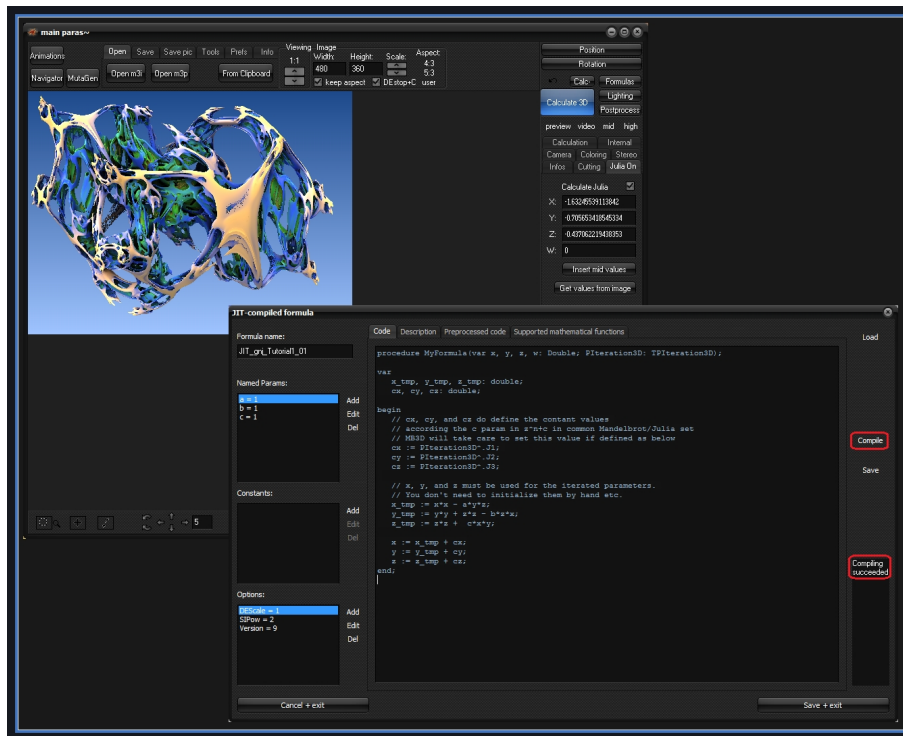


Mandelbulb3D v1.9.0 – Very First Experiences with the new Just-In-Time Compiler and the Formula Editor

by gannjondal
gannjondal.deviantart.com



This tutorial is thought for people with at least some basic experience in using Mandelbulb3D. However, very specific knowledge about any exotic settings, or features is not necessary.

Thanks to the work of Andreas Maschke alias thargor6, and some other people, as well as some thorough discussions on fractalforums.com within the last months we have now again some major move in the development of Mandelbulb3D.

One of the new features is the just-in-time compiler, and the attached formula editor. I want to share my very first experiences in using this new feature.

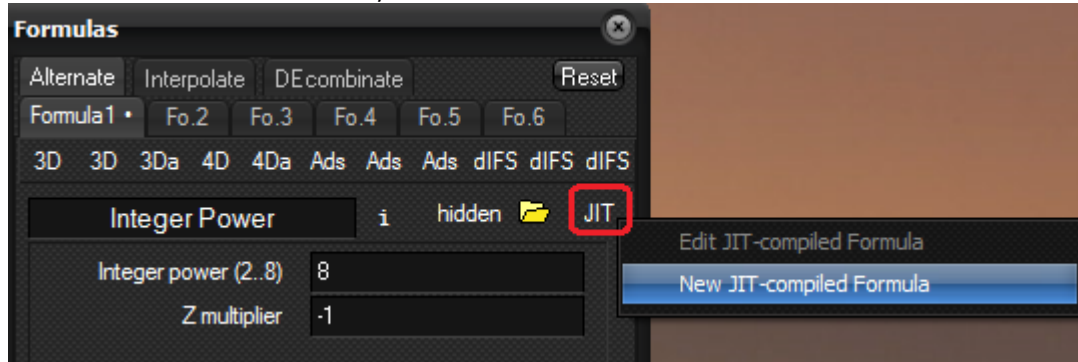
Please don't expect too much. I have decided not to use the triplex numbers for this first tutorial.

I know, this type of numbers allow to create more elaborate forms, especially details. However, to use them in a correct way it needs some special mathematical knowledge. Please review the old discussions on fractalforums.com if you are interested.

