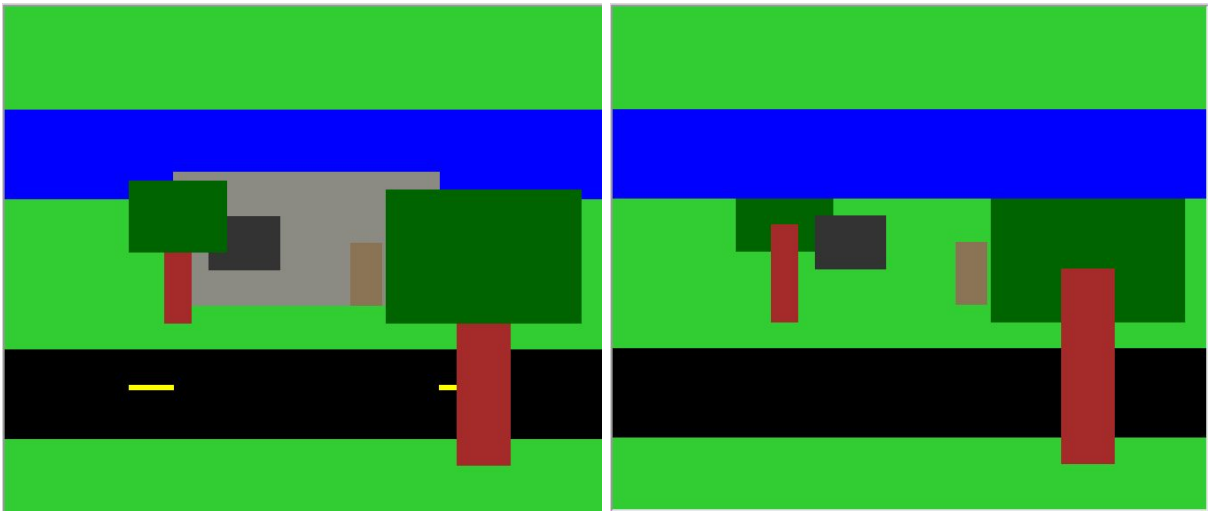
CSCI 4041, Fall 2018, Programming Assignment 1

Due Tuesday, 9/11/18, 10:30 AM (submission link on Canvas)

This is not a collaborative assignment; you must design, implement and test the solution(s) on your own. You may not consult or discuss the solution with anyone other than the course instructor or TAs. In addition, you may not include solutions or portions of solutions obtained from any source other than those provided in class. Obtaining or sharing solutions to any programming assignment for this class is considered academic misconduct. If you are not sure what this means, consult the class syllabus or discuss it with the course instructor.

The painter's algorithm is a simple technique used in computer graphics for rendering a scene in which some elements obscure parts of other elements. The idea is that if you sort the objects in your scene by their depth into the screen, and then draw them in order, starting with the most distant and ending with the closest, you will naturally draw things in the foreground over the top of things in the background.

For example, below are two renderings of a scene that consists of overlapping rectangles. On the left is the intended image. The rectangles representing the background are overlapped by those in the foreground, giving a sense of depth. On the right is an image generated by drawing the same rectangles in a different order. You'll notice that the building has been completely obscured by the background, and that the river has been drawn over the top of parts of the trees.



In this assignment, you'll be implementing the painter's algorithm using Insertion Sort. Download the template PA1.py from the class website. The template includes a `Rectangle` class, including a draw method to sketch a given `Rectangle` object using Python's `turtle` module, along with some test cases, including the sample scene above. You'll need to implement the `Insertion_Sort` method, which takes one argument, a list of `Rectangle` objects, and uses the Insertion Sort algorithm presented in the textbook to sort that list based on

each `Rectangle`'s `d` attribute, which represents the depth, from high `d` to low `d` (that is, furthest away to closest). See the image on the next page for the `d` values for the example scene. If done correctly, running PA1.py should result in the image on the left.

Requirements:

- You must download the template file PA1.py and edit the `Insertion_Sort` function. Do not edit any other part of the file.
- Your program must run without errors on the version of Python 3 installed on the CSELabs machines, Python 3.5.2 (if you're testing this on CSELabs, you need to type python3 or idle3 instead of python or idle to start the correct version from the terminal)
- You are not permitted to use any built-in Python sorting routines like the `sorted()` function or the `.sort()` list method. You are also not allowed to use any Python function that asks for user input, such as `input()`, as this will break the grading script.
- You must implement the Insertion Sort algorithm as described in the textbook. Any other sorting algorithm will receive very little credit, even if you pass every test case.
- However, note that while the Insertion Sort algorithm in the textbook describes how to sort a list of numbers in non-decreasing order, you must sort a list of Rectangle objects in non-increasing order by depth, so some adjustments to the algorithm will be needed.
- Note that there are some test cases that include multiple `Rectangles` in the list with the same `d` value. The Insertion Sort algorithm presented in the textbook is stable: that is, if two items are "equal" for the purposes of sorting, and one appears before the other in the original list, they will still appear in that order in the list after it has been sorted. To pass all test cases, your Insertion Sort must also be stable.
- This assignment will be graded automatically based on the number of test cases your program passes. There will be several secret test cases in addition to the ones included in the template to ensure you're not hard-coding in the solutions.
- The grading breakdown for this assignment is as follows:
    - 30%: File runs without syntax errors
    - 70%: Passing test cases without breaking any requirements.
- The unedited template file already runs without syntax errors and passes 3/6 test cases (because 3 of the 6 test cases are lists that are already in sorted order). This means that if your program causes syntax errors or passes less than 3/6 test cases, you will get a better score by just submitting the original template unedited.
- Submit your edited PA1.py file to the Programming Assignment 1 link on Canvas before 10:30 AM on 9/11/18. No credit will be given for late submissions.

For reference, here are the depth (`self.d`) values of each Rectangle in the sample scene: