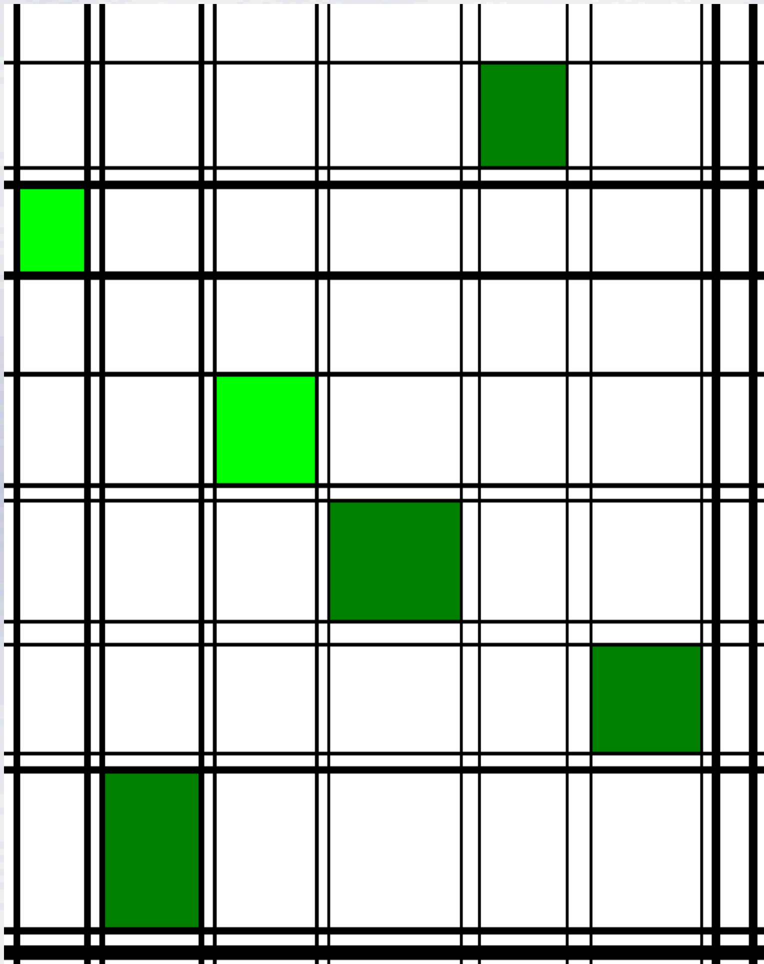
Generating Abstract Art with Perl



- Duncan C. White
- Systems Administrator, Dept of Computing, Imperial College, London.
- http://www.doc.ic.ac.uk/~dcw/
- email: dcw@doc.ic.ac.uk

Introduction

- Perl is already well used in the linguistic arts – eg generating artificial poetry (haikus etc).
- So, what about the visual arts? Can Perl be used to produce high quality artwork?
- Well, Perl has excellent support for drawing 2D images – GD, Tk, the Gimp and PerlMagick.
- But there's a problem – very few of the visual arts lend themselves easily to an algorithmic treatment. How could we describe (for instance) painting a portrait algorithmically?

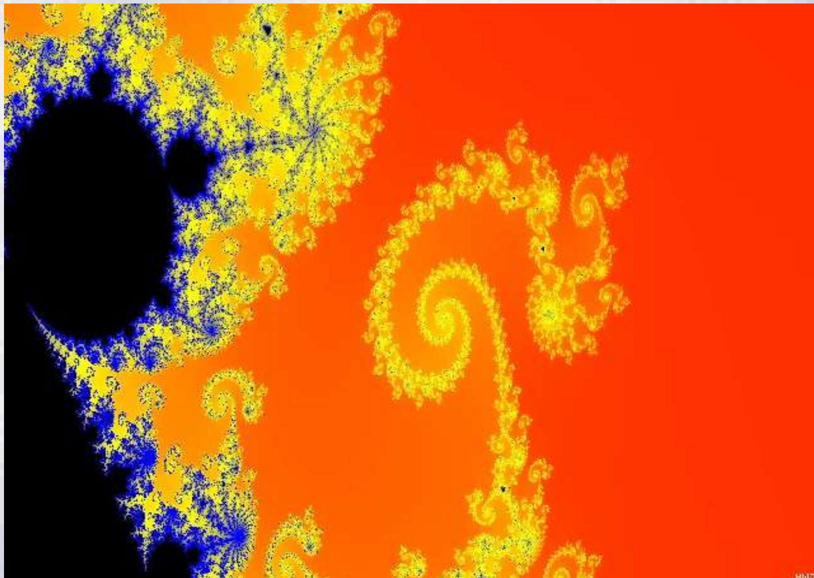
Too Complex? Start Simple!