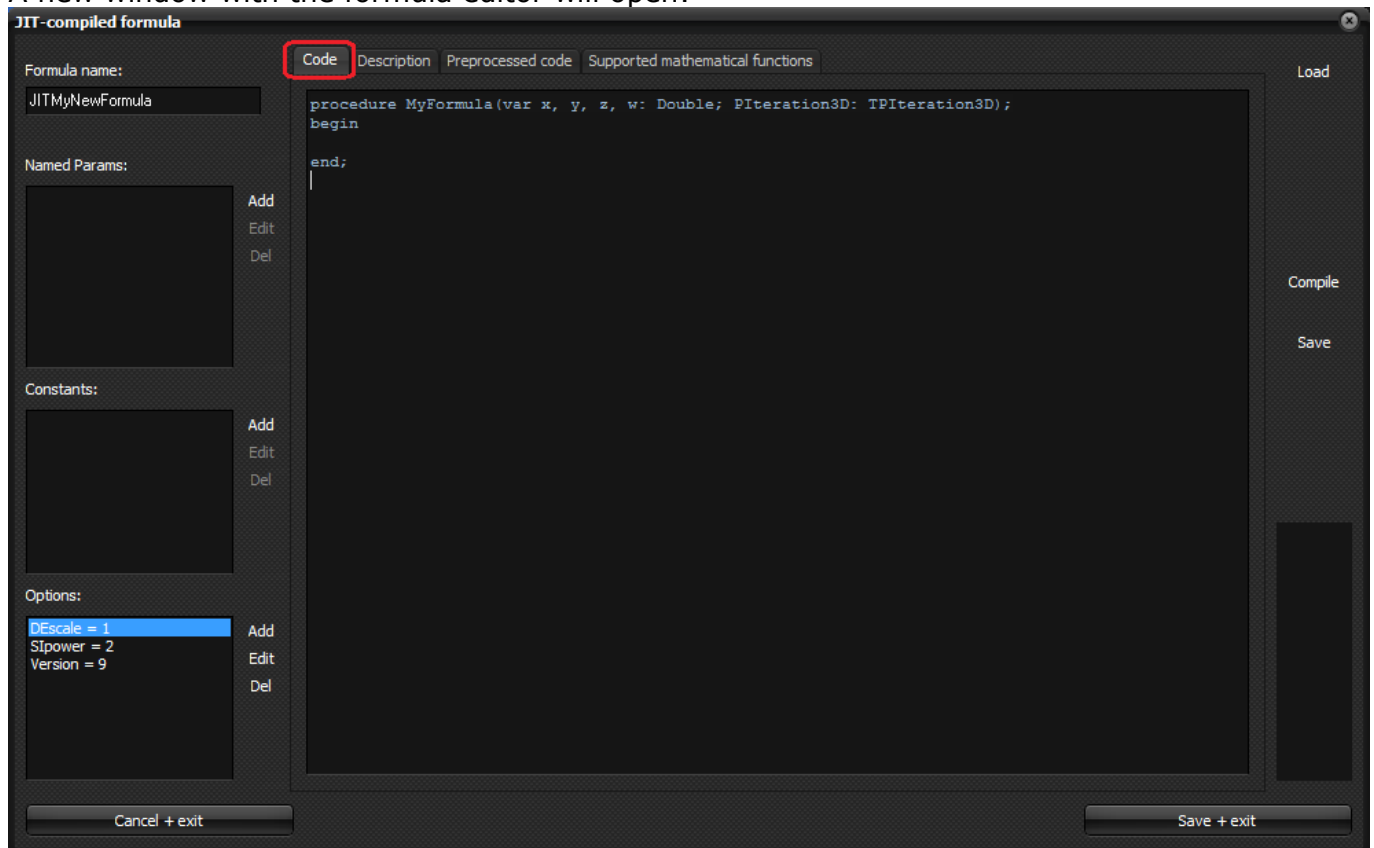
Ok, let's start.

1. First Steps

Start the new MB3D 1.9.0, and switch to the formula window:



A new window with the formula editor will open:

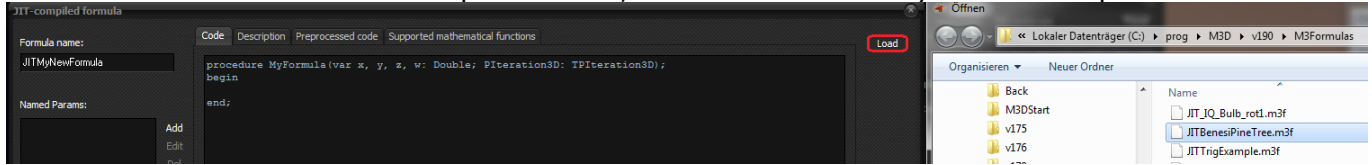


2. Create the Formula

One option to follow up would be to load one of the sample JIT formulas which are shipped with the program.

The JIT formulas are all named JIT*.m3f . Only these formulas can be opened, because JIT formulas are stored almost in source code while all other formulas contain machine code.

I have decided not to use a sample formula, but I want to show you how to open:



a) Copy the formula to the code window:

Instead I want to start with a self-made sample program.

Please ensure that you have opened the 'code' tab, and copy the below code into the main window:

```
procedure MyFormula(var x, y, z, w: Double; PIteration3D: TPIteration3D);

var
    // announce the variables you want to use below
    x_tmp, y_tmp, z_tmp: double;
    cx, cy, cz: double;

begin
    // cx, cy, and cz do define the constant values
    // according the c param in z^n+c in common Mandelbrot/Julia set
    // MB3D will take care to set this value if defined as below
    cx := PIteration3D^.J1;
    cy := PIteration3D^.J2;
    cz := PIteration3D^.J3;

    // x, y, and z do represent the iterated variable.
    // You don't need to initialize them by hand etc. Just read and write them as you want.
    x_tmp := x*x - a*y*z;
    y_tmp := y*y + z*z - b*z*x;
    z_tmp := z*z + c*x*y;

    x := x_tmp + cx;
    y := y_tmp + cy;
    z := z_tmp + cz;
end;
```

A few notes:

- x, y, and z are reserved values which contain the iterated variable
- You could iterate w as well to calculate with 4-dimensional numbers.
Note: Even if you don't use w you need to leave it in the procedure definition.
- The `TPIteration3D` type does refer to a record which defines a larger set of parameters, and properties used internally within MB3D. At least some of them can be used here in the formula editor.
I'm not programmer enough to tell you (correct) details. I think I know where I can find the type definitions, and some explanations, but I don't want to tell you wrong things. Check fractalforums.com, or ask Andreas (or a few other of the programming gurus) if you want to know more.

