Московский авиационный институт (национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Компьютерная графика»

Студент: И. П. Моисеенков
Преподаватель: Г. С. Филиппов
Группа: М8О-308Б-19

Дата: 18.11.2021
Оценка:
Подпись:

Москва, 2021

Лабораторная работа №3

Основы построения фотореалистичных изображений

Задача: Используя результаты Л.Р.№2, аппроксимировать заданное тело выпуклым многогранником. Точность аппроксимации задается пользователем. Обеспечить возможность вращения и масштабирования многогранника и удаление невидимых линий и поверхностей. Реализовать простую модель закраски для случая одного источника света. Параметры освещения и отражающие свойства материала задаются пользователем в диалоговом режиме.

Вариант 11: Прямой усеченный круговой конус.

1 Описание

Для выполнения данного задания я воспользовался библиотекой matplotlib для python.

Вместо построения прямого усеченного конуса будем строить прямую усеченную пирамиду. Количество граней пирамиды - параметр аппроксимации - задается пользователем. Чем больше параметр, тем более пирамида похожа на конус.

Принцип построения фигуры полностью совпадает с описанным в предыдущем отчете. Единственное отличие в том, что нам необходимо уметь строить неизвестное заранее количество граней.

В программе имеется возможность показать и скрыть невидимые линии.

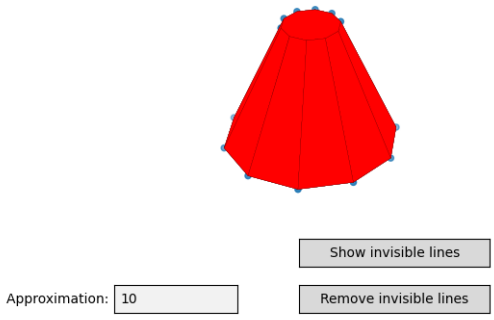
2 Исходный код

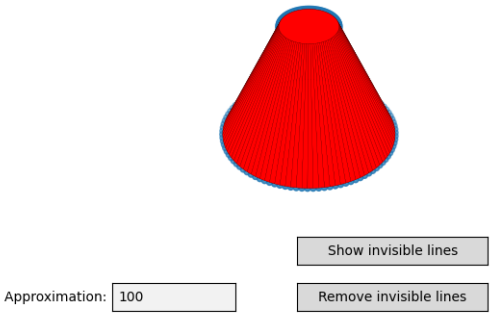
```
1  import numpy as np
2  from matplotlib import pyplot as plt
3  from mpl_toolkits.mplot3d.art3d import Poly3DCollection
4  from matplotlib.widgets import Button, TextBox
5
6
7  def draw_figure(new_approximation):
8      global ax, approximation, r_lower, r_upper, sides, alpha
9      approximation = new_approximation
10
11     ax.clear()
12     v = [] # vertices of cone
13     for i in range(approximation):
14         v.append([r_lower * np.cos(2 * np.pi * i / approximation), r_lower * np.sin(2 *
               np.pi * i / approximation), 0])
```

```
15          v.append([r_upper * np.cos(2 * np.pi * i / approximation), r_upper * np.sin(2 *
                np.pi * i / approximation), 1])
16
17      v = np.array(v)
18      ax.scatter3D(v[:, 0], v[:, 1], v[:, 2]) # adding vertices to plot
19
20      sides = [[v[i % (2 * approximation)],
21                v[(i + 1) % (2 * approximation)],
22                v[(i + 3) % (2 * approximation)],
23                v[(i + 2) % (2 * approximation)]] for i in range(0, approximation * 2 - 1,
                      2)]
24
25      sides.append([v[i] for i in range(0, approximation * 2 - 1, 2)])
26      sides.append([v[i] for i in range(1, approximation * 2, 2)])
27
28      # adding sides to plot
29      collection = Poly3DCollection(sides, alpha=alpha, edgecolors='black', linewidth
            =0.1, facecolors='red')
30      ax.add_collection3d(collection)
31
32      ax.grid(None)
33      ax.axis('off')
34      plt.draw()
35
36
37  fig = plt.figure()
38  fig.subplots_adjust(bottom=0.2)
39  fig.canvas.mpl_disconnect(fig.canvas.manager.key_press_handler_id)
40  ax = fig.add_subplot(111, projection='3d')
41
42  r_upper = 1 # radius for upper side of truncated cone
43  r_lower = 3 # radius for lower side of truncated cone
44
45  approximation = 10 # amount of sides for approximation of a truncated cone
46  alpha = 0.5 # 0.5 -> show invisible lines; 1 -> delete invisible lines
47
48  draw_figure(approximation)
49
50
51  def button_callback_remove(event):
52      global alpha
53      alpha = 1
54      ax.add_collection3d(Poly3DCollection(sides, alpha=alpha, edgecolors='black',
            linewidth=0.1, facecolors='red'))
55      plt.draw()
56
57
58  button_ax_remove = fig.add_axes([0.5, 0.05, 0.31, 0.06])
59  button_remove = Button(button_ax_remove, "Remove invisible lines")
```

```
60 button_remove.on_clicked(button_callback_remove)
61
62
63 def button_callback_show(event):
64     global alpha
65     alpha = 0.5
66     ax.add_collection3d(Poly3DCollection(sides, alpha=alpha, edgecolors='black',
           linewidth=0.1, facecolors='red'))
67     plt.draw()
68
69
70 button_ax_show = fig.add_axes([0.5, 0.15, 0.31, 0.06])
71 button_show = Button(button_ax_show, "Show invisible lines")
72 button_show.on_clicked(button_callback_show)
73
74
75 def submit_fn(value):
76     draw_figure(int(value))
77
78
79 axbox = fig.add_axes([0.2, 0.05, 0.2, 0.06])
80 text_box_B = TextBox(axbox, "Approximation: ")
81 text_box_B.on_submit(submit_fn)
82 text_box_B.set_val(approximation)
83
84
85 plt.show()
```

3 Результат работы





4 Выводы

Выполнив третью лабораторную работу по компьютерной графике, я изучил способ аппроксимации тела вращения выпуклым многогранником. В моей работе телом вращения был усеченный конус, который я аппроксимировал n-гранной усеченной пирамидой. Уже при 50 гранях я получил фигуру, очень похожую на конус.