Project 2

Shiny!

Due December 11, 2020 at 5pm

Project 2 gives you a chance to show off your programming skills combined with your creative inspiration. This project is designed to take you a solid two weeks to complete; please plan accordingly.

This project can be completed in three phases:

1. the parts that don't require recursion (are approachable now)
2. the parts that require recursion (can start now, all material by Nov 24)
3. extra credit (don't start until after you've finished phases 1 and 2)

You should start the first phase now and not worry about the second until we cover it in class. Pieces of the program that you can implement in phase one are:

1. Generation of an html file containing rectangles
2. Design of logic for randomly deciding whether to split a rectangle and where
3. Selection of a random color for the rectangles (method described below)

In order to encourage you to start planning (and implementing) during the first two weeks of the assignment, we are also offering additional credit for anyone who shows a member of the course staff a Python program that generates an html file containing single randomly located rectangle before 5pm on Dec 4, 2020. This counts as an Engagement Activity.

Learning Goals

1. Design and implement a more complex program than previously.
2. Make good choices about which functions to design and their inputs and outputs.
3. Use recursion to create elements at different geometric scales.
4. Use random number generation to create interesting variation.
5. Identify and fix errors.

The Assignment

Piet Mondrian (March 7, 1872 – February 1, 1944) was a Dutch painter who created numerous famous paintings in the early half of the previous century that consisted of a white background, prominent black horizontal and vertical lines, and regions colored with red, yellow and blue. Three examples are shown below:

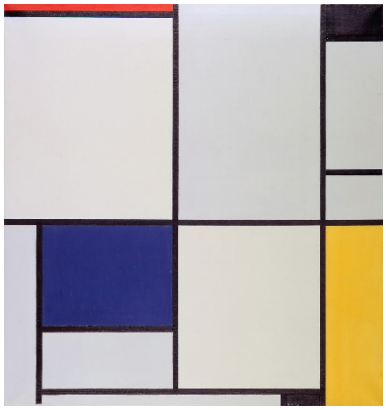
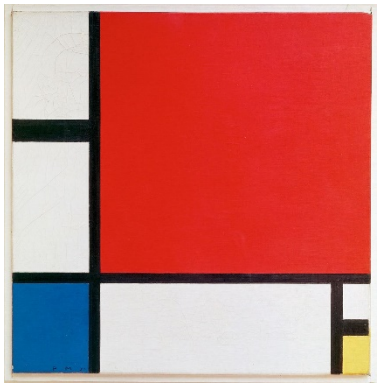
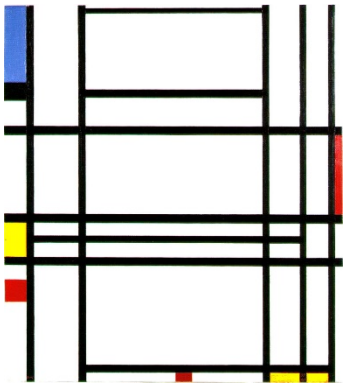


Tableau I, 1921



Composition II in Red, Blue, and Yellow, 1930



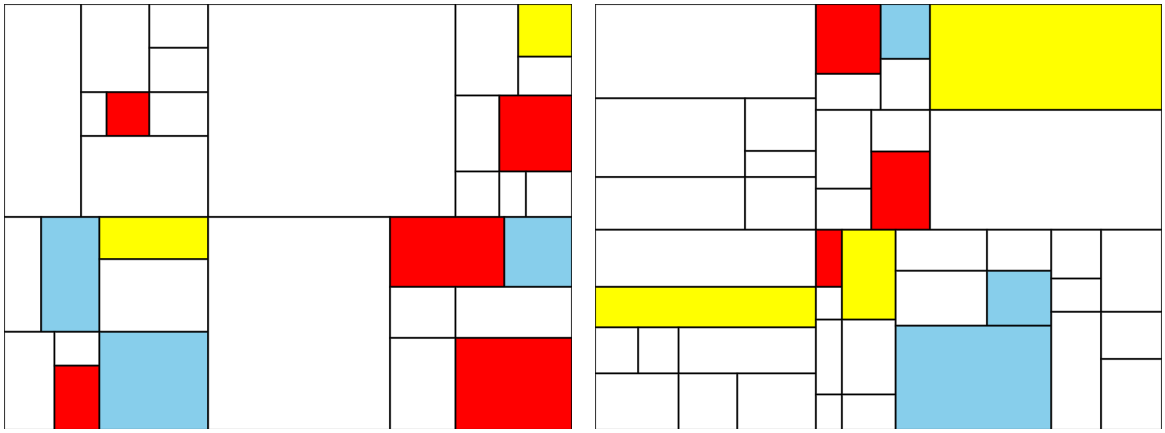
Composition No. 10, 1939-1942

Your task is to write a program that uses recursion to generate pseudo-random "art" in a Mondrian style. Your program's output will be an HTML document that contains rectangle primitives (perhaps among others) within an SVG tag. The following general strategy will be used to generate art in a Mondrian style:

if the region is really big, always split
If the region is wider than half the initial canvas size and taller than half the initial canvas height:
    Use recursion to split the region into four smaller regions (a vertical split and a horizontal split) with both split locations chosen randomly.
Else if the region is taller than half the initial canvas size:
    Use recursion to randomly split the region into two smaller.
Else if the region is wider than half the initial canvas size:
    Use recursion to randomly split the region into two smaller regions.

