Pretty Pictures:
Simple Mathematics and Computer Graphics

David Malone (TCD)

November 2000

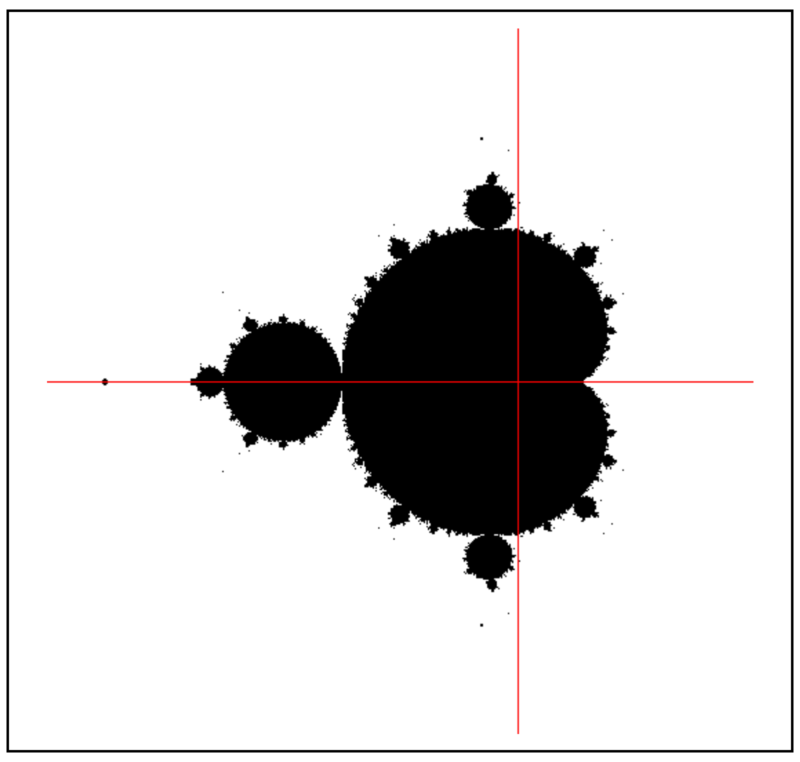
The Plan

To show how a little mathematics can go a long way in computer graphics and produce pretty pictures.

- Mandelbrot Sets,
- Ray Tracing,
- Colours,
- Image Compression.

Mandelbrot Sets

'Most everyone has seen the Mandelbrot set:



It is drawn on the complex plane $\mathbb{C}$ by looking at the sequence:

$$z_{n+1} = z_n^2 + z_0$$

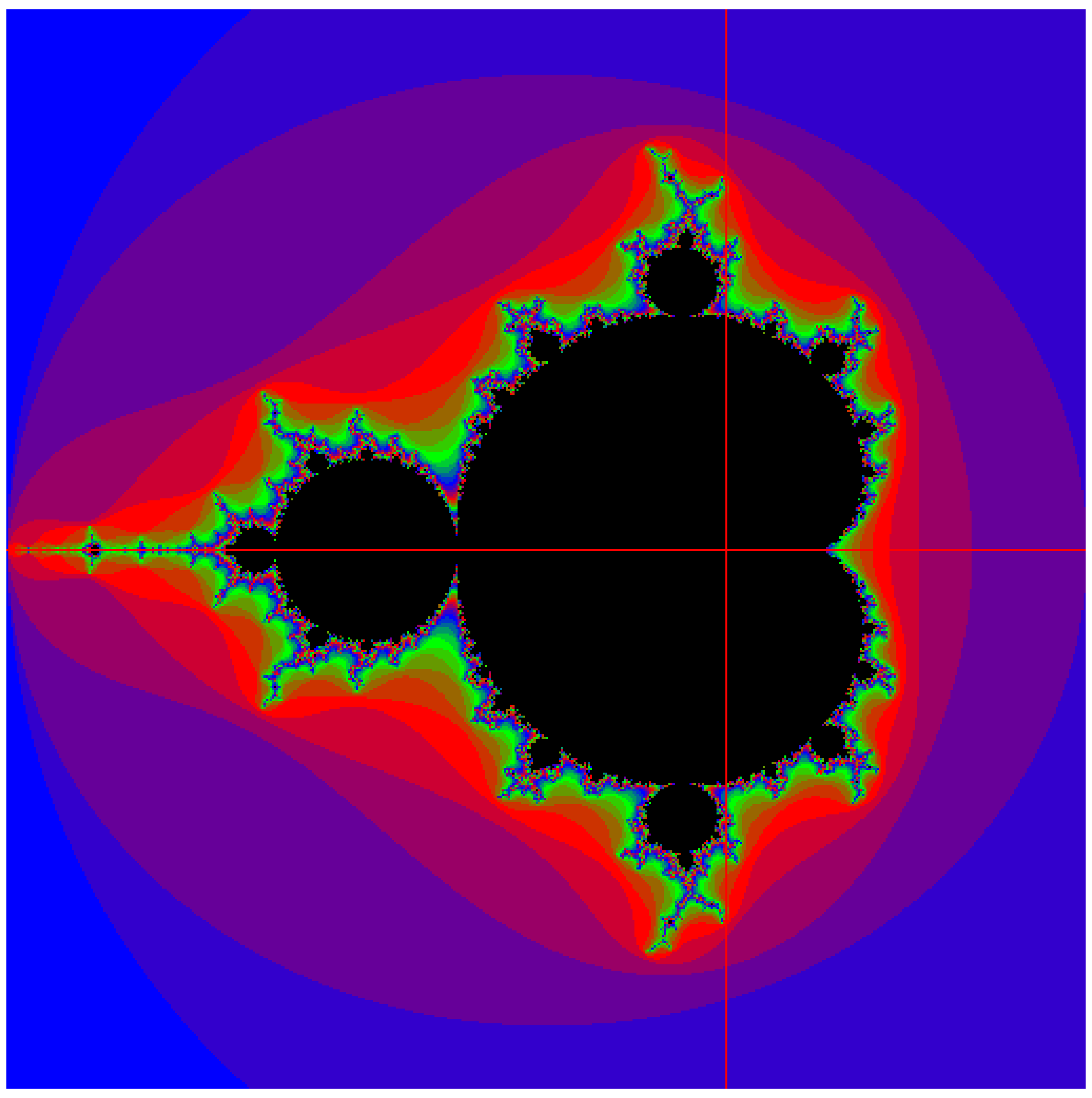
where z_0 is point you want to colour in. If $|z_n| \not\to \infty$ then colour it in black.

