FANTASY CITY: GENERATING TERRAINS AND CITIES FROM BASIC GEOMETRIES

Thu Hang Nguyen
S5080488
CATA Level 4

I. INTRODUCTION

Fantasy City is a Maya Python script that creates a surreal, futuristic-looking cities from basic 3D geometries such as cones, cubes, cylinders, pipes,...etc. Its algorithm is based on the control of geometry index rules in Maya, and maps generating from noise. The script allows user to pass in the size of base terrain; coverage, intensity and level of three map layers; and some parameters for the port around the city.

II. ALGORITHM DESIGN

1. Main idea

The fantasy city generated from this script contains a main (base) terrain with small similar terrains around it, a port circling around the main terrain with bridges connecting the two objects, streets and highways running across the terrain, and buildings.

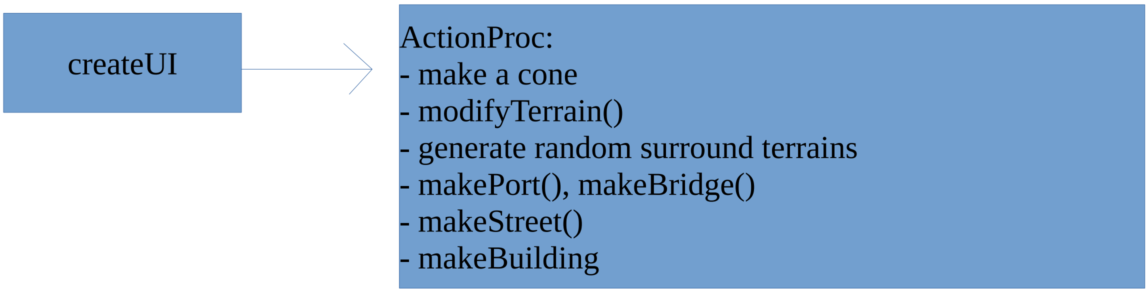
To create the terrain, *Fantasy City* applies Perlin noise on a cone geometry. It retrieves height map values from Perlin noise library, gives the values more variety by calculating simple operations with input from user. Finally, the program stores them in a list called *layer*, then applies height maps on each part of the cone separately.

For the port, this script builds a curve from the terrain's vertices, then extrudes a cube along this curve. The bridges, highways and streets are also created using this method. Therefore, we know the position of each point on the bridges, highways and streets based on the position of the terrain's vertices.

To generate buildings, the script picks up the vertices on the terrain that are not used to make curves (so that buildings will not collide with streets and highways), then makes buildings in those positions. The building is an external object imported from *Artefacts* folder.

2. Algorithm flow

Fantasy City is coded in Procedural Programming paradigm, every controls and calculations are wrapped up in functions. The main program first enters createUI to open GUI window. After user clicking OK button, the program feeds input to actionProc and generates artefacts. Some small functions are defined inside a function for short repeating codes.



createUI

ActionProc:
- make a cone
- modifyTerrain()
- generate random surround terrains
- makePort(), makeBridge()
- makeStreet()
- makeBuilding

3. External Library

The script makes use of Perlin noise – a type of gradient noise that has a lot of application in computer graphics. The full noise library used in this script can be downloaded here: https://github.com/caseman/noise

External library is stored in the directory named *Artefacts*.

III. USER MANUAL

1. Load the script

Place Fantansy City script folder in your directory. Then load the script into Maya Script Editor's Python tab. In order to load external library and objects successfully, user need to change the path to their *Artefacts* folder manually:

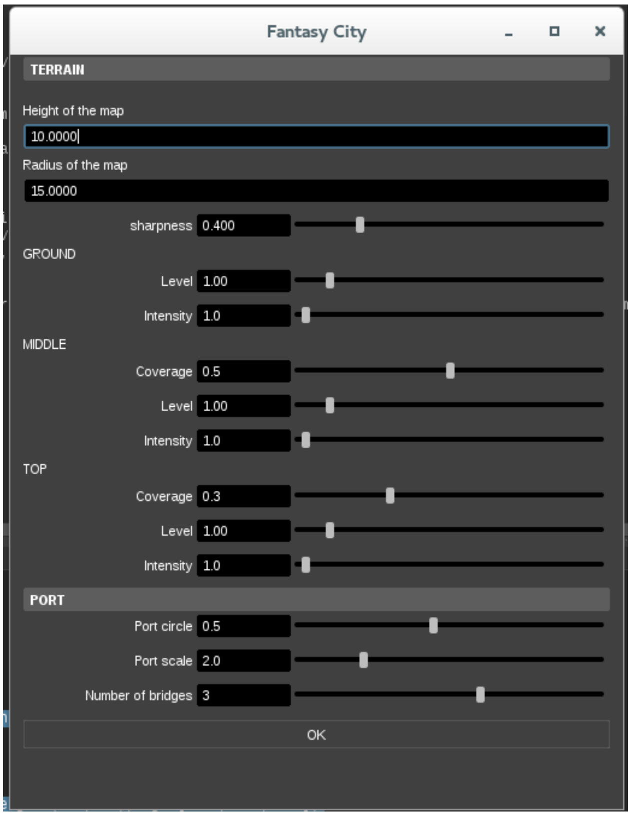
at the beginning of the script:

```
#please change your directory here
sys.path.append('YOUR PATH HERE')
```