- $P_{iteration3D}^{J1} - J3$ are at least 3 of those parameters defined within `TPIteration3D` which you really need. They represent the c in $z^n + c$ as of the common Mandelbrot / Julia set.
For a better readability I have mapped them to the newly defined variables cx , cy , and cz
- The 'Constants' section can be used if you need a value again and again in your formula without recalculating while each iteration; for instance `rooteight=2.8284271247`
- I cannot tell much about the 'Options' sections right now. I have seen them in the type definitions – but without comments. Feel free to experiment.
- My formula above has no mathematical meaning.
It's just simple enough to be relatively fast – and the resulting forms are interesting enough (similar to PseudoXDB).

b) Change the name of the formula:

You will also need to change the name of the formula (see the red marks in the screenshot below).

For this example I choosed the name `JIT_gnj_Tutorial1_01` as of below name schema:

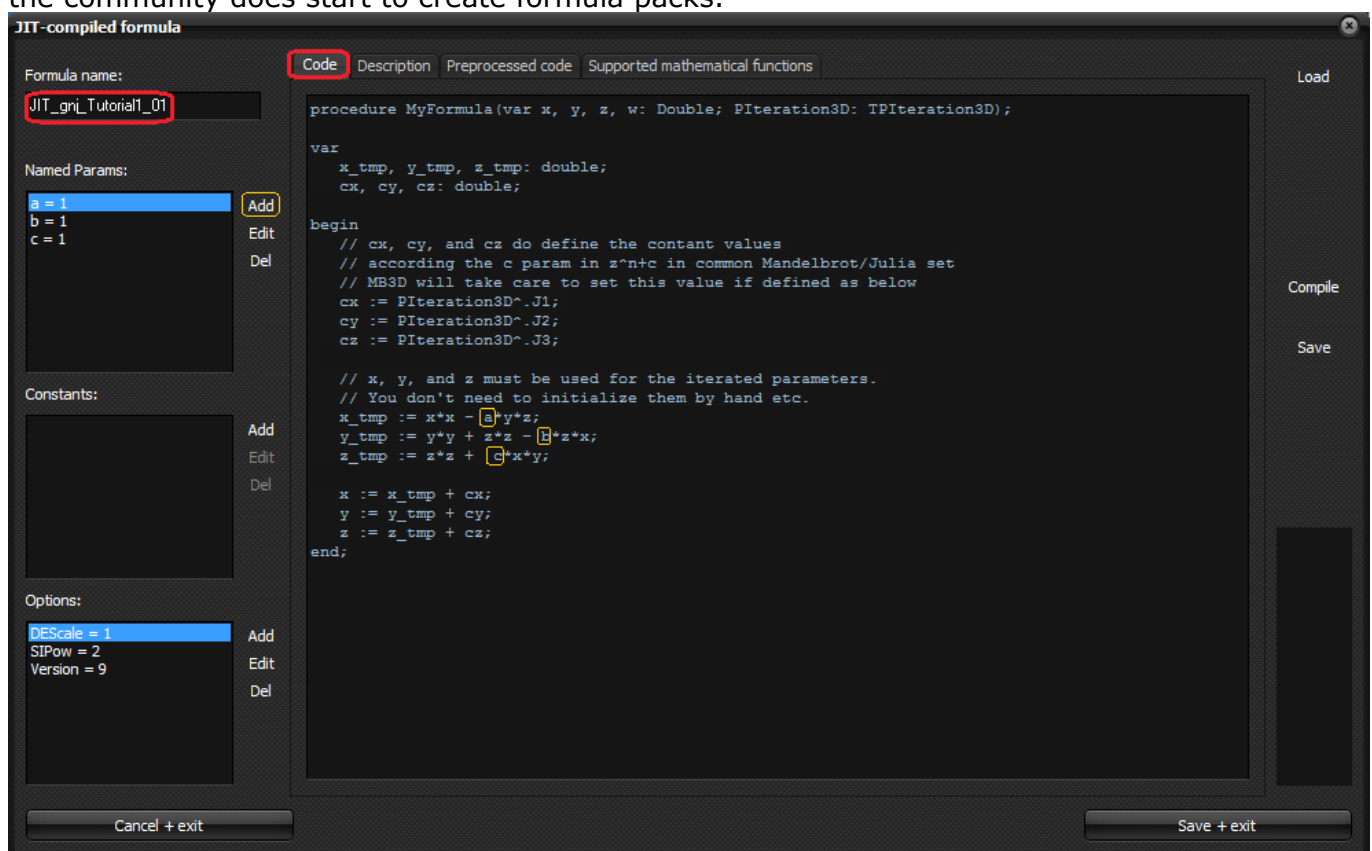
JIT – a mark for all JIT formulas (as in the sample formulas)

gnj – 3 letter acronym like in UF which cover the name of the author (gnj -> gannjondal)

Tutorial1 – Just a name for the formula

01 – a version number. One of the less perfect things of old M3D formulas is the almost non-existing versioning which caused me to create a dozend of program directories containing different formula packs.

That's just an example how name schema could look like – but I think we need one as soon as the community does start to create formula packs.

