

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Математический факультет

Кафедра теории функций и геометрии

Программная реализация (на языке JavaScript) алгоритмов
генерации ФОС по геометрии в 2024 году

Курсовая работа

Направление 010501 Фундаментальные математика и механика

Зав.кафедрой _____ д.физ.-мат.н., проф. Е.М. Семёнов

Обучающийся _____ А.С. Суматохина

Руководитель _____ д.физ.-мат.н., проф. Е.М. Семёнов

Воронеж 2024

Содержание

Введение	3
1 Планиметрия	4
1.1 Вспомогательные функции	4
1.1.1 Функции для работы с массивами	4
1.1.2 Функции для работы с числами	5
1.1.3 Функции для работы с canvas	5
1.1.4 Функции вызова ошибок	8
1.2 Этапы разработки шаблоны с вспомогательным чертежом по теме «Планиметрия»	9
2 Стереометрия	13
2.1 Разработка библиотек с помощью Gpt-Chat	13
2.2 Применение ООП для разработки шаблонов	16
2.3 Вспомогательные функции	17
2.3.1 Функции для работы с координатами	17
2.3.2 Функции для работы с canvas	17
2.4 Этапы разработки шаблоны с вспомогательным чертежом по теме «Стереометрия»	18
Заключение	24
Приложение	26

Введение

Единый государственный экзамен (ЕГЭ) — централизованно проводимый в Российской Федерации экзамен в средних учебных заведениях — школах, лицеях и гимназиях, форма проведения ГИА (Государственный Итоговая Аттестация) по образовательным программам среднего общего образования. Служит одновременно выпускным экзаменом из школы и вступительным экзаменом в вузы.

Но за время обучения в 10 и 11 классе при подготовке к ЕГЭ школьники сталкиваются с дефицитом заданий по определённым категориям. Так в конце 2021 года в список заданий ЕГЭ были добавлены новые задания под номером 11 по теме «Графики функции», а в конце 2023 - задание №2 по теме «вектора», количество которых, для прорешивания было очень мало. А по теме «Производная и первообразная» банк заданий расходуется при подготовке с невероятной скоростью: Так как это преимущественно графические задания, решение их занимает менее минуты, а их составление вручную занимает несоразмерно много времени.

ЕГЭ является относительно неизменяемым экзаменом, поэтому все материалы, которые уже были выложены в открытый доступ, имеют полные решения, что приводит к списыванию учениками.

При этом существуют задания с вспомогательным чертежом. Чаще всего для целого ряда заданий используется одна и та же иллюстрация, которая не всегда соответствует условиям задачи, а иногда отвлекает от решения. Проект «Час ЕГЭ» позволяет решить все эти проблемы.

«Час ЕГЭ» — компьютерный образовательный проект, разрабатываемый при математическом факультете ВГУ в рамках «OpenSource кластера» и предназначенный для помощи учащимся старших классов при подготовке к тестовой части единого государственного экзамена. Задания в «Час ЕГЭ» генерируются случайным образом по специализированным алгоритмам, называемых шаблонами, каждый из которых охватывает множество вариантов соответствующей ему задачи. Для пользователей предназначены четыре оболочки (режима работы): «Случайное задание», «Тесты на печать», «Полный тест» и «Мини-интеграция». «Час ЕГЭ» является полностью открытым (код находится под лицензией GNU GPL 3.0) и бесплатным. В настоящее время в проекте полностью реализованы тесты по математике с кратким ответом (бывшая «часть В»). [3] Планируется с течением времени включить в проект тесты по другим предметам школьной программы.

Первая глава этой работы посвящена обзору вспомогательных функций, которые ускоряют написание шаблонов по теме «Планиметрия». А также приведён алгоритм написания шаблона с чертежом.

Вторая глава представляет решение проблемы отрисовки фигур в трёхмерном пространстве на языке программирования JavaScript; рассказывает о применении объектно-ориентированного программирования для упрощения написания шаблонов с чертежом; приводит обзор вспомогательных функций и алгоритм написания шаблона по теме «Стереометрия».

1. Планиметрия

В этой главе мы приводим вспомогательные функции и алгоритм написания шаблона по планиметрии

1.1. Вспомогательные функции

1.1.1. Функции для работы с массивами

`Array.prototype.permuteCyclic = function(repeat)`

Возвращает массив после циклической перестановки. В листинге 3 в строке 10 функция используется для перестановки букв в названии угла.

```
1   let array = [1, 2, 3, 4, 5];
2
3   array.permuteCyclic(1);
4   // [5, 1, 2, 3, 4]
5
6   array.permuteCyclic(-2);
7   // [3, 4, 5, 1, 2]
8
9   array.permuteCyclic(0);
10  // [1, 2, 3, 4, 5]
11
```

`Array.prototype.mt_coordinatesOfIntersectionOfTwoSegments = function()`

Возвращает координаты пересечения двух отрезков, задаваемых первыми парами точек из массива. В листинге 4 в строке 85 функция используется для нахождения точки пересечения рёбер составного многогранника.

```
1   let array = [{x:0,y:5},{x:-4,y:4},{x:1,y:10},{x:-3,y:6}];
2
3   array.mt_coordinatesOfIntersectionOfTwoSegments()
4   //{ x: -5.333333333333333, y: 3.6666666666666667, status: false }
5   //Отрезки не пересекаются, но прямые проходящие через них пересекаются в точке {x,y}
6
7   array = [{x:0,y:5},{x:-4,y:4},{x:1,y:1},{x:-3,y:6}];
8   array.mt_coordinatesOfIntersectionOfTwoSegments()
9   //{ x: -1.8333333333333333, y: 4.541666666666667, status: true }
10  //Отрезки пересекаются в точке {x,y}
```

`Array.prototype.shuffleJoin = function(separator)`

Перемешивает и соединяет массив с разделителем `separator`. `separator` по умолчанию пустая строка. Функция используется в 5 в строке 74 для отображения условий задачи в случайном порядке.

```
1   let array = ['A', 'B', 'C', 'D',];
2   array.shuffleJoin();
```

```

3 //ADBC
4
5 array.shuffleJoin('; ');
6 //C; D; B; A

```

`Array.prototype.joinWithConjunction = function(separator)`

Соединяет массив запятыми и соединяет два последних элемента союзом «и».

```

1 let array = ['A', 'B', 'C', 'D',];
2
3 array.joinWithConjunction();
4 //A, B, C и D

```

1.1.2. Функции для работы с числами

`Number.prototype.perfectCubicMultiplier = function()`

Возвращает максимальный делитель данного числа, куб которого также является делителем данного числа.

```

1 let number = 81;
2
3 number.perfectCubicMultiplier()
4 //3
5
6 number = 36;
7 number.perfectCubicMultiplier()
8 //1
9
10 number = -27;
11 number.perfectCubicMultiplier()
12 //3

```

`Number.prototype.texcbirt = function(p1, p2)`

TeX-представление кубического корня из данного числа.

Если данное число - полный куб, то корень из числа.

Если p1, то из-под корня будут вынесены возможные множители.

Если p1, p2 и из-под корня выносятся единица, то она будет опущена

1.1.3. Функции для работы с canvas

`CanvasRenderingContext2D.prototype.drawSection = function(vertex, fillStyle)`

Заполняет область цветом fillStyle по вершинам из массива vertex.

```

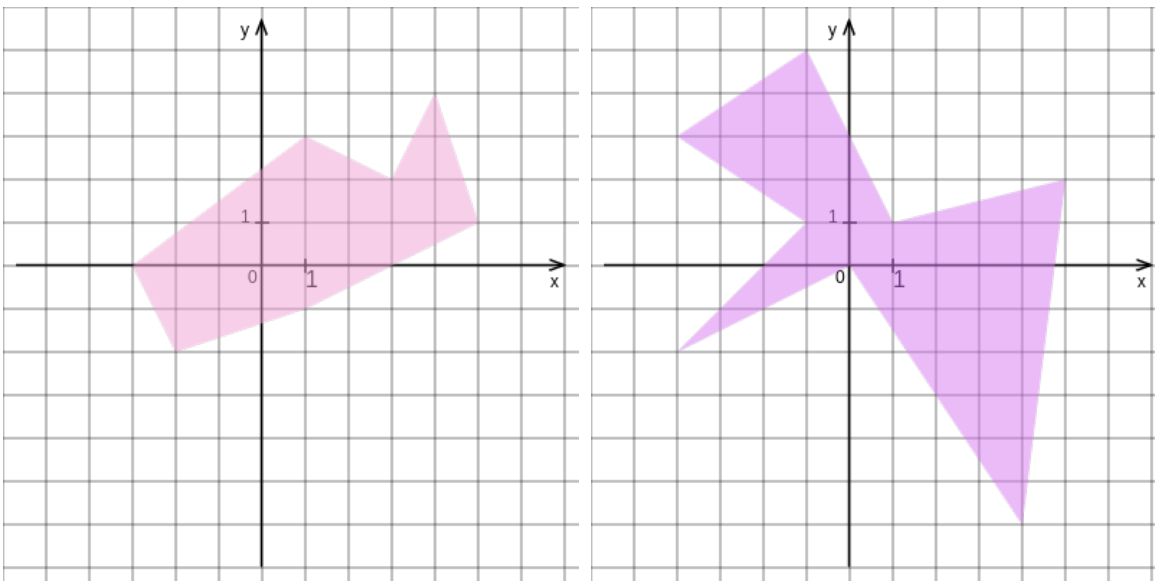
1 let paint1 = function(ctx) {
2   let h = 400;
3   let w = 400;
4   ctx.drawCoordinatePlane(w, h, {
5     hor: 1,

```

```

6         ver: 1
7     }, {
8         x1: '1',
9         y1: '1',
10        sh1: 16,
11    }, 30);
12    ctx.scale(30, -30);
13    ctx.drawSection([[1, 3], [-3, 0], [-2, -2], [1, -1], [5, 1], [4, 4], [3,
14    2]]);
15
16    ctx.drawSection([[-2, 0], [-1, 1], [-4, 3], [-1, 5], [1, 1], [5, 2], [4,
17    -6], [0, 0], [-4, -2],]);
18    };

```



`CanvasRenderingContext2D.prototype.drawLineAtAngle = function(x, y, angle, length)`

Рисует отрезок длины `length` под углом `angle` (в радианах). Пример использования в листинге 6 в строках 19 и 25 (используется для отрисовки биссектрисы).

`CanvasRenderingContext2D.prototype.strokeInMiddleOfSegment = function(x1, y1, x2, y2, length, quantity)`

Ставит штрихи длины `length` на середине отрезка перпендикулярно ему. Функция используется в листинге 6 в строках 21-22 для обозначения равных по длине сторон треугольника.

`CanvasRenderingContext2D.prototype.markSegmentWithLetter = function(x, y, angle, letter, length, maxLength)`

Вспомогательная функция для отрисовки текста около некоторого отрезка.

