Computer Generated Images of Positive Axes Systems in 3-Space

Robert Duncan

I've written many programs to explore positive axes systems, some of them are dedicated to one task, such as imaging the track of the position of an address at each step of the iteration formula, others are purely numerical, giving a table of values. Many programs started with a simple objective and evolved into a general tool for investigation of a range of characteristics of the 3D self-similar structures in complex 3-space. As you can image displaying a very strange three dimensional object on a flat computer screen is a bit of a challenge. To actually display such a structure in a manner like Blender does with rotations, translations, and scaling, would require far greater computer and display equipment than I have available.

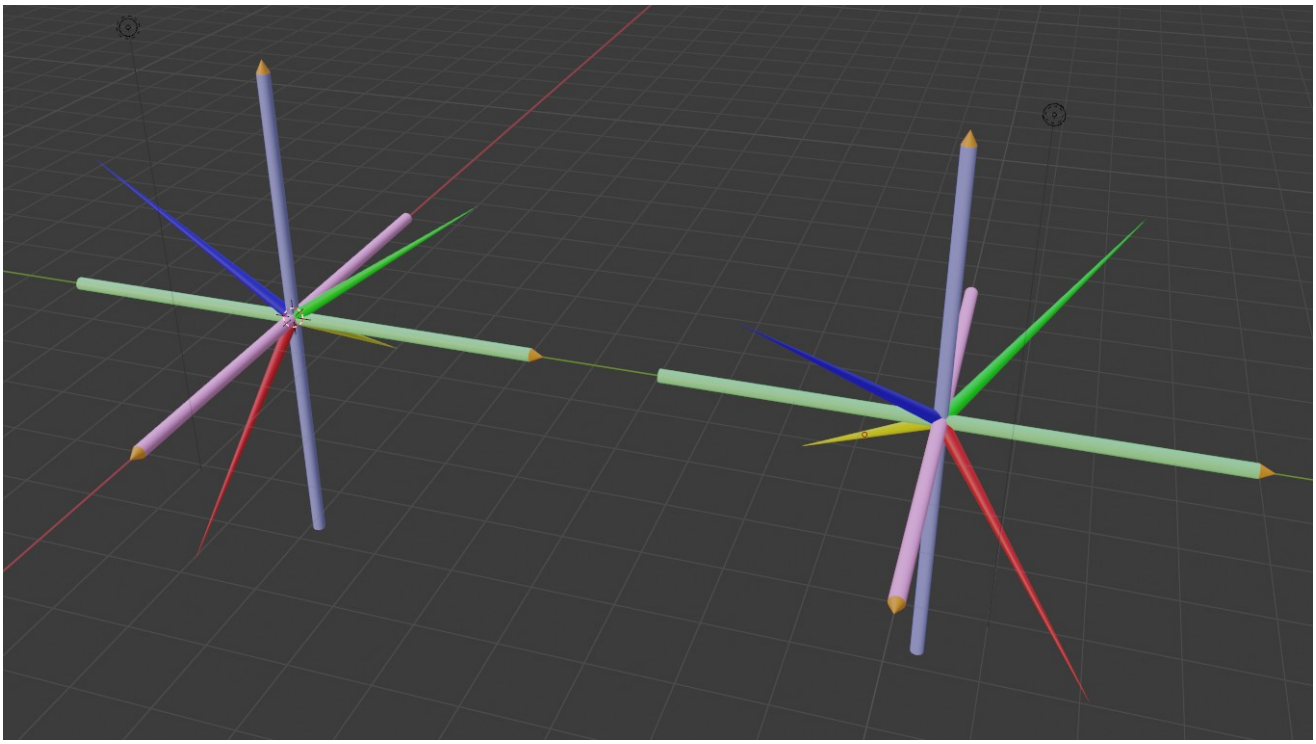
The 3D structure is, of course, made up of those points in 3-space that are 'in the set'. In the set means that no matter how many times the iteration formula is iterated for that point's address, the address stays finite. The points in the set generally, but not always, form an object that is connected and is a solid. Although programmatically we must select discrete points to iterate, any point within the boundary of the set is in the set. The 'boundary' is fractal and self similar, and therefore rather difficult to define.

