the sampled bitmap.

Voxels can be cubes, rounded (chamfered) cubes or spheres.

The **Size** parameter defines the size of the edges of the voxel cubes or the diameter of the voxel spheres.

The **Scale** parameter defines the scale at which the voxels are generated. This is NOT the same as the **Size**. The **Size** parameter defines the size of the voxels and the grid where they are placed. The **Scale** parameter simply affects the final size of the voxels but not the positions on the grid where they are placed.
A final size of the voxels is a result of the **Scale** parameter multiplied by the **Size** parameter.

If the voxel **Type** is set to **Rounded Cube**, the **Fillet Radius** and **Fillet Steps** parameters become available. Always use the smaller values possible for the **Fillet Steps** as this increases the density of the generated geometry.

If the voxel **Type** is set to **Sphere**, the **Subdivisions** parameter become available. Always use the smaller values possible for the **Subdivisions** as this increases the density of the generated geometry.

All generated geometry is faceted. If you need to smooth the geometry display, add a **Phong** tag to the **Spriter** generator.

The generated geometry has their internal **Editor Color** parameter set to the color of the pixel from which it was generated from. This means that the generated sprite will show up in the editor and will also be rendered with the correct color.

However, that color is not in any material. It is merely interpreted from the **Editor Color** and it is the one used for the rendering by default when no material is assigned to the **Spriter** object.

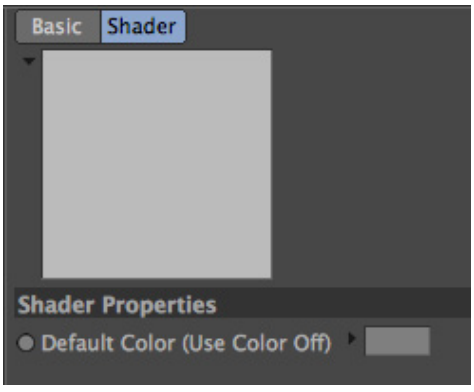
But you may need more complex materials, with **Transparency**, or **Reflection**, or **Bump** or whatever. If you create and assign a material to the **Spriter** object, the color set in the **Color** channel will be the one used for ALL voxels. This means that you would loose all the sampled colors for each voxel.

That is why the **Spriter** plugin includes the **EditorColorShader** shader.

The EditorColorShader

This shader, when used in a material, will generate the color that is in the editor **Display Color** parameter of the object whose material is assigned to.
So, the same material can produce different outputs depending on the value of the editor **Display Color** parameter of each object the material is assigned to.

This shader only has one option:



It is the color that is used by default (in the editor or in the render) when the **Use Color** parameter of the object whose material is assigned to, is set to Off.

This shader will generate the color that is set in the editor **Display Color** parameter when the **Use Color** parameter is set to **Automatic** or **On**. It will generate the color associated with the layer of object when the **Use Color** parameter is set to **Layer**.