How do we actually test this? Well, if $|z_0| > 2$ then it definitely blows up. It $|z_0| < 2$ then we keep looking at z_n:

$$|z_{n+1}| > |z_n^2| - |z_0| > |z_n^2| - 2 > |z_n|,$$

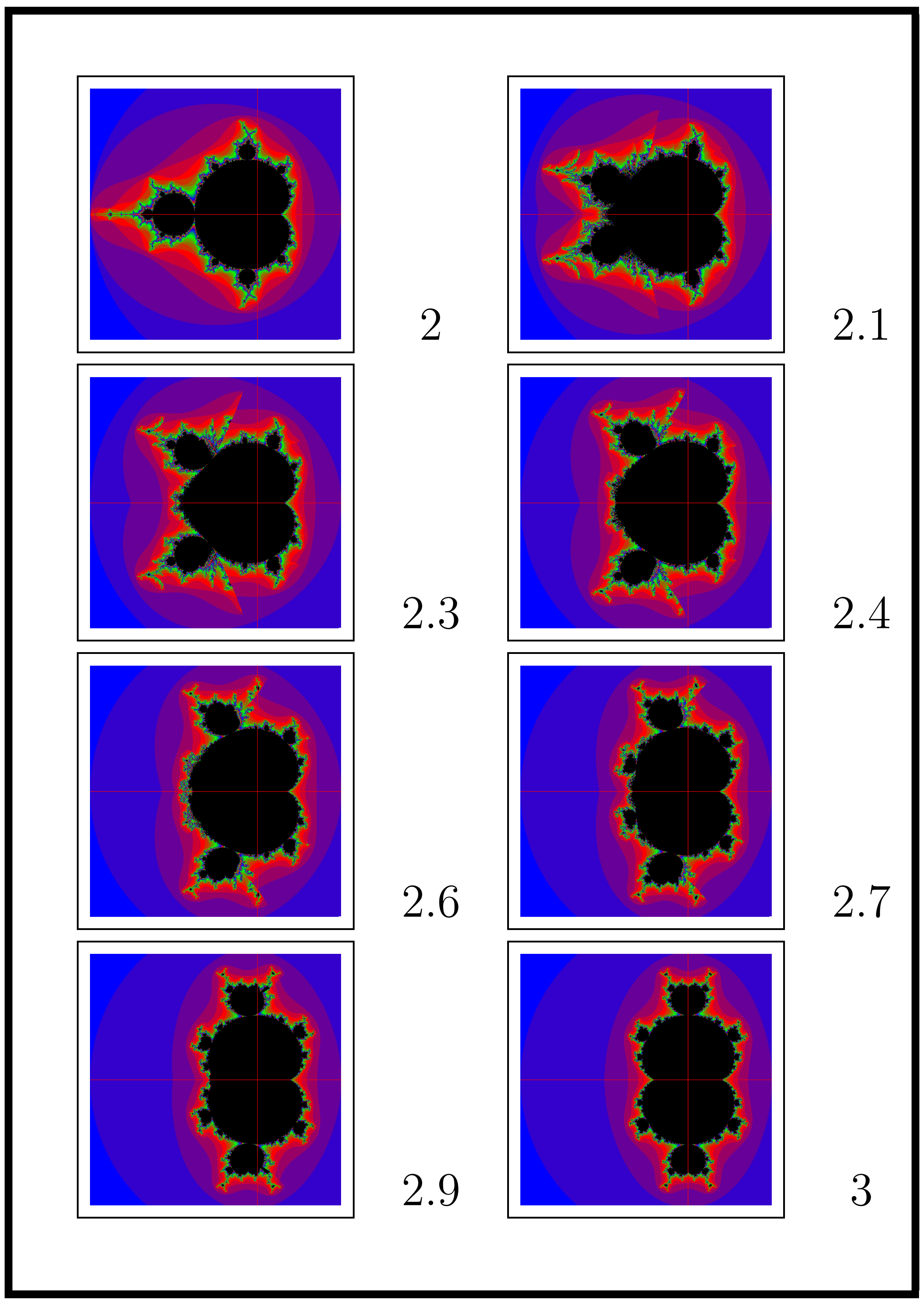
if $|z_n| > 2$. If we colour points according to how many steps it takes for $|z_n| > 2$ we get the colour version of the Mandelbrot set.

