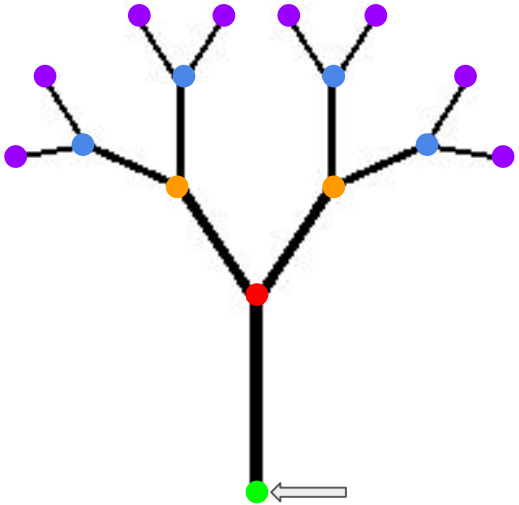
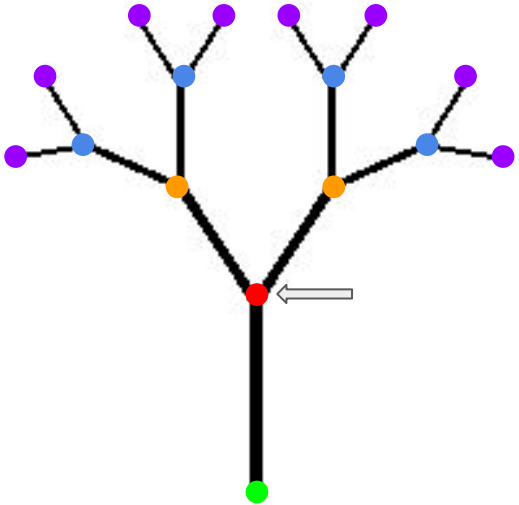


