Отчёт о модификации проекта «Изображение проекции полиэдра»

Г.Е. Жмелев

12 июня 2024 года

Содержание

*

Постановка задачи

Назовём точку в пространстве «хорошей», если её проекция находится строго правее прямой $x = -2$.

Модифицируйте эталонный проект таким образом, чтобы определялась и печаталась следующая характеристика полиэдра: сумма площадей граней, ровно одна вершина которых — «хорошая» точка.

Ход решения

Вычисление площади

В процесе решения задачи потребуется отличать «хорошую» точку от обычной. Для этого определим для объектов класса `R3` атрибут `is_good`. Он принимает значение `True`, если точка «хорошая», а иначе — `False`. В том же классе определим два метода:

- `set_is_good()` — метод, который определяет, является ли точка «хорошей» (присваивая соответствующее значение атрибуту `is_good`).
- `magnitude()` — метод, который возвращает модуль вектора с началом в центре координат и концом в данной точке.

Для подсчёта площади грани определим метод `area()` в классе `Facet`. В начале следует создать функцию, которая будет возвращать площадь треугольной грани, заданной двумя векторами. Такая площадь определяется

по следующей формуле:

$$S = \frac{1}{2}|\vec{a} \times \vec{b}|$$

где $\vec{a}$, $\vec{b}$ — векторы, исходящие из одной точки. Получая на вход функции грань из произвольного числа вершин, следует разбить её на несколько треугольников и подсчитать площадь каждого. Суммарная площадь — искомая.

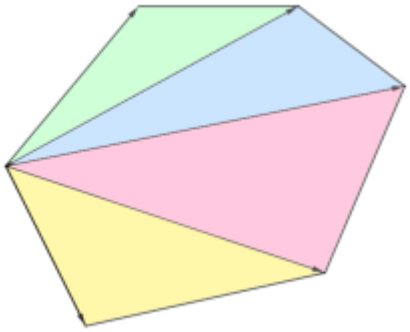

Рис. 1: Площадь данного многоугольника суммируется из площадей составляющих его треугольников (выделены разными цветами). Каждый из треугольников построен на двух векторах, исходящих из одной точки.

Реализация метода `area()`:

```
# Площадь грани
def area(self):
    # площадь треугольника, построенного на двух векторах
    def __triangle_area(edge_1, edge_2):
        return edge_1.cross(edge_2).magnitude() / 2
    area = 0.0
    # грань может иметь любое число вершин
    for i in range(1, len(self.vertexes) - 1):
        edge_1 = self.vertexes[0] - self.vertexes[i]
        edge_2 = self.vertexes[i+1] - self.vertexes[i]
        area += __triangle_area(edge_1, edge_2)
    return area
```

При чтении данных из файла, содержащего информацию о полиэдре, у каждой точки будем вызвать метод `is_good()`. Делать это требуется перед умножением точки на коэффициент гомотетии, т.к. он не влияет на её проекцию. Заметим, что при умножении точки на коэффициент гомотетии, нам возвращается новый объекта класса `R3`. По этой причине добавим в конструктор данного класса аргумент, присваивающий значение атрибуту `is_good`. Таким образом, даже после домоножения точки на коэффициент гомотетии мы будем знать, является ли она «хорошей».

Для решения задачи требуется найти суммы площадей граней, а не их проекций. Добавим дополнительный список `_vertexes`, который будет содержать вершины полиэдра без преобразований. Данный список будет заполняться вместе со списком преобразованных вершин полиэдра, поэтому порядок точек в нём будет такой же.

При обработке грани, будем записывать количество «хороших» точек в переменную `good_points_num`. Если значение данной переменной равно 1, посчитаем площадь грани построенной на заданных вершинах (используя при этом вершины без преобразований из списка `_vertexes`). Найденную площадь прибавим к новому атрибуту `good_area` класса `Polyedr`, который обозначает искомую площадь.

Модификация части кода, отвечающей за обработку граней:

```
# вспомогательный массив
buf = line.split()
# количество вершин очередной грани
size = int(buf.pop(0))
# кол-во хороших точек в грани
good_points_num = 0
# массив вершин этой грани
vertexes = []
for n in buf:
    if self.vertexes[int(n) - 1].is_good:
        good_points_num += 1
    vertexes.append(self.vertexes[int(n) - 1])
# задание рёбер грани
for n in range(size):
    self.edges.append(Edge(vertexes[n - 1], vertexes[n]))
# задание самой грани
self.facets.append(Facet(vertexes))
if good_points_num == 1:
    orig_vertexes = [_vertexes[int(n) - 1] for n in buf]
    self.good_area += Facet(orig_vertexes).area()
```

Модификация класса `TkDrawer`

Добавим в класс `TkDrawer` метод `draw_line()`, который рисует линию с учётом коэффициента гомотетии из файла полиэдра и переменной `SCALE`.

Реализация метода `draw_line()`:

```
# Рисование линии
def draw_line(self, p, q, homothety=1.0, color="black"):
    self.canvas.create_line(x(p, homothety), y(p, homothety),
                            x(q, homothety), y(q, homothety),
                            fill=color, width=1)
    self.root.update()
```

Написание тестов

Для проверки правильности решения задачи были написаны тесты с использованием простых полиэдров, имеющих несложную структуру (например, две плоскости). Также специально для тестов были созданы полиэдры: параллелепипед, тетраэдр. Тесты, проверяющие корректность вычисления площади размещены в файле `test_good_vertexes.py`.

На данных изображениях представлен результат работы программы для одной и той же проекции параллелепипеда с разными коэффициентами гомотетии:

Рис. 2: Большой коэффициент гомотетии.

Рис. 3: Маленький коэффициент гомотетии.

Можно заметить, что изменение коэффициента гомотетии меняет положение красной вертикальной прямой, разделяющей точки на «хорошие» и обычные. При этом суммарная площадь подходящих граней не меняется.

Команды, с помощью которых получены итоговые отчёты в заданных форматах

Представленные команды следует исполнять в директории, содержащей файл `report.md`.

- Для получения отчёта в формате `.pdf` используется следующая команда:

```
pandoc --template default.latex -s --toc --lua-filter
./include-code-files.lua --metadata-file=metadata.yaml
report.md -o report.pdf
```

- Для получения отчёта в формате `.html` используется следующая команда:

```
pandoc -o report.html -f markdown -t html -s --template
default.html5 -s --toc --lua-filter ./include-code-files.lua
--metadata-file=metadata.yaml --mathjax report.md
```

- Для получения отчёта в формате `.docx` используется следующая команда:

```
pandoc -o report.docx -f markdown -t docx --lua-filter
./include-code-files.lua --metadata-file=metadata.yaml report.md
```