0	1	2
3	4	5
6	7	8
9	10	11
12	13	14



This simple idea can be varied to produce other interesting sets. One easy extension is to look at:

$$z_{n+1} = z_n^3 + z_0$$

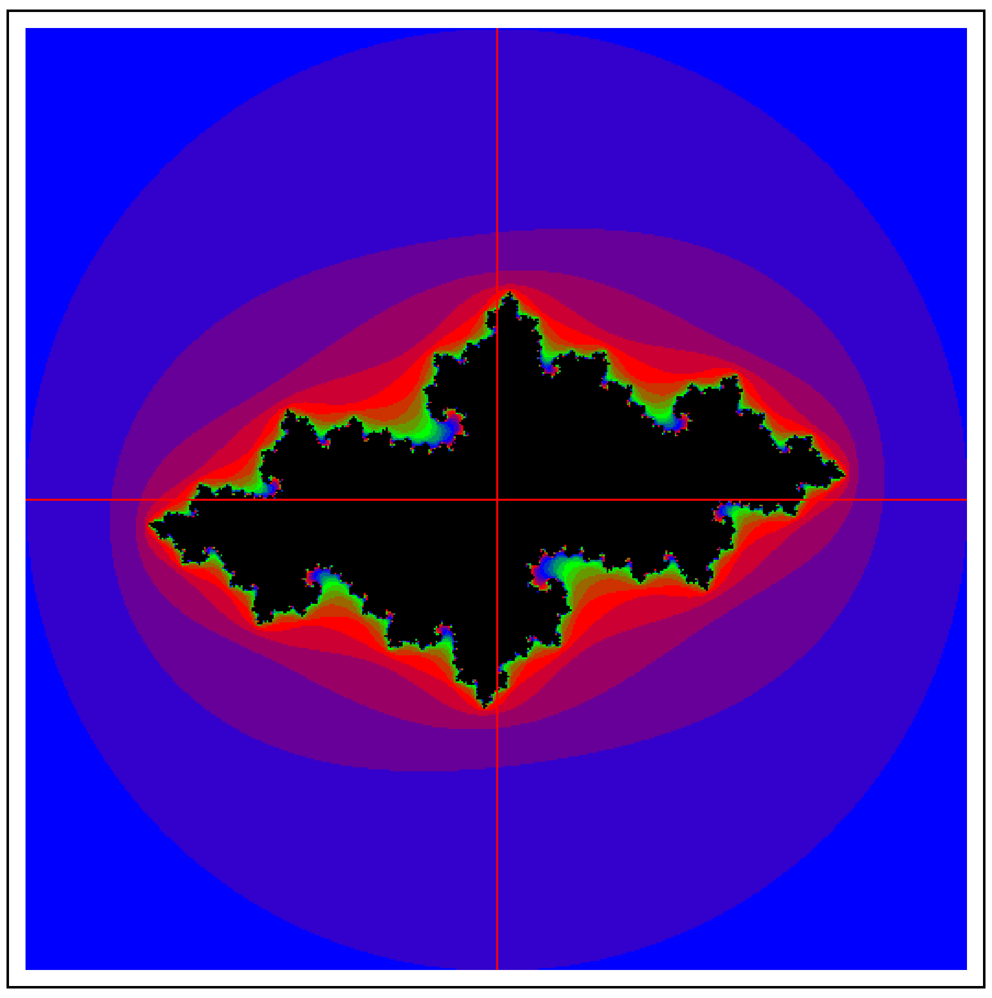

2
2.1
2.3
2.4
2.6
2.7
2.9
3

Julia Sets

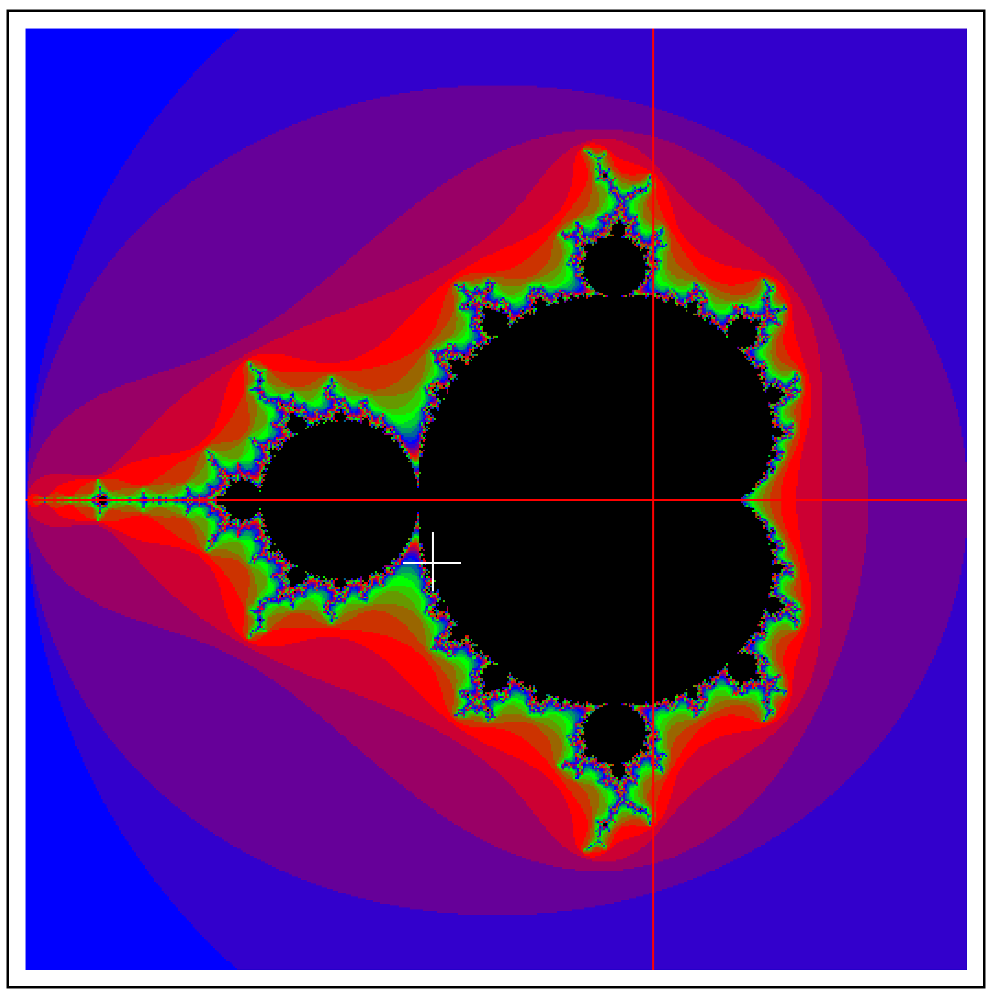
These are another variant. This time choose c and keep it fixed:

$$z_{n+1} = z_n^2 + c$$

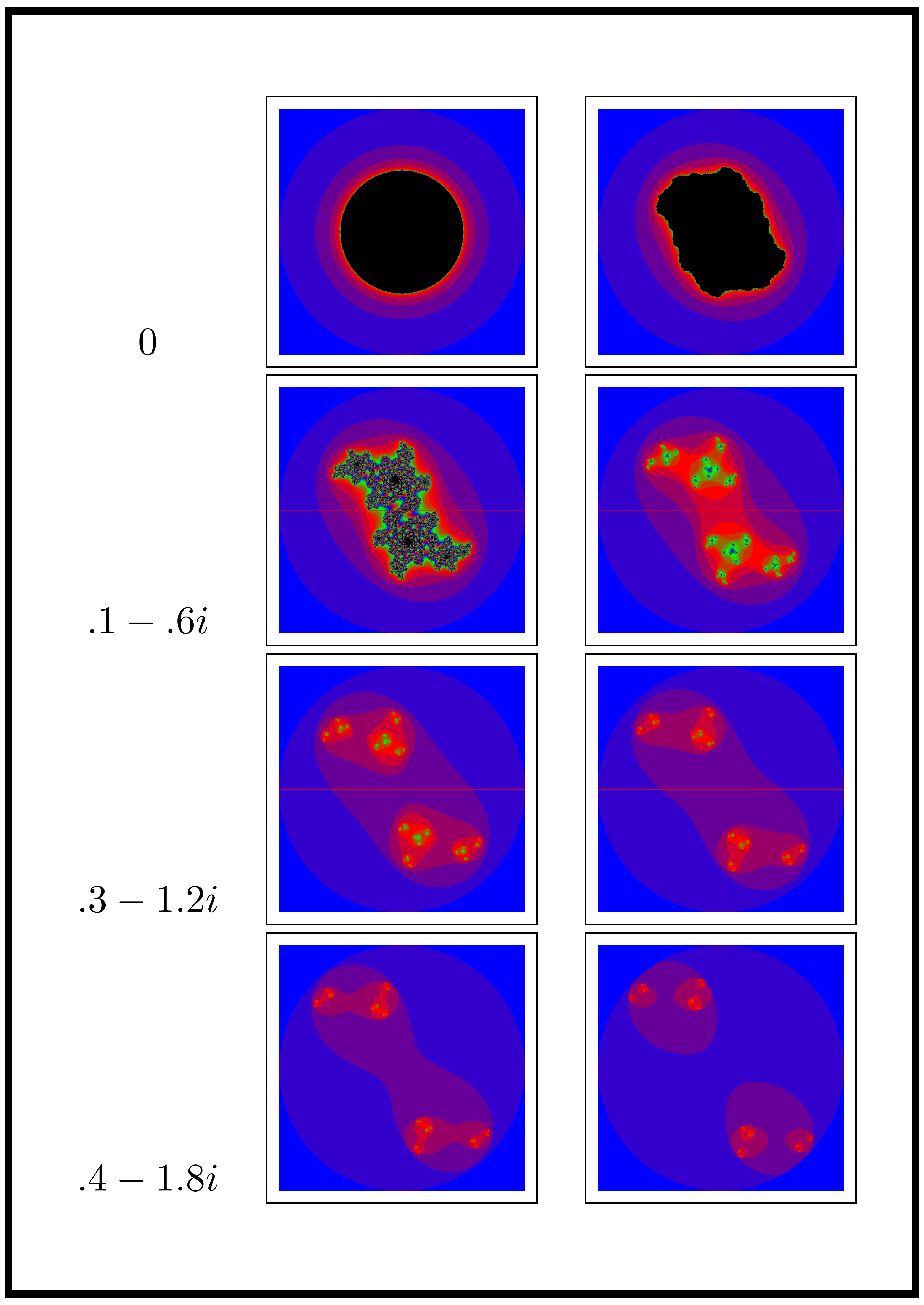
For $c = -0.7 + 0.2i$ we get:



What a lot of people don't know is that the Mandelbrot set is like a telephone directory of Julia sets.



We can actually 'deform' a circle into a Julia set!


0
$.1 - .6i$
$.3 - 1.2i$
$.4 - 1.8i$

Fixed Point Arithmetic

When computers were slow people used fixed point instead of floating point.

Write non-integers as:

$$m \times 2^n$$

Addition is easy:

$$m_1 \times 2^n + m_2 \times 2^n = (m_1 + m_2) \times 2^n.$$

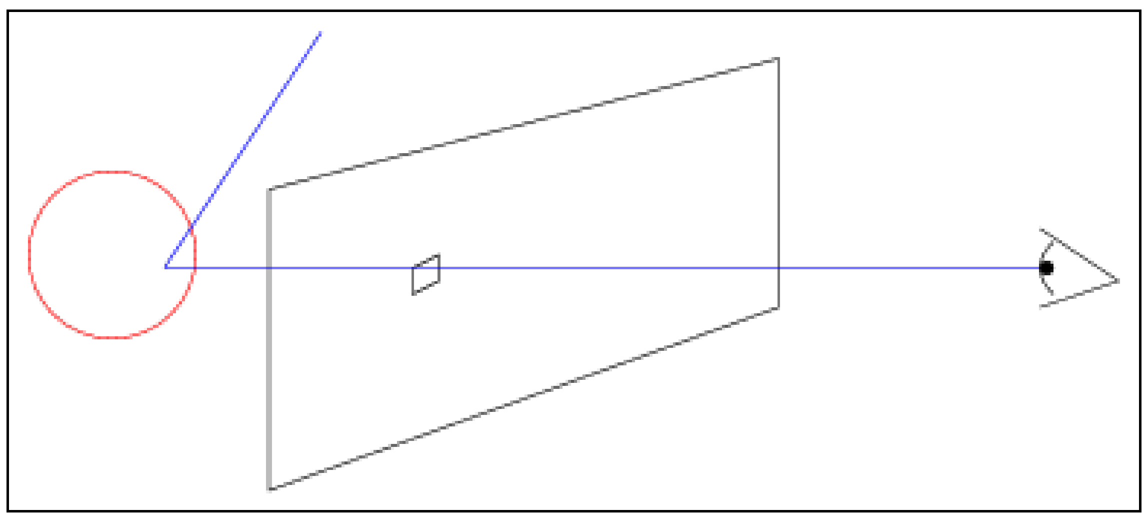
Multiplication is a little harder:

$$(m_1 \times 2^n) * (m_2 \times 2^n) = (m_1 m_2 2^n) \times 2^n.$$

You don't have to remember n 'cos it is fixed. In the floating point n varies and normalisation is complicated.