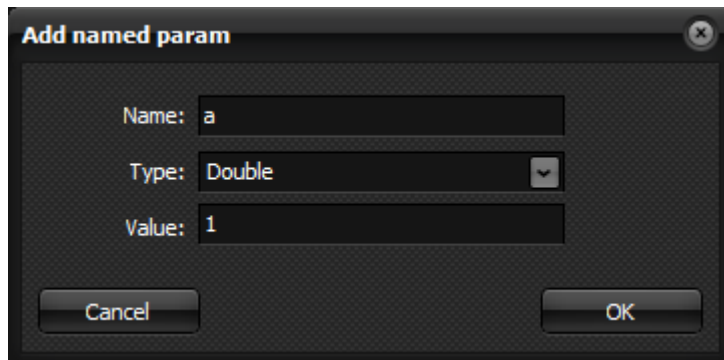


c) Add the formula parameters, and set the default values:

Now define the formula parameters which will be shown later at the formula tab.

The yellow marks above show how to define them:

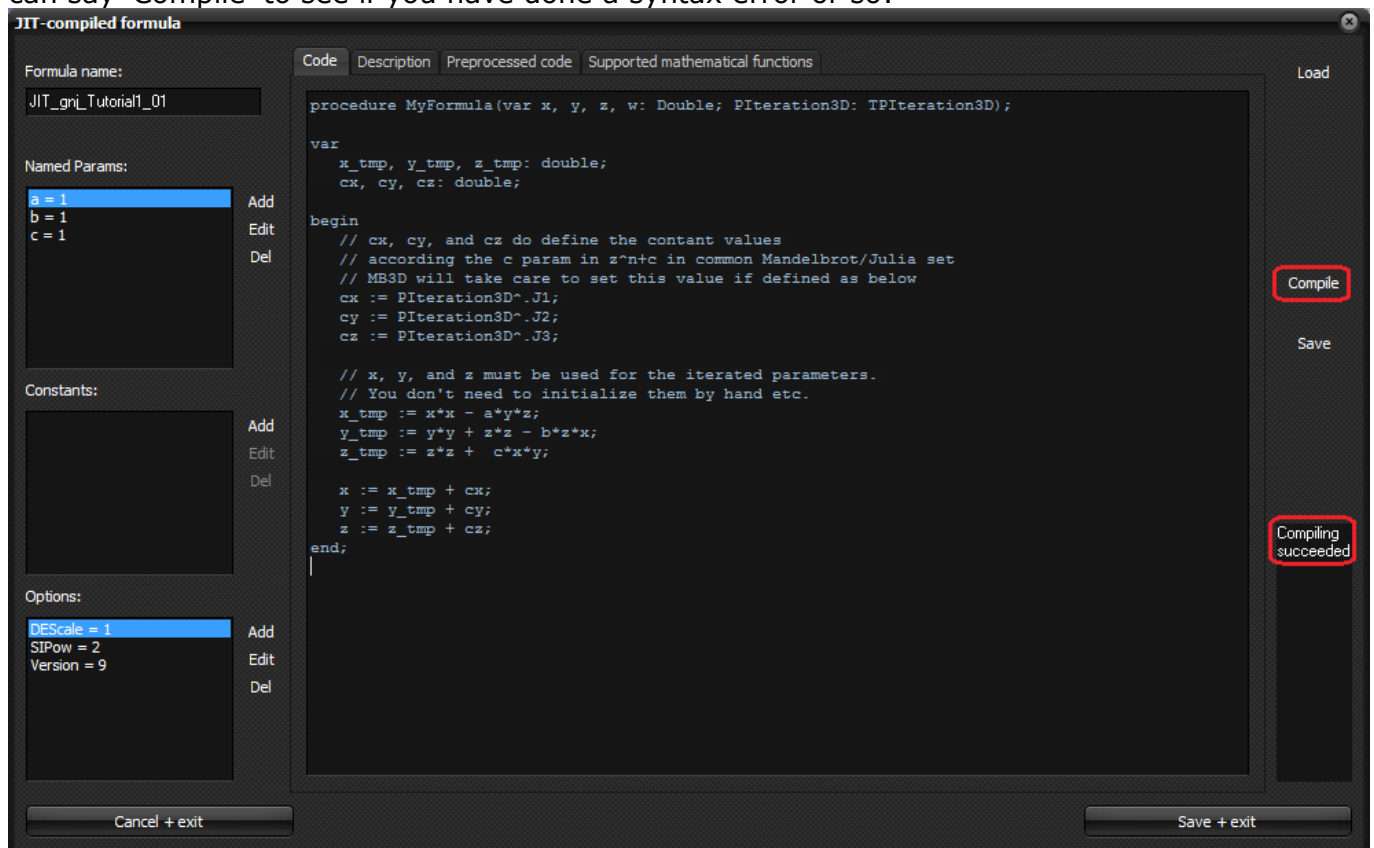
Click 'Add' besides the 'Named param' section - The below window will be opened.
Set the name of the parameter, choose the correct type, and set the value field which will be the default value of the parameter.



The dialog box titled "Add named param" contains three input fields: "Name:" with the value "a", "Type:" with a dropdown menu showing "Double", and "Value:" with the value "1". At the bottom, there are two buttons: "Cancel" and "OK".

d) Last step on actual formula definition:

After the three named parameters have been added the major work is done already – and you can say 'Compile' to see if you have done a syntax error or so:



