Rewrite, critique, and commentary.   - Bill “Bald Eagle” Walker

Granite_21 macro - Documentation - version #1
[Layout and purpose of the macro sections: the color maps, pattern masks, etc.]

Introduction

All igneous [define] rocks are composed of different assemblages of juxtaposed and closely packed minerals that show different forms, colours, sizes, and compositions. The only exceptions are volcanic glasses, such as obsidian, that are the amorphous product of a rapidly cooling magma which solidifies before crystallization can occur.
Granites are in general rather coarse-grained, and so clearly show the different minerals composing their assemblages. The chemical composition of those minerals, and their relative proportions thoughout the rock, determine the type of granite.

The overall colour of the rock (white, pink, or grey) is controlled by the relative proportion of each mineral, which facilitates the optical classification of a particular granite. Sometimes particularly large crystals (phenocrysts, generally plagioclase [add a definition]), are present and can assist in the identification. Figure 1 shows a QAPF diagram (Quartz, Alkali feldspar, Plagioclase, Feldspathoid, also known as a Streckeisen diagram) which is used by the scientific community for the classification of igneous rocks. More information on this can be found here (1) and for granites in general, here (2).

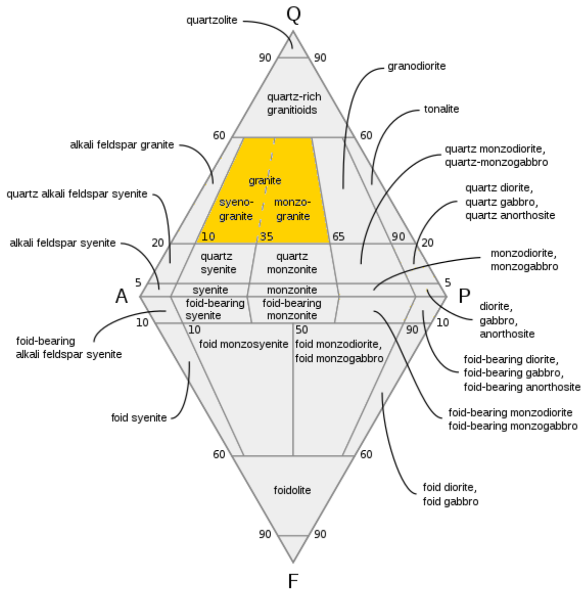

Q
A
P
F
quartzolite
quartz-rich granitoids
granodiorite
tonalite
alkali feldspar granite
granite
syeno-granite
monzo-granite
quartz alkali feldspar syenite
quartz monzodiorite, quartz-monzogabbro
quartz diorite, quartz gabbro, quartz anorthosite
alkali feldspar syenite
quartz syenite
quartz monzonite
monzodiorite, monzogabbro
syenite
monzonite
foid-bearing syenite
foid-bearing monzonite
diorite, gabbro, anorthosite
foid-bearing alkali feldspar syenite
foid monzosyenite
foid monzodiorite, foid monzogabbro
foid-bearing diorite, foid-bearing gabbro, foid-bearing anorthosite
foid-bearing monzodiorite foid-bearing monzogabbro
foid syenite
foidolite
foid diorite, foid gabbro
90
60
20
10
35
65
5
50

Illustration 1: Fig. 1: QAPF diagram with granite field highlighted in yellow (afterKent G. Budge).

Granite maps and masks

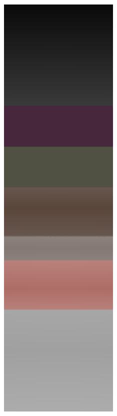
To simulate real world granites in POV-Ray, we make use of colour_maps (i.e. the different minerals/colours composing the granite pigment) and colour masks (i.e. colour_map-based pigment_patterns used within the granite texture) which control the distribution of those same minerals/colours. Necessarily, there is a close relationship between maps and masks as mineral/colour boundaries within a particular granite must corres-pond to their mask boundaries. The final texture is pretty complex, as additional differential scaling perturbes the final aspect of the texture. In the following paragraphs we shall explain how this is achieved by the macro, illustrated by comprehensive examples and diagrams. To make things easier to understand, the different parts of the macro are described separately.
(1) https://en.wikipedia.org/wiki/QAPF_diagram
(2) https://en.wikipedia.org/wiki/Granite

Granite pigment creation
Sections 1 through 3 of the macro concern the creation of granite pigments (Fig. 2). The input to this part of the macro - independently from the different parameters which will be described later - consists of three different colour map arrays (prefixed with A_) for each granite type file:

A_Granite_map1: describing the different minerals/colours;
A_Granite_map2: describing the veins crossing the granites (discussed separately below);
A_Granite_mask_map: describing the (spatial) distribution of the different minerals/colours.

