Отчет по модификации проекта “Изображение проекции полиэдра”

Новиков Г. Д. Б24-215 10.05.25 Ссылка на репозиторий на GitHub github.com/Eaglem-greg/polyhedron

---

Описание задачи

Сущность задачи

Необходимо найти сумму площадей граней, которые имеют ровно одну “хорошую” точку, точку назовем “хорошей”, если ее проекция x > -2.

Математические аспекты

Гранью полиэдра является произвольный выпуклый многоугольник, для отыскания площади которого нам необходимо находить центр грани, затем строить вектора от этой точки до каждой вершины. Таким образом мы разбиваем многоугольник на несколько треугольников, затем, используя свойство векторного произведения, находим модуль векторного произведения для каждого треугольника и суммируем их площади. Сумма площадей этих треугольников и будет являться площадью данной грани. Площадь треугольника через векторное произведение можно найти таким образом:

$$\vec{a} = (x_a, y_a, z_a)$$
$$\vec{b} = (x_b, y_b, z_b)$$
$$S = \frac{1}{2} \cdot \|\vec{a} \times \vec{b}\|$$
$$\|\vec{a} \times \vec{b}\| = \sqrt{(y_a z_b \quad z_a y_b)^2 + (x_a z_b \quad x_b z_a)^2 + (x_a y_b \quad x_b y_a)^2}$$

Модификация проекта

Нам необходимо учитывать проекцию каждой точки по x, для этого добавим метод **check** в класс **Facet**, извлекая каждую вершину из массива, мы будем проверять первую ее координату, если $x > 2$, то увеличиваем счетчик на 1, будем возвращать **True**, в случае, когда счетчик будет равен 1, в противном случае **False**.

Код метода **check**:

```
def check(self):
    s = 0
    for egde in self.vertexes:
        if egde.x > -2:
            s += 1
    return s == 1
```

Затем реализуем методы, принцип которых описан в математических аспектах, для вычисления площади грани в классах **Facet** и **R3**.

Код метода **area**:

```
    def area(self):
        if len(self.vertexes) < 3:
            return 0.0
        normal = self.h_normal()
        total_area = 0.0
        center = self.center()
        for i in range(len(self.vertexes)):
            v1 = self.vertexes[i] - center
        v2 = self.vertexes[(i + 1) % len(self.vertexes)] - center
            total_area += 0.5 * v1.cross(v2).length()
        return total_area
```

Код метода **length**:

```
    def length(self):
       return sqrt(self.x ** 2 + self.y ** 2 + self.z ** 2)
```

Приступим к модификации класса **Polyedr**, создадим метод, который извлекает из первой строки текстового документа значение коэффициента гомотетии, он необходим чтобы в определенное количество раз удлинять ребра при их построении для наглядности.

Код метода **take_homotetia_koef**:

```
    def take_homotetia_koef(self):
        with open(self.file) as f:
            first_line = f.readline()
            return float(first_line.split()[0])
```

Осталось только посчитать сумму площадей граней, которые удовлетворяют нашему условию, и поделить ее на квадрат коэффициента гомотетии, для этого в реализованном методе **total_area** пройдемся по списку граней. Если грань подходит под условие, то прибавляем ее площадь. В конце разделим на квадрат коэффициента гомотетии, который мы получаем из метода **take_homotetia_koef**.

Код метода **total_area**:

```
    def total_area(self):
        ch = self.take_homotetia_koef()
        return sum(facet.area() for facet in self.facets
                if facet.check()) / (ch ** 2)
```

Фрагменты тестов

Сначала рассмотрим новые тесты для класса R3:

```python
def test_length0(self):  Eaglem-greg
    unit_x = R3( x: 1.0,  y: 0.0,  z: 0.0)
    unit_y = R3( x: 0.0,  y: 1.0,  z: 0.0)
    unit_z = R3( x: 0.0,  y: 0.0,  z: 1.0)
    self.assertEqual(unit_x.length(),  second: 1.0)
    self.assertEqual(unit_y.length(),  second: 1.0)
    self.assertEqual(unit_z.length(),  second: 1.0)
```

Для класса `Polyedr` были добавлены тесты для методов `total_area` и `take_homotetia_koef`:

```python
def test_total_area(self):  Eaglem-greg *
    with patch.object(Polyedr,  attribute: 'take_homotetia_koef', return_value=200.0), \
         patch.object(self.polyedr.facets[0],  attribute: 'check', return_value=False), \
         patch.object(self.polyedr.facets[1],  attribute: 'check', return_value=False), \
         patch.object(self.polyedr.facets[2],  attribute: 'check', return_value=False), \
         patch.object(self.polyedr.facets[3],  attribute: 'check', return_value=False):
         expected_area = 0.0
         result = self.polyedr.total_area()
         self.assertTrue(isclose(result, expected_area))

def test_take_homotetia_koef(self):  Eaglem-greg
    self.assertEqual(self.polyedr.take_homotetia_koef(),  second: 200)
```

Для класса `Facet` были добавлены тесты для методов `area` и `check`:

```python
def test_check(self):  Eaglem-greg
    f = Facet([R3(-2.0,  y: 2.0,  z: 0.0), R3(-2.0, -2.0,  z: 0.0), R3(-4.0,  y: 0.0,  z: 0.0), R3( x: 0.0,  y: 0.0,  z: 0.0)])
    self.assertTrue(f.check())

def test_check1(self):  Eaglem-greg
    f = Facet([R3( x: 3.0,  y: 2.0,  z: 0.0), R3(-2.0, -2.0,  z: 0.0), R3(-4.0,  y: 0.0,  z: 0.0), R3( x: 0.0,  y: 0.0,  z: 0.0)])
    self.assertFalse(f.check())

def test_area(self):  Eaglem-greg
    f = Facet([R3( x: 0.0,  y: 0.0,  z: 0.0), R3( x: 1.0,  y: 0.0,  z: 0.0), R3( x: 0.0,  y: 1.0,  z: 0.0)])
    expected_area = 0.5
    self.assertTrue(isclose(f.area(), expected_area))

def test_area1(self):  Eaglem-greg
    f = Facet([R3(-2.0,  y: 2.0,  z: 0.0), R3(-2.0, -2.0,  z: 0.0)])
    expected_area = 0.0
    self.assertTrue(isclose(f.area(), expected_area))
```