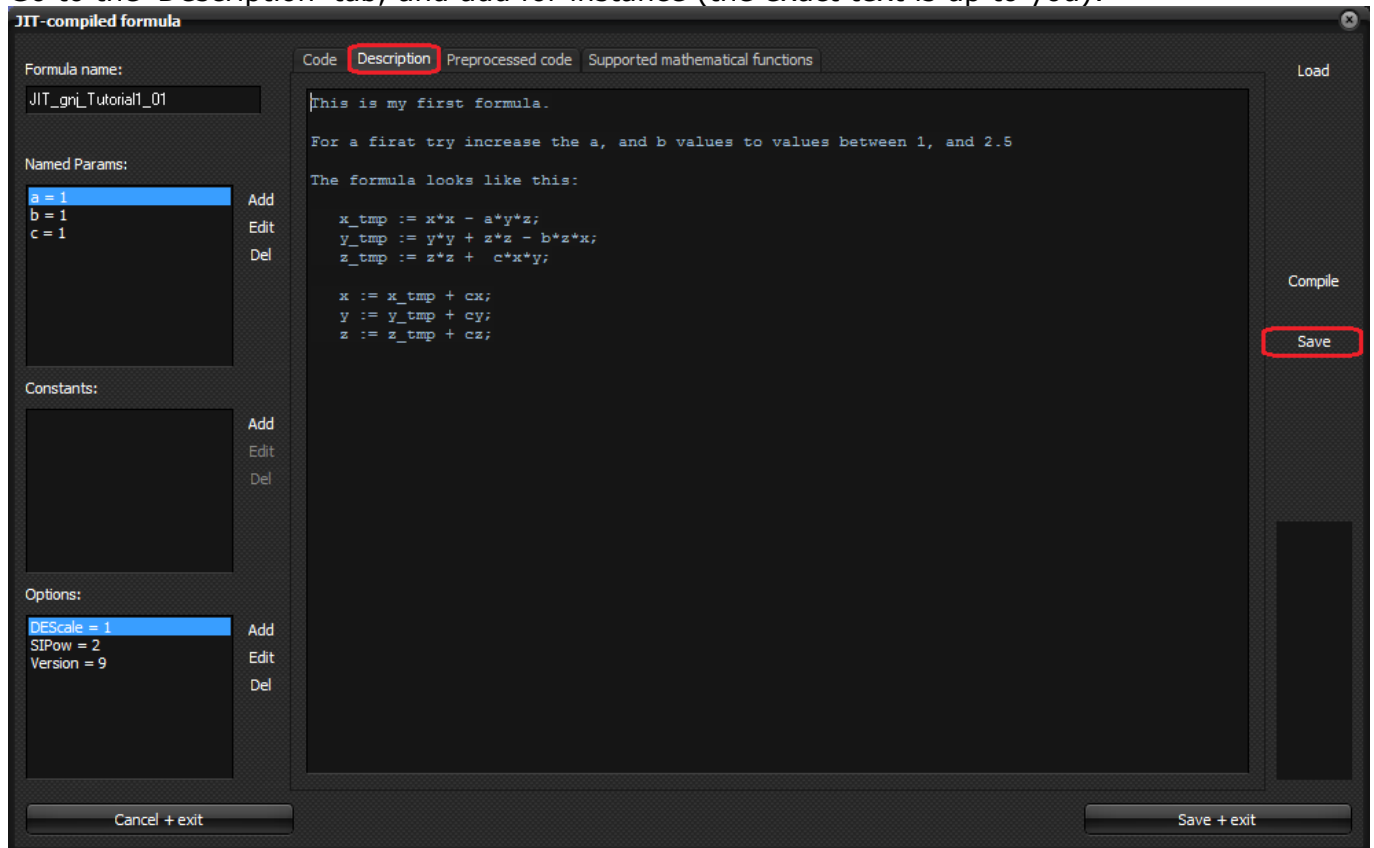
The "JIT-compiled formula" window has a tabbed interface with "Code" selected. The left sidebar contains "Named Params" (a=1, b=1, c=1) and "Options" (DEScale=1, SIPow=2, Version=9). The main code editor shows a procedure definition for a Mandelbrot set iteration. On the right, there are buttons for "Compile", "Save", and "Compiling succeeded".

```
procedure MyFormula(var x, y, z, w: Double; PIteration3D: TPIteration3D);  
  
var  
  x_tmp, y_tmp, z_tmp: double;  
  cx, cy, cz: double;  
  
begin  
  // cx, cy, and cz do define the constant values  
  // according the c param in z^n+c in common Mandelbrot/Julia set  
  // MB3D will take care to set this value if defined as below  
  cx := PIteration3D.J1;  
  cy := PIteration3D.J2;  
  cz := PIteration3D.J3;  
  
  // x, y, and z must be used for the iterated parameters.  
  // You don't need to initialize them by hand etc.  
  x_tmp := x*x - a*y*z;  
  y_tmp := y*y + z*z - b*z*x;  
  z_tmp := z*z + c*x*y;  
  
  x := x_tmp + cx;  
  y := y_tmp + cy;  
  z := z_tmp + cz;  
end;
```

3. Add a Description

Now a well defined formula should contain a valid description.

Go to the 'Description' tab, and add for instance (the exact text is up to you):



You will finally find this text if you press the 'i' button at the formula tab finally.