Examples are from the (default) Dakota Red Granite data. ("not_0" is a small value (1/265) which is small enough to render a completely black pixel, but which retains the ability to contribute to the multiplicative result of color and intensity calculations resulting from strong illumination, radiosity, etc.)
A_Granite_map1 is a mixed array and has the following format:

```
#declare Map1_entries = 18;
#declare A_Granite_map1 =
array mixed
[Map1_entries][2]
{
{0.00, <not_0, not_0, not_0>},
{0.25, <0.059, 0.059, 0.059>},
{0.25, <0.086, 0.027, 0.059>},
{0.35, <0.086, 0.027, 0.059>},
/...snip.../
{0.63, <0.769, 0.329, 0.298>},
{0.70, <0.669, 0.229, 0.198>},
{0.75, <0.769, 0.329, 0.298>},
{0.75, <0.600, 0.600, 0.600>},
{0.85, <0.550, 0.550, 0.550>},
{1.00, <0.650, 0.650, 0.650>}
}
```



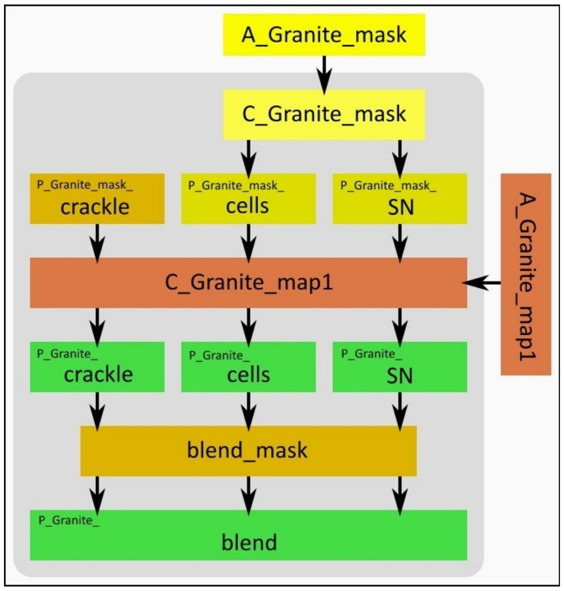


Illustration 2: Fig. 2: Pigment generation (sections 1 to 3).

I think I would like to see a visual representation of the color map to more clearly illustrate the blended and discrete regions of color. Maybe render a simple 1-unit high box with a gradient y and this color_map, as shown.

Since granites consist of close-packed, discrete assemblages of minerals, this can be simulated by sharply-bounded groups of colours, with each group representing a different mineral. Groups/minerals are graphically shown here separated by an empty line. The array is read into the macro's *C_Granite_map1* colour map, which is then incorporated into one of the granite pigment patterns: crackle, cells, step noise (SN). These patterns are generated by the macro's *C_Granite_mask* colour map, which is in turn fed by the *A_Granite_mask* mixed array. The structure of the mask array is similar to that of the color_map array:

```
#declare Mask_entries = 14;
#declare A_Granite_mask =
array mixed
[Mask_entries][2]
{
{0.00, <not_0, not_0, not_0>},
{0.25, <not_0, not_0, not_0>},

{0.25, <0.100, 0.100, 0.100>},
{0.35, <0.100, 0.100, 0.100>},
/...snip.../
{0.63, <0.700, 0.700, 0.700>},
{0.75, <0.700, 0.700, 0.700>},
{0.75, <0.900, 0.900, 0.900>},
{1.00, <0.900, 0.900, 0.900>}
}
```