And inside makeBuiding function:

```
def makeBuilding(terrain, sX, sZ, coverage):
…
        #please change your directory here
        pathName = 'YOUR PATH HERE'
```

Then run the script using Maya Script Editor, it will open a GUI window.



TERRAIN

Height of the map 10.0000

Radius of the map 15.0000

sharpness 0.400

GROUND

Level 1.00

Intensity 1.0

MIDDLE

Coverage 0.5

Level 1.00

Intensity 1.0

TOP

Coverage 0.3

Level 1.00

Intensity 1.0

PORT

Port circle 0.5

Port scale 2.0

Number of bridges 3

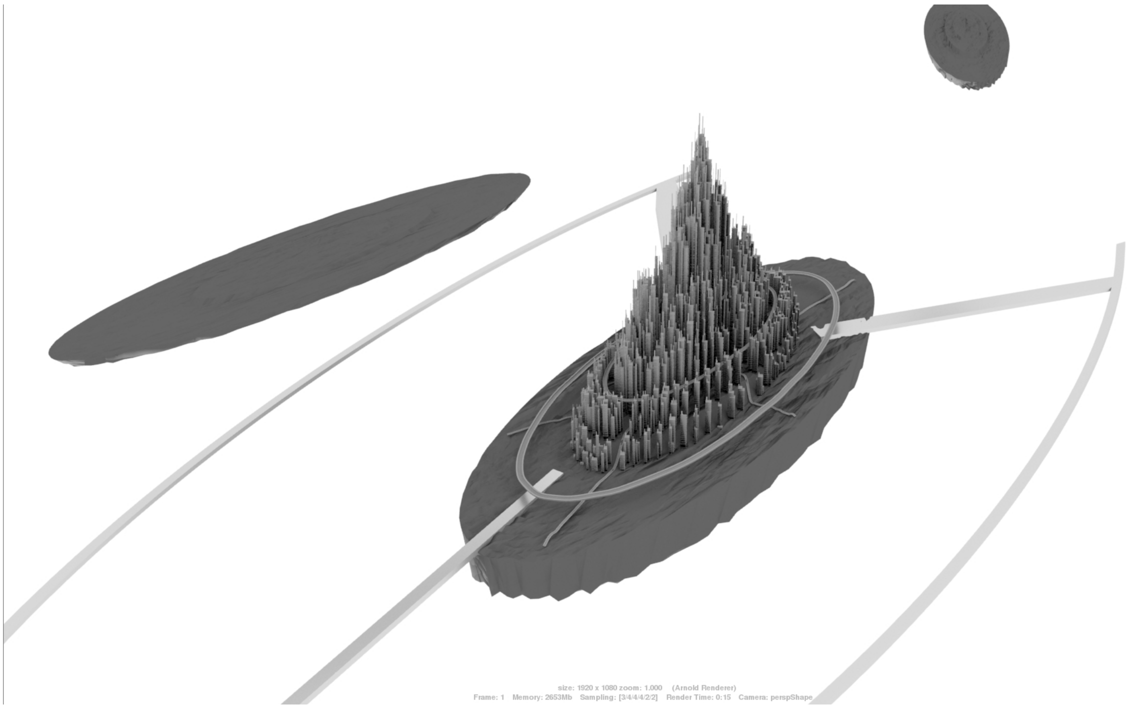
2. Enter input

Users are allowed to pass parameters controlling the shape of terrains and port.

For a more dramatic result, users can increase *level* and *intensity* of middle and top layer – these values will raise the centre area to form a sharper looking terrain.

After clicking OK button, the city will be generated and put under '*City*' group. The script will print out 'Build City Done' if it runs successfully.

IV. RESULTS


size: 1920 x 1080 zoom: 1.000 (Arnold Renderer)
Frame: 1 Memory: 2653Mb Sampling: [3/4/4/4/2/2] Render Time: 0:15 Camera: perspShape