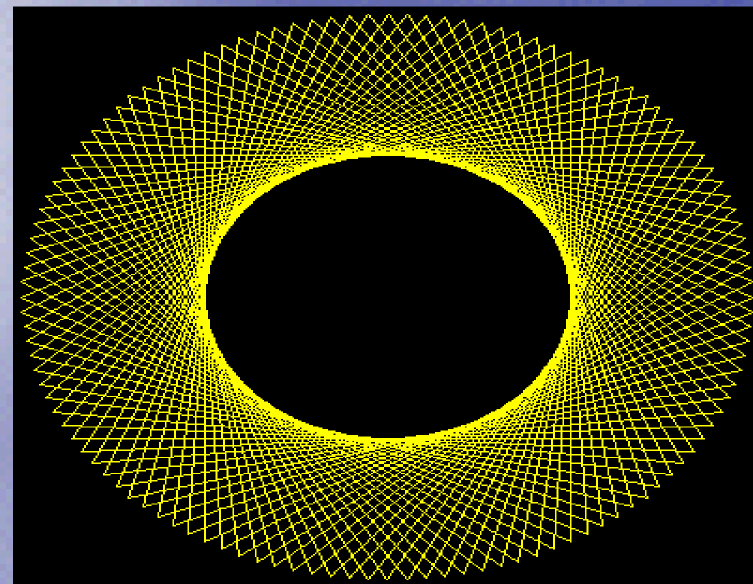
- Let’s look for particular styles of visual art that are easier to treat algorithmically.
- The more formalized and constrained forms are likely to be easier; so let’s look at some of those:
- First, let’s dismiss one obvious area – as soon as we think of computers and vision we probably think of highly realistic computer animations like Jurassic Park and Toy Story.
- While these are brilliant pieces of work, they tend to produce photo–realistic images much more like a photograph than an artwork.

First Stylized Artform – Geometrical Art

- There are several forms of art that are directly based on mathematics and geometry.
- Most obvious of these are *fractals* ranging from the stunningly complex Mandelbrot and Julia Sets, to very simple Koch snowflake curves.
- Many natural forms (especially plants) have a fractal structure, so fractal systems such as Lindemayer Systems have been used to produce semi–realistic plant growth images.
- Geometrical Art also includes Islamic tiling patterns, practically all of M.C. Escher's work, lacework doilies and pin–and–thread needlework.