Again, maybe render a simple 1-unit high box...
Note that the mask map does not necessarily have the same number of entries as the colour map, but the entries representing the boundaries between each discrete group of minerals (also separated by an empty line) necessarily have identical values. This could be explained a bit better. The map index values? Note also that the crackle pattern is a stock POV-Ray pigment pattern, and its parameters are not controlled with macro arguments in this implementation.
Blend (developed originally by Tekno Frannansa - aka “Tek”), the second pattern in the macro (blend_mask, Fig. 2) takes the generated pigments and provides a blended version of them. This can be very useful to simulate a weathered aspect of the granite.
A_Granite_map2, which controls the apperance of the (quartz) veins intersecting a given granite, is a mixed array with the following format:

```
#declare Map2_entries = 5;
#declare A_Granite_map2 =
array mixed
[Map2_entries][3]
{
{0.000, <0.800, 0.800, 0.800>, 0.150},
{0.005, <0.800, 0.800, 0.800>, 0.000},
{0.010, <0.800, 0.800, 0.800>, 0.150},
{0.011, <1.000, 1.000, 1.000>, 1.000},
{1.000, <1.000, 1.000, 1.000>, 1.000}
}
```

This array describes the colour map of the quartz. It is a bit different from the minerals colour map in that it also contains transmit information. When enabled, the vein texture will be layered over the main granite texture. Remember that this feature is still in an experimental phase. It is advised to not change anything to this array, and to copy it verbatim from one granite include file to the next. Things may change in the future though.
Perhaps you should add #warning “This texture uses veins which is an experimental feature….”
In section 4 of the macro, the data from this array are fed into the internal *C_Granite_map2* colour map, which in turn is the core of the *P_Granite_veins* pigment that employs a marble pattern.

Granite texture creation

In section 5 of the macro (Fig. 3), the texture generation, we employ the same pigment patterns used in section 3 (Fig. 2). However, they are implemented for a different reason, and in a different way. This time they control the distribution of different scales of the granite pigments throughout the granite texture. An array (*A_Granite_var*) serves as input for the pigment map. It has the following format:

```
#declare Var_entries = 8;
#declare A_Granite_var =
array [Var_entries][3]
{
{0.20, 0.50, 0.15},
{0.25, 1.00, 0.18},
{0.35, 1.00, 0.18},
{0.40, 0.50, 0.15},
{0.60, 1.00, 0.15},
{0.65, 0.50, 0.18},
{0.75, 1.00, 0.18},
{0.80, 0.50, 0.15}
}
```

The first elements are the pigment map entries; the second are used in the normal map to control the strength of the granite normal pattern (frosted version of the granite); the third elements uniformly scale both the pigments in the pigment map and the normal patterns in the normal map.

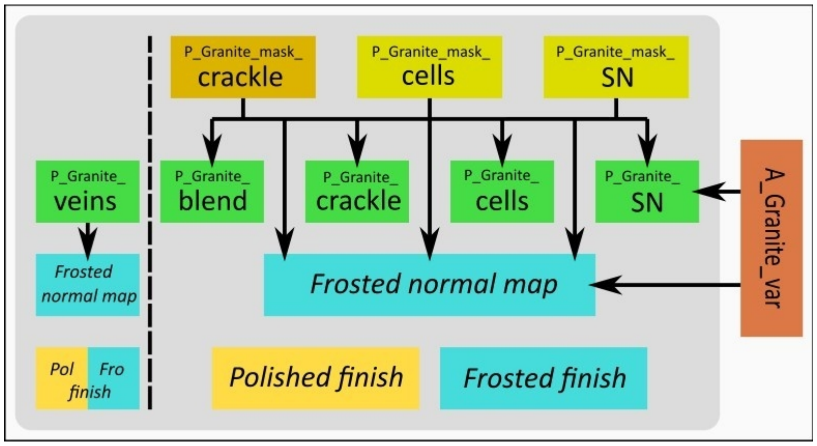


Illustration 3: Fig 3: Texture generation (section 5).

Also in this section, different finish blocks are generated, corresponding to either the polished or the frosted versions of the granite, and whether or not the granite contains veins (layered texture). Finally, the normal block for the frosted version of the granite is generated.