`CanvasRenderingContext2D.prototype.signSegmentInMiddle = function(x1, y1, x2, y2, letter, length, maxLength)`

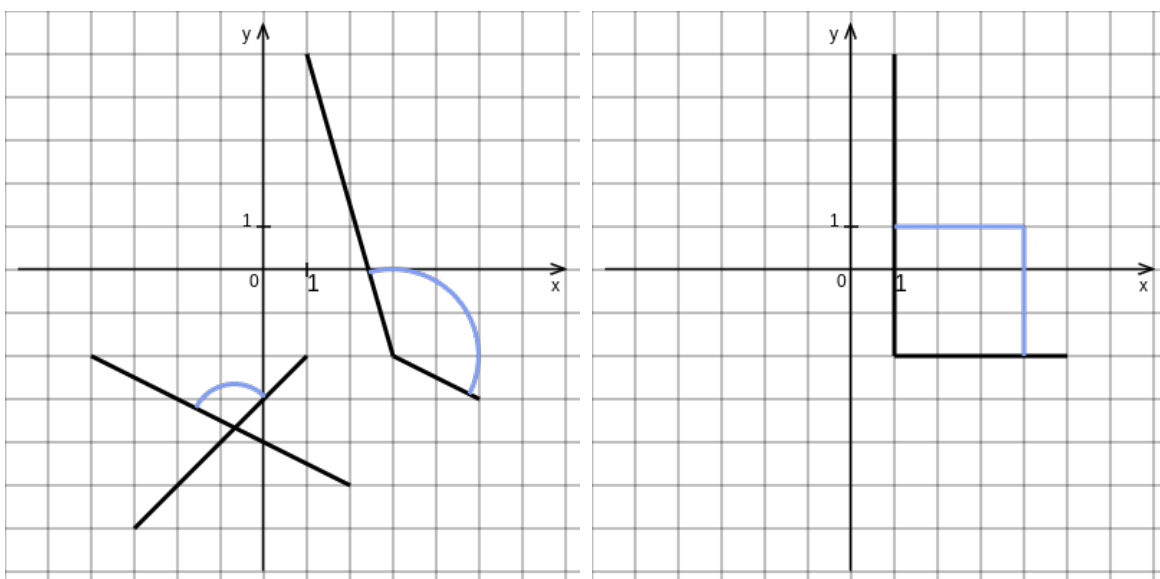
Рисует строку `letter` на середине отрезка. В листинге 4 в строках 94 - 98 функция используется для отображения длин рёбер многогранника.

`CanvasRenderingContext2D.prototype.arcBetweenSegments`

```
= function(coordinates, radius)
```

Рисует знак угла между двумя отрезками в месте их пересечения. `coordinates` - массив вида `[x1, y1, x2, y2]`.

```
1  let paint1 = function(ctx) {
2    let h = 400;
3    let w = 400;
4    ctx.drawCoordinatePlane(w, h, {
5      hor: 1,
6      ver: 1
7    }, {
8      x1: '1',
9      y1: '1',
10     sh1: 16,
11   }, 30);
12   ctx.scale(30, -30);
13
14   ctx.lineWidth = 2 / 30;
15   ctx.drawLine(1, 5, 3, -2);
16   ctx.drawLine(3, -2, 5, -3);
17   ctx.arcBetweenSegments([1, 5, 3, -2, 5, -3], 2);
18
19   ctx.drawLine(2, -5, -4, -2);
20   ctx.drawLine(1, -2, -3, -6);
21   ctx.arcBetweenSegments([2, -5, -4, -2, -3, -6, 1, -2,], 1);
22
23   ctx.drawLine(1, 5, 1, -2);
24   ctx.drawLine(1, -2, 5, -2);
25   ctx.strokeStyle = om.secondaryBrandColors.iz();
26   ctx.arcBetweenSegments([1, 5, 1, -2, 5, -2], 3);
27
28   };
```



```
CanvasRenderingContext2D.prototype.arcBetweenSegmentsCount
= function(coordinates, radius, number, step)
```

Рисует знак угла между двумя отрезками в месте их пересечения `number` раз с

отступом `step`. В листинге 7 в строках 27 - 28 используется для обозначения двух равных углов.

```
CanvasRenderingContext2D.prototype.drawEllipse
= function(x, y, radiusX, radiusY, rotation, startAngle, endAngle,
anticlockwise)
```

Рисует эллипс.

```
CanvasRenderingContext2D.prototype.drawArc
= function(x, y, radius, startAngle, endAngle, anticlockwise)
```

Рисует дугу.

1.1.4. Функции вызова ошибок

Во время разработки шаблонов по теме «Графики функции» требовалось много раз переопределять коэффициенты функций через циклы `while` или `do...while`, пока они не начнут соответствовать заданным условиям (видимость графика, сливание его с осями, видимость целых точек). Это часто приводило к бесконечной работе шаблона, при этом сложно было определить, какое условие не выполняется.

Для этого было разработано окружение `retryWhileUndefined` для шаблонов, которое бы перезапускало их не более `maxIterations` раз, если одно из условий не удалетворено.

```
function retryWhileUndefined(theFunction, maxIterations)
```

Но всё равно было тяжело определить, почему шаблон перезапускается. Для этого было разработано более совершенное окружение `retryWhileError`, которое не только могло бы ограничивать количество перезапусков, но и фиксировать, какие проверки не были пройдены и выводить их на экран.

```
function retryWhileError(theFunction, maxIterations, maxCollectedErrors)
```

Для окружения были написаны функции-утверждения, которые имеют структуру : условие не выполнено - записать ошибку - перезапустить шаблон. Если максимальное количество повторений достигнуто, то вывести накопившиеся ошибки и количество их появлений.

```
function genAssert(condition, message)
```

Если условие `condition` - `false`, то шаблон перезапускается.

```
function genAssertNonempty(array, message)
```

Если массив `array` пуст, то шаблон перезапускается.

```
function genAssertZ1000(number, message)
```

Если число `number` имеет более 3 знаков после запятой, то шаблон перезапускается.

```
function genAssertIrreducible(numerator, denominator, message)
```

Если дробь `numerator/denominator` сократима, то шаблон перезапускается.

```
function genAssertSaneDecomposition(number, maxFactor, message)
```

Если `number` число не раскладывается на простые множители, не более одного

из которых превосходит `maxFactor`.

1.2. Этапы разработки шаблоны с вспомогательным чертежом по теме «Планиметрия»

Для примера возьмём задание №19416 [2].

Заготовка шаблона имеет вид.

```
1 (function () {  
2   retryWhileError(function () {  
3     NAinfo.requireApiVersion(0, 2);  
4  
5     let paint1 = function (ctx) {  
6       };  
7  
8     NATask.setTask({  
9       text: 'В треугольнике ABC AC=BC, AB=15, AHвысота-, BH=5. Найдите косинус  
10      ABC',  
11      answers: 0,  
12      author: ['Суматохина Александра']  
13    });  
14    NATask.modifiers.addCanvasIllustration({  
15      width: 400,  
16      height: 400,  
17      paint: paint1,  
18    });  
19  }, 1000);  
20 })();
```

1. Начнём с отрисовки чертежа для задания. Отметим стороны треугольника так, чтобы он лежал в центре холста, а до краёв оставалось 10-20px. При отрисовке используем функцию `drawLine`. Добавим высоту

```
1 (function () {  
2   retryWhileError(function () {  
3     NAinfo.requireApiVersion(0, 2);  
4  
5     let paint1 = function (ctx) {  
6  
7       ctx.lineWidth = 2;  
8       ctx.strokeStyle = om.secondaryBrandColors;  
9  
10      ctx.drawLine(10, 370, 390, 370);  
11      ctx.drawLine(10, 370, 180, 50);  
12      ctx.drawLine(180, 50, 390, 370);  
13  
14      ctx.drawLine(280, 200, 10, 370);  
15    };  
16  
17    NATask.setTask({
```

```

18     text: 'В треугольнике ABC AC=BC, AB=15, ANвысота-, BNНайдите=5.
косинус ABC.',
19     answers: 0,
20     author: ['Суматохина Александра']
21   });
22   NATask.modifiers.addCanvasIllustration({
23     width: 400,
24     height: 400,
25     paint: paint1,
26   });
27 }, 1000);
28 })();

```

2. Добавим на рисунок штрихи, указывающие на равенство сторон и обозначение прямого угла при помощи функций `strokeInMiddleOfSegment` и `arcBetweenSegments` соответственно. И подпишем вершины и точку перпендикуляра. Добавим модификатор `NATask.modifiers.variativeABC(vertices)`, который заменяет все буквы в задании на случайные.

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let vertices = ['A', 'B', 'C', 'H'];
6
7     let paint1 = function (ctx) {
8
9       ctx.lineWidth = 2;
10      ctx.strokeStyle = om.secondaryBrandColors;
11
12      ctx.drawLine(10, 370, 390, 370);
13      ctx.drawLine(10, 370, 180, 50);
14      ctx.drawLine(180, 50, 390, 370);
15
16      ctx.drawLine(280, 200, 10, 370);
17
18      ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
19      ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);
20
21      ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
22      25);
23
24      ctx.font = "23px liberation_sans";
25      ctx.fillText(vertices[0], 10 - 5, 370 + 25);
26      ctx.fillText(vertices[1], 180, 50 - 10);
27      ctx.fillText(vertices[2], 390 - 10, 370 + 25);
28      ctx.fillText(vertices[3], ver2.x + 10, ver2.y);
29    };
30
31    NATask.setTask({
32      text: 'В треугольнике ABC AC=BC, AB=15, ANвысота-, BNНайдите=5.

```

```

    косинус ABC.',
32     answers: 0,
33     author: ['Суматохина Александра']
34   });
35   NATask.modifiers.variativeABC(vertices);
36   NATask.modifiers.addCanvasIllustration({
37     width: 400,
38     height: 400,
39     paint: paint1,
40   });
41 }, 1000);
42 })();

```

3. Теперь добавим ответ в задание. Проверим при помощи `genAssertZ1000`, что ответ целый (если иначе шаблон запускается заново). Поместим все буквы и числа в `$...$`. Все условия из задачи преобразуем в массив и соединим случайным образом с помощью функции `shuffleJoin`.

```

1  (function() {
2    retryWhileError(function() {
3      NAinfo.requireApiVersion(0, 2);
4
5      let a = sl(2, 89);
6      let b = slKrome(a, 1, a - 1);
7      genAssertZ1000(b / a, 'Нецелый ответ');
8
9      let vertices = ['A', 'B', 'C', 'H'];
10
11     let paint1 = function(ctx) {
12
13       ctx.lineWidth = 2;
14       ctx.strokeStyle = om.secondaryBrandColors;
15
16       ctx.drawLine(10, 370, 390, 370);
17       ctx.drawLine(10, 370, 180, 50);
18       ctx.drawLine(180, 50, 390, 370);
19
20       ctx.drawLine(280, 200, 10, 370);
21
22       ctx.strokeInMiddleOfSegment(180, 50, 10, 370, 10);
23       ctx.strokeInMiddleOfSegment(180, 50, 390, 370, 10);
24
25       ctx.arcBetweenSegments([180, 50, 280, 200, 280, 200, 10, 370],
26       25);
27
28       ctx.font = "23px liberation_sans";
29       ctx.fillText(vertices[0], 10 - 5, 370 + 25);
30       ctx.fillText(vertices[1], 180, 50 - 10);
31       ctx.fillText(vertices[2], 390 - 10, 370 + 25);
32       ctx.fillText(vertices[3], 280+10, 200);
33     };

```

```

33
34   NATask.setTask({
35     text: 'В треугольнике $ABC$ '+'[$AC=BC$', '$AB='+a+'$',
'$АНвысота-$', '$BH='+b+'$'].shuffleJoin(' ', '+'). Найдите косинус
$BAC$.',
36     answers: b / a,
37     author: ['Суматохина Александра']
38   });
39   NATask.modifiers.variativeABC(vertices);
40   NATask.modifiers.addCanvasIllustration({
41     width: 400,
42     height: 400,
43     paint: paint1,
44   });
45   }, 1000);
46 })();

```

2. Стереометрия

2.1. Разработка библиотек с помощью Gpt-Chat

На данный момент в языке JavaScript отсутствуют встроенные средства для изображения трёхмерных фигур. И существует только одна подходящая библиотека `Three.js`, которая могла бы выполнить проецирование координат фигуры на плоскость с учётом положения наблюдателя. Но подключение библиотеки значительно увеличивает вес проекта (на 2 GB). При этом для создания любого объекта необходима не только камера, но и сцена с рендерингом, что значительно замедляет работу проекта.

