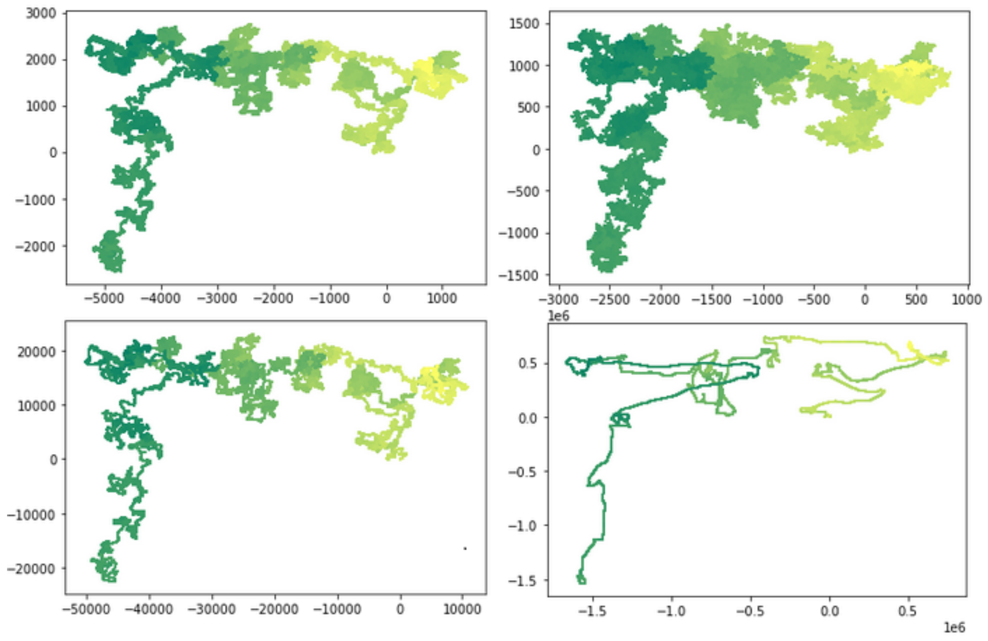
Gaussian randomization in a random walker and world weight

SJ

December 16, 2020



Contents

Introduction

A short introduction to an expansion to the random walker made before, a walker with gaussian randomization. The file SingleStep_Map_generator.pdf goes into detail about the idea of the program and why it wont be used, this is merely a practice exercise in understanding randomization.

What Are Gaussian Randomizers

This section is oversimplified as Gaussian randomizers are just a means to an end for this walker. There are a lot of good tutorials on gaussian functions and randomizers online here is a simple tutorial for example `https://www.youtube.com/watch?v=IIuXF5QRBTY`.

Gaussian functions create vaues in a way they would occur in nature, with values that happen more often and values that dont happen often. naturally they occur in a bell curve(**??**). Things like creature heights and even test scores ofen occur with such a curve .

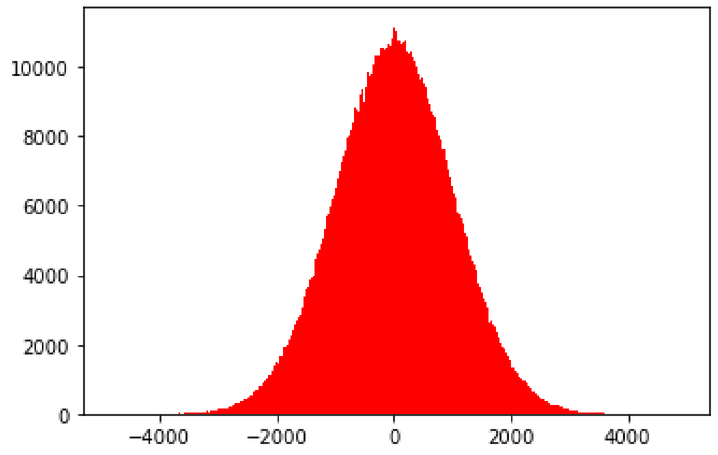


Figure 1: A standard bell curve

Let's Get gaussian

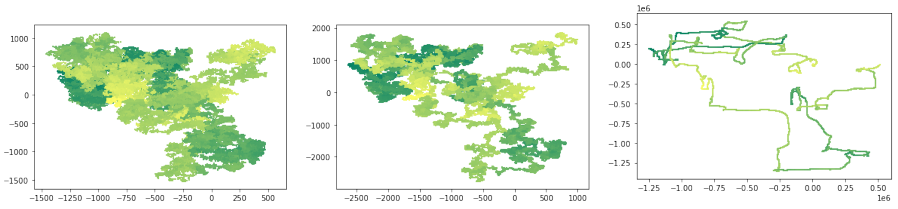
A Gaussian randomizer is added to the walker to set the distance the walker can jump to a direction in one turn. The pyton version of gaussian randomizer returns floats which I convert to ints in order to make the table with all numbers not to big to run.

Where the first version only jumped 1 in any direction this version jumps from a standard . This version picks a direction and jumps a distance using the follwing python formula for both x and y. The colouring formula is the same as in the previous version as the random walker. The formula is the following, the only change in the working version being that we cast it into an int.

$$n1 = gauss(self.world\_compression * -self.x\_change\_previous, 10) \tag{1}$$

Gauss is a randomizer in python that picks random numbers based on a bell curve. The 10 is the standard deviation and the $self.world\_compression * -self.x\_change\_previous$ is the starting point, always starting out at zero.

The x_change zero is a check what the result of the formula was the previous time we ran it. The world_compression is a number between 1 and -1 that decides how compressed(how close together is the landmass.) the world generation is. if the number is larger then 0 the world becomes more compressed due to the prevous sum being removed from the formula. If the number is lower then zero the world becomes less compressed. in the figure below(**??**) the landmass diffrent forms of landmass are shown.



(a) World_compression = 1 (b) World_compression = 0 (c) World_compression = -1

Figure 2: several world compressions shown

conclusion

The conclusion drawn from this expansion is the following, adding complexety to the movement algortihm makes control of world generation a lot easier. The final version of the randomizer needs to have a good balance between complexity and control needs to be researched in the final version.