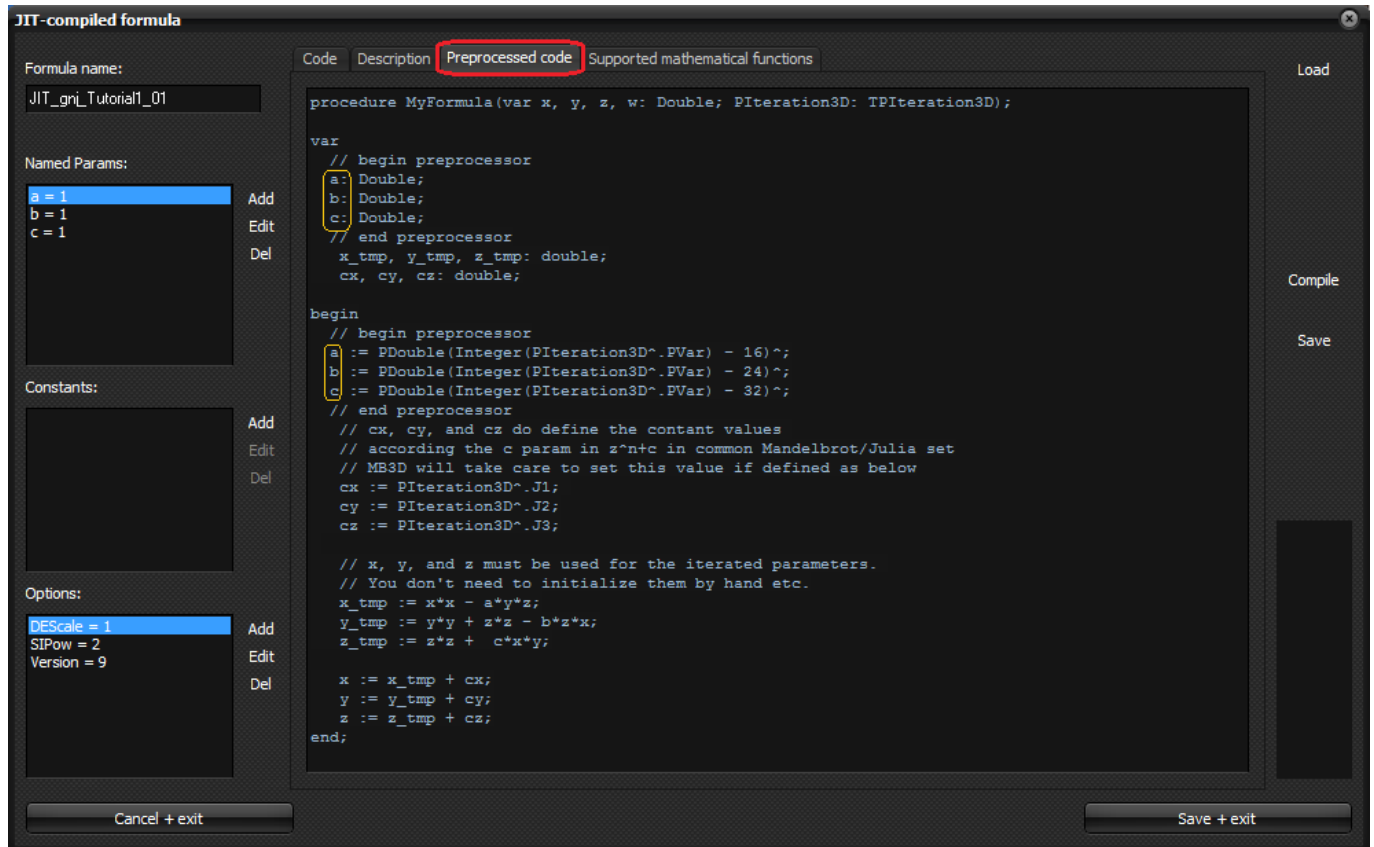
You can now press 'Save' (marked red). The formula will be saved to your configured formula directory as [Formula name].m3f .

After you have closed the formula editor it should be available in the formula choice lists, sorted alphabetically. If that should not be the case you can load the formula manually, or of course restart MB3D.

4. Informational Windows

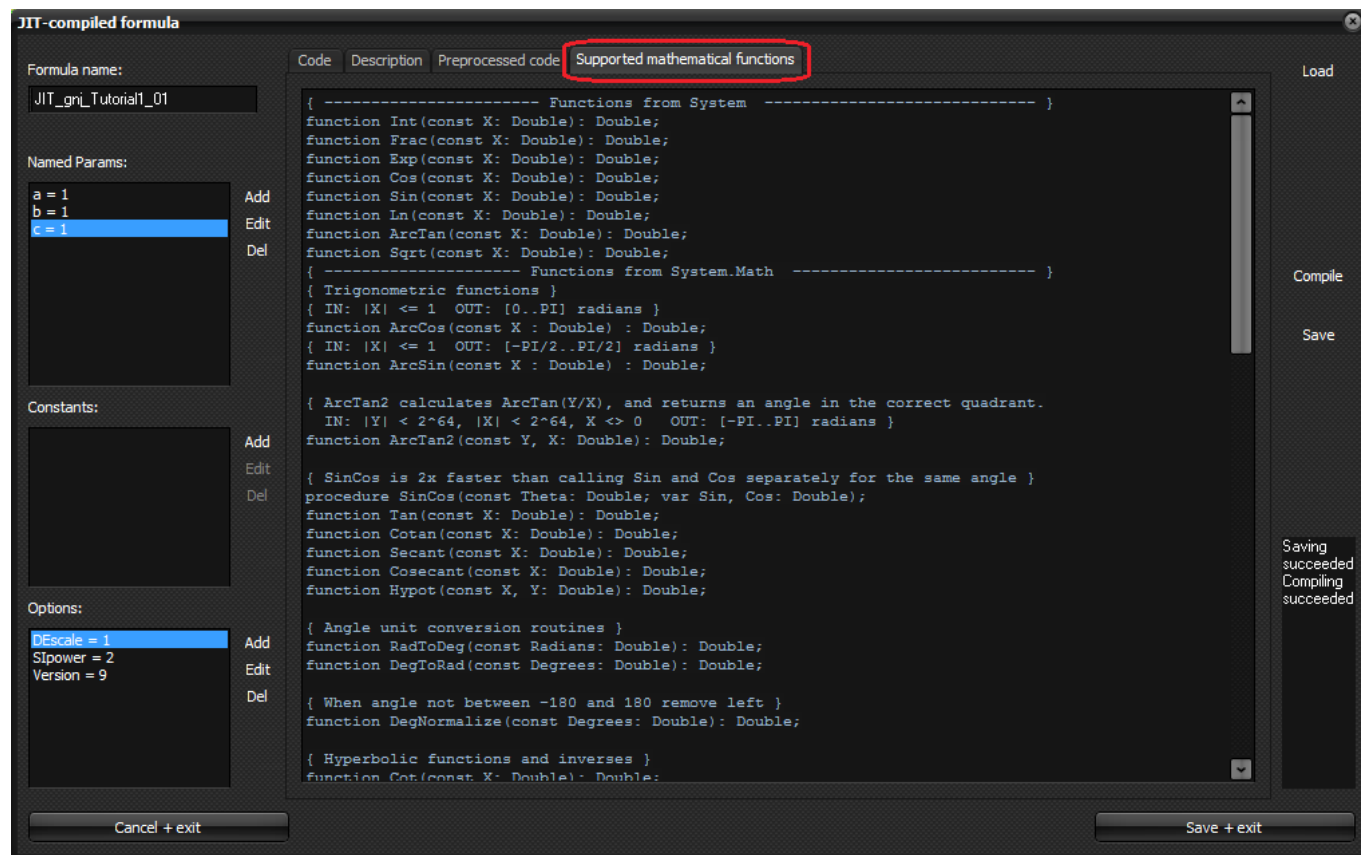
The other two tabs in editor window ('Preprocessed code', and 'Supported mathematical functions') are mainly informative.