```
1 import * as THREE from 'three';
2
3 // Создать сцену, камеру и рендер
4 const scene = new THREE.Scene();
5 const camera = new THREE.PerspectiveCamera(75, window.innerWidth /
   window.innerHeight, 0.1, 1000);
6 camera.position.z = 2;
7 const renderer = new THREE.WebGLRenderer();
8 renderer.setSize(window.innerWidth, window.innerHeight);
9
10 // Создать геометрию куба
11 const geometry = new THREE.BoxGeometry(1, 1, 1);
12
13 // Создать материал куба
14 const material = new THREE.MeshBasicMaterial({ color: 0x00ff00 });
15
16 // Создать куб
17 const cube = new THREE.Mesh(geometry, material);
18
19 // Добавить куб в сцену
20 scene.add(cube);
21
22 // Отрендерить сцену
23 renderer.render(scene, camera);
```

Листинг 1: Код необходимый для отрисовки куба

Подобные ей библиотеки проводят проецирование на плоскость с поворотом только вокруг осей Ox и Oz , что при отрисовке более сложных чертежей является недостаточным.

С учётом самых современных технологий, встал закономерный вопрос, сможет ли нейросеть ChatGPT 3.5 сгенерировать код проектора. Ниже представлен диалог с нейросетью:

```
write a 2D to 3D projector in JS without using libraries. input: point = {x,y,z},
camera= {x,y,z, angleX, angleY, angleZ, scale}
```

```

// Define the point in 2D
const point = { x: 100, y: 50, z: 0 };

// Define the camera parameters
const camera = { x: 0, y: 0, z: 0, angleX: 0, angleY: 0, angleZ: 0,
  scale: 1 };

// Function to project 2D point to 3D
function project2Dto3D(point, camera) {
  // Apply rotation around X axis
  const rotatedY = point.y * Math.cos(camera.angleX) - point.z *
    Math.sin(camera.angleX);
  const rotatedZ = point.y * Math.sin(camera.angleX) + point.z *
    Math.cos(camera.angleX);

  // Apply rotation around Y axis
  const rotatedX = point.x * Math.cos(camera.angleY) + rotatedZ *
    Math.sin(camera.angleY);
  const rotatedZ2 = -point.x * Math.sin(camera.angleY) + rotatedZ *
    Math.cos(camera.angleY);

  // Apply rotation around Z axis
  const rotatedX2 = rotatedX * Math.cos(camera.angleZ) - rotatedY *
    Math.sin(camera.angleZ);
  const rotatedY2 = rotatedX * Math.sin(camera.angleZ) + rotatedY *
    Math.cos(camera.angleZ);

  // Apply scaling
  const scaledX = rotatedX2 * camera.scale;
  const scaledY = rotatedY2 * camera.scale;

  // Translate to camera position
  const translatedX = scaledX + camera.x;
  const translatedY = scaledY + camera.y;
  const translatedZ = rotatedZ2 * camera.scale + camera.z;

  return { x: translatedX, y: translatedY, z: translatedZ };
}

// Project the point from 2D to 3D
const projectedPoint = project2Dto3D(point, camera);

// Output the result
console.log(»Projected 3D point:», projectedPoint);

```

shorten function project2Dto3D

```
function project2Dto3D(point, camera) {
  const { x, y, z } = point;
  const { angleX, angleY, angleZ, scale, x: cx, y: cy, z: cz } =
camera;

  const cosX = Math.cos(angleX);
  const sinX = Math.sin(angleX);
  const cosY = Math.cos(angleY);
  const sinY = Math.sin(angleY);
  const cosZ = Math.cos(angleZ);
  const sinZ = Math.sin(angleZ);

  const rx = x * cosY * cosZ - y * cosY * sinZ + z * sinY;
  const ry = (x * (sinX * sinY * cosZ + cosX * sinZ) + y * (sinX *
sinY * sinZ - cosX * cosZ) + z * sinX * cosY) * scale;
  const rz = (-x * (cosX * sinY * cosZ - sinX * sinZ) - y * (cosX *
sinY * sinZ + sinX * cosZ) + z * cosX * cosY) * scale;

  return { x: rx + cx, y: ry + cy, z: rz + cz };
}
```

add comments in code

```
function project2Dto3D(point, camera) {
  // Extract point coordinates
  const { x, y, z } = point;
  // Extract camera parameters
  const { angleX, angleY, angleZ, scale, x: cx, y: cy, z: cz } =
camera;

  // Precompute trigonometric values for rotation angles
  const cosX = Math.cos(angleX);
  const sinX = Math.sin(angleX);
  const cosY = Math.cos(angleY);
  const sinY = Math.sin(angleY);
  const cosZ = Math.cos(angleZ);
  const sinZ = Math.sin(angleZ);

  // Apply rotation around X axis
  const rx = x * cosY * cosZ - y * cosY * sinZ + z * sinY;
  // Apply rotation around Y axis and scale
  const ry = (x * (sinX * sinY * cosZ + cosX * sinZ) + y * (sinX *
sinY * sinZ - cosX * cosZ) + z * sinX * cosY) * scale;
  // Apply rotation around Z axis and scale
  const rz = (-x * (cosX * sinY * cosZ - sinX * sinZ) - y * (cosX *
sinY * sinZ + sinX * cosZ) + z * cosX * cosY) * scale;
```

```

// Translate to camera position
const translatedX = rx + cx;
const translatedY = ry + cy;
const translatedZ = rz + cz;

return { x: translatedX, y: translatedY, z: translatedZ };
}

```

За несколько шагов удалось получить корректный, оптимизированный код.

2.2. Применение ООП для разработки шаблонов

Банк заданий содержит большое количество разнообразных задач по теме «Стереометрия». Поэтому одной из первостепенных задач было сократить код шаблонов и исключить вычислительные ошибки. Для этого были разработаны классы многогранников, которые содержат в себе длины рёбер, объем, площади оснований, а так же тернарную матрицу связности и канонические координаты вершин.

Матрица может содержать значения: 1, 0, либо специальное значение, указывающий на отображении ребра пунктиром.

Пример канонической матрицы связей:

```

[
  [1],
  [0, 1],
  [1, 0, 1],
  [0, 0, 0, 1],
  [1, 0, 0, 0, 1],
  [0, 1, 0, 0, 0, 1],
  [0, 0, 1, 0, 1, 0, 1],
];

```

Листинг 2: Каноническая матрица связей для параллелепипеда

Мы можем опускать конец матрицы, если он состоит только из нулей. И нам не

Определение. Каноническим положением Рис. 1-Рис. 2 будем называть такое расположение многогранника, когда его высота, проходящая через центр масс его основания, совпадает с осью аппликат и началом координат делится пополам.

При таком расположении, начало координат можно расположить в центре иллюстрации. Тогда чертёж не будет чрезмерно смещён ни в одну из сторон.

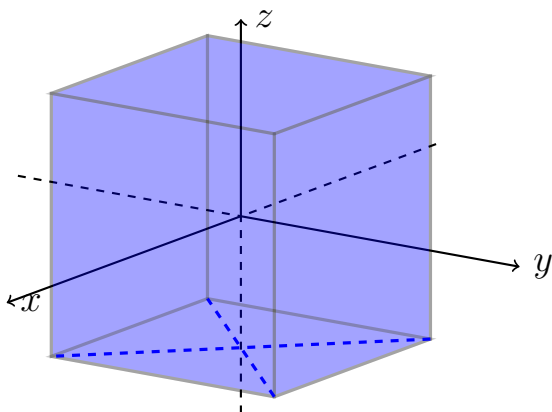


Рис. 1: Каноническое положение для

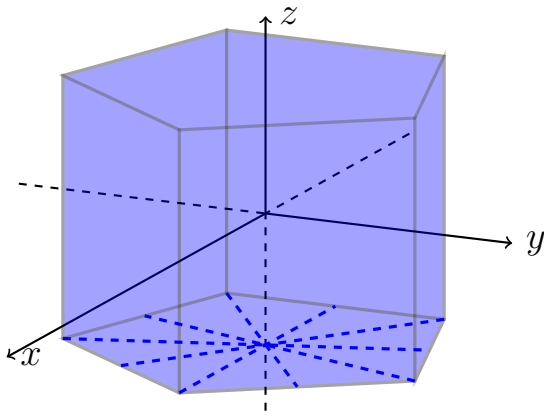


Рис. 2: Каноническое положение для правильной пятиугольной призмы

2.3. Вспомогательные функции

2.3.1. Функции для работы с координатами

function **verticesInGivenRange**(vertex, startX, finishX, startY, finishY)

Возвращает **true**, если двухмерная координата точки **vertex** вида **{x,y}** находится в некоторой прямоугольной области, иначе **false**.

function **autoScale**(vertex3D, camera, vertex2D, startX, finishX, startY, finishY, step, maxScale)

Увеличивает свойство объекта **camera.scale**, до тех пор пока все двухмерные координаты **vertex2D** вида **{x,y}** находится в некоторой прямоугольной области. **step** по умолчанию 0.1.

function **distanceFromPointToSegment**(point, segmentStart, segmentEnd)

Возвращает длину перпендикуляра между двухмерной точкой **point** вида **{x,y}** до отрезка с концами в **segmentStart** и **segmentEnd**.

2.3.2. Функции для работы с canvas

CanvasRenderingContext2D.prototype.drawFigure = **function**(vertex, matrixConnections)

Соединяет линиями точки массива **vertex** с элементами **{x,y}** в соответствии с матрицей связей **matrixConnections**, которая является массивом, содержащим в себе 0, 1 или массив **step**, указывающий на отрисовку пунктиром.

Пример матрицы связей:

```
1  let matrixConnections = [
2      [1],
3      [strok, strok],
4      [0, 0, strok],
5      [1, 0, 0, 1],
6      [0, 1, 0, 1, 1]
7  ];
8
```

```
CanvasRenderingContext2D.prototype.drawFigureVer2 = function()
```

vertex, matrixConnections Соединяет линиями точки массива **vertex** с элементами {x,y} в соответствии с матрицей связей **matrixConnections**. Эта матрица представляет собой объект, где каждое числовое поле соответствует номеру вершины в массиве **vertex**. В каждом поле находится массив номеров других вершин, с которыми должна быть соединена данная вершина.

Пример матрицы связей:

```
1 let matrixConnections = {
2   0: [1, [3, stroke], 5],
3   2: [1, [3, stroke], 7],
4   4: [[3, stroke], 5, 7],
5   9: [1, 8, 10],
6  11: [8, 10, 12],
7  13: [5, 8, 12],
8  15: [7, 10, 12],
9 };
10
```

2.4. Этапы разработки шаблоны с вспомогательным чертежом по теме «Стереометрия»

Для примера возьмём задание №27074 [5].

Заготовка шаблона имеет вид.