Some Examples of Geometrical Art



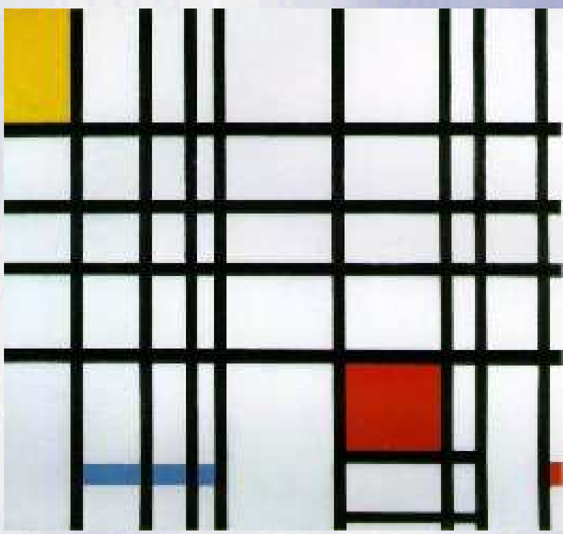


Second Stylized Form – Abstract Art

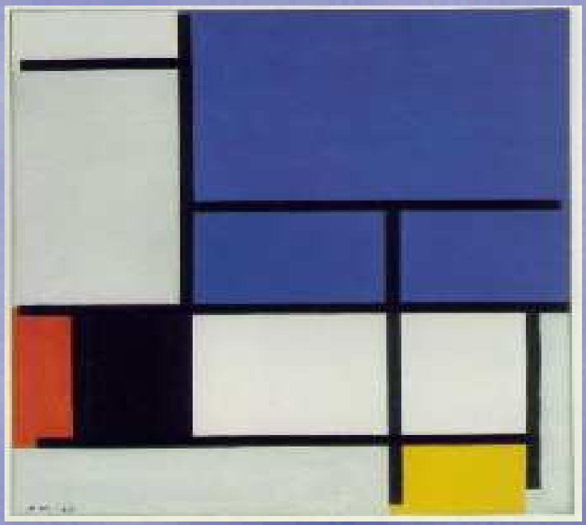
- Many forms of abstract art are highly stylized and thus may be amenable to computerization.
- Of course, there are many different abstract artistic styles. I want to look at the work of a particular artist:
- Piet Mondrian, who lived from 1872–1944, mostly in Paris and New York.
- After a successful career as a conventional landscape artist, from around 1912 Mondrian developed and experimented with a wholly original abstract style:

Mondrian's Abstract Style

- Paint an irregular grid of horizontal and vertical lines, dividing the abstract space into regions. Then fill a small number of the regions with bold primary colours.
- Here are some examples of Mondrian's work:



Composition with Red, Yellow and Blue, 1920



Composition with Large Blue Plane, Red, Black, Yellow, and Gray, 1921

Computerising Mondrian?

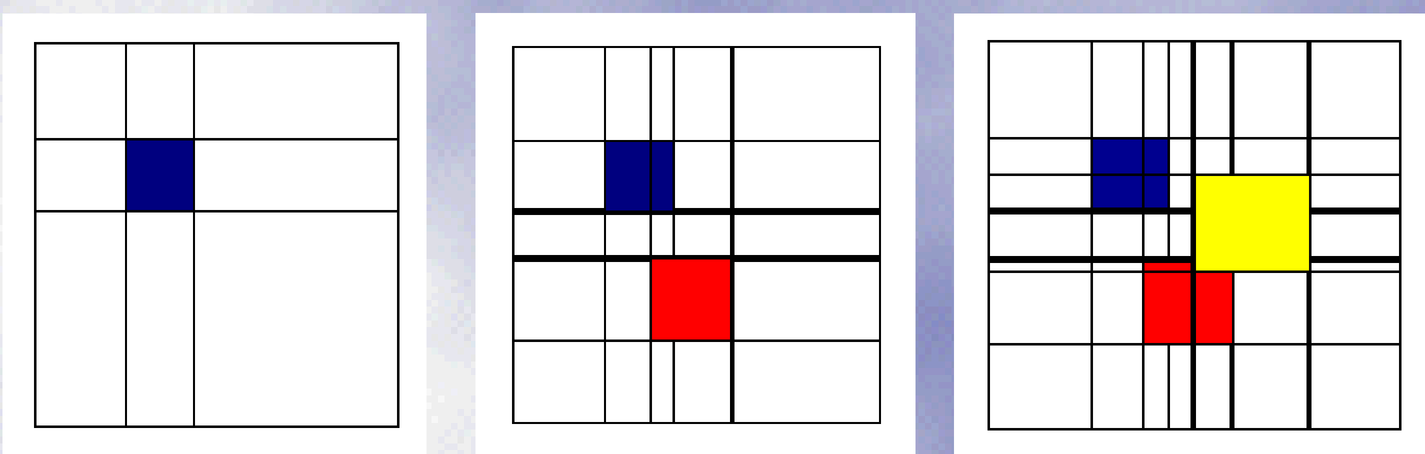
- Computerising Mondrian’s style is graphically very simple: we only need to be able to draw lines and fill rectangles!
- What is less clear is what constraints we should put on the placement of the grid and rectangles for maximal aesthetic effect.
- To this end, in 1999, I decided to investigate this problem, partially as a way of learning to use the GD module.
- I wrote three web–based experimental programs to explore various different algorithms. Let’s look at them in turn.

