Procedural Terrain Generation

NSD Final Project Presentation

311552013 林予晨

Table of Contents

Introduction

Procedural Generation

Procedural Generation is a method of creating data algorithmically as opposed to manually, typically through a combination of human-generated assets and algorithms coupled with computer-generated randomness and processing power.

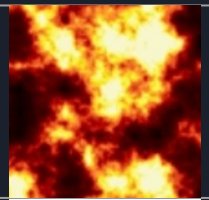
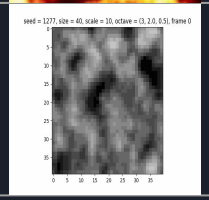
In video games, **Procedural Terrain Generation** is very useful where you want to generate natural terrain (caves, hills, rivers, etc.) that has a smooth feel, but is still random.

Perlin Noise

Perlin Noise is a very popular algorithm for procedural generation developed by Ken Perlin in 1983. It can be used for any kind of wave-like, undulating material, texture, or terrain.

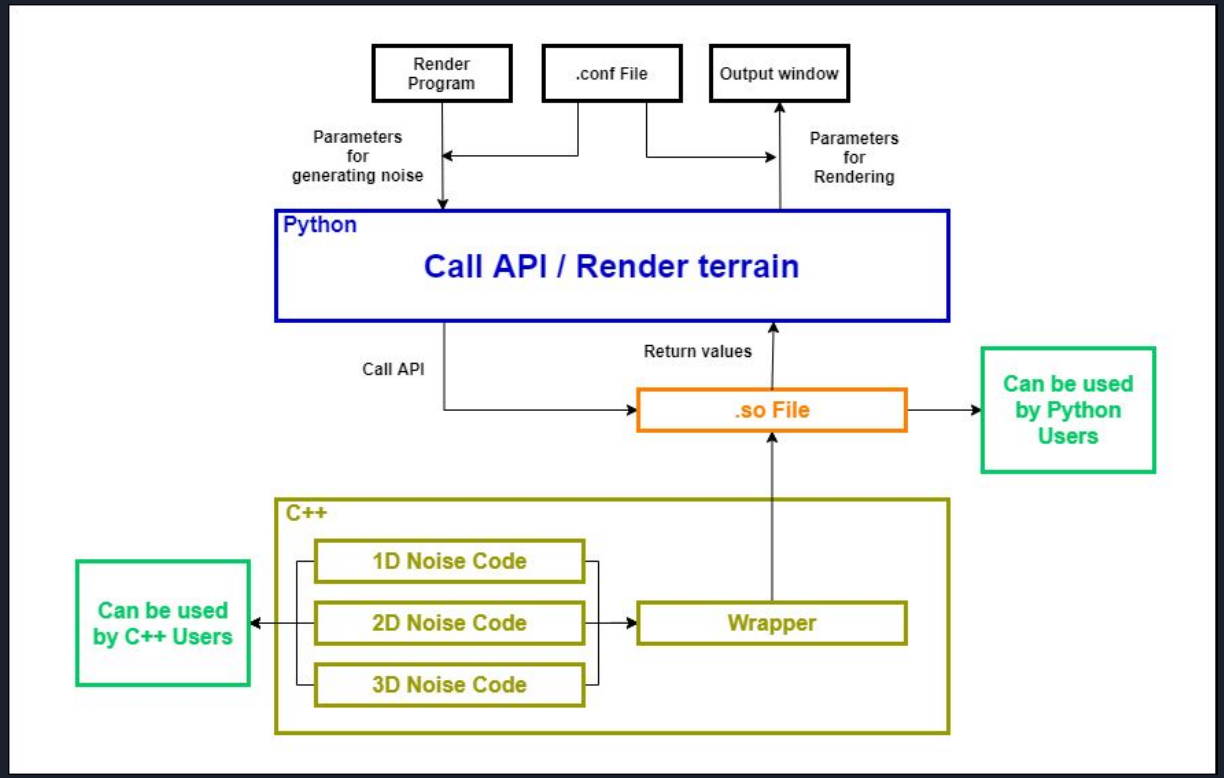
Compared with just some random values, Perlin Noise can generate values very smoothly and continuously, which looks more realistic in terrain generation.

Application Examples

1D		Handwritten effect		
2D		Terrain generation		Procedural texture
3D		3D object generation		2D animation

Implemetation

System Architecture



Engineering Infrastructure

- Automatic build system : GNU Make
- Version control : git
- Testing framework : pytest
- Documentation : README.md in the github repository

Build System

This system is built by makefile. Makefile targets include :

Target	Description
(all)	compile a `noise.so` in `noise/` folder.
test	use `pytest` to test python API.
graph1	display 1D noise testing graph.
graph2	display 2D noise testing graph.
graph3	display 3D noise testing graph and generate a `noise3d.gif` file in `test/` folder.
render	Take parameters in `render.conf`, generate `heightmap.png` and `colormap.png`, and render terrain by `Ursina`.
clean	remove all generated files (*.so, __pycache__/, etc).