```
1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let paint1 = function (ctx) {
6       };
7
8     NATask.setTask({
9       text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем
треугольной пирамиды ABCA_1.',
10      answers: 0,
11      author: ['Суматохина Александра']
12    });
13    NATask.modifiers.addCanvasIllustration({
14      width: 400,
15      height: 400,
16      paint: paint1,
17    });
18  }, 100000);
19 })();
```

1. Создадим объект класса **Parallelepiped** со случайной высотой, шириной и глубиной в заданном диапазоне.

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let par = new Parallelepiped({
6       depth: sl(10, 50),
7       height: sl(10, 50),
8       width: sl(10, 50),
9     });
10
11    let paint1 = function (ctx) {
12    };
13
14    NATask.setTask({
15      text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите
16      объем треугольной пирамиды ABCA_1.',
17      answers: 0,
18      author: ['Суматохина Александра']
19    });
20    NATask.modifiers.addCanvasIllustration({
21      width: 400,
22      height: 400,
23      paint: paint1,
24    }, 100000);
25  })();

```

2. Определим переменную **camera**, которая будет отвечать за положение наблюдателя. И спроецируем канонические координаты параллелепипеда на двумерную плоскость при помощи функции **project3DTo2D**. И отмасштабируем полученные координаты так, чтобы они занимали максимально заполняли спрайт, функцией **autoScale**.

```

1 (function () {
2   retryWhileError(function () {
3     NAinfo.requireApiVersion(0, 2);
4
5     let par = new Parallelepiped({
6       depth: sl(10, 50),
7       height: sl(10, 50),
8       width: sl(10, 50),
9     });
10
11    let camera = {
12      x: 0,
13      y: 0,
14      z: 0,
15      scale: 5,
16
17      rotationX: -Math.PI / 2 + Math.PI / 14,

```

```

18         rotationY: 0,
19         rotationZ: 2 * Math.PI / 3,
20     };
21
22     let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
23
24     autoScale(par.verticesOfFigure, camera, point2DPar, {
25         startX: -180,
26         finishX: 160,
27         startY: -160,
28         finishY: 160,
29         maxScale: 50,
30     });
31
32     point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34     let paint1 = function (ctx) {
35     };
36
37     NATask.setTask({
38         text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9.
Найдите объем треугольной пирамиды ABCA_1.',
39         answers: 0,
40         author: ['Суматохина Александра']
41     });
42     NATask.modifiers.addCanvasIllustration({
43         width: 400,
44         height: 400,
45         paint: paint1,
46     });
47     }, 100000);
48 })();

```

3. Перемещаемся в середину иллюстрации. Отрисовываем фигуру функцией **drawFigure**, передав в неё матрицу связей для параллелепипеда.

```

1 (function () {
2     retryWhileError(function () {
3         NAinfo.requireApiVersion(0, 2);
4
5         let par = new Parallelepiped({
6             depth: sl(10, 50),
7             height: sl(10, 50),
8             width: sl(10, 50),
9         });
10
11         let camera = {
12             x: 0,
13             y: 0,
14             z: 0,

```

```

15         scale: 5,
16
17         rotationX: -Math.PI / 2 + Math.PI / 14,
18         rotationY: 0,
19         rotationZ: 2 * Math.PI / 3,
20     };
21
22     let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
23
24     autoScale(par.verticesOfFigure, camera, point2DPar, {
25         startX: -180,
26         finishX: 160,
27         startY: -160,
28         finishY: 160,
29         maxScale: 50,
30     });
31
32     point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34     let paint1 = function (ctx) {
35         let h = 400;
36         let w = 400;
37         ctx.translate(h / 2, w / 2);
38         ctx.lineWidth = 2;
39         ctx.strokeStyle = om.secondaryBrandColors;
40         let strok = [5, 4];
41         ctx.drawFigure(point2DPar, [
42             [strok],
43             [0, 1],
44             [strok, 0, 1],
45             [0, 0, 0, 1],
46             [strok, 0, 0, 0, 1],
47             [0, 1, 0, 0, 0, 1],
48             [0, 0, 1, 0, 1, 0, 1],
49         ]);
50     };
51
52     NATask.setTask({
53         text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9.
Найдите объем треугольной пирамиды ABСA_1.',
54         answers: 0,
55         author: ['Суматохина Александра']
56     });
57     NATask.modifiers.addCanvasIllustration({
58         width: 400,
59         height: 400,
60         paint: paint1,
61     });
62     }, 100000);
63 })();

```

4. Далее вырезаем из условия значения и заменяем их данными из класса. Впишем ответ. Обособляем имена фигур в `$...$`. Добавляем буквы на вершины параллелепипеда. Добавим модификаторы `NAtask.modifiers.assertSaneDecima` (исключает нецелый ответ) и `NAtask.modifiers.variativeABC(letter)` (заменяет все буквы в задании на случайные).

```
1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let par = new Parallelepiped({
6       depth: sl(10, 50),
7       height: sl(10, 50),
8       width: sl(10, 50),
9     });
10
11     let pyr = new Pyramid({
12       height: par.height,
13       baseArea: 0.5 * par.baseArea,
14     });
15
16     let camera = {
17       x: 0,
18       y: 0,
19       z: 0,
20       scale: 5,
21
22       rotationX: -Math.PI / 2 + Math.PI / 14,
23       rotationY: 0,
24       rotationZ: 2 * Math.PI / 3,
25     };
26
27     let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
28
29     autoScale(par.verticesOfFigure, camera, point2DPar, {
30       startX: -180,
31       finishX: 160,
32       startY: -160,
33       finishY: 160,
34       maxScale: 50,
35     });
36
37     point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
38
39     let letter = ['A', 'B', 'C', 'D', '⊠D', '⊠A', '⊠B', '⊠C',];
40
41     let paint1 = function(ctx) {
```

```

42     let h = 400;
43     let w = 400;
44     ctx.translate(h / 2, w / 2);
45     ctx.lineWidth = 2;
46     ctx.strokeStyle = om.secondaryBrandColors;
47     let strok = [5, 4];
48     ctx.drawFigure(point2DPar, [
49         [strok],
50         [0, 1],
51         [strok, 0, 1],
52         [0, 0, 0, 1],
53         [strok, 0, 0, 0, 1],
54         [0, 1, 0, 0, 0, 1],
55         [0, 0, 1, 0, 1, 0, 1],
56     ]);
57
58     ctx.font = "25px liberation_sans";
59     point2DPar.forEach((elem, i) => ctx.fillText(letter[i],
elem.x, elem.y + ((i < point2DPar.length / 2) ? 15 : -10)));
60     };
61
62     NATask.setTask({
63         text: 'Объем параллелепипеда $ABCD A_1 B_1 C_1 D_1$ равен
64 $'+par.volume+'$. Найдите объем треугольной пирамиды $ABCA_1$.',
65         answers: par.volume/6,
66         author: ['Суматохина Александра']
67     });
68
69     NATask.modifiers.assertSaneDecimals();
70     NATask.modifiers.variativeABC(letter);
71
72     NATask.modifiers.addCanvasIllustration({
73         width: 400,
74         height: 400,
75         paint: paint1,
76     });
77     }, 100000);
78 })();

```

Заключение

За этот год был полностью покрыт открытый банк заданий ФИПИ по темам:

- Планиметрия — 26 шаблонов принято.
- Вектора — 18 шаблонов (10 принято, 8 на рецензировании).
- Стереометрия — 56 шаблонов (7 принято, 49 на рецензировании).
- Теория вероятности — 10 шаблонов на рецензировании.
- Теория вероятности (повышенной сложности) — 11 шаблонов (1 принят 10 на рецензировании).

В ядро проекта добавлены:

- Функции, упрощающие написание шаблонов по темам «Планиметрия» и «Стереометрия».
- Класс многогранников.
- Линейный проектор из $\mathbb{R}^3 \rightarrow \mathbb{R}^2$.

А также сокращён технический долг проекта.

Все добавленные в проект задания можно использовать для составления контрольных работ, проведения текущего контроля знаний учащихся, подготовки к ЕГЭ.

В будущем планируется добавить в проект класс плоских геометрических фигур и использовать в заданиях по теме «Планиметрия» динамические изображения.

Список литературы

- [1] Момот Е. А., Арахов Н. Д. Разработка и внедрение ПО для сбора статистики результатов подготовки к ЕГЭ по математике профильного уровня //Актуальные проблемы прикладной математики, информатики и механики. – 2021. – С. 1-2.
- [2] Открытый банк задач ЕГЭ по Математике.Профильный уровень. – URL: <https://prof.mathege.ru/>
- [3] Федеральный институт педагогических измерений. – URL: <https://fipi.ru/ege/otkrytyy-bank-zadaniy-ege>
- [4] Единый государственный экзамен. – URL: https://ru.wikipedia.org/wiki/Единый_государственный_экзамен
- [5] Решу ЕГЭ - Сдам ГИА. – URL: <https://ege.sdamgia.ru/problem?id=27074>

Приложение

```
1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let a = sl(2, 89);
6     let b = slKrome(a, 1, a - 1);
7
8     let vertices = window.latbukv.iz(4);
9
10    let angle = sl1() ? vertices.slice(0, 3).permuteCyclic(1) :
vertices.slice(0, 3).permuteCyclic(2);
11
12    genAssertZ1000(b / a, 'Кривой ответ');
13
14    let paint1 = function(ctx) {
15      ctx.lineWidth = 2;
16      ctx.strokeStyle = om.primaryBrandColors [0];
17
18      ctx.drawLine(10, 370, 390, 370);
19      ctx.drawLine(10, 370, 180, 200);
20      ctx.drawLine(180, 200, 390, 370);
21
22      //высота
23      ctx.lineWidth = 1.2;
24      ctx.drawLine(180 - 40, 200 - 33, 390, 370);
25      ctx.drawLine(180 - 40, 200 - 33, 10, 370);
26
27      //прямой угол
28      ctx.strokeStyle = om.secondaryBrandColors.iz();
29      ctx.drawLine(180 - 52, 200 - 13, 180 - 35, 200 - 1);
30      ctx.drawLine(180 - 35, 200 - 1, 180 - 23, 200 - 17);
31
32      //штрихи
33      ctx.drawLine(275 + 10, 210 + 90, 300, 200 + 90);
34      ctx.drawLine(80, 200 + 90, 105 - 10, 210 + 90);
35
36      ctx.font = "23px liberation_sans";
37      ctx.fillText(vertices[0], 10 - 5, 370 + 25);
38      ctx.fillText(vertices[1], 180, 200 - 10);
39      ctx.fillText(vertices[2], 390 - 10, 370 + 25);
40      ctx.fillText(vertices[3], 180 - 40, 200 - 40);
41
42    };
43
44    NATask.setTask({
45      text: 'В треугольнике $' + vertices.slice(0, 3).shuffle().join('') +
'$ $' +
46      vertices.slice(0, 2).shuffle().join('') +
```

```

47     '=' + vertices.slice(1, 3).shuffle().join('') + '$, $' +
vertices[0] + vertices[2] + '=' + a + '$, высота $' +
48     [vertices[0], vertices[3]].shuffle().join('') +
49     '$ равна $' + b + '$. Найдите $\sin \angle ' + angle.join('') +
'$.',
50     answers: b / a,
51     analys: '',
52   });
53   NATask.modifiers.addCanvasIllustration({
54     width: 400,
55     height: 400,
56     paint: paint1,
57   });
58   }, 1000);
59   })();
60   // 105

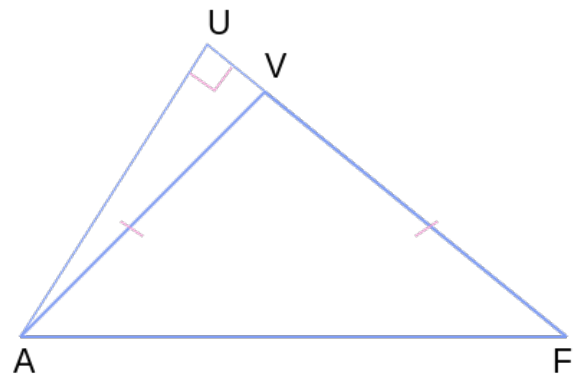
```

Листинг 3: 105.js

Примеры генерируемых задач 105.js

В треугольнике FAF $VA = VF$,
 $AF = 75$, высота AU равна 39. Най-
дите $\sin \angle VFA$.