However, methods exist to give some idea of the shape of a 3D object, and, if the programmatic parameters are selected carefully, a mental image can be built. This paper examines one particular program to explain the images produced and how to interpret them.

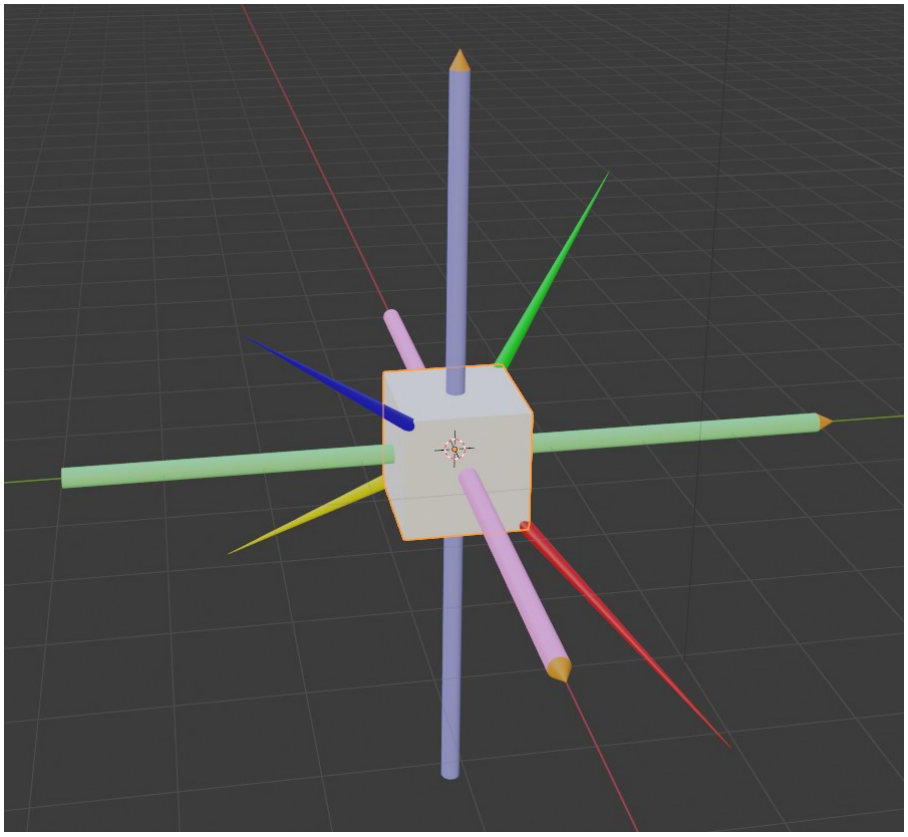
The program effectively takes a 2D slice through the 3D structure and displays that slice. The 'four positive axes complex 3-space address' of each point contained on the plane of that slice is iterated using the selected formula. Defining the slice in the positive coordinate system was very difficult, as dot and cross products as well as trigonometric functions are not yet defined. Early programs tried to allow arbitrarily placed and angled slices, but this became very cumbersome to program. After much analysis of the images, a simpler program evolved based on the standard Cartesian 3-space coordinate system. The slice was restrained to a plane perpendicular to one of the Cartesian axes, (that is, a plane parallel to either the XY, XZ, or YZ planes) at a settable distance out from the origin on that axis. The addresses of the points to be iterated were determined in the Cartesian system. These addresses were then translated into the positive axes coordinate system and iterated. The result of the iteration was just a color indicating if the point stayed within the set (black) or how quickly it 'went to infinity'

(the other colors). The colors are then displayed using the Cartesian addresses of the plane of the slice.

This means that the orientation of the positive axes system against the Cartesian axes system becomes critical as to how the slices penetrated the structure. After much experimentation the best orientation was found to be that shown below on the right and is used throughout these papers. Again, the X axis is pink, with the positive direction coming towards the viewer, Y is green, and Z is lavender. Perhaps a bit easier to understand, the second image below shows both the Cartesian and positive axes systems protruding from a cube.

