Второй отборочный этап

Индивидуальная часть

Задача II.1.1. Equidistant shell (100 баллов)

Эквидистантной оболочкой трехмерного тела называется множество внешних по отношению к нему точек, удаленных от него не более чем на заданное расстояние δ. На рисунке можно видеть пример эквидистантной оболочки для куба, представленной в разрезе.

Даны произвольный выпуклый многогранник и значение δ. Требуется написать программу, вычисляющую объем эквидистантной оболочки.

Формат входных данных

В начале входных данных хранится исходный многогранник, записанный в следующем виде.

Вначале идет целое число V, за которым следует ровно $3 \cdot V$ вещественных чисел, задающих координаты вершин.

Далее целое число E, за которым следует ровно $2 \cdot E$ номеров вершин, попарно задающих ребра.

Далее идет целое число F, за которым следует ровно F граней, записанных в следующем виде.

Вначале целое число ребер N, за которым следует N номеров ребер этой грани в произвольном порядке.

Нумерация всех указанных элементов начинается с нуля.

В конце входных данных записано вещественное значение δ.

Формат выходных данных

Выходные данные должны содержать объем с точностью не менее 5 знаков после запятой.

Ограничения

Все грани являются невырожденными и выпуклыми.

Координаты вершин лежат в диапазоне от -10 до 10.

Число элементов каждого вида не превосходит 100.

$0 \leqslant \delta \leqslant 1$.

Примеры

Пример №1

Стандартный ввод

```
8
-1.00000 -1.00000 -1.00000
 1.00000 -1.00000 -1.00000
-1.00000  1.00000 -1.00000
 1.00000  1.00000 -1.00000
-1.00000 -1.00000  1.00000
 1.00000 -1.00000  1.00000
-1.00000  1.00000  1.00000
 1.00000  1.00000  1.00000

12
1 0
2 0
4 0
3 1
5 1
3 2
6 2
7 3
5 4
6 4
7 5
7 6

6
4 3 5 1 0
4 4 8 2 0
4 6 9 2 1
4 7 4 10 3
4 7 11 6 5
4 10 11 9 8

0.25
```

Стандартный вывод

```
7.24355
```

Решение

Для начала разложим оболочку исходного тела на составляющие. Построим призмы высоты δ, путем вытягивания многоугольных граней в направлении их внешних нормалей. Выберем нормали двух граней, стыкующихся по ребру.

Несложно убедиться, что проведенная через них плоскость будет ортогональна соответствующему ребру.

Построим цилиндры радиуса δ, оси которых совпадают с ребрами исходного многогранника, а основания представляют собой сектора, проведенные между соответствующими нормалями.

Оставшиеся части оболочки примут вид шаровых секторов с центрами в вершинах исходного многогранника, в основании которых лежат сферические многоугольники, центральные углы которых соответствуют углам между нормалями смежных граней, сходящихся в текущих вершинах:

Несложно убедиться, что объединение всех указанных частей дает нам исходную оболочку.

В свою очередь, выпуклость исходного многогранника гарантирует, что никакие две части не пересекаются между собой.

Таким образом, объем эквидистантной оболочки, может быть получен как сумма объемов указанных частей. Объем каждой призмы будет равен $V_p = S_p \cdot \delta$, где S_p — площадь плоской грани.

Объем цилиндра будет равен $V_c = S_c \cdot |E|$, где S_c — площадь сектора, $|E|$ — длина соответствующего ребра.

Можно попробовать доказать, что для выпуклого многогранника сумма объемов шаровых секторов равна объему полного шара радиуса δ, т. е. $V = 4\pi \cdot \delta^3/3$.

С другой стороны, можно найти объем каждого такого сектора, как $V_s = S_s \cdot \delta/3$, где S_s — площадь сферического многоугольника, для нахождения которой можно воспользоваться известной формулой.

Задача II.1.2. Скворечник (100 баллов)

Вам необходимо разработать упрощенную 3D модель скворечника.

Центр нижней грани основания модели скворечника должен находится в точке начала координат. Толщина досок должна быть равной 1 см. Высота скворечника — 36 см, включая толщину крыши. Лицевая сторона скворечника должна быть параллельна плоскости XZ и расположена со стороны положительной координаты оси Y. Центр отверстия летка скворечника находится на высоте 25 см, центр жердочки — на высоте 20 см. Дырки под жердочку быть не должно. В модели меш жердочки должен соприкасаться со стенкой скворечника. Другие размеры и взаимное расположение досок показаны на рис. II.1.1.

Имеются рендеры из *viewport* с нескольких ракурсов в ортогональной проекции — рис. II.1.2, рис. II.1.3. Обратите внимание, что рендеры также несут дополнительную информацию о форме, расположении и размерах модели. Модель не должна содержать текстур.

Модель проверяется на основе попиксельного сравнения рендеров с указанных во входном файле ракурсов. К модели применяется автоматически созданный материал со случайно подобранным оттенком. В качестве метрики для сравнения рендеров моделей используется величина dssim по каждому цветовому каналу. Баллы за каждый тест начисляются в зависимости от величины метрики.

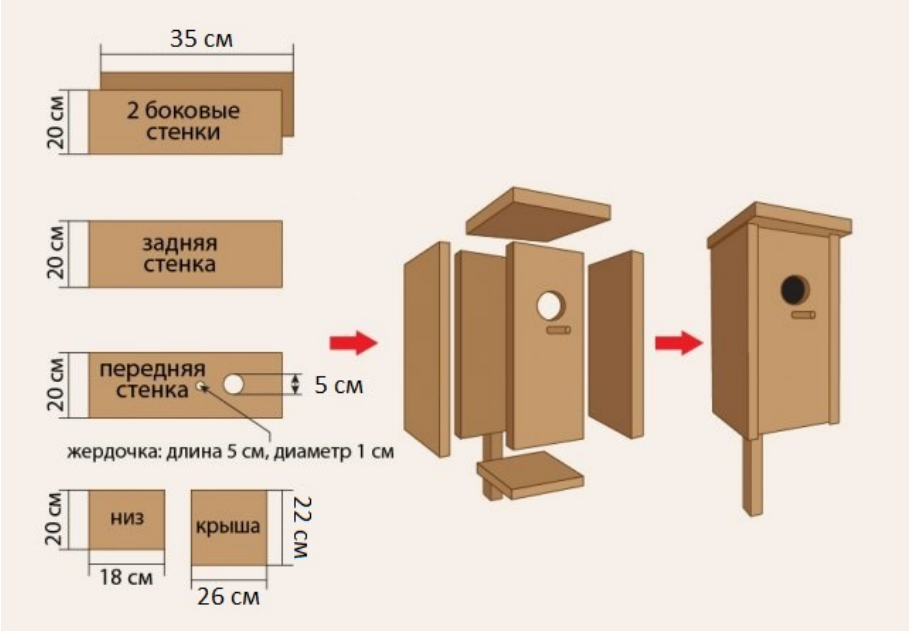

Рис. II.1.1:

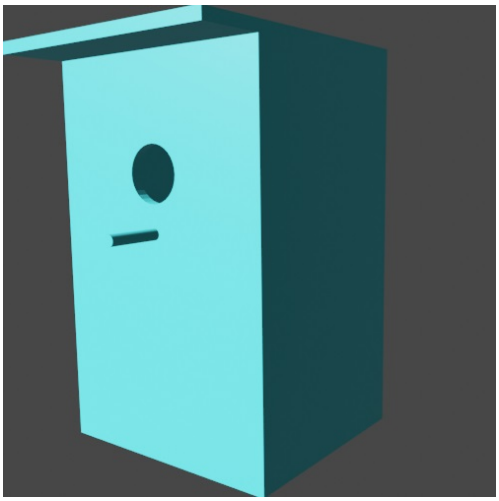

Рис. II.1.2:

Рис. II.1.3:

Источник света:
Type Sun
Color (255, 255, 255)
Specular 1.00
Strength 10.0
Location (10 m, -10 m, 20 m)
Rotation (30, 30, 0)

Камера:
Focal Length 50mm
Render Engine: Eevee

Формат входных данных

Во входном файле содержатся x, y, z — координаты камеры в метрах и rx, ry, rz — углы поворота в радианах.

Формат выходных данных

В качестве решения следует отправлять файл формата OBJ (расширение .obj). Размер файла не должен превышать 999997 Байт. Единицы измерения должны соответствовать физическим величинам. Координаты вершин модели должны быть указаны в метрах.

Примеры

Пример №1

Стандартный ввод
`-0.408464 0.499149 0.265191` `1.429877 0.014555 -2.484780`
Стандартный вывод
`Смотри рис. 3.2`

Пример №2

Стандартный ввод
`-0.056628 0.104147 -0.006480` `-0.272270 -3.141593 0.689107`
Стандартный вывод
`Смотри рис. 3.3`

Решение

Прежде чем моделировать скворечник необходимо определить из каких частей он состоит и какие примитивы можно использовать для этих целей.

Левая и правая стенки идентичны по форме и размерам. Можно трансформировать куб.

Крышу и дно скворечника можно сделать по отдельности трансформируя куб.

Задняя и лицевая стенки одинаковы по форме и размерам. Также можно трансформировать куб. На лицевой части стенки имеется круглое отверстие для птиц, которое можно вырезать, используя модификатор Boolean и цилиндр соответствующего размера.

Жёрдочку можно сделать из цилиндра.

Каждую полученную часть скворечника необходимо правильно расположить в соответствии с условием задачи. Важно не забыть про пивот. И перед экспортом

модели в OBJ формат необходимо объединить все части скворечника путем CTRL+J, если используется Blender.

Командная часть

Задача II.2.1. Four-dimensional polytope (100 баллов)

Пусть имеется некоторый выпуклый четырехмерный политоп, заданный в виде сетки, составленной из таких топологических элементов, как вершины, ребра, грани и трехмерные тела, для которых выполняются следующие условия.

Каждая вершина такого политопа задается четырьмя своими координатами (x, y, z, w). Каждое ребро представляет собой прямолинейный отрезок и задается парой соединяемых им вершин. Каждая грань представляет собой выпуклый многоугольник и задается набором своих ребер. Каждое тело представляет собой выпуклый многогранник и задается набором своих граней.

Сам политоп задается набором ограничивающих его трехмерных тел.

Требуется написать программу, вычисляющую объем сечения указанного политопа трехмерным подпространством при $w = 0$.

Формат входных данных

Во входных данных последовательно хранятся наборы сеточных элементов.

Вначале идет целое число V, за которым следует ровно $4 \cdot V$ вещественных чисел, задающих координаты вершин.

Далее целое число E, за которым следует ровно $2 \cdot E$ номеров вершин, попарно задающих ребра.

Далее идет целое число F, за которым следует ровно F граней, записанных в следующем виде.

Вначале целое число ребер N, за которым следует N номеров этих ребер.

Аналогичным образом записываются тела, заданные номерами своих граней.

Нумерация сеточных элементов каждого вида начинается с нуля.

Формат входных данных

Выходные данные должны содержать объем с точностью не менее 5 знаков после запятой.

Ограничения

Гарантируется, что все сеточные элементы являются невырожденными.

Координаты вершин лежат в диапазоне от -10 до 10.

Число элементов каждого вида не превосходит 1000.

Примеры

Пример №1

Стандартный ввод

```
5
-1.00000 -1.00000 -1.00000 -0.50000
 0.00000  1.00000 -1.00000 -0.50000
 1.00000  0.00000 -1.00000 -0.50000
 0.00000  0.00000  1.00000 -0.50000
 0.00000  0.00000  0.00000  0.50000

10
0 1
0 2
0 3
1 2
1 3
2 3
0 4
1 4
2 4
3 4

10
3 0 1 3
3 1 2 5
3 0 2 4
3 3 4 5
3 0 6 7
3 1 6 8
3 2 6 9
3 3 7 8
3 4 7 9
3 5 8 9

5
4 0 1 2 3
4 0 4 5 7
4 1 5 6 9
4 2 4 6 8
4 3 7 8 9
```

Стандартный вывод

```
0.12500
```

Решение

Для начала разделим вершины исходного политопа относительно секущего подпространства:

1. верхние — лежащие в верхней половине подпространства ($z > 0$);
2. нижние — лежащие в нижней половине подпространства ($z < 0$);

3. внутренние — лежащие в подпространстве (z = 0).

Добавляем все внутренние вершины в результирующий список вершин V. Далее из ребер исходного политопа выделим следующие разновидности:

1. секущие — пересекающие подпространство (обе вершины разного вида);
2. верхние — обе вершины верхние;
3. нижние — обе вершины нижние;
4. внутренние — лежащие в подпространстве (обе вершины внутренние).

На каждом секущем ребре определим точку, в которой оно пересекает подпространство.

Сформируем на ее основе новую вершину и привяжем ее к содержащему ее ребру.

Добавим новую вершину в результирующий список вершин V.

Добавим внутренние ребра в результирующий список ребер E.

Далее рассмотрим двумерные грани и разделим их на:

1. секущие — если они содержат хотя бы одно секущее ребро;
2. верхние — если все их ребра верхние;
3. нижние — если все их ребра нижние;
4. внутренние — если все их ребра внутренние.

На каждой секущей грани определим отрезок прямой, проходящий через секущие ребра.

В силу выпуклости, таких ребер всегда должно быть либо два, либо ни одного.

Сформируем на его основе новое ребро и привяжем его к текущей грани.

Добавим новое ребро в результирующий список ребер E.

Добавим внутренние грани в результирующий список граней F.

Действуя аналогичным образом, рассмотрим все трехмерные тела.

Добавим недостающие грани, привязанные к секущим телам.

Добавим внутренние тела в результирующий список тел B.

Дополним их телами, сформированными из новых граней (возникших в результате сечения исходных тел).

В результате получим набор элементов, формирующих выпуклый многогранник либо набор многогранников.

Все, что остается, это найти их объемы. по расширению файла Выберите файл Файл не выбран отправить последняя попытка

Задача II.2.2. Звезды под водой (100 баллов)

При разработке систем распознавания изображений часто большую проблему представляет получение размеченного набора данных (датасета). Сбор реальных изображений может быть затратным по времени и ресурсам. Технологии конструирования реалистичных виртуальных сцен могут помочь в такой ситуации.

В этой задаче требуется разработать программу, которая находит объекты на

изображениях, сделанных под водой. Для тестирования мы подготовили несколько тысяч рендеров подводных сцен при помощи проекта на Unity, который можно скачать по ссылке https://github.com/Supermagzzz/NTI2020_DATASET. Требуется определить, есть ли на этих картинках морские звезды и если есть, то находить их количество и положение.

Пример одной из таких картинок

На этой картинке находится две морских звезды, положения которых отмечено красными прямоугольниками на картинке ниже.

Формат входных данных

Датасет доступен по этой ссылке https://drive.google.com/drive/u/0/folders/16gvLNcL1IzoJN0gm6kzIF2zz2H-Qa3Cr. Он представляет из себя архив с картинками в формате png. Название картинки соответствует номеру теста. На каждой картинке может быть от нуля до трех морских звезд.

Формат выходных данных

В первой строке выходного файла выведите количество тестов t, на которые вы собираетесь дать ответ. После этого выведите ответы на t тестов в следующем формате:

В первой строке каждого ответа выведите два одно целое число – номер теста на который вы собираетесь дать ответ. Во второй строке каждого ответа выведите количество звезд в этом тесте (если звезд на картинке не было, выведите ноль). **Запрещается выводить больше чем три звезды для каждого ответа, иначе тест будет проигнорирован**.

Для каждой звезды в отдельной строке выведите координаты (в пикселях) ограничивающего прямоугольника (https://ru.wikipedia.org/wiki/AABB) для этой звезды в формате $x1$, $y1$, $x2$, $y2$, где $x1$, $y1$ – координаты левого верхнего угла этого прямоугольника, а $x2$, $y2$ – координаты нижнего правого угла, разделенные одним пробелом. Выводить координаты ограничивающих прямоугольников звезд можно в любом порядке (соблюдая формат). Стоит считать, что начало координат находится в левом верхнем углу картинки. В файле не должно быть лишних пустых строк.

Будем считать, что звезда найдена верно, если:

- $X1$, $Y1$, $X2$, $Y2$ – координаты реального ограничивающего прямоугольника;
- $x1$, $y1$, $x2$, $y2$ – координаты найденного вами ограничивающего прямоугольника;
- Величина $|x1 - X1| + |x2 - X2|$ составляет менее чем 10% от общей ширины картинки;
- Величина $|y1 - Y1| + |y2 - Y2|$ составляет менее чем 10% от общей высоты картинки.

Для оценки будет использоваться величина, равная количеству правильно найденных звезд, поделенному на количество реально существующих звезд. Эта величина будет умножена на 100 и округлена вверх до целого числа.

Для проверки вашего решений отошлите файл с ответами в указанном выше формате.

В качестве среды разработки следует выбирать Answer text (расширение .txt).

Для генерации обучающей выборки рекомендуется воспользоваться нашим генератором (https://github.com/Supermagzzz/NTI2020_DATASET).

Решение

Для определения границ морской звезды с высокой точностью необходимо обучить Object Detector на основе датасета, который возможно сгенерировать, используя проект, ссылка на который указана в задаче. Один из способов решения заключается в обучении нейросетей. В качестве модели можно выбрать R-CNN, Fast R-CNN, YOLO и другие. Рассмотрим подробнее обучение модели YOLOv4.

1. Подготовка данных
   1.1. Генерация обучающей выборки
   Используя проект, расположенный по ссылке, необходимо сгенерировать датасет. Количество сгенерированных данных должно быть не менее 2000 на категорию. Лучше 4-8 тыс. размеченных изображений.

1.2. Разметка
В первую очередь, необходимо разметить данные. Несмотря на то, что в генераторе помимо изображений генерируется сама разметка, необходимо привести разметку к нужному формату, в зависимости от используемой модели. Для обучения модели YOLOv4 необходимо чтобы возле каждого изображения image_0001.jpg находились размеченные данные в текстовом файле, название которого должно совпадать с названием изображения image_00001.txt.
image_0001.jpg image_0001.txt
image_0002.jpg image_0002.txt
...
image_0003.jpg image_0003.txt
Формат разметки для YOLOv4 должен выглядеть следующим образом:
id xc yc w h, где
id – номер категории,
xc, yc – координаты центра прямоугольной области,
w, h – высота и ширина прямоугольной области.
В рамках текущей задачи категория всего одна, поэтому здесь будет всегда 0, так как нумерация категорий – от 0 до n-1.

1.3. Аугментация
Для повышения точности распознавания датасет рекомендуется расширить путём аугментации обучающей выборки. Для аугментации данных можно использовать библиотеку Imgaug на питоне, которая позволяет не просто разнообразить изображения, но и в соответствии с изменениями генерирует соответствующую разметку.

1.4. Смешивание
После аугментации рекомендуется перемешать сгенерированные и аугментированные данные.

2. Обучение модели
Для обучения модели можно использовать фреймворк, например, Tensorflow, Keras, Darknet.

Задача II.2.3. Позиционирование VR-шлема (100 баллов)

Начинающий программист Вася участвует в проекте по разработке собственного шлема виртуальной реальности.

Прототип шлема оснащен датчиками, измеряющими время отклика сигнала от n базовых станций, расположенных в комнате. По задержке откликов можно узнать расстояние от шлема до каждой базовой станции.

Помогите Васе по известным координатам базовых станций и расстояниям до них однозначно определить текущие координаты шлема, либо выяснить, что это невозможно (поскольку данных недостаточно).

Решение считается верным, если оно отличается от истинного положения шлема не более чем в 5-м знаке после запятой.

Формат входных данных

Входные данные содержат натуральное n, за которым следует набор из $4 \times n$ вещественных чисел, определяющих координаты базовых станций (xi, yi, zi) и расстояния до них di.

Формат выходных данных

Если задача имеет решение, выходные данные должны содержать целое число 1, за которым следуют координаты шлема, указанные с точностью не менее 5 знаков после запятой.

В противном случае выходные данные должны содержать единственное число 0.

Ограничения

Все расстояния корректны и указаны с точностью до 10-го знака после запятой.

$-10 \leqslant (xi, yi, zi) \leqslant 10, 0 \leqslant di \leqslant 50, 1 \leqslant n \leqslant 105$.

Примеры

Пример №1

Стандартный ввод
4 -5.30710 7.24030 -1.23090 5.8376287360 -1.01140 2.40600 0.50000 1.6557115600 0.35000 1.40000 3.56100 3.6476015421 1.98000 1.75000 -2.05000 5.5483426363

Стандартный вывод
1 -2.00000 3.28010 1.50000

Пример №2

Стандартный ввод
4 -1.09670 2.46050 -3.25040 4.3513370980 0.53010 -1.37090 -3.25040 3.9144142014 2.75010 -1.37090 -3.25040 4.3513370980 -2.44060 0.76030 -3.25040 4.7208526804

Стандартный вывод
0

Задача II.2.4. Пушка и рождественское дерево (100 баллов)

Вася решил позаниматься НЕРЕАЛЬНЫМ занятием. Он решил пострелять из пушки по рождественскому дереву в новом году. Пушка уже направлена точно на цель по оси X, но рассчитать точную траекторию полета ядра Вася сам не может.

Вася просит помощи в этом сложном занятии, потому что уже поздняя ночь и он забыл все алгоритмы.

Пушка находится в статическом состоянии и управлять можно только начальным углом траектории полета снаряда к плоскости оси X. Проект (https://github.com/dvfu/NTI-UE4-CannonAndChristmasTree), который написал Вася, уже содержит игру и умеет считывать и выводить файлы нужных форматов. Вам необходимо реализовать класс UCannonSceneComponent только для управления начальным углом полета снаряда. Ваша задача найти такой угол траектории полета снаряда, чтобы попасть в как можно более высокую точку дерева.

Формат входных данных

Файл с решением должен содержать реализацию класса UCannonSceneComponent.