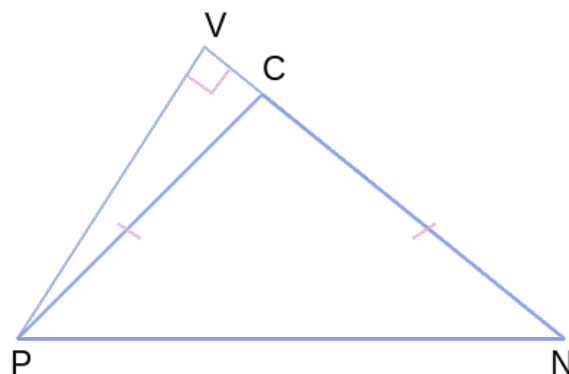
Ответ: 0,52



Приложение. 1

В треугольнике NCP $CP = CN$, $PN = 40$, высота VP равна 6. Найдите $\sin \angle CNP$.

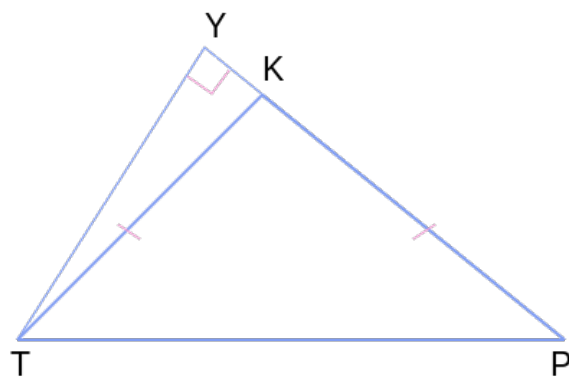
Ответ: 0,15



Приложение. 2

В треугольнике PKT $KT = PK$, $TP = 4$, высота TU равна 2. Найдите $\sin \angle KPT$.

Ответ: 0,5



Приложение. 3

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let stroke = [4, 5];
6
7     let matrixConnections = {
8       0: [1, [3, stroke], 5],
9       2: [1, [3, stroke], 7],
10      4: [
11        [3, stroke],
12        [5, stroke],
13        [11, stroke]
14      ],
15      6: [1, 7, 9],

```

```

16      8: [5, [11, stroke], 13],
17      10: [7, 9, 15],
18      12: [
19          [11, stroke], 13, 15
20      ],
21      14: [9, 13, 15],
22  };
23
24  let par1 = new Parallelepiped({
25      depth: sl(10, 20),
26      height: sl(5, 20),
27      width: sl(10, 20),
28  });
29
30  let par2 = new Parallelepiped({
31      depth: par1.depth,
32      height: sl(5, 20),
33      width: slKrome(par1.width, 5, par1.width - 5),
34  });
35
36  let vertex3D =
37  par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
38  shiftCoordinate3D(elem, {
39      x: 0,
40      y: 0,
41      z: -0.5 * (par1.height + par2.height),
42  })));
43
44  vertex3D = vertex3D.map((elem) => shiftCoordinate3D(elem, {
45      x: 0,
46      y: 0,
47      z: 0.5 * par2.height,
48  })));
49
50  let camera = {
51      x: 0,
52      y: 0,
53      z: 0,
54      scale: 1,
55
56      rotationX: -Math.PI / 2 + Math.PI / 14,
57      rotationY: 0,
58      rotationZ: Math.PI / sl(10, 14),
59  };
60
61  let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
62  camera));
63
64  autoScale(vertex3D, camera, point2D, {
65      startX: -180,
66      finishX: 160,
67      startY: -160,

```

```

65     finishY: 160,
66     maxScale: 100,
67 });
68
69 point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
70 genAssert((point2D[4].x - point2D[8].x).abs() > 20);
71 genAssert((point2D[4].y - point2D[13].y).abs() > 50);
72 genAssert((point2D[12].x - point2D[14].x).abs() > 40);
73
74 let rand = sl1();
75
76 let paint1 = function(ctx) {
77     let h = 400;
78     let w = 400;
79     ctx.translate(w / 2, h / 2);
80     ctx.lineWidth = 2;
81     ctx.strokeStyle = om.secondaryBrandColors;
82     ctx.drawFigureVer2(point2D, matrixConnections);
83
84     if (point2D[4].x > point2D[8].x) {
85         let point = [point2D[4], point2D[5], point2D[8],
point2D[13]].mt_coordinatesOfIntersectionOfTwoSegments();
86         ctx.drawLine(point2D[5].x, point2D[5].y, point.x, point.y);
87     } else {
88         ctx.drawLine(point2D[4].x, point2D[4].y, point2D[5].x,
point2D[5].y);
89         let point = [point2D[4], point2D[11], point2D[8],
point2D[13]].mt_coordinatesOfIntersectionOfTwoSegments();
90         ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
91     }
92
93     ctx.font = "20px liberation_sans";
94     ctx.signSegmentInMiddle(point2D[2].x, point2D[2].y, point2D[7].x,
point2D[7].y, par1.height, 10, 20);
95     ctx.signSegmentInMiddle(point2D[10].x, point2D[10].y,
point2D[15].x, point2D[15].y, par2.height, 10, 20);
96     ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
point2D[15].x, point2D[15].y, par2.width, -10, 20);
97     ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
point2D[1].y, par1.width, 18, 20);
98     ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
point2D[2].y, par1.depth, 18, 20);
99 };
100 NATask.setTask({
101     text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] +
102         ' многогранника, изображённого на рисунке все( двугранные углы -
прямые).',
103     answers: [par1.surfaceArea + par2.surfaceArea - 2 * par2.baseArea,
par1.volume + par2.volume][rand],
104 });
105 NATask.modifiers.addCanvasIllustration({
106     width: 400,

```

```

107         height: 400,
108         paint: paint1,
109     });
110 },
111     1000);
112
113 })();
114 //27193 25671 25673 25675 25677 25679

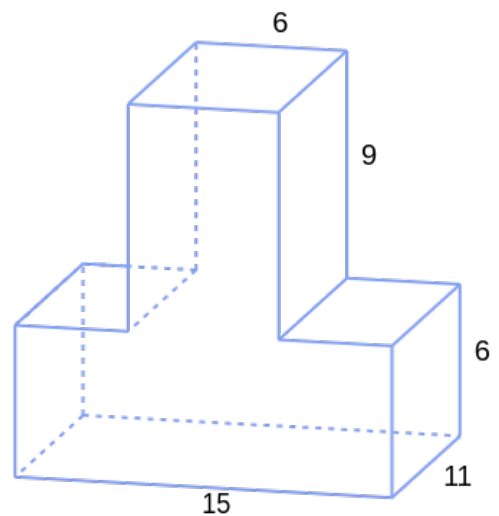
```

Листинг 4: 29193.js

Примеры генерируемых задач 27193.js

Найдите площадь поверхности многогранника, изображённого на рисунке (все двугранные углы – прямые).

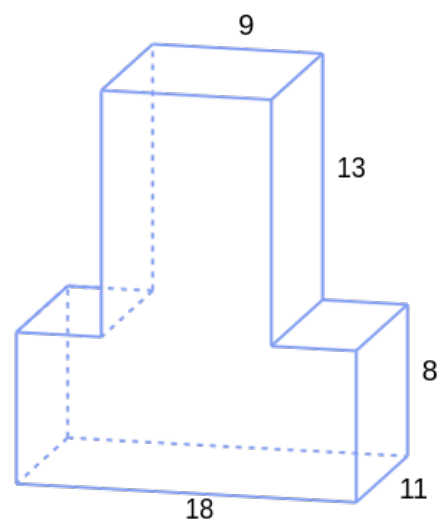
Ответ: 948



Приложение. 4

Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

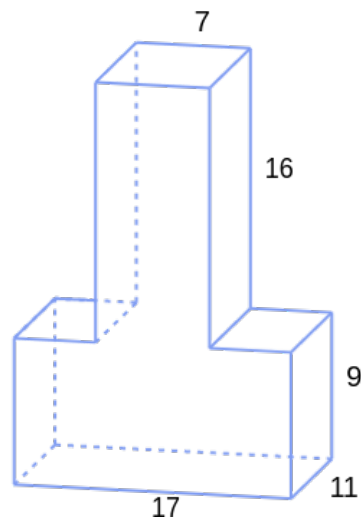
Ответ: 2871



Приложение. 5

Найдите площадь поверхности многогранника, изображённого на рисунке (все двугранные углы – прямые).

Ответ: 1454



Приложение. 6

```

1 (function() {
2   retryWhileError(function() {
3     lx_declareClarifiedPhrase('сторона', 'основания');
4     NAinfo.requireApiVersion(0, 2);
5
6     let pyr = new RegularPyramid({
7       height: sl(20, 50),
8       baseSide: sl(20, 40),
9       numberSide: 4
10    });
11
12    pyr.verticesOfFigure.push({
13      x: 0,
14      y: 0,
15      z: pyr.verticesOfFigure[0].z
16    });
17
18    let question = [
19      [sklonlxkand('боковое ребро'), pyr.sideEdge],
20      [sklonlxkand('объём'), pyr.volume],
21    ].shuffle();
22    question.unshift([sklonlxkand('сторона основания'), pyr.baseSide]);
23
24    let camera = {
25      x: 0,
26      y: 0,
27      z: 0,
28      scale: 5,
29
30      rotationX: -Math.PI / 2 + Math.PI / 14,
31      rotationY: 0,
32      rotationZ: [1, 2].iz() * Math.PI / 3,
33    };

```



```

34
35   let point2DPyr = pyr.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
36
37   autoScale(pyr.verticesOfFigure, camera, point2DPyr, {
38     startX: -180,
39     finishX: 160,
40     startY: -160,
41     finishY: 160,
42     maxScale: 50,
43   });
44
45   point2DPyr = pyr.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
46
47   let letters = ['A', 'B', 'C', 'D', 'S', 'O'];
48
49   let strok = [5, 4];
50
51
52   let paint1 = function(ctx) {
53     let h = 400;
54     let w = 400;
55     ctx.translate(h / 2, w / 2);
56     ctx.lineWidth = 2;
57     ctx.strokeStyle = om.secondaryBrandColors;
58     ctx.drawFigure(point2DPyr, [
59       [1],
60       [strok, strok],
61       [1, strok, strok],
62       [1, 1, strok, 1, 0, strok],
63     ]);
64
65     ctx.font = "30px liberation_sans";
66     point2DPyr.forEach((elem, i) => ctx.fillText(letters[i], elem.x,
elem.y + ((i != point2DPyr.length - 2) ? 15 : -
67       10)));
68   };
69
70   NATask.setTask({
71     text: 'В правильной четырёхугольной пирамиде $SABCD$ с ' + 'основанием
$ABCD$ ' +
72       [question[0][0].ie + ' рав' + ['ен', 'на',
'но']][question[0][0].rod] + ' $' + question[0][1].pow(2).texsqrt(1) +
'$ ',
73     question[1][0].ie + ' рав' + ['ен', 'на', 'но']][question[1][0].rod]
+ ' $' + question[1][1].pow(2).texsqrt(1) + '$ '
74     ].shuffleJoin(', ') +
75     '. Найдите ' + question[2][0].ve + ' пирамиды.',
76     answers: question[2][1],
77     author: ['Суматохина Александра'],
78   });

```

```

79   NAtask.modifiers.variativeABC(letters);
80   NAtask.modifiers.multiplyAnswerBySqrt(13);
81   NAtask.modifiers.allDecimalsToStandard(true);
82   NAtask.modifiers.assertSaneDecimals();
83   NAtask.modifiers.addCanvasIllustration({
84     width: 400,
85     height: 400,
86     paint: paint1,
87   });
88   },10);
89 })();
90 //https://ege314.ru/8-stereometriya-ege/reshenie-3011/
91 //3011

