Procedural generation with Perlin Noise in Python

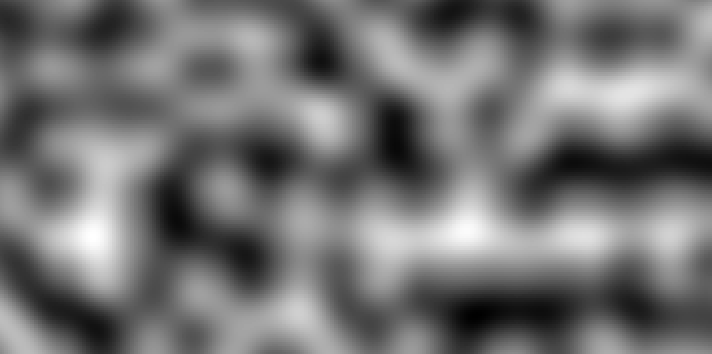
Abel - 2021年3月10日

介绍Introduction

We use this project mainly to simulate a 2d map. In this map, there are blue oceans, brown-yellow beaches, green grass, and gray mountains behind the grass.

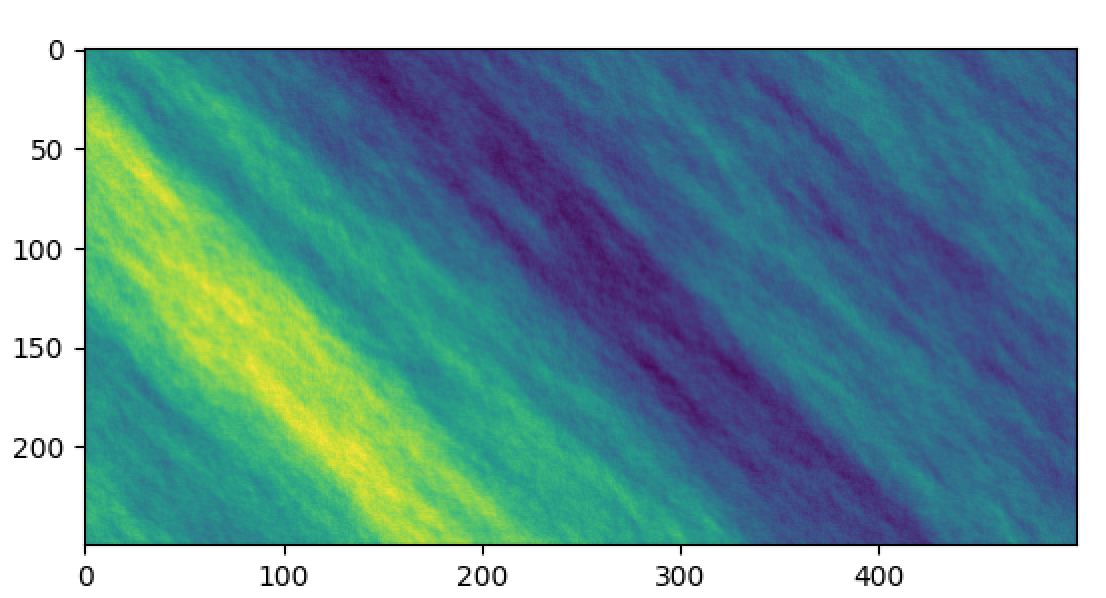
The environment is mainly an amphibious environment: part of the ecology is grass, forests and mountains, and part of the ecology is intertwined with beaches and oceans.



1. We first generate noise background map. I meanly use the ideal from the blog(https://medium.com/inspired-to-program-%E3%85%82-%D9%88-%CC%91%CC%91/procedural-generation-in-python-7b75127b2f74) . I defined a function I have created to create a layer of ‘noise’, and then to normalise that noise so it all sits between 0–1. I use this elsewhere in my code to assign an elevation to my maps.The length of each sub array is therefore the width of the map, and the total number of sub arrays is it’s height (which are also the two arguments that this function makes, to ensure that the noise map generated is the same size as the map itself).
2. It’s not enough, we need colors. So I add color range into it .generate\_noise starts by creating an array of arrays, and filling it with 0s. I don’t technically need to do this, as the way in which this function adds noise is always from the top left of the array, across and down, so could just append. However, this ensures that there’s the right number of elements in the array, and also allows me to tweak the following code so that I add noise from various points.