The veins texture (Fig. 3) has its own finish blocks (polished or frosted), and a normal block where the normal map is controlled by the same pigment (*P_Granite_veins*) used in the pigment pattern.

Still experimental is the use of subsurface light transmission. The macro parameter SubS switches this on in this section's finish blocks. [to be developed further]

Again, maybe add #warning “This texture uses veins which is an experimental feature….”

The granite textures have been structured and scaled in such a way that they correspond to a fairly fine-grained real-world granite. This means applying this texture to an arbitrarily-sized object without any further scaling will show a real-world-scale granite texture. One POV-unit corresponds to 100 cm. The user is free to resize the texture via the macro parameter *M_scale*, which defaults to <1,1,1>.

Perhaps show a code example using M_scale to change the texture

Granite material creation

In section 6 of the macro, the polished or frosted granite material blocks are generated, with or without veins. Additionally and finally, the user has the liberty to apply any transformation to the material.

Example?

Really very nicely organized diagrams that you have. Some of the sentences were a bit wordy, so I condensed them for clarity.

Other than that, assume to some extent that the paper may be read “in a vacuum”. Terms or abbreviations that get used should be defined/explained/given context the first time they are used, and then used freely after that. Provide enough information for the user to understand anything presented and that can easily and brieflyaccomplished – anything outside the scope of the paper can be referenced for “further reading”.

Now, you have
 scale M_scale
 rotate M_rotat
 translate M_trans
Are these really necessary in the macro?
They can just as easily be applied once the texture is defined, unless the point you want to make is that adjusting these attributes inside the macro allows memory savings. (I have NO idea how much memory any given texture takes up)

I think that an annotated diagram of the final texture should accompany this document, and perhaps some specialized renders of the array results – showing how the texture elements correspond to the array entries, especially the mask. Maybe in a similar way to what I did here
http://news.povray.org/povray.binaries.images/thread/
%3Cweb.5a7774f01a0c102c5cafe28e0%40news.povray.org%3E/

Presumably the page layout will get “prettied up” in further revisions, so not important to address here.

Macro comments:

Line 26: Why not just expand “Typ” to “Type”?
Line 391 – spelling “aligne” → align

color_map gradient scene:

```
// +w128 +h480 +A0.3


#version 3.8;
global_settings{ assumed_gamma 1.0 }


camera {
orthographic
location <0.0, 0.5, -2>
right x
up y
look_at <0, 0.5, 0>
}

light_source { <0, 0.5, -5> rgb 1 }

sky_sphere {pigment {rgb 1}}

#declare not_0 = 1/265;

#declare Map1_entries = 18;
#declare A_Granite_map1 =
array mixed [Map1_entries][2] {
{0.00, <not_0, not_0, not_0>},
{0.25, <0.059, 0.059, 0.059>},

{0.25, <0.086, 0.027, 0.059>},
{0.35, <0.086, 0.027, 0.059>},

{0.35, <0.118, 0.118, 0.078>},
{0.45, <0.118, 0.118, 0.078>},

{0.45, <0.200, 0.137, 0.110>},
{0.50, <0.150, 0.087, 0.060>},
{0.57, <0.200, 0.137, 0.110>},

{0.57, <0.400, 0.337, 0.310>},
{0.60, <0.350, 0.287, 0.260>},
{0.63, <0.400, 0.337, 0.310>},

{0.63, <0.769, 0.329, 0.298>},
{0.70, <0.669, 0.229, 0.198>},
{0.75, <0.769, 0.329, 0.298>},

{0.75, <0.600, 0.600, 0.600>},
{0.85, <0.550, 0.550, 0.550>},
{1.00, <0.650, 0.650, 0.650>}
}
```

```
#local C_Granite_map1 = //the granite colour_map proper
color_map {
#if (version >= 3.8)
blend_mode 2
#end

#for (J, 0, Map1_entries-1, 1)
#local Entry = A_Granite_map1[J][1];
[A_Granite_map1[J][0] rgb Entry]
#end
}

#declare Pigment = pigment {gradient -y color_map {C_Granite_map1}}

#declare W = 1;
box {<-W, 0, 0>, <W, 1, 0.1> pigment {Pigment}}
```