First Algorithm – mondrian1

- My first Mondrian program was very simple:
- It decided (randomly between limits) how many rectangles to colour in, and placed each rectangle completely randomly across the canvas.
- After some experimentation, I decided that no rectangle should be bigger than 20% of the canvas width or height, or narrower than 10%.
- Whenever the program placed a rectangle, it placed horizontal and vertical gridlines of random thickness right by the rectangle.

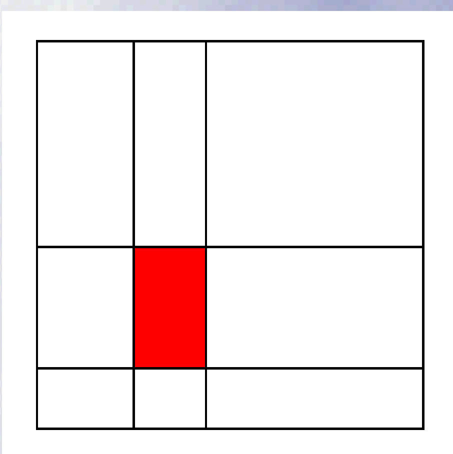
Mondrian1 in action

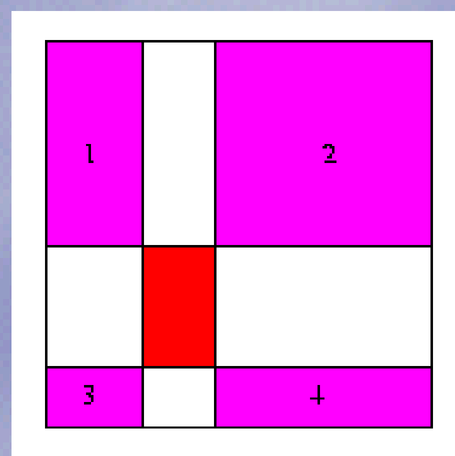
- Here’s mondrian1 in operation. In this example, the edges that extend from the red (second) rectangle intersect the blue (original) rectangle, as do the edges extending from the yellow rectangle. Because the blue rectangle was drawn first, the later insecting edges appear over it.
- Mondrian1 doesn’t really divide up the plane!



Second Algorithm – Mondrian2

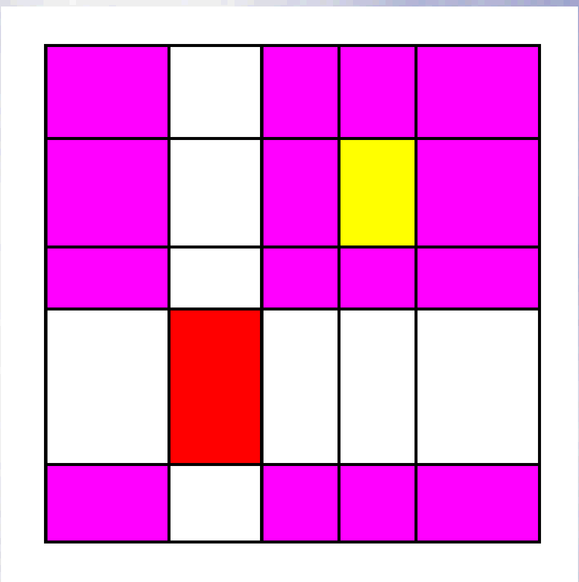
- Mondrian2 prevents any gridline from intersecting any rectangle. It does this using a list of available regions as follows:
- Place the first filled rectangle randomly. Extend the rectangle's sides to form the first gridlines.
- Now cut the space along the gridlines. Only the labelled four regions are still available.