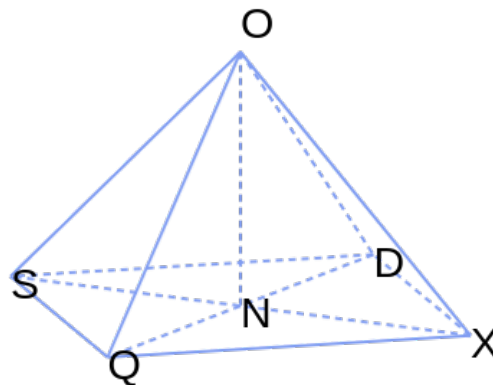
```

Листинг 5: 3011.js

Примеры генерируемых задач 3011.js

В правильной четырёхугольной пирамиде $OQSDX$ с основанием $QSDX$ боковое ребро равно $\sqrt{1489,5}$, сторона основания равна 39. Найдите объём пирамиды.

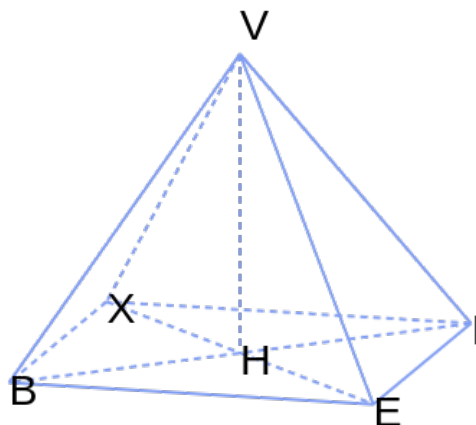
Ответ: 13689



Приложение. 7

В правильной четырёхугольной пирамиде $VEBXI$ с основанием $EBXI$ боковое ребро равно $\sqrt{848,5}$, сторона основания равна 27. Найдите объём пирамиды.

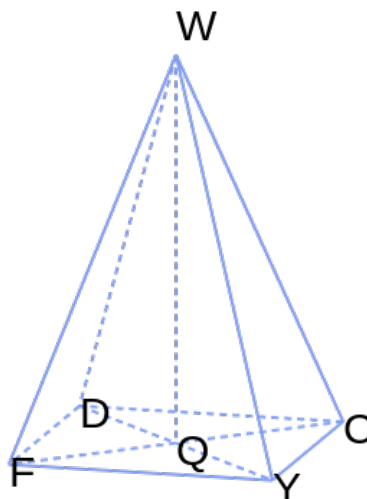
Ответ: 5346



Приложение. 8

В правильной четырёхугольной пирамиде $WYFDC$ с основанием $YFDC$ боковое ребро равно $\sqrt{1513}$, сторона основания равна 24. Найдите объём пирамиды.

Ответ: 6720



Приложение. 9

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let angle = sl(2, 44);
6     let condition = [
7       ['большой', 45 + angle],
8       ['меньший', 45 - angle]
9     ].iz();
10
11     let paint1 = function(ctx) {
12       ctx.lineWidth = 2;
13
14       let angle = Math.PI/2.9;
15

```

```

16     ctx.strokeStyle = om.secondaryBrandColors.iz();
17     ctx.drawLine(10, 250, 390-8, 250);
18     let ver = ctx.drawLineAtAngle(10, 250, -angle, 200-25);
19     ctx.drawLineAtAngle(ver.x, ver.y, -angle+Math.PI/2, 350-20);
20     //штрихи
21     ctx.strokeInMiddleOfSegment(10, 250, (390-8)/2, 250, 10);
22     ctx.strokeInMiddleOfSegment(10, 250, 3*(390-8)/2, 250, 10);
23     //биссектриса
24     ctx.strokeStyle = om.primaryBrandColors [0];
25     let bis = ctx.drawLineAtAngle(ver.x, ver.y,
(-angle+Math.PI/2)+Math.PI/4, 160+2);
26     //медиана
27     ctx.strokeStyle = om.primaryBrandColors [1];
28     ctx.drawLine(ver.x, ver.y,(390-8)/2, 250);
29
30     ctx.strokeStyle = om.primaryBrandColors [0];
31     ctx.arcBetweenSegments([10, 250,ver.x, ver.y, bis.x, bis.y], 30);
32     ctx.arcBetweenSegments([390-8, 250,ver.x, ver.y, bis.x, bis.y], 38);
33
34 };
35
36 NATask.setTask({
37     text: 'Угол между биссектрисой и медианой прямоугольного треугольника, ' +
38         'проведёнными из вершины прямого угла, равен $' + angle + '~{\circ}$.'
39     'Найдите ' + condition[0] + ' угол прямоугольного треугольника. Ответ
дайте в градусах.',
40     answers: condition[1],
41     analys: '',
42 });
43 NATask.modifiers.addCanvasIllustration({
44     width: 400,
45     height: 400,
46     paint: paint1,
47 });
48 }, 1000);
49 })();
50 //2069

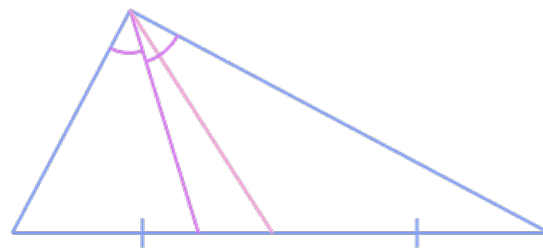
```

Листинг 6: 2069.js

Примеры генерируемых задач 2069.js

Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен 38° . Найдите больший угол прямоугольного треугольника. Ответ дайте в градусах.

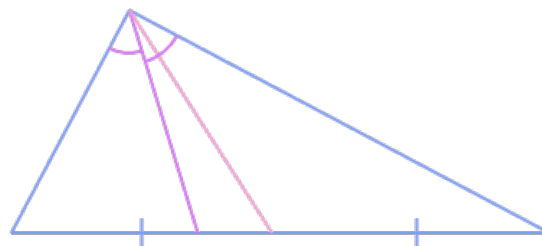
Ответ: 83



Приложение. 10

Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен 8° . Найдите меньший угол прямоугольного треугольника. Ответ дайте в градусах.

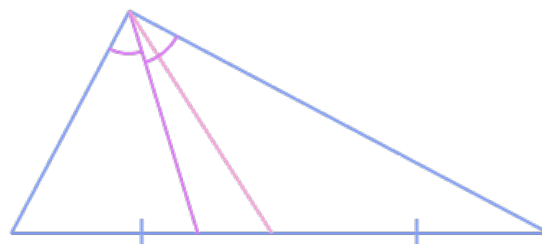
Ответ: 37



Приложение. 11

Угол между биссектрисой и медианой прямоугольного треугольника, проведёнными из вершины прямого угла, равен 31° . Найдите меньший угол прямоугольного треугольника. Ответ дайте в градусах.

Ответ: 14



Приложение. 12

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let angle = sl(2, 89);
6
7     let vertices = window.latbukv.iz(6);
8
9     let rand = sl1();
10
11    let paint1 = function(ctx) {
12      ctx.lineWidth = 2;
13
14      let angle = -Math.PI / 3.2;
15      ctx.strokeStyle = om.secondaryBrandColors.iz();
16
17      let vertex = ctx.drawLineAtAngle(10, 370, angle, 400);
18      ctx.drawLine(10, 370, 390, 370);
19      ctx.drawLine(390, 370, vertex.x, vertex.y);
20
21      //Биссектрисы
22      let bisector1 = ctx.drawLineAtAngle(10, 370, angle / 2, 345);
23      let bisector2 = ctx.drawLineAtAngle(390, 370, Math.atan2(-370 +
vertex.y, -390 + vertex.x) / 2 - Math.PI / 2, 317);
24
25      //Углы
26      ctx.strokeStyle = om.primaryBrandColors[rand];
27      ctx.arcBetweenSegmentsCount([vertex.x, vertex.y, 10,
370].concat([bisector1.x, bisector1.y]), 30, 2);
28      ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
370, 390, 370])), 40, 2);
29
30      ctx.strokeStyle = om.primaryBrandColors[rand];

```

```

31     ctx.arcBetweenSegments([10, 370, 390, 370].concat([bisector2.x,
bisector2.y]), 30);
32     ctx.arcBetweenSegments(([bisector2.x, bisector2.y].concat([390, 370,
vertex.x, vertex.y])), 40);
33
34     ctx.strokeStyle = om.primaryBrandColors[1 - rand];
35     ctx.arcBetweenSegmentsCount([bisector1.x, bisector1.y].concat([10,
370]).concat([bisector2.x, bisector2.y]).concat([390, 370]), 25, 3);
36
37     ctx.font = "23px liberation_sans";
38     ctx.fillText(vertices[0], vertex.x, vertex.y - 10);
39     ctx.fillText(vertices[1], 10 - 5, 370 + 20);
40     ctx.fillText(vertices[2], 390 - 20, 370 + 20);
41
42     ctx.fillText(vertices[3], bisector1.x, bisector1.y);
43     ctx.fillText(vertices[4], bisector2.x - 20, bisector2.y);
44
45     ctx.fillText(vertices[5], 210, 250);
46 };
47
48 NATask.setTask({
49     text: 'В треугольнике $' + vertices.slice(0, 3).join('') + '$ угол $'
+ vertices[0] + '$ равен $' + angle +
50     '~{\c} $, углы $' + vertices[1] + '$ и $' + vertices[2] + '$ -
острые, ' +
51     'биссектрисы $' + [vertices[1], vertices[3]].shuffleJoin() + '$ и
$' + [vertices[2], vertices[4]].shuffleJoin() +
52     '$ пересекаются в точке $' + vertices[5] + '$. Найдите угол $' +
vertices[2] + vertices[5] + vertices[1] +
53     '$. Ответ дайте в градусах.',
54     answers: 90 + 0.5 * angle,
55     analys: '',
56 });
57 NATask.modifiers.addCanvasIllustration({
58     width: 400,
59     height: 400,
60     paint: paint1,
61 });
62 }, 1000);
63 })();
64 //27764 628357 628475 47369 47371 47373 47375 47377 47379 47381 47383
47385 47387 47389 47391 47393 47395 47397 47399 47401 47403 47405 47407
47409 47411 47413 47415 47417 47419 47421 47423 47425 47427 47429 47431
47433 47435 47437 47439 47441 47443 47445 47447 47449 47451 47453