Ray Tracing

Ray tracing is a way of using a computer to produce a picture of a ‘scene’ based on a geometric description of it.



The idea: For each point in the picture trace a ray back from the eye of the viewer through that point to find out what they see.

Rays & Intersections

Representing a Ray is easy:

$$\vec{r}(t) = \vec{o} + t\vec{d}.$$

When a ray hits something we solving an equation. Usually the equation of an object looks something like:

$$f(\vec{x}) = 0$$

and finding the intersection with the ray involves solving:

$$f(\vec{r}(t)) = 0$$

We're looking for the smallest positive solution.

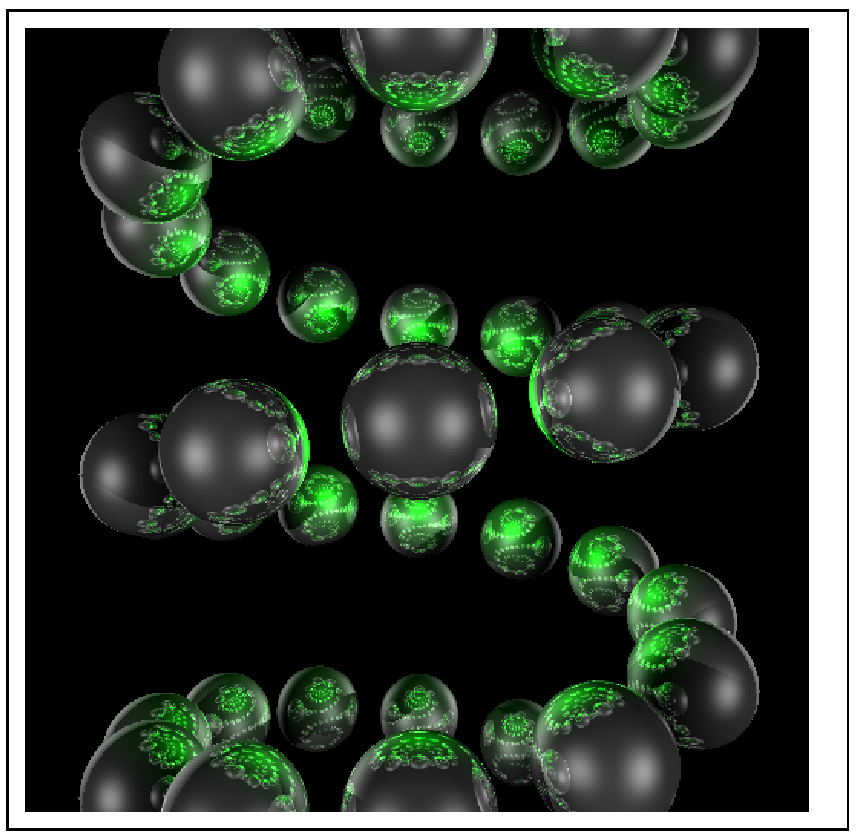
Example: sphere

The equation of a sphere is:

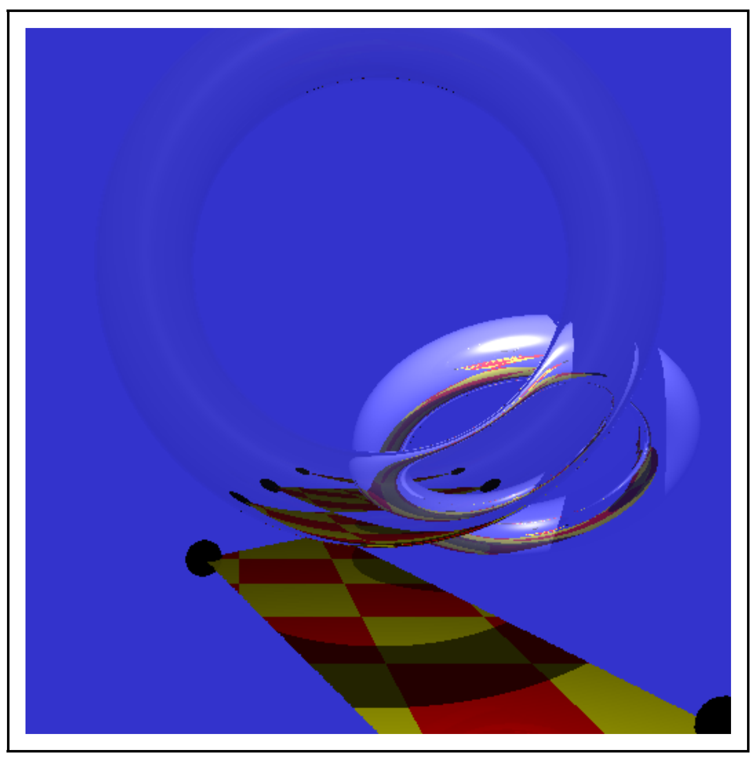
$$(\vec{x} - \vec{c}) \cdot (\vec{x} - \vec{c}) = r^2.$$