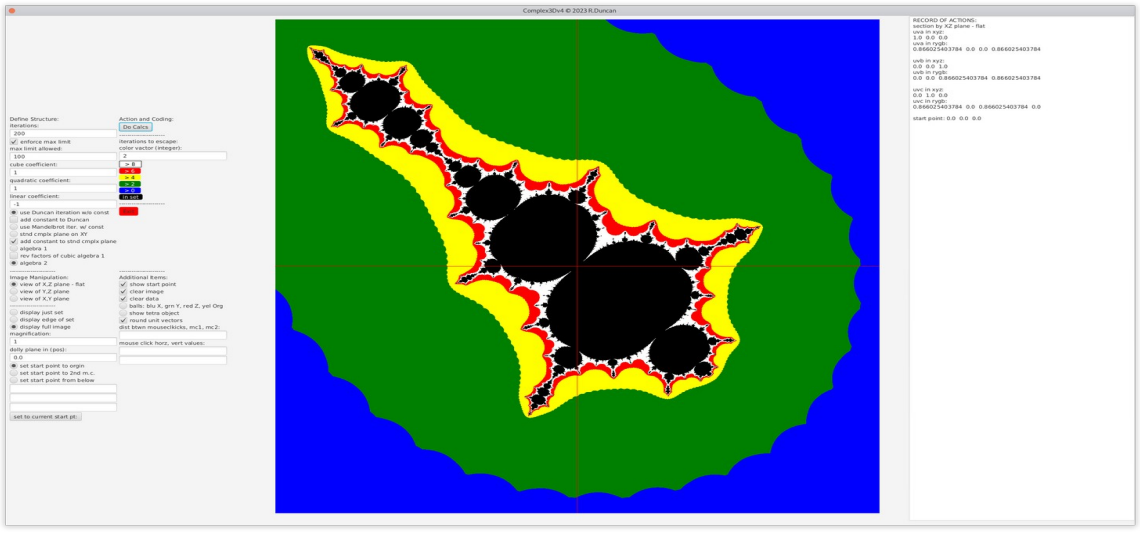


Two possible orientations of both axes systems (Blender)



The preferred orientation of positive to Cartesian axes systems (Blender)

Having decided on the proper orientation almost all the rest of the parameters that can be varied are available to the user at run time. Here is the view of the program's interface:



Program Complex3Dv4 interface

On the next page is a blowup of the parameter settings on the left of the above image. These settings control both how the image is displayed and what the iteration formula and parameters are.

The controls in the left column start with ‘iterations’. This is the number of times the reiterative formula will be iterated. If too low a number is set, many addresses that would have moved away from the set will not have had time to do so and are then, incorrectly, counted as in the set. Too high a value takes the computer longer to run the program without any increase in precision. Most of these structures are fully formed by as low as 20 iterations. I generally use 200 iterations to be safe. Technically, we should let the program run for an infinite number of iterations for each point, which would take too long!

Next is the max limit value and whether or not to apply it. This is a limit on how large the value of each iteration can become before the program decides that it is outside the set. This value may be reached before the 200 iterations. It simply lowers the run time of the program. In the complex plane the Mandelbrot programs usually use 2 for the maximum value; once an iteration is larger than 2 we know the address is not in the Mandelbrot set.

The next three fields describe the iteration formula. The numbers are the coefficients of the cubic, quadratic, and linear terms. If zero is entered that term is not part of the formula. The radio buttons (the selectable circles) and the check boxes (the squares) that follow modify the iteration formula defined by the three coefficients.

The two basic modifications are using the Duncan iteration or the Mandelbrot iteration. The difference is whether or not a constant is added to the formula. The Mandelbrot standard formula is $z_{n+1} = z_n^2 + c$, where c, the constant, is just the starting address. The option to add the constant to the Duncan formula actually equates the two forms.