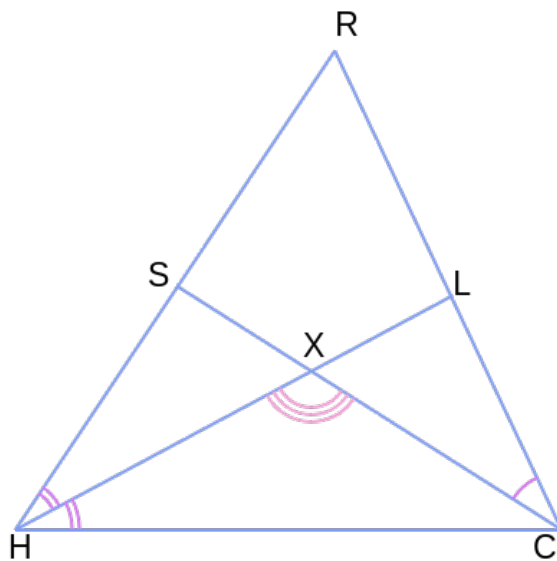
```

Листинг 7: 27764.js

Примеры генерируемых задач 27764.js

В треугольнике RHC угол R равен 67° , углы H и C – острые, биссектрисы LH и SC пересекаются в точке X . Найдите угол CXH . Ответ дайте в градусах.

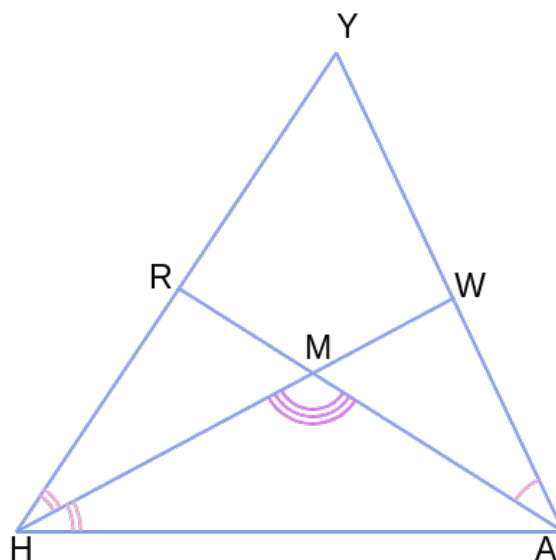
Ответ: 123,5



Приложение. 13

В треугольнике YHA угол Y равен 45° , углы H и A – острые, биссектрисы HW и AR пересекаются в точке M . Найдите угол AMH . Ответ дайте в градусах.

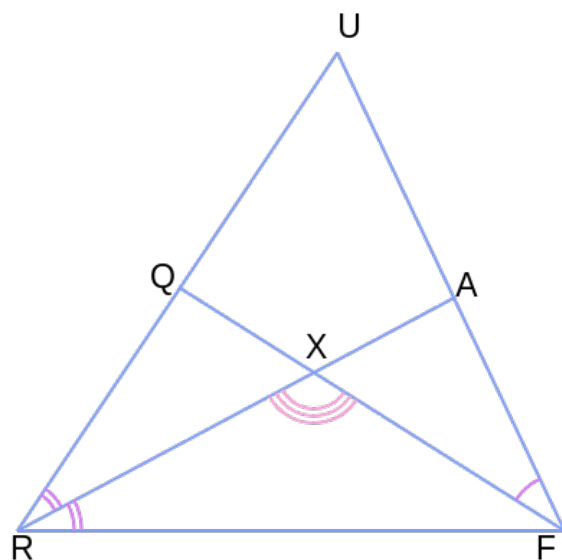
Ответ: 112,5



Приложение. 14

В треугольнике URF угол U равен 7° , углы R и F – острые, биссектрисы AR и FQ пересекаются в точке X . Найдите угол FXR . Ответ дайте в градусах.

Ответ: 93,5



Приложение. 15

Шаблоны по теме «Стереометрия»

```
1 (function() {
2   retryWhileError(function() {
3
4     let pyr1 = new RegularPyramid({
5       height: sl(30, 70),
6       baseSide: sl(20, 50),
7       numberSide: 4
8     });
9
10    let pyr2 = new Pyramid({
11      height: 0.5 * pyr1.height,
12      baseArea: 0.5 * pyr1.baseArea,
13    });
14
15    pyr1.verticesOfFigure =
coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
pyr1.verticesOfFigure[4]);
16
17    let camera = {
18      x: 0,
19      y: 0,
20      z: 0,
21      scale: 5,
22
23      rotationX: -Math.PI / 2 + Math.PI / 14,
24      rotationY: 0,
25      rotationZ: [1, 2].iz() * Math.PI / 3,
26    };
27
28    let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
29
30    autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
31      startX: -180,
32      finishX: 160,
33      startY: -160,
34      finishY: 160,
35      maxScale: 50,
36    });
37
38    point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
39
40    let letters = ['A', 'B', 'C', 'D', 'S', 'E'];
41    let strok = [5, 4];
42
43    let paint1 = function(ctx) {
44      let h = 400;
45      let w = 400;
46      ctx.translate(h / 2, w / 2);
```

```

47     ctx.lineWidth = 2;
48     ctx.strokeStyle = om.secondaryBrandColors;
49     ctx.drawFigure(point2DPyr, [
50         [1],
51         [0, strok],
52         [1, strok, strok],
53         [1, 1, strok, 1, ],
54         [0, 1, 0, 1, 0]
55     ]);
56
57     ctx.font = "30px liberation_sans";
58     point2DPyr.forEach((elem, i) => ctx.fillText(letters[i], elem.x,
59         elem.y + ((i <= point2DPyr.length - 3) ? 15 : -
60             10)));
61
62
63     NATask.setTask({
64         text: 'Объём правильной четырёхугольной пирамиды $SABCD$ равен $' +
65         pyr1.volume.pow(2).texsqrt(1) + '$. ' +
66         'Точка $E$ - середина ребра $SA$. Найдите объём треугольной пирамиды
67         $EABD$.',
68         answers: pyr2.volume,
69         author: ['Суматохина Александра'],
70     });
71     NATask.modifiers.variativeABC(letters);
72     NATask.modifiers.multiplyAnswerBySqrt(13);
73     NATask.modifiers.allDecimalsToStandard(true);
74     NATask.modifiers.assertSaneDecimals();
75     NATask.modifiers.addCanvasIllustration({
76         width: 400,
77         height: 400,
78         paint: paint1,
79     });
80 //27114 75015 75063 519535 75017 75019 75021 75023 75025 75027 75029 75031
81     75033 75035 75037 75039 75041 75043 75045 75047 75049 75051 75053 75055
82     75057 75059 75061

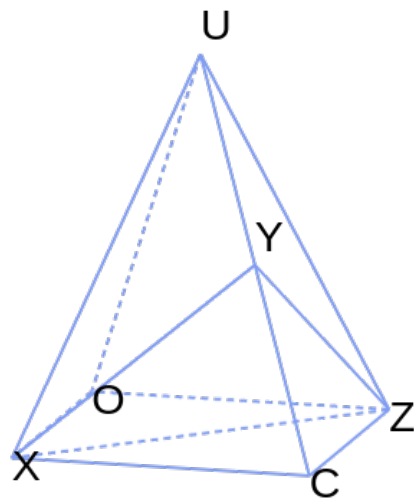
```

Листинг 8: 27114.js

Примеры генерируемых задач 27114.js

Объём правильной четырёхугольной пирамиды $UCXOZ$ равен 31164. Точка Y – середина ребра UC . Найдите объём треугольной пирамиды $YCXZ$.

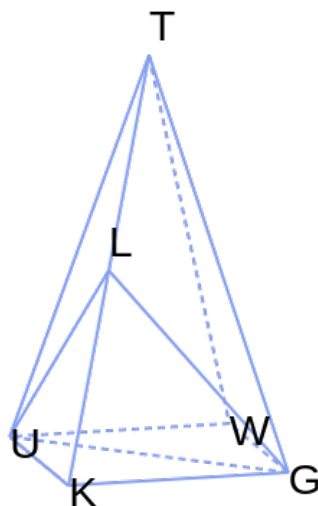
Ответ: 7791



Приложение. 16

Объём правильной четырёхугольной пирамиды $TKUWG$ равен 4800. Точка L – середина ребра TK . Найдите объём треугольной пирамиды $LKUG$.

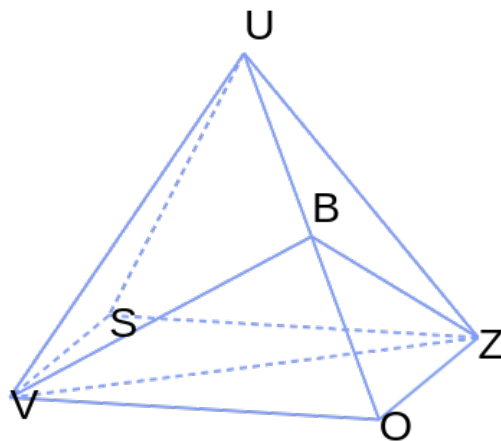
Ответ: 1200



Приложение. 17

Объём правильной четырёхугольной пирамиды $UOVSZ$ равен 25650. Точка B – середина ребра UO . Найдите объём треугольной пирамиды $BOVZ$.

Ответ: 6412,5



Приложение. 18

```

1 (function() {
2   retryWhileError(function() {
3
4     let pyr1 = new RegularPyramid({
5       height: sl(10, 30)*(3).sqrt(),
6       baseSide: sl(20, 50),
7       numberSide: 3
8     });
9
10    let pyr2 = new RegularPyramid({
11      height: pyr1.height,
12      baseSide: 0.5 * pyr1.baseSide,
13      numberSide: 3
14    });
15
16    pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
17      pyr1.verticesOfFigure[1]));
18
19    pyr1.verticesOfFigure.push(coordinatesMiddleOfSegment3D(pyr1.verticesOfFigure[0],
20      pyr1.verticesOfFigure[2]));
21
22    let strok = [5, 4];
23
24    let camera = {
25      x: 0,
26      y: 0,
27      z: 0,
28      scale: 5,
29
30      rotationX: -Math.PI / 2 + Math.PI / 14,
31      rotationY: 0,
32      rotationZ: sl(1,2)* Math.PI / 8,
33    };
34  });
35 }());

```

```

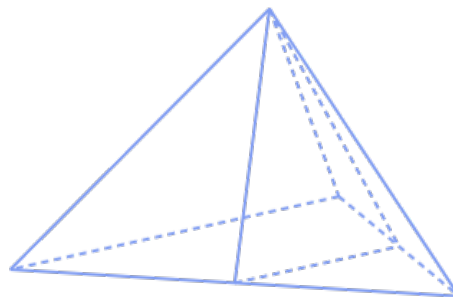
30 };
31
32 let point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
33
34 autoScale(pyr1.verticesOfFigure, camera, point2DPyr, {
35     startX: -180,
36     finishX: 160,
37     startY: -160,
38     finishY: 160,
39     maxScale: 50,
40 });
41
42 point2DPyr = pyr1.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));
43
44 let paint1 = function(ctx) {
45     let h = 400;
46     let w = 400;
47     ctx.translate(h / 2, w / 2);
48     ctx.lineWidth = 2;
49     ctx.strokeStyle = om.secondaryBrandColors;
50     ctx.drawFigure(point2DPyr, [
51         [1],
52         [strok, strok],
53         [1, 1, strok, 0, 1, strok],
54         [0, 0, 0, 0, 0, strok]
55     ]);
56 };
57
58 NATask.setTask({
59     text: ' Объем треугольной пирамиды равен $' + pyr1.volume + '$. ' +
60         'Через вершину пирамиды и среднюю линию её основания проведена плоскость
см(. рисунок). ' +
61         'Найдите объем отсечённой треугольной пирамиды.',
62     answers: pyr2.volume,
63     author: ['Суматохина Александра'],
64 });
65 NATask.modifiers.multiplyAnswerBySqrt(13);
66 NATask.modifiers.allDecimalsToStandard(true);
67 NATask.modifiers.assertSaneDecimals();
68 NATask.modifiers.addCanvasIllustration({
69     width: 400,
70     height: 400,
71     paint: paint1,
72 });
73 }, 1000);
74 })();
75 //27115 75065 75109 75113 514460 75067 75069 75071 75073 75075 75077 75079
75081 75083 75085 75087 75089 75091 75093 75095 75097 75099 75101 75103
75105 75107 75111

```

Примеры генерируемых задач 27115.js

Объём треугольной пирамиды равен 2560. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

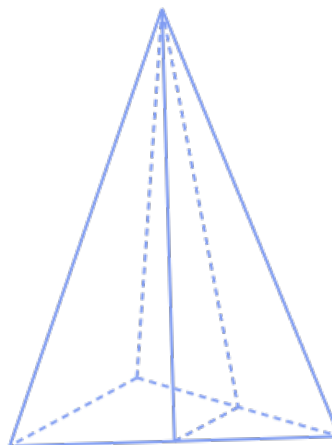
Ответ: 640



Приложение. 19

Объём треугольной пирамиды равен 4950. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

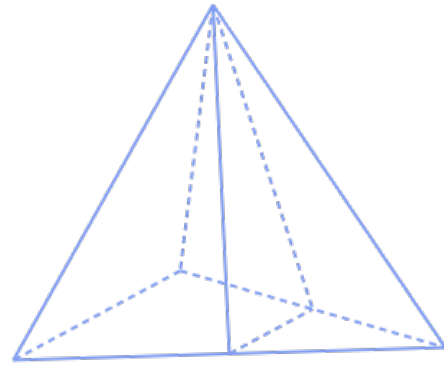
Ответ: 1237,5



Приложение. 20

Объём треугольной пирамиды равен 12096. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.

Ответ: 3024



Приложение. 21