APIs

The APIs will have both C++ and Python version in 1D/2D/3D.

- Create class object
- Create class object with detail options (scale, octaves, lacunarity, persistance)
- Get noise value at specific position
- Get noise value list
- Get single parameter value (seed, xsz, ysz, zsz, scale, octaves, lacunarity, persistance)

Testing

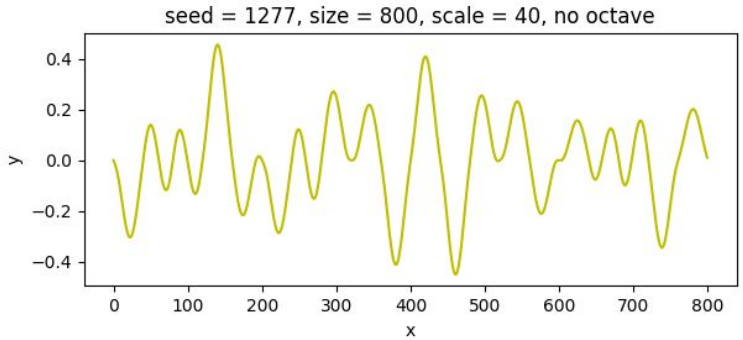
Using Pytest

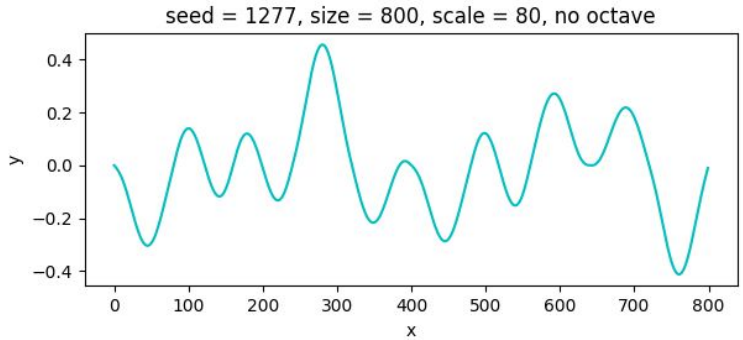
- testAttributes() : check if functions return correct values
- testSeed() : check if two same seeds generate same noise values
- drawGraph() : display noise values with different parameters

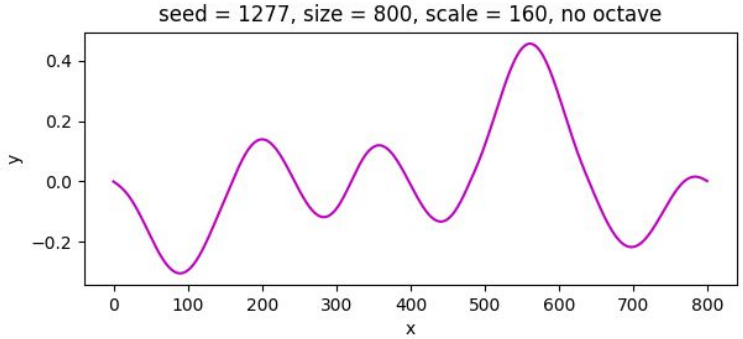
```
ubuntu20@ubuntu20-VirtualBox:~/Desktop/nsd/procedural-terrain-generation$ make test
python3 -m pytest test/ -v
============================================================================================= test session starts ====
platform linux -- Python 3.8.10, pytest-7.2.2, pluggy-1.0.0 -- /usr/bin/python3
cachedir: .pytest_cache
rootdir: /home/ubuntu20/Desktop/nsd/procedural-terrain-generation
collected 6 items

test/test_1d.py::testAttributes PASSED
test/test_1d.py::testSeed PASSED
test/test_2d.py::testAttributes PASSED
test/test_2d.py::testSeed PASSED
test/test_3d.py::testAttributes PASSED
test/test_3d.py::testSeed PASSED

============================================================================================== 6 passed in 1.61s =====
ubuntu20@ubuntu20-VirtualBox:~/Desktop/nsd/procedural-terrain-generation$
```

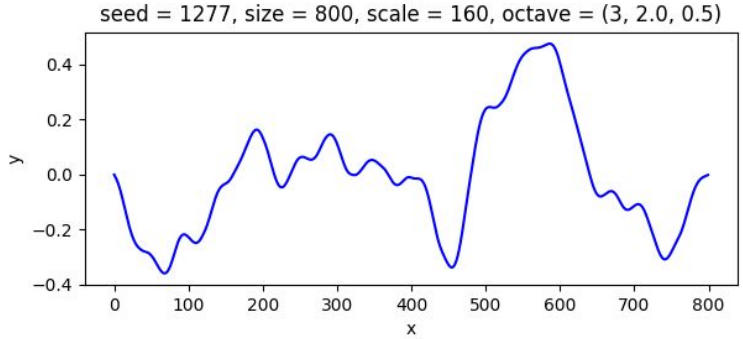
Result

1D Noises


seed = 1277, size = 800, scale = 40, no octave
y
x
0.4
0.2
0.0
−0.2
−0.4
0
100
200
300
400
500
600
700
800