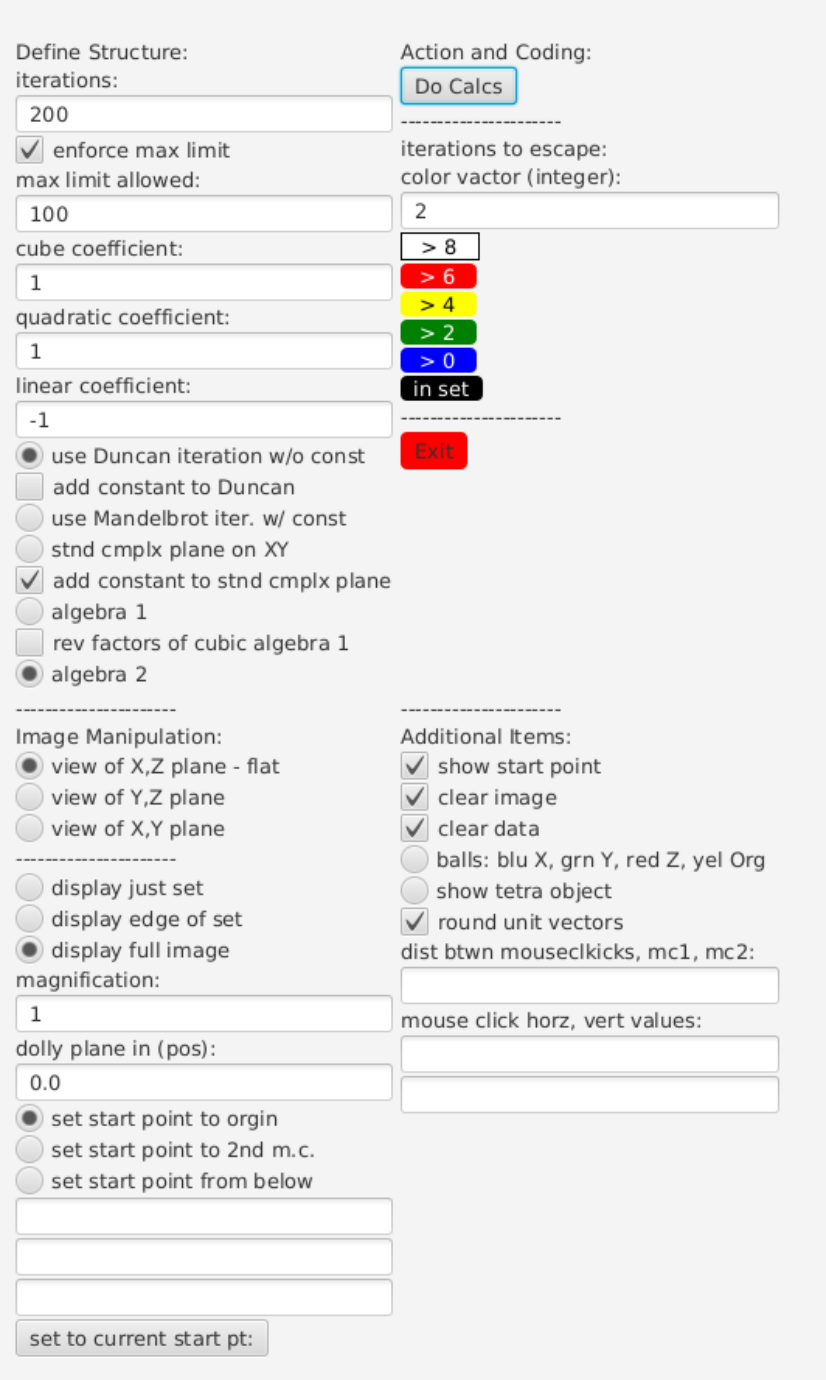
Another radio button allows the program to abandon the 3-space positive 4-axes space for the normal 2-space complex plane. This allows comparisons between what the structures look like in 2D and 3D. This became very useful early on in the exploration of these sets.

There are two different algebras available, selectable by the radio buttons algebra 1 and algebra 2. The check box ‘rev factors of cubic algebra 1’ has to do with how the cubic is handled in a non-commutative algebra, reversing the order of multiplications.

The rest of the left column deals with where in the structure the 2D slice is taken; which plane (XY, XZ, or YZ) and how far from the origin the plane is taken (the dolly value). The center point can be selected with the mouse or by hand and the magnification of the image is settable.

The right column sets the colors outside the set, allows for drawing circles and lines on the images by mouse clicks, and some testing of the program.

This program is the main tool used to explore the 3-space complex positive 4-axes system. The discoveries and remaining mysteries (there are many) are described in the next paper on this web site.



User settable parameters