After all this done correctly , I use matplotlib.pyplot lib’s drawing function to

draw the back-noise-map. It looks like:



3.Different environments in above image (ocean, beach, forest, mountains) are not sufficiently distinguished. My analysis should be that the gradient of the color needs to be boosted.

while I have never observed natural map near terrain that looks like this if we look at any one part of the map it seems ‘realistic.’ Let’s take it a step further and add mountains/beachs, make them more easy to discover regions from sky.

**blue = [65,105,225]**

**green = [34,139,34]**

**beach = [238, 214, 175]**

**snow = [255, 250, 250]**

**mountain = [139, 137, 137]**

**house\_positions = []**

**def add\_color(world):**

**color\_world = np.zeros(world.shape+(3,))**

**for i in range(shape[0]):**

**for j in range(shape[1]):**

**if world[i][j] < -0.05:**

**color\_world[i][j] = snow**

**elif world[i][j] < 0.1:**

**color\_world[i][j] = mountain**

**elif world[i][j] < .20:**

**color\_world[i][j] = green**

**house\_positions.append((j,i))**

**elif world[i][j] < 0.35:**

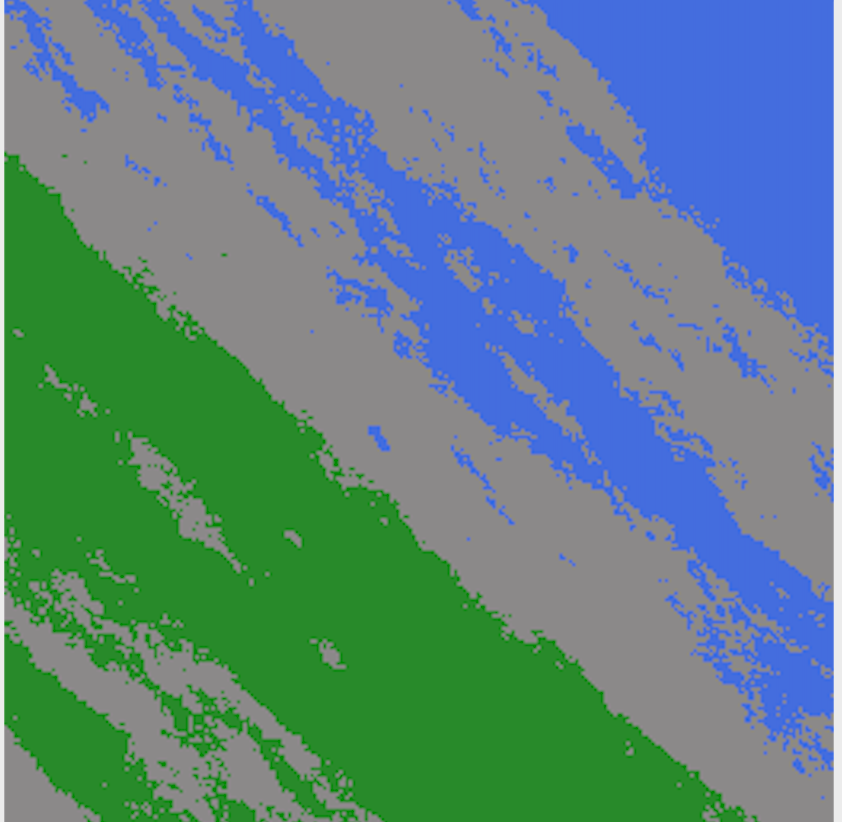
**color\_world[i][j] = beach**

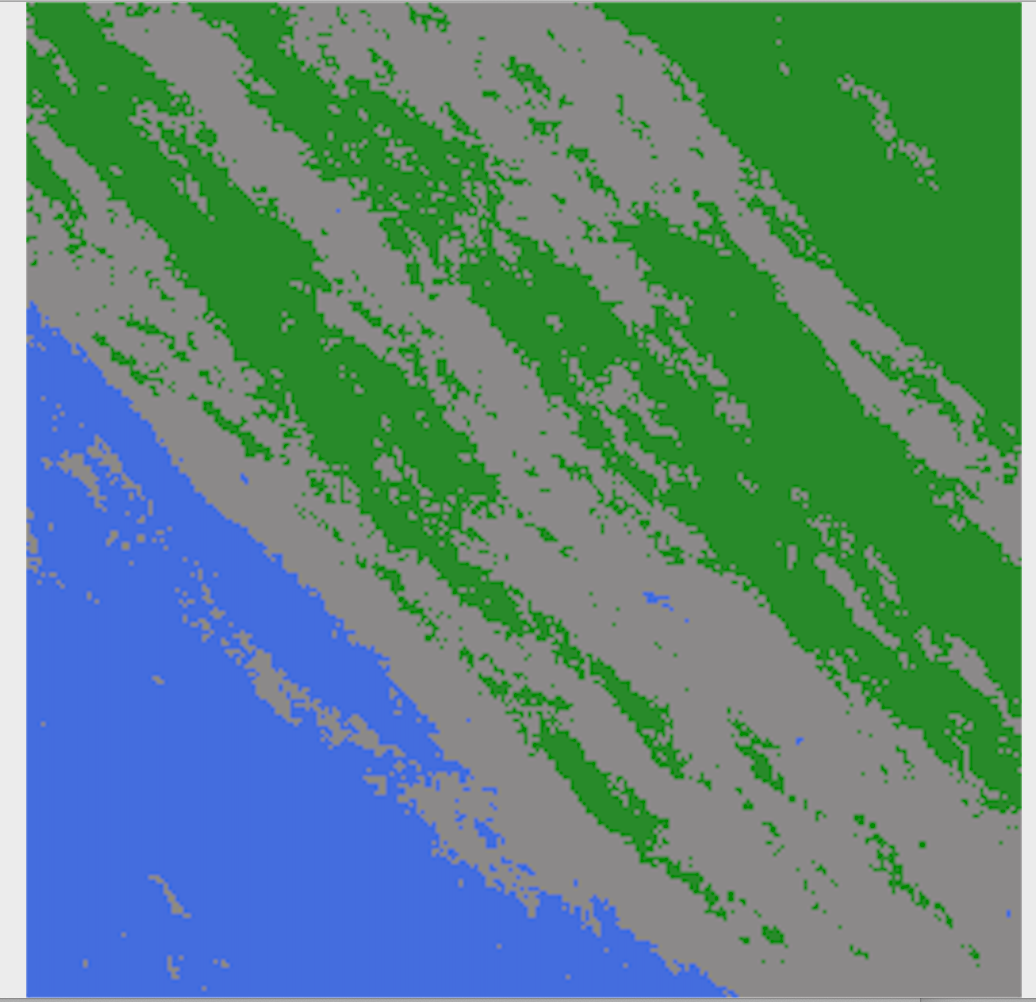
