Отчёт по модификации проекта Полиэдр

Соколов Пётр, Б23-205

2024

Данный отчёт посвящён модификации эталонного проекта “Полиэдр”. Решалась по нахождению суммы длин рёбер, удовлетворяющих определённому условию.

# Содержание

* [Содержание](#содержание)
* [Команды для преобразования формата](#команды-для-преобразования-формата)
* [Постановка задачи](#постановка-задачи)
* [Описание решения](#описание-решения)
  + [Изменения в классе R3](#изменения-в-классе-r3)
  + [Изменения в классе Polyedr](#изменения-в-классе-polyedr)
  + [Изменения в data и run\_shadow.py](#изменения-в-data-и-run_shadowpy)
* [Тесты](#Тесты)

# Команды для преобразования формата

1. pandoc -s report.md -o report.docx — для преобразования в *docx* файл;
2. pandoc report.md -s --self-contained --template=default.html5 --mathjax -o report.html — для преобразования в *html* файл;
3. pandoc -s report.md -o report.pdf --template default.latex — для преобразования в *pdf* файл.

# Постановка задачи

В данном проекте выполнялась задача 17 (список задач предоставлен по [ссылке](https://edu-support.mephi.ru/materials/293/html/labs/10/problems.md)).

Условие задачи звучит так: Назовём точку в пространстве «хорошей», если её проекция находится строго внутри окружности , но строго вне окружности . Модифицируйте эталонный проект таким образом, чтобы определялась и печаталась следующая характеристика полиэдра: сумма длин рёбер, оба из концов которых не являются «хорошими» точками.

# Описание решения

Для решения поставленной задачи требовалось отслеживать попадание проекции каждой из вершин полиэдра в открытое кольцо с внутренним и внешним радиусами, равными 1 и 2 соотвественно (см. рис.1). После, если две данные вершины, имеющие общее ребро не попали в кольцо, то нужно было находить длину ребра, которе их соединяло (рассматриваются именно пары смежных вершин) и добавлять в общую сумму.

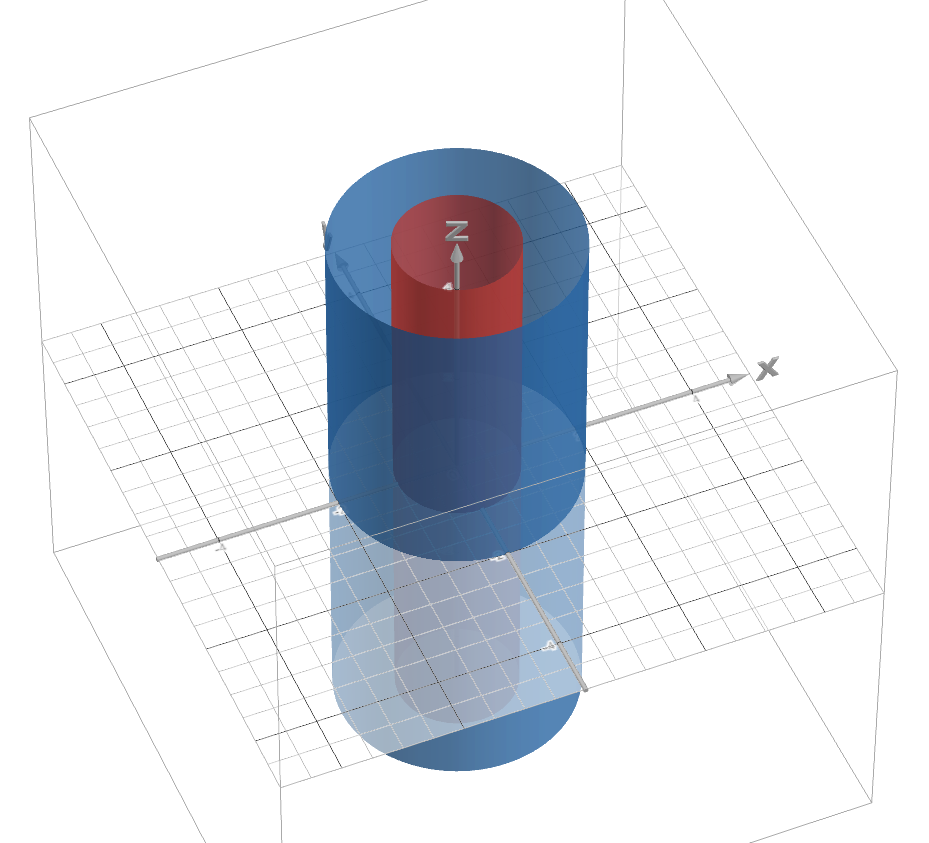


Рисунок 1

## Изменения в классе R3

Для этого в классе R3, в файле r3.py, добавили два метода:

1. in\_reg, метод проверяющий попадание заданной точки в открытое кольцо;
2. dist, метод находящий расстояние между точками (оно и есть длина ребра).

Ниже представлен код данных методов.

код *первого метода*:

def in\_reg(self):  
 if ((self.x\*\*2 + self.y\*\*2)<4) and ((self.x\*\*2 + self.y\*\*2)>1):  
 return True  
 else:  
 return False

код *второго метода*:

def dist(self, other):  
 x\_l = (self.x - other.x)\*\*2  
 y\_l = (self.y - other.y)\*\*2  
 z\_l = (self.z - other.z)\*\*2  
 return sqrt((x\_l + y\_l + z\_l))

## Изменения в классе Polyedr

Так как производилась модификация не оптимизированного кода эталонного проекта, то некоторые рёбра обрабатывались несколько раз. В таком случае, при работе с ними, мы бы получали неправильный результат.

Поэтому в класс был добавлен отдельный метод edges\_lenght, который сначала удалял все повторяющиеся рёбра, оставляя лишь один экземпляр, а после обрабатывал оставшиеся и длины тех, что соответствовали поставленному условию, добавлял уже в итоговую сумму. Стоит отметить, что данный метод работал с неизменёнными координатами вершин, так как по условию проеобразование координат не изменяет сам полиэдр.

ниже приведён код данного метода:

def edges\_lenght(self):  
 edges = {}  
 for e in self.edgest:  
 a = e.beg  
 b = e.fin  
 if (a, b) not in edges and (a, b) not in edges:  
 edges[(a, b)] = e  
 if not (a.in\_reg()) and not (b.in\_reg()):  
 self.s += a.dist(b)  
 return self.s

## Изменения в data и run\_shadow.py

Был добавлен ещё один полиэдр new\_box.geom, который потребовался для тестирования правильности работы метода из класса Polyedr.

В run\_shadow.py теперь отстраивается новый полиэдр и две окружности, составляющие открытое кольцо (см. рис.2), а также после каждого построения выводится искомая сумма.

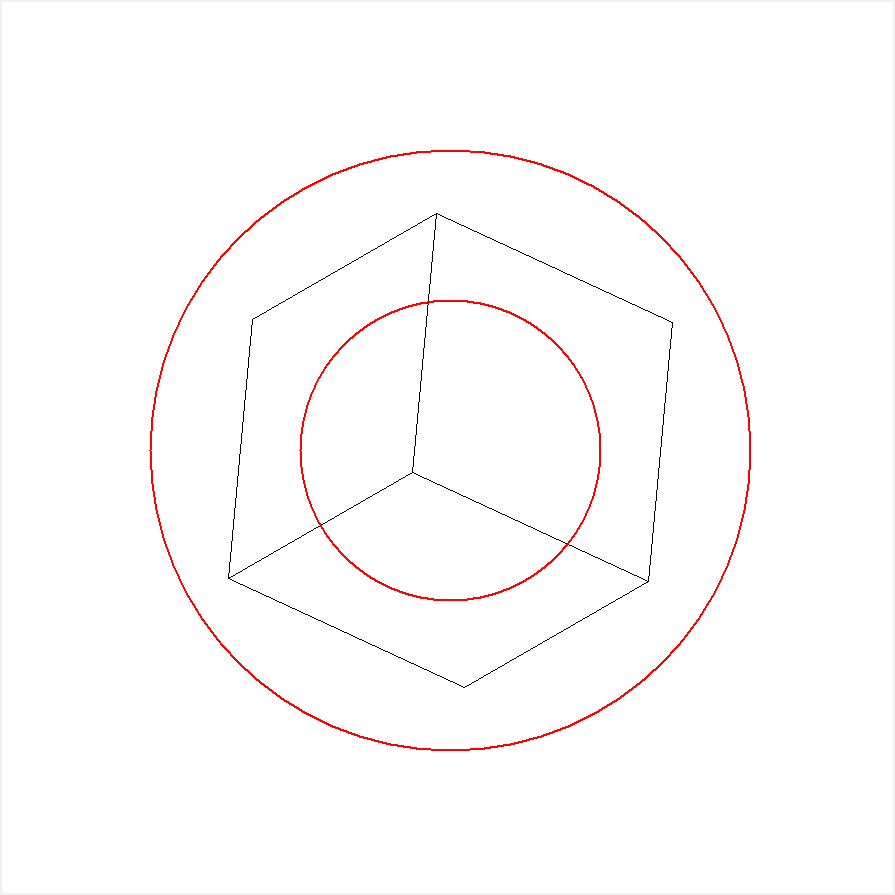


Рисунок 2

# Тесты

Добавленные тесты для класса R3, проверяющие высчитываемое расстояние и условие принадлежности вершин открытому кольцу.

def test\_dist01(self):  
 self.assertEqual(self.a1.dist(self.a2), 1)  
  
 def test\_dist02(self):  
 self.assertAlmostEqual(self.a1.dist(self.a3), sqrt(3))  
  
 def test\_in\_reg1(self):  
 self.assertTrue(self.a4.in\_reg())  
  
 def test\_in\_reg2(self):  
 self.assertFalse(self.a.in\_reg())

Добавленные тесты для класса Polyedr. Ниже приведены тесты, проверяющие правильность высчитывания суммы для каждого из данных полиэдров, включая новый new\_box.

def test\_sum01(self):  
 p = Polyedr('data/box.geom').edges\_lenght()  
 self.assertEqual(p, 12.0)  
  
 def test\_sum02(self):  
 p = Polyedr('data/ccc.geom').edges\_lenght()  
 self.assertEqual(p, 30.0)  
  
 def test\_sum03(self):  
 p = Polyedr('data/king.geom').edges\_lenght()  
 a = 35.29536857  
 self.assertAlmostEqual(p, a)  
  
 def test\_sum04(self):  
 p = Polyedr('data/babem.geom').edges\_lenght()  
 a = 2324.05376815  
 self.assertAlmostEqual(p, a)  
  
 def test\_sum05(self):  
 p = Polyedr('data/cow.geom').edges\_lenght()  
 b = 976.4371654314  
 self.assertAlmostEqual(p, b)  
  
 def test\_sum06(self):  
 p = Polyedr('data/new\_box.geom').edges\_lenght()  
 self.assertAlmostEqual(p, 0)