The window 'Preprocessed code' does show the same code you have added under the 'Code' tab plus the internal used definitions for named params (yellow marks), and constants. This code gets filled internally; so far I see there's nothing to change for you. Probably good news for everyone who is (like me) not a Delphi programmer. Pointers to content of packed records as at the second yellow mark are really ... specific (or ancient? - Sorry :-))



The window **'Supported mathematical functions'** does list all the functions you can use in your code plus some explanation.

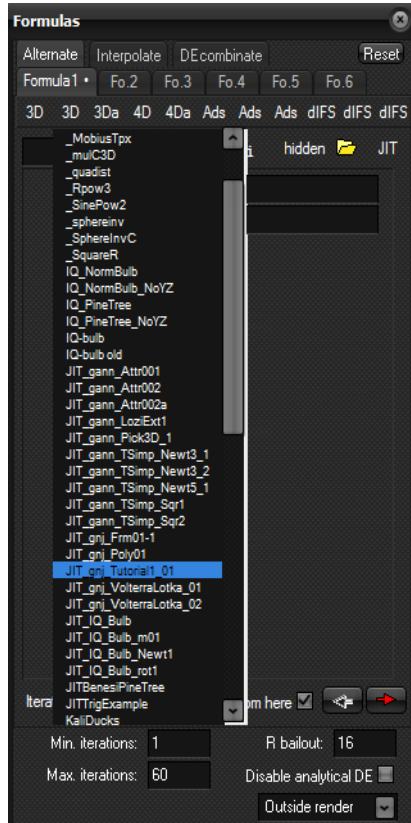
Note: This list does currently not contain a function for the absolute value of a real number, but `abs()` seems to work.



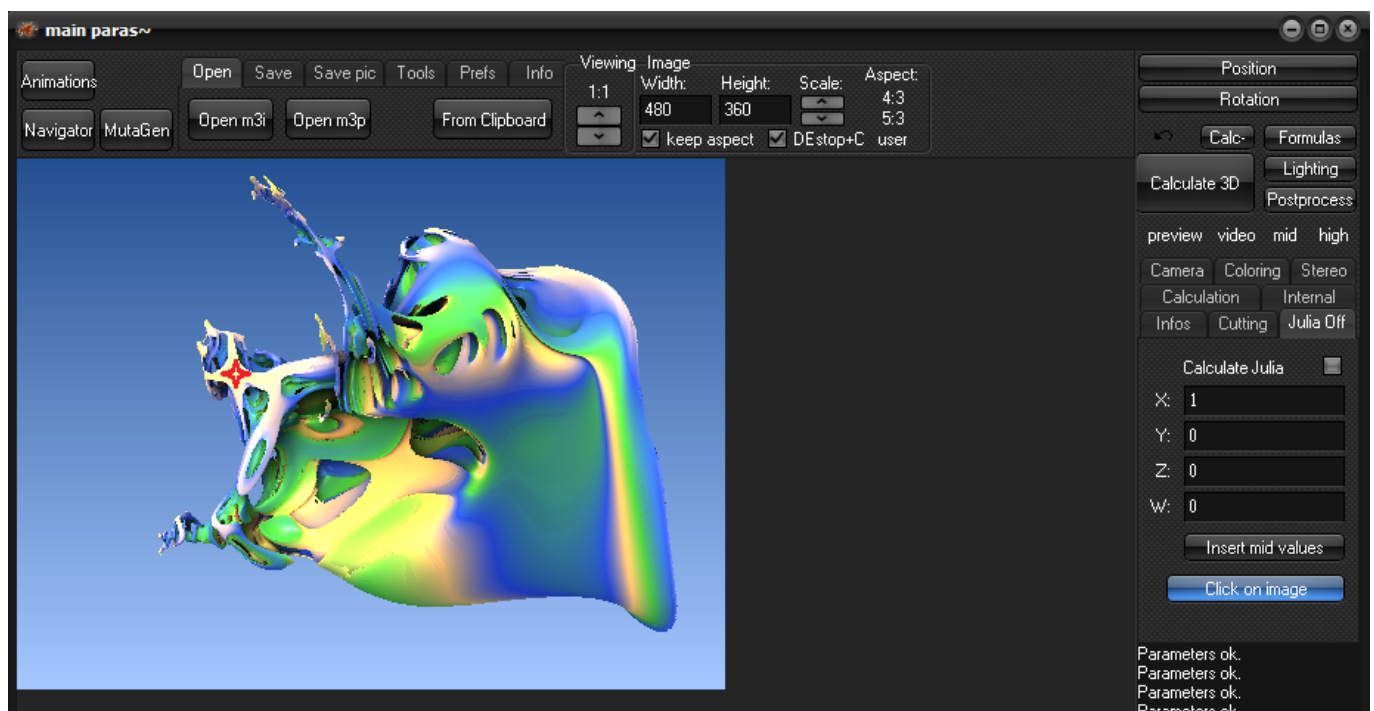
Now you can close the formula editor already!