Dans la fonction n°1, appel de la fonction n°2

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

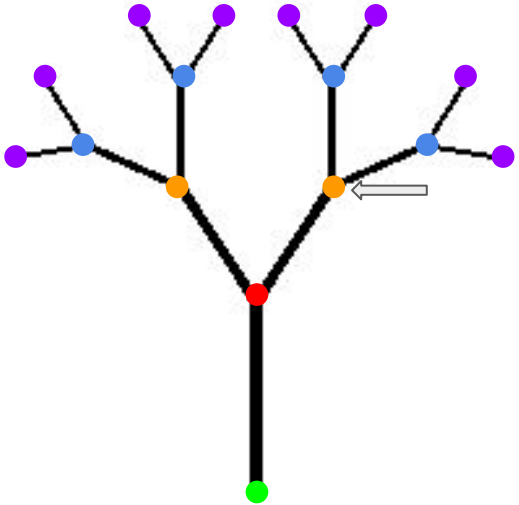
Code exécuté



Dans la fonction n°2, appel de la fonction n°3

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

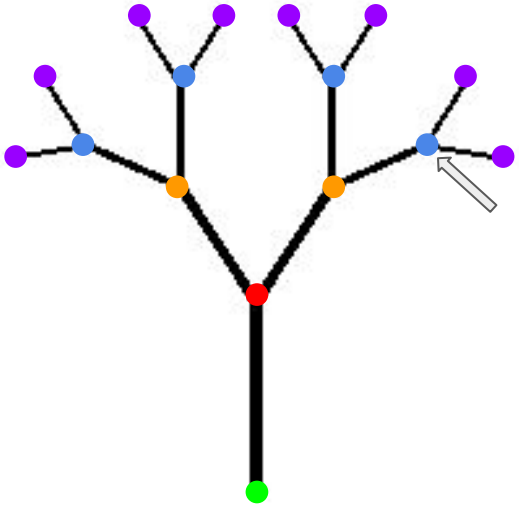
Code exécuté



Dans la fonction n°4, appel de la fonction n°5

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

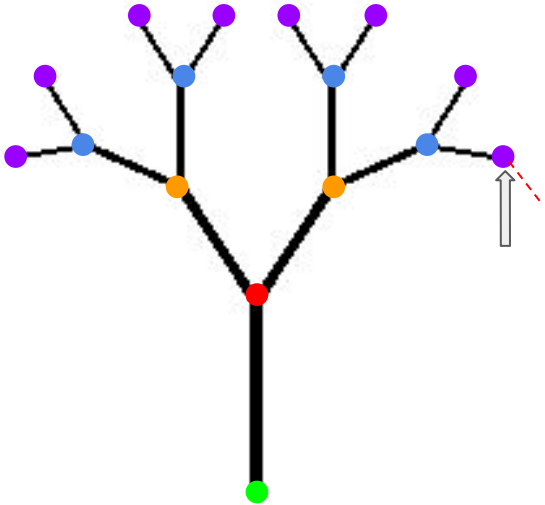
Code exécuté



Dans la fonction n°5, appel de la fonction n°6

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

Code exécuté

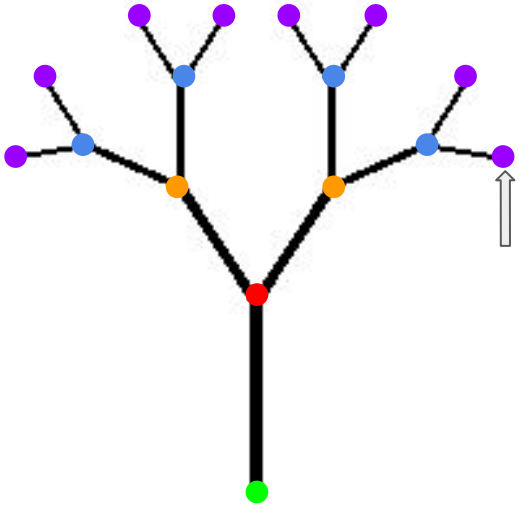


Dans la fonction n°6, pas d'appel

La condition est vraie, on n'exécute pas le code après donc pas de dessin de la branche de droite enfant

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

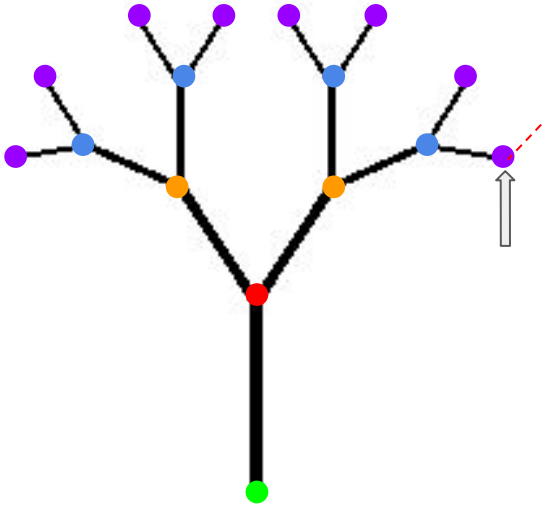
Code exécuté



Dans la fonction n°5, appel de la fonction n°7

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):

    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

Code exécuté

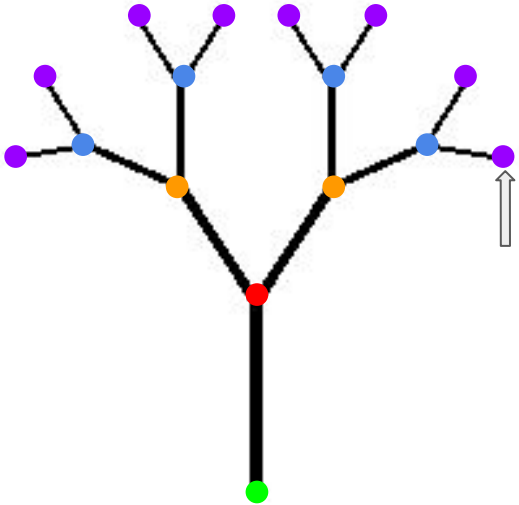


Dans la fonction n°7, pas d'appel

La condition est vraie, on n'exécute pas le code après donc pas de dessin de la branche de droite enfant

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):
    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

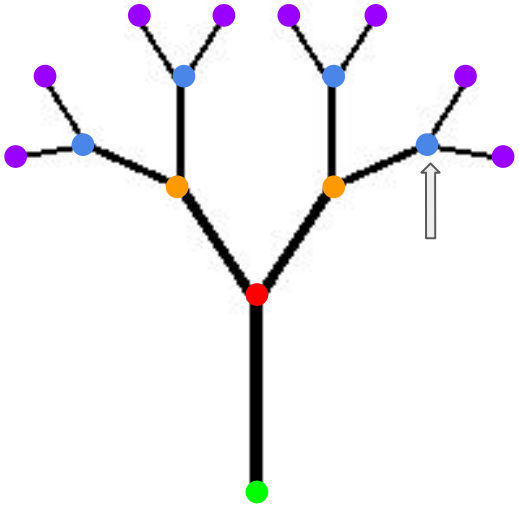
Code exécuté



Dans la fonction n°5

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):

    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

Code exécuté



Fermeture de la fonction n°5, on continue le code de la n°4: appel de la fonction n°8…

```python
def fractal_canopy(turtle, branch_length, branch_width, ratio, depth, angle):

    if depth == 0:
        return None

    turtle.width(branch_width)

    turtle.forward(branch_length)

    turtle.right(angle)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.left(angle * 2)

    fractal_canopy(turtle, branch_length * ratio, branch_width * ratio, ratio, depth - 1, angle)

    turtle.right(angle)

    turtle.backward(branch_length)
```

Code exécuté