Subbing in $\vec{x} = \vec{r}$, we get a formula for t:

$$t^2 + 2t(\vec{o} - \vec{c}) \cdot \vec{d} + (\vec{o} - \vec{c})^2 - r^2 = 0$$



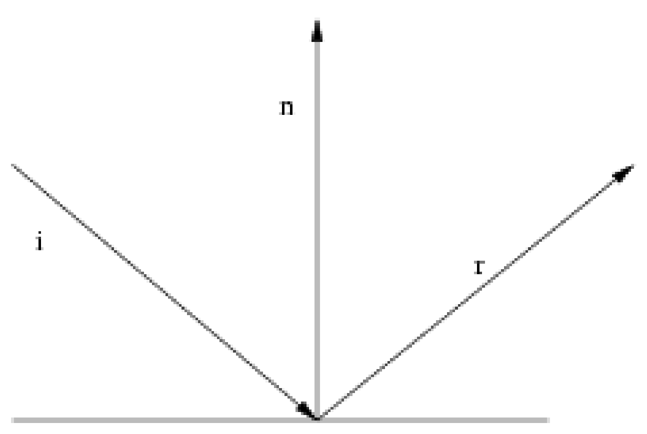
Errors in solving equations are obvious!



Reflections & Refractions

Ray tracing is *recursive* as at each stage you may have a reflected or refracted ray to trace.

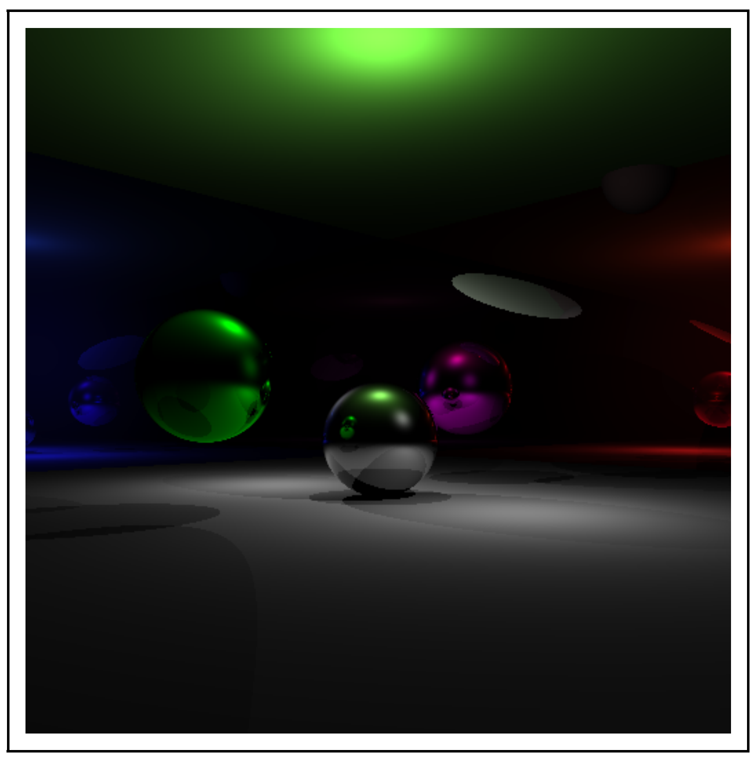
Reflection:

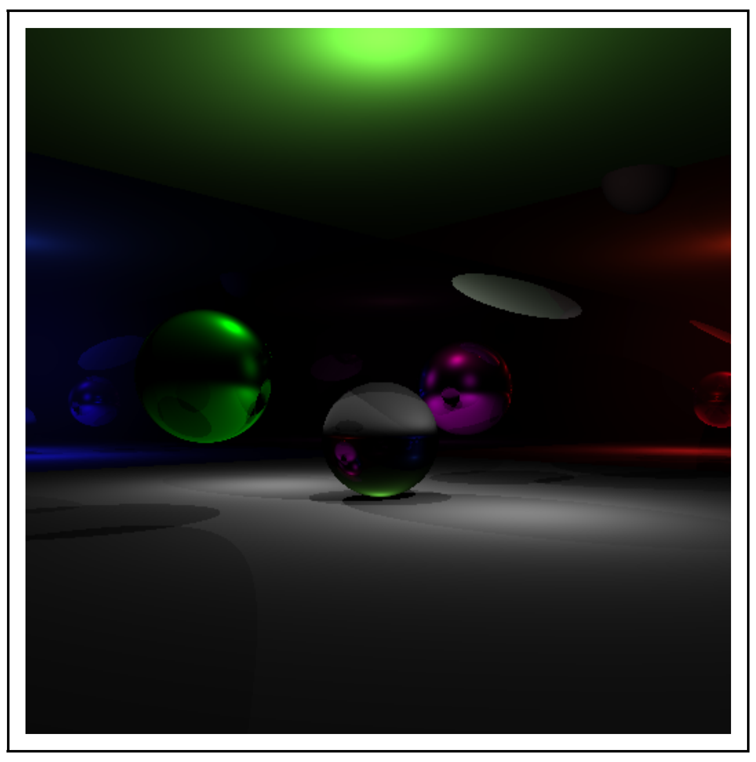


$$\vec{r} = \vec{i} + 2(\vec{i} \cdot \vec{n})\vec{n}$$

Refraction:

$$\vec{r} = \eta\vec{i} + \left(\eta(\vec{i} \cdot \vec{n}) - \sqrt{1 - \eta^2(1 - (\vec{i} \cdot \vec{n})^2)} \right) \vec{n}$$





The Normal

There are two ways of finding the normal to a surface. One is to use geometry and common sense. For the sphere the normal at $\vec{x}_0$ is:

$$\frac{\vec{x}_0 - \vec{c}}{r}.$$

The other option is to apply some mathematics to $f(\vec{x}) = 0$ and you'll find the normal is in the direction:

$$\nabla_{\vec{x}} f(\vec{x})\big|_{\vec{x}_0}.$$

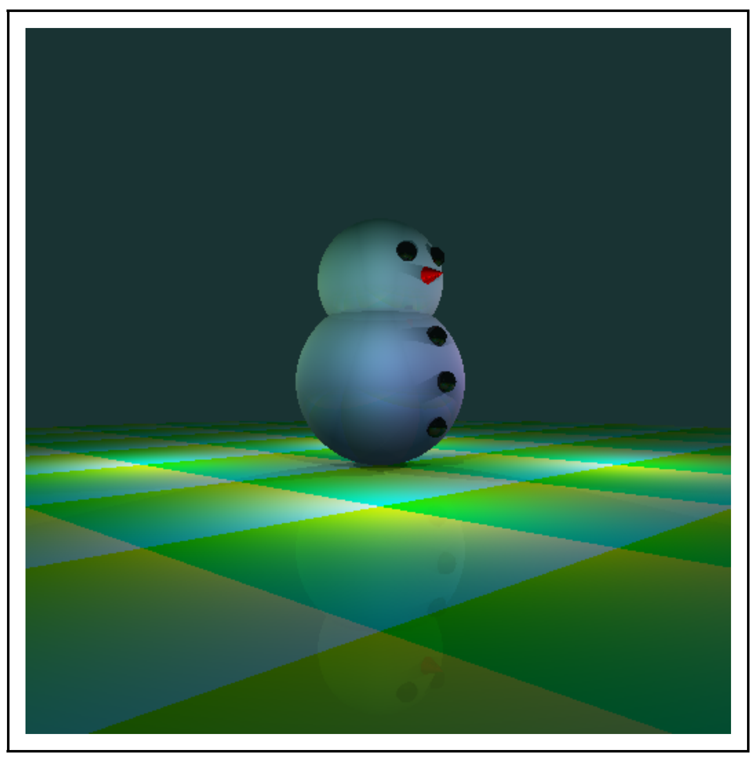
Either way, once you have an intersection and normal function you can ray trace an object.

Colour

Colour are often represented as tripple indicating red, green and blue:

$$\text{Wheat} = 0.96 + 0.87 + 0.70$$

There are other schemes: HSV, Pantone,



Not all things can produce all colours. The set of colours a display device can produce is known as a gamut.

Image Compression

Image compression is relatively big business, the idea is to make an image as small as possible for transmission.

One classic trick is run-length encoding.

001001001001...001

becomes

001×78

More modern techniques are 'lossy', for example jpg and mpg

Similar ideas used in audio for minidisk, mp3 and mobile phones.

0% (65020)

25% (532461)

50% (654198)

75% (725158)

100% (801369)

Orig (883538)