# Python User Manual

The Python script `Generating.py` generates an image file with random colors in the PPM format. The image will contain a sequence of 'windows', with each window getting progressively smaller based on the input percentages. The colors of each window follow a normal distribution, with the standard deviation for each window's colors adjusted according to the relevant percentage. ## Requirements

- Python 3.5 or later.

Usage

1. Navigate to the directory containing the script in your terminal or command prompt.
2. Run the script with the required arguments in the following format:

```
python Generating.py output_image_name.ppm w h percentage1 percentage2 ... percentageN
```

The arguments are explained as follows:

- `output_image_name.ppm`: The name of the output PPM file.
- `w`: The width of the output image, in pixels.
- `h`: The height of the output image, in pixels.
- `percentage1`, `percentage2`, …, `percentageN`.

The sequence of percentages which determine the size of each window in the image. The number of percentages also determines the number of windows in the image. For example, if `percentage1` is 0.5 and `percentage2` is 0.4, there will be three windows: the first will be the full size of the image, the second will be half the size of the image, and the third will be 40% of the size of the second window.

Here is an example of running the script:

```
python Generating.py random_image.ppm 1000 1000 0.5 0.4
```

This will generate a PPM image named `random_image.ppm` of size 1000x1000 pixels, with three windows. The first window is of full size (1000x1000), the second window is half the size of the image (500x500), and the third window is 40% the size of the second window (200x200).

After running the script, you should find the generated PPM file in the same directory as the script. You can open this file with any software that supports the PPM format.

# C ++ User Manual

The C++ program generates an image file with random colors in the PPM format. The image will contain a sequence of 'windows', with each window getting progressively smaller based on the input percentages. The colors of each window follow a normal distribution, with the standard deviation for each window's colors adjusted according to the relevant percentage.

The program also supports parallel computing through OpenMP, which can significantly speed up execution on systems with multiple cores or processors.

Requirements

- A modern C++ compiler that supports C++11 or later.
- OpenMP library for parallel computing (optional).

Usage

1. Navigate to the directory containing the script in your terminal or command prompt.
2. Compile the C++ code. If you want to enable OpenMP for parallel computing, you need to set the appropriate flag when compiling:

   ```
   g++ -fopenmp -std=c++11 generation_image.cpp -o your_program
   ```

   If you don't want to use OpenMP, you can compile without the `-fopenmp` flag:

   ```
   g++ -std=c++11 generation_image.cpp -o your_program
   ```

3. Run the compiled program with the required arguments:

   ```
   ./your_program output_image_name.ppm width height percentage1 percentage2 ... percentag
   ```

   The arguments are explained as follows:

   - `output_image_name.ppm`: The name of the output PPM file.
   - `width`: The width of the output image, in pixels.
   - `height`: The height of the output image, in pixels.
   - `percentage1`, `percentage2`, …, `percentageN`: The sequence of percentages which determine the size of each window in the image.

Here is an example of running the program:

```
./your_program random_image.ppm 1000 1000 0.5 0.4
```

This will generate a PPM image named `random_image.ppm` of size 1000x1000 pixels, with three windows. The first window is of full size (1000x1000), the second window is half the size of the image (500x500), and the third window is 40% the size of the second window (200x200).

4. After running the program, you should find the generated PPM file in the same directory as the script. You can open this file with any software that supports the PPM format.

Notes on OpenMP

OpenMP is a library for parallel computing in C++, among other languages. If you compile your program with the `-fopenmp` flag, the program will use OpenMP to fill the matrix with random colors in parallel, using multiple threads, by default this program uses half of the threads of the machine.

This can potentially speed up execution of your program if you are running on a system with multiple cores or processors, as different parts of the matrix can be filled in parallel.

If you compile without the `-fopenmp` flag, the program will run without parallel computing, and only one core will be used to fill the matrix. For small images, the time it takes to generate the image does not vary significantly.