5. Test the formula

As mentioned the new formula should now be available in the formula choice window:



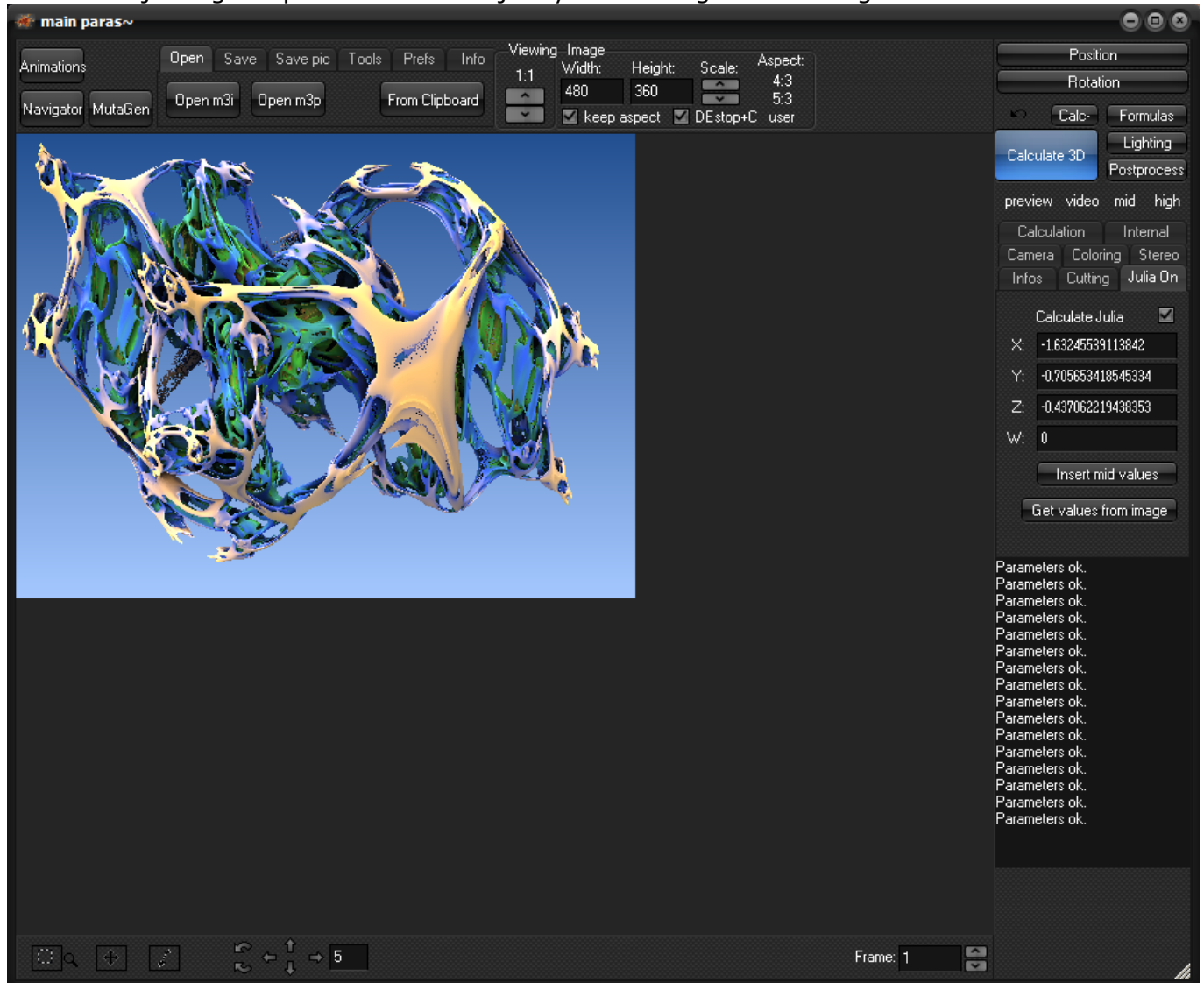
To get below window set the formula parameter to about 0.725, and b to 2.5, and optimize the position of the figure, for instance using the Navigator (the values I have used for the screenshot must have been slightly different – but the picture is similar enough).



6. Check the Julia ability

Go to the Julia tab of the MB3D main window, click on 'Get values from image', choose the point which is marked red above, and calculate again.

After re-adjusting the position of the object you should get something like:



I guess that's nice enough for a first trial.

Have fun!