

Project 2
I'm an Artist!
Due December 9, 2022 at 5pm
(5% bonus for submission before 5pm Dec. 8)

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
2. the parts that require recursion
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 your functions from the first phase work. Pieces of the program that you can implement in phase one are:

1. Use of the graphics program to draw 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)

You have the knowledge of recursion that you need now, but it might be better to start with the parts that don't require recursion.

In order to encourage you to start planning (and implementing) during the first week of the assignment, you have a **mandatory design review due before 5pm Friday December 2**. This design review consists of your design to a member of the course staff. More details about expectations and course staff office hours are available below.

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:

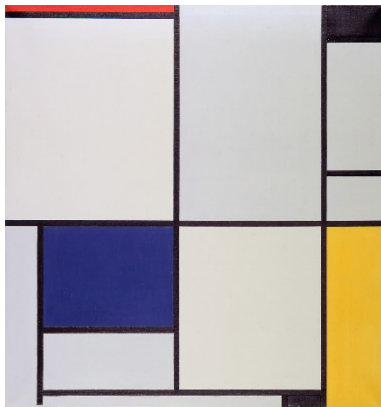
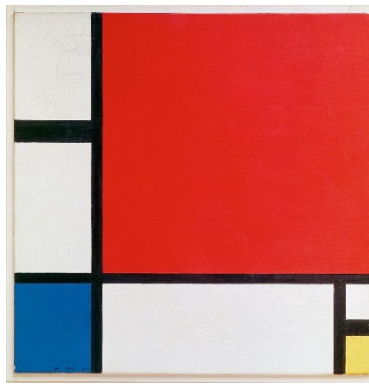
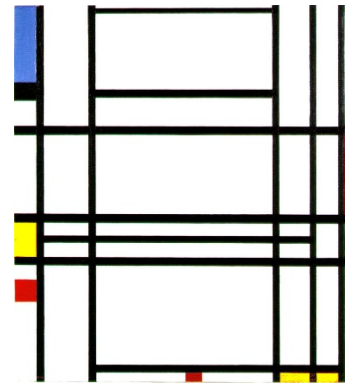


Tableau I, 1921



Composition II in Red,
Blue, and Yellow, 1930



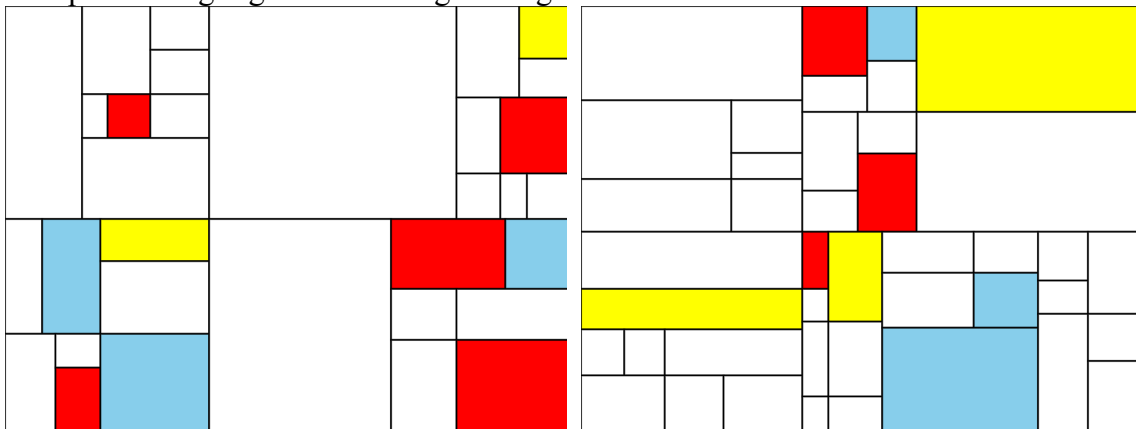
Composition No. 10,
1939-1942

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 graphics images that contains rectangle primitives (perhaps among others). The following general strategy will be used to generate art in a Mondrian style:

Here are some guidelines for splitting:

1. If the region dimension (either width or height) is larger than half of the initial canvas size, always split at randomly chosen location (if both are bigger, split in both directions)
2. If the region dimension (either width or height) less than half the initial canvas size and larger than 90 pixels, randomly decide whether to split at randomly chosen location (if both are big enough, potentially split in both directions)
3. Otherwise, fill the region with a randomly colored rectangle

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 90 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 90 as the lower bound for the random number, we ensure that we never split a region that is less than 90 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 31% and 68% 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 $r < 0.9$ then fill the region with yellow

Else if $r < 0.15$ 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.

Extra Credit

Once you have the basic 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.
- [4 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.

Drawing module

You will use the module `graphics.py` (created by John Zelle) to create your display. The most recent version of the library can be obtained at <http://mcsp.wartburg.edu/zelle/python>. Go to that site and download the file “`graphics.py`”. The site also contains documentation in html and pdf format. A simple program to draw a single rectangle might look like this:

```
from graphics import *

def main():
    win = GraphWin("My Drawing", 600, 600)
    aRect = Rectangle(Point(150,150), Point (325, 440))
    aRect.setOutline("black")
    aRect.setFill("purple")
    aRect.draw(win)
    win.getMouse() # pause for click in window
    win.close()
main()
```

Preliminary design review

The preliminary design review is worth 10% of this project. For the preliminary design review, you should write a description of the project design structure as you did in Lab 8. Specifically, you should sketch the contents of the program, breaking it down into **data** and **functions** and then order the functions into a program structure (basically an outline of the program). Your sketch should contain the following elements:

1. Data: what data is required by the project
Variable names w/ description of its purpose
Data type of each variable (string, int, list, etc.)
2. Algorithms:
High-level processes the program needs to have.
3. Full Program pseudocode:
 - Arrange the various functions in order that they need to happen.
 - Include loops and some notion of when they terminate.
 - Place the necessary functions in the loops.
 - How the recursive calls work
 - The pseudocode should contain a mix of high- and low-level items. Some aspects you might not solve until you start coding, so it's fine to give a high-level description.

Explain your project design structure to a member of the COS 125 staff during office hours or before/after your lab. Your design is due before 5pm on Friday December 2.

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 or around the building, then please send an email to penny.rheingans@maine.edu clearly stating such when you submit your assignment.