**elif world[i][j] < 1.0:**

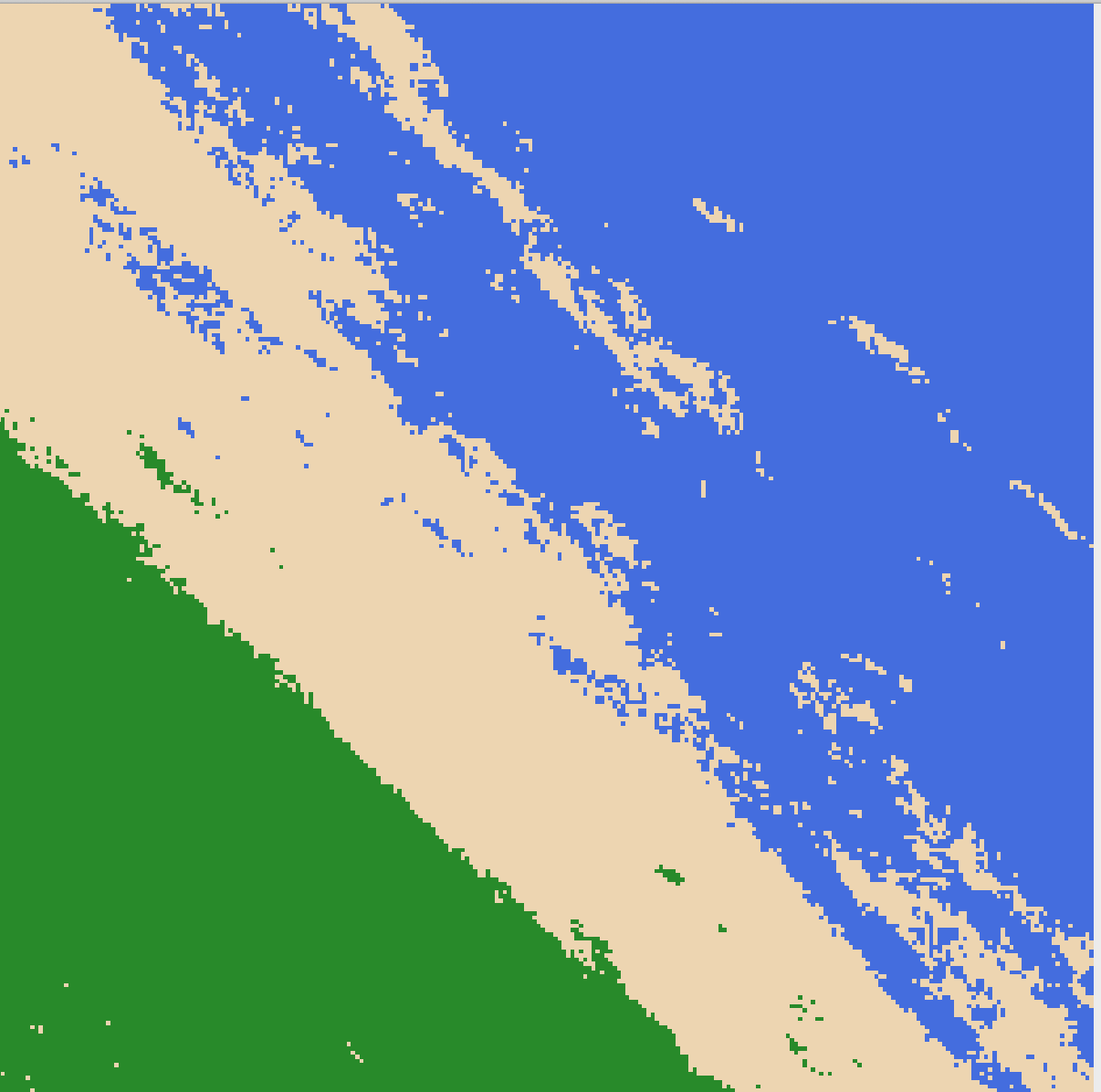
**color\_world[i][j] = blue**

**return color\_world**

After make the map into different group of things, we make the map more divided, like nature did:





4. we fixed some noise range in noise function , make color range more splited:

5. we then Small different colored rectangles to indicate houses . In this steps we need make sure that all houses on beachs/ green grass / moutains, but not in oceans.

so we need to pick random N(in this script , N=5) positions from those type of areas:

**house\_count = 5**

**real\_house\_positions = []**

**colors = ['red', 'blue', 'DarkKhaki', 'yellow', 'pink']**

**for i in range(house\_count):**

**# Create a Rectangle patch**

**rect = patches.Rectangle(house\_positions[len(house\_positions)\*i//house\_count], 10, 10, linewidth=5,**

**edgecolor=colors[i], facecolor='none')**

**real\_house\_positions.append(house\_positions[len(house\_positions)\*i//house\_count])**

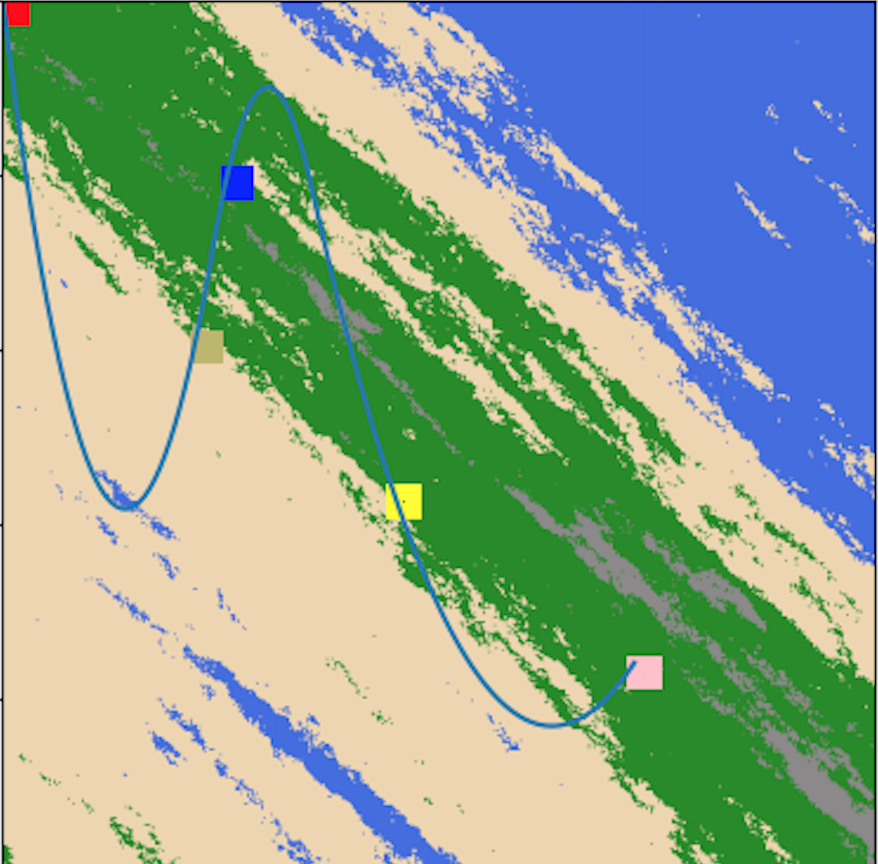
**# Add the patch to the Axes**

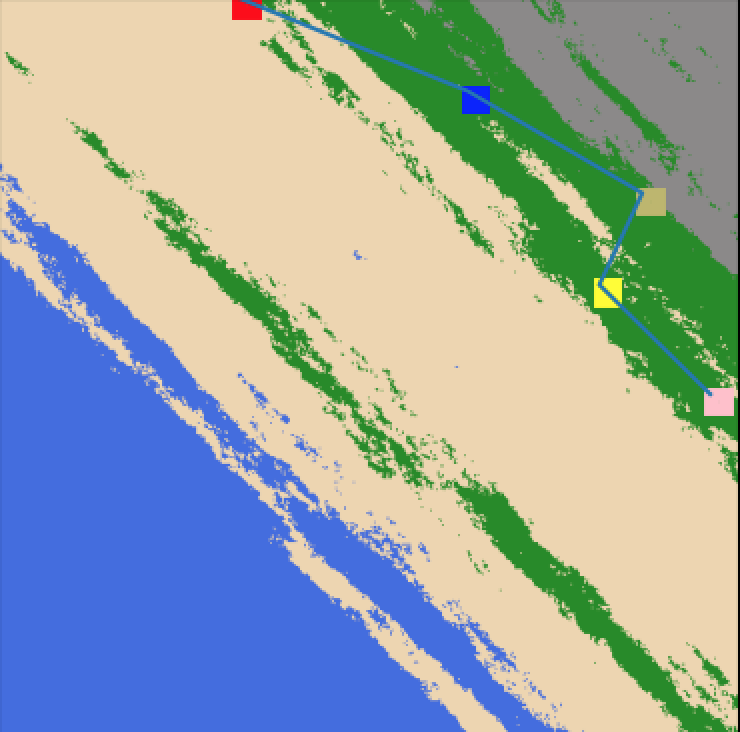
**ax.add\_patch(rect)**

6. The roads are connected (the roads will only pass on beaches, mountain peaks, and grassland without ocean separation), and all the houses are built in non-ocean areas, and the houses can be accessed from each other through roads

We then make road ways connecting said houses .In this steps ,we add two type of paths: smooth curves , and more direct lines.

The result of these 2 type of lines and houses append is like





**Conclusion**

So we started with some simple noise idea and ended up with a way to generate a realtively unlimited number of unique and natural looking archipelagos(with ocean, banch , grass, mountain, houses, roads with houses), 7 elements. Randomness is already consided, the result of my program shall not be deterministic, so that it can be used to generate different maps.