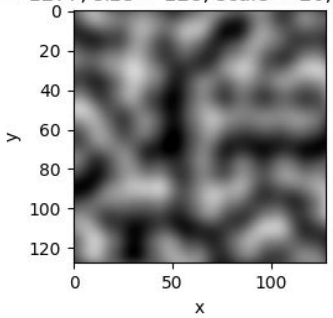

seed = 1277, size = 800, scale = 80, no octave
y
x
0.4
0.2
0.0
−0.2
−0.4
0
100
200
300
400
500
600
700
800


seed = 1277, size = 800, scale = 160, no octave
y
x
0.4
0.2
0.0
−0.2
0
100
200
300
400
500
600
700
800

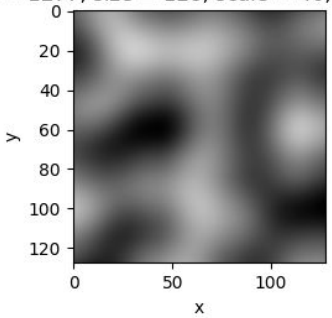

seed = 1277, size = 800, scale = 160, octave = (3, 2.0, 0.5)
y
x
0.4
0.2
0.0
−0.2
−0.4
0
100
200
300
400
500
600
700
800

2D Noises

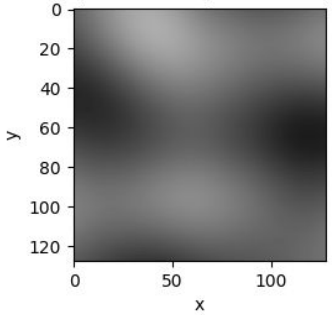
seed = 1277, size = 128, scale = 20, octave = 1



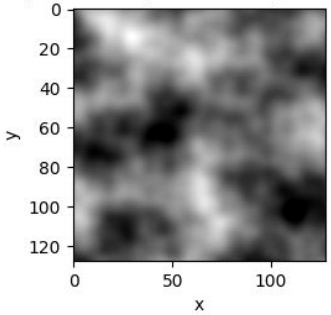
seed = 1277, size = 128, scale = 40, octave = 1



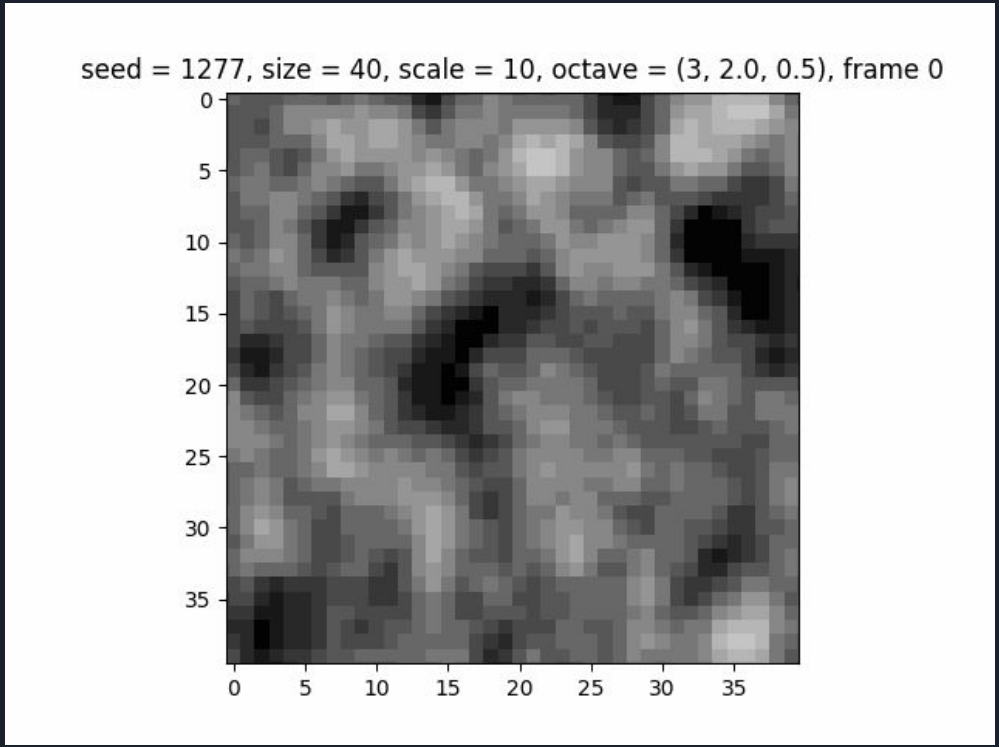
seed = 1277, size = 128, scale = 80, octave = 1



seed = 1277, size = 128, scale = 40, octave = (3, 2.0, 0.5)



3D Noises


seed = 1277, size = 40, scale = 10, octave = (3, 2.0, 0.5), frame 0
0
5
10
15
20
25
30
35
0
5
10
15
20
25
30
35

Terrain

Thank You