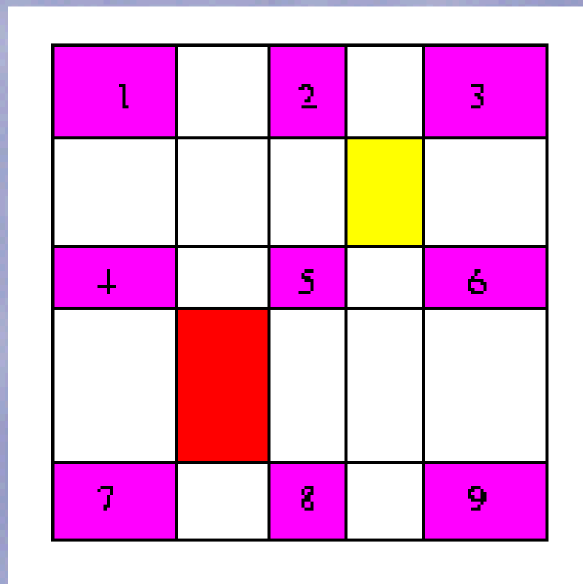




Mondrian2 cont

- Now we place the second rectangle as follows: Pick one of the available regions randomly. Position a (yellow) rectangle randomly within it.
- Now clip the chosen region into 4 diagonal pieces again, and clip all other available regions against the new grid edges:

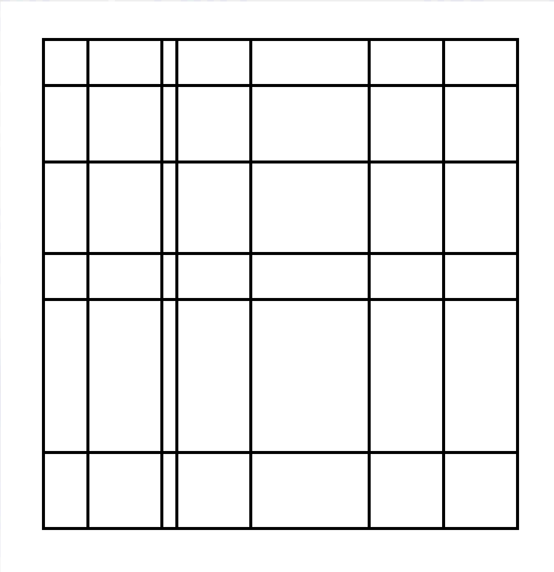


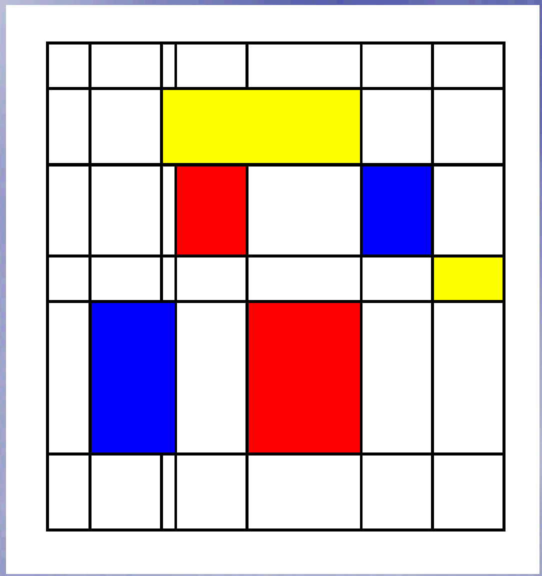


Final Algorithm – Mondrian3

- This version discards the available region list. Instead it places the entire irregular grid first, deciding how many horizontal and vertical gridlines to have, and where and how thick each gridline should be.
- Then it chooses how many rectangular cells to fill in, and chooses grid coordinates and colours for each one.
- Occasionally, it chooses not a single grid cell, but several adjacent cells, filling them in and thus partially hiding some gridlines.

* So, for example, we might choose 5 horizontal lines and 6 vertical lines, then choose 6 cells at random for colouring in:





Conclusion

- This is a start! We've seen that relatively simple Perl programs can produce a single form of abstract art reasonably well.
- However, even in narrow focus, it's true that my programs don't fully capture all Mondrian's talent and complexity.
- Perhaps that's just as well?
- All 3 programs mentioned here (and more) are available at http://www.doc.ic.ac.uk/~dcw/
- A final thought – could certain forms of landscape art such as a Whistler Nocturne also could be simulated?