if the region is big enough, maybe split
Else if the region is big enough to split both horizontally and vertically, and both a horizontal and vertical split are randomly selected:
    Use recursion to split the region into four smaller regions (a vertical split and a horizontal split) with both split locations chosen randomly.
Else if the region is tall enough to split vertically, a vertical split is randomly selected:
    Use recursion to split the region into two smaller regions using a horizontal line with the split location chosen randomly.
Else if the region is wide enough to split horizontally, and a horizontal split is randomly selected:
    Use recursion to split the region into two smaller regions using a vertical line with the split location chosen randomly.
Else:
    Fill the current region (randomly, either white or colored, and if colored, with a random determination of red, blue or yellow).

A couple of images generated using this algorithm are shown below.



Use the following strategy when randomly deciding whether or not to split a region:

Generate a random integer between 120 and the width of the region * 1.5.
If the random integer is less than the width of the region then split the region.

While this strategy works, you might find yourself asking: Why is the random number between 120 and the width of the region * 1.5? By using 120 as the lower bound for the random number, we ensure that we never split a region that is less than 120 pixels wide (or tall when splitting in the other direction), and as such, we meet the constraint that the region is big enough to split (for my arbitrary definition of big enough). Selecting a random value that could be up to 1.5 * the width of the region, but then only performing a split when the random value is less than the width of the region, provides a random chance that a larger region will not be split into smaller regions.

Use the following strategy when splitting a region, either because it is so big that it will always get split, or because it was randomly selected to be split:

Choose the split point, randomly, somewhere between 33% and 67% across the region (or down the region if splitting in the other direction). Choose two random split points when splitting both horizontally and vertically.
Split the region into two smaller regions, one on the left and one on the right (or one on top and one on the bottom), or four smaller regions if splitting both horizontally and vertically
Use recursion to fill / further split each new region

Use the following strategy to decide which color will be used to fill a region that will not be split further:

Select a random value, r,
If r < 0.0833 then fill the region with yellow
Else if r < 0.1667 then fill the region with blue
Else if r < 0.25 then fill the region with red
Else fill the region with white

Save a copy of your program using a different name once you have the standard algorithm working. You will turn in this program as the base assignment. If you make the extra credit extensions, you will turn them in as a whole separate program.

Once you have the provided algorithm working you are encouraged to adjust / expand / adapt this algorithm to generate art with your own specific style provided that is still at least vaguely Mondrian in style (meaning that it largely consists of horizontal and vertical lines and colored regions, at least the majority of which are rectangular). Ideas for customizing your work that you might want to consider include:

- [1 pt] Using your favorite colors rather than red, yellow and blue for the filled regions
- [2 pts] Changing the distribution of random numbers used when selecting the sizes of regions, colors (or anything else random). For example, instead of allowing all possible values, reduce the space to numbers that are evenly divisible by 10 (or 20 or some other number) so that the random lines have more regular spacing to them.
- [2 pts] Using a patterned fill for some regions instead of only using solid fills
- [2 pts] Occasionally split a region into three smaller regions instead of two or four
- [3 pts] Soliciting user input to control some of the choices made. Be sure to prompt appropriately for any user input you expect.

These extensions will be worth a maximum of 10 points of extra credit and will be graded on how interesting the resulting images are.

Generating an html file

Your program should use file I/O to generate an html file containing the image description. The general format of a very simple html file that draws a single red rectangle looks like this:

```
<html>
<head></head>
<body>
<svg width="800" height="800">

<rect width="50" height="100"
style="fill:rgb(255,0,0);stroke-width:3;stroke:rgb(0,0,0)"
/>

you can put more rectangles here…..

</svg>
</body>
</html>
```

How to turn in your homework

Turn in each program (the standard implementation and the extra credit version, if any) in its own file. When turning in your own assignment make sure to add your last name to the file name (for example: Rheingans_p2.py or Rheingans_p2_XC.py). You should also submit an html file of your favorite image because we will probably not see the same image when we test your program (because of the random elements).

This assignment includes a creative and artistic element. As a result, we are hoping to receive numerous interesting submissions that will be worthy of showing off. We plan to post the images that are created on webpages or around the building so that others can view them. Your image will be posted anonymously, unless you choose to include your name as part of the image that you create. Please do not put your student number on your image. If you are **not** willing to have your image included on the website then please send an email to penny.rheingans@maine.edu clearly stating such when you submit your assignment.