ray tracing programmes: *ix*

M. Fernández-Guasti 

e-mail: mfg@xanum.uam.mx, url: http://luz.izt.uam.mx

March 25, 2016

1 introduction

It is possible to perform visualization (in 2D) of the three dimensional structure via ray tracing. It is essential to use open source programmes in order to be able to modify them. Furthermore, open source programmes, in addition to being free, are best suited for colaboration and the advancement of human knowledge.

2 programme for *ix*

In order to include the *fractal with imaginary scators under the quadratic iteration*, codenamed *ix*, (pronounced 'eesh') it is necessary to modify the recurrence relationship and the magnitude according to imaginary scator algebra.

The iteration now involves three functions. It should be performed with an efficient and fast language, for example C++. The lines here below outline the recurrence relationships. It would be preferable to use the notation, `s` for the scalar, and `x,y` for the director components:

```
{
        double s2 = z.s * z.s;
        double x2 = z.x * z.x;
        double y2 = z.y * z.y;

        double news = s2 - x2 - y2 + (x2 * y2) / s2;
        double newx = 2.0 * z.s * z.x * (1 - y2 / s2 );
        double newy = 2.0 * z.s * z.y * (1 - x2 / s2 );

        z.s = news;
        z.x = newx;
```

```
        z.y = newy;
}
```

3 modified Mandelbulber to include *ix*

In this example, the Mandelbulber programme v 2.07-1 was used. It is developed by the project leader Krzysztof Marczak and programmers Krzysztof Marczak, Sebastian Jennen, Graeme McLaren, Bernardo Martelli.

- add in Mandelbulber, v 2.07-1file /src/fractal_formulas.cpp

```
/** quadratic iteration in imaginary scator algebra */
void ImaginaryscatorPower2Iteration(CVector3 &z)
{
        double x2 = z.x * z.x;
        double y2 = z.y * z.y;
        double z2 = z.z * z.z;

        double newx = x2 - y2 - z2 + (y2 * z2) / x2;
        double newy = 2.0 * z.x * z.y * (1 - z2 / x2 );
        double newz = 2.0 * z.x * z.z * (1 - y2 / x2 );

        z.x = newx;
        z.y = newy;
        z.z = newz;
}
```

- add in fractal_formulas.hpp

```
void ImaginaryscatorPower2Iteration(CVector3 &z);
```

- add in fractal_list.cpp

```
fractalList->append(sFractalDescription("Imaginary scator Power 2",
                                "imagsca_power_2",
                                fast_imagsca_power2,
                                analyticDEType,
                                logarithmicDEFunction,
                                cpixelEnabledByDefault,
                                10));
```

- add in fractal_list.hpp

```
                fast_imagsca_power2 = 152,
```

- add in compute_fractal.cpp

```
case fast_imagsca_power2:
        {
                ImaginaryscatorPower2Iteration(z);
                break;
        }
```

- also modify and add in line 711

```
// r calculation
// r = sqrt(z.x * z.x + z.y * z.y + z.z * z.z + w * w);
        switch(fractal->formula)
        {
                default:
                {
                        r = sqrt(z.x * z.x + z.y * z.y + z.z * z.z + w * w);
                        break;
                }
                //scator magnitudes
                // magnitude in imaginary scator algebra
                case fast_imagsca_power2:
                {
                        r = sqrt(z.x * z.x + z.y * z.y + z.z * z.z
                                + (z.y * z.y *  z.z * z.z) / (z.x * z.x) );
                        break;
                }
        }
```

- also add in: switch (formula) line 836

```
case fast_imagsca_power2:
```

- add in /usr/share/mandelbulber2/language/ qt_data_en.ts (in two places) just after "../qt_data/fractal_mandelbulb_power_2.ui"

```
<location filename="../qt_data/fractal_imagsca_power_2.ui" line="14"/>

<location filename="../qt_data/fractal_imagsca_power_2.ui" line="20"/>
```

- copy file fractal_mandelbulb_power_2.ui in /usr/share/mandelbulber2/qt_data with name

```
fractal_imagsca_power_2.ui
```

- to compile and install

```
cd makefiles
qmake mandelbulber.pro
make all
cd ..
./install
```

4 settings

The most relevant parameteers are:

1. to stop at maximum iteration
2. to force the Delta DE distance estimation method
3. better, but not esential to have a large bailout number

In the lines here below, typical settings of a `.fract` Mandelbulber file

---

```
# Mandelbulber settings file
# version 2.07
# only modified parameters
[main_parameters]
aux_light_enabled_1 true;
aux_light_intensity_1 1;
aux_light_position_1 1 -3 -2;
bailout 100000;
camera 0.1 -3 0;
camera_distance_to_target 3.001666203960727;
camera_rotation 1.909152432996377 0 0;
camera_top 0 0 1;
delta_DE_method 1;
flight_last_to_render 0;
formula_1 152;
iteration_threshold_mode true;
keyframe_last_to_render 0;
use_default_bailout false;
```