```

1 (function() {
2   retryWhileError(function() {
3     NAinfo.requireApiVersion(0, 2);
4
5     let stroke = [4, 5];
6
7     let matrixConnections = {
8       0: [1, [3, stroke], 5],
9       2: [1, [3, stroke], 7],
10      4: [[3, stroke], 5, 7],
11      9: [1, 8, 10],
12      11: [8, 10, 12],
13      13: [5, 8, 12],
14      15: [7, 10, 12],
15    };
16
17    let par1 = new Parallelepiped({
18      depth: sl(10, 20),
19      height: sl(10, 20),
20      width: sl(10, 20),
21    });
22
23    let par2 = new Parallelepiped({
24      depth: slKrome(par1.depth, 5, par1.depth - 3),
25      height: slKrome(par1.height, 5, par1.height - 3),
26      width: slKrome(par1.width, 5, par1.width - 5),
27    });
28
29    let vertex3D =
30    par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
31      shiftCoordinate3D(elem, {
32        x: -0.5 * (par1.width - par2.width),
33        y: -0.5 * (par1.depth - par2.depth),

```



```

32     z: -0.5 * (par1.height - par2.height),
33   }));
34
35   let camera = {
36     x: 0,
37     y: 0,
38     z: 0,
39     scale: 1,
40
41     rotationX: -Math.PI / 2 + Math.PI / 14,
42     rotationY: 0,
43     rotationZ: Math.PI / sl(10, 14),
44   };
45
46   let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
47     camera));
48
49   autoScale(vertex3D, camera, point2D, {
50     startX: -180,
51     finishX: 160,
52     startY: -160,
53     finishY: 160,
54     maxScale: 100,
55   });
56
57   point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
58   genAssert((point2D[3].y - point2D[11].y).abs() > 20);
59   genAssert((point2D[3].y - point2D[8].y).abs() > 20);
60   genAssert((point2D[13].x - point2D[14].x).abs() > 20);
61
62   let rand = sl1();
63
64   let paint1 = function(ctx) {
65     let h = 400;
66     let w = 400;
67     ctx.translate(w / 2, h / 2);
68     ctx.lineWidth = 2;
69     ctx.strokeStyle = om.secondaryBrandColors;
70     ctx.drawFigureVer2(point2D, matrixConnections);
71
72     ctx.font = "20px liberation_sans";
73     ctx.signSegmentInMiddle(point2D[4].x, point2D[4].y, point2D[7].x,
74       point2D[7].y, par1.height, -10, 20);
75     ctx.signSegmentInMiddle(point2D[10].x, point2D[10].y,
76       point2D[15].x, point2D[15].y, par2.height, -20, 20);
77     ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
78       point2D[15].x, point2D[15].y, par2.width, 20, 20);
79     ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
80       point2D[1].y, par1.width, 18, 20);
81     ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
82       point2D[2].y, par1.depth, 18, 20);
83     ctx.signSegmentInMiddle(point2D[9].x, point2D[9].y, point2D[10].x,

```

```

78     point2D[10].y, par2.depth, 15, 20);
79     };
80     NATask.setTask({
81         text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] +
            ' многогранника, изображённого на рисунке все( двугранные углы -
            прямые).',
82         answers: [par1.surfaceArea, par1.volume - par2.volume][rand],
83     });
84     NATask.modifiers.addCanvasIllustration({
85         width: 400,
86         height: 400,
87         paint: paint1,
88     });
89 },
90 1000);
91
92 })();
93 //27193 25671 25673 25675 25677 25679

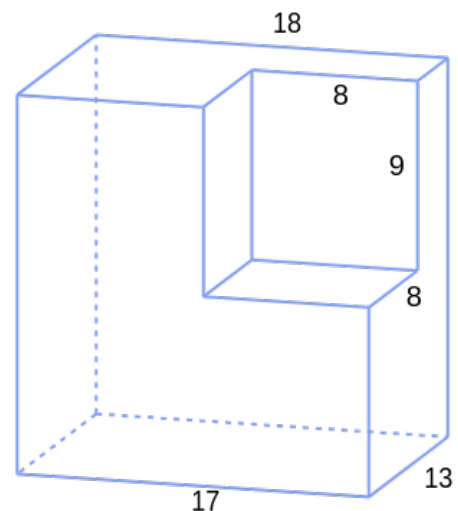
```

Листинг 10: 12.js

Примеры генерируемых задач 12.js

Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

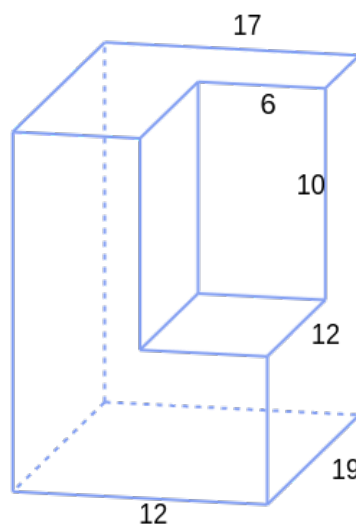
Ответ: 3402



Приложение. 22

Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

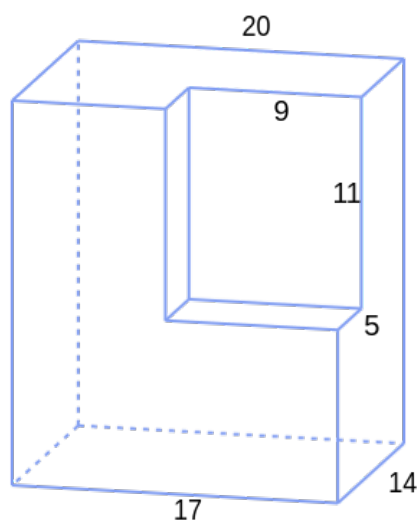
Ответ: 3156



Приложение. 23

Найдите объём многогранника, изображённого на рисунке (все двугранные углы – прямые).

Ответ: 4265



Приложение. 24

```
1 (function() {  
2   retryWhileError(function() {  
3     NAinfo.requireApiVersion(0, 2);  
4  
5     let stroke = [4, 2];  
6  
7     let matrixConnections = {  
8       0: [1, [3, stroke], 5],  
9       2: [1, [3, stroke], 7],  
10      4: [  
11        [3, stroke], 5, [11, stroke]  
12      ],  
13      6: [1, 7, 9],  
14      8: [5, [11, stroke], 13],  
15      10: [  
16        [3, stroke], 5, [11, stroke]  
17      ],  
18      12: [5, 13, 15],  
19      14: [15, [11, stroke], 13],  
20      16: [15, 13, 15],  
21      18: [15, [11, stroke], 13],  
22      20: [15, 13, 15],  
23      22: [15, [11, stroke], 13],  
24      24: [15, 13, 15],  
25      26: [15, [11, stroke], 13],  
26      28: [15, 13, 15],  
27      30: [15, [11, stroke], 13],  
28      32: [15, 13, 15],  
29      34: [15, [11, stroke], 13],  
30      36: [15, 13, 15],  
31      38: [15, [11, stroke], 13],  
32      40: [15, 13, 15],  
33      42: [15, [11, stroke], 13],  
34      44: [15, 13, 15],  
35      46: [15, [11, stroke], 13],  
36      48: [15, 13, 15],  
37      50: [15, [11, stroke], 13],  
38      52: [15, 13, 15],  
39      54: [15, [11, stroke], 13],  
40      56: [15, 13, 15],  
41      58: [15, [11, stroke], 13],  
42      60: [15, 13, 15],  
43      62: [15, [11, stroke], 13],  
44      64: [15, 13, 15],  
45      66: [15, [11, stroke], 13],  
46      68: [15, 13, 15],  
47      70: [15, [11, stroke], 13],  
48      72: [15, 13, 15],  
49      74: [15, [11, stroke], 13],  
50      76: [15, 13, 15],  
51      78: [15, [11, stroke], 13],  
52      80: [15, 13, 15],  
53      82: [15, [11, stroke], 13],  
54      84: [15, 13, 15],  
55      86: [15, [11, stroke], 13],  
56      88: [15, 13, 15],  
57      90: [15, [11, stroke], 13],  
58      92: [15, 13, 15],  
59      94: [15, [11, stroke], 13],  
60      96: [15, 13, 15],  
61      98: [15, [11, stroke], 13],  
62      100: [15, 13, 15],  
63      102: [15, [11, stroke], 13],  
64      104: [15, 13, 15],  
65      106: [15, [11, stroke], 13],  
66      108: [15, 13, 15],  
67      110: [15, [11, stroke], 13],  
68      112: [15, 13, 15],  
69      114: [15, [11, stroke], 13],  
70      116: [15, 13, 15],  
71      118: [15, [11, stroke], 13],  
72      120: [15, 13, 15],  
73      122: [15, [11, stroke], 13],  
74      124: [15, 13, 15],  
75      126: [15, [11, stroke], 13],  
76      128: [15, 13, 15],  
77      130: [15, [11, stroke], 13],  
78      132: [15, 13, 15],  
79      134: [15, [11, stroke], 13],  
80      136: [15, 13, 15],  
81      138: [15, [11, stroke], 13],  
82      140: [15, 13, 15],  
83      142: [15, [11, stroke], 13],  
84      144: [15, 13, 15],  
85      146: [15, [11, stroke], 13],  
86      148: [15, 13, 15],  
87      150: [15, [11, stroke], 13],  
88      152: [15, 13, 15],  
89      154: [15, [11, stroke], 13],  
90      156: [15, 13, 15],  
91      158: [15, [11, stroke], 13],  
92      160: [15, 13, 15],  
93      162: [15, [11, stroke], 13],  
94      164: [15, 13, 15],  
95      166: [15, [11, stroke], 13],  
96      168: [15, 13, 15],  
97      170: [15, [11, stroke], 13],  
98      172: [15, 13, 15],  
99      174: [15, [11, stroke], 13],  
100     176: [15, 13, 15],  
101     178: [15, [11, stroke], 13],  
102     180: [15, 13, 15],  
103     182: [15, [11, stroke], 13],  
104     184: [15, 13, 15],  
105     186: [15, [11, stroke], 13],  
106     188: [15, 13, 15],  
107     190: [15, [11, stroke], 13],  
108     192: [15, 13, 15],  
109     194: [15, [11, stroke], 13],  
110     196: [15, 13, 15],  
111     198: [15, [11, stroke], 13],  
112     200: [15, 13, 15],  
113     202: [15, [11, stroke], 13],  
114     204: [15, 13, 15],  
115     206: [15, [11, stroke], 13],  
116     208: [15, 13, 15],  
117     210: [15, [11, stroke], 13],  
118     212: [15, 13, 15],  
119     214: [15, [11, stroke], 13],  
120     216: [15, 13, 15],  
121     218: [15, [11, stroke], 13],  
122     220: [15, 13, 15],  
123     222: [15, [11, stroke], 13],  
124     224: [15, 13, 15],  
125     226: [15, [11, stroke], 13],  
126     228: [15, 13, 15],  
127     230: [15, [11, stroke], 13],  
128     232: [15, 13, 15],  
129     234: [15, [11, stroke], 13],  
130     236: [15, 13, 15],  
131     238: [15, [11, stroke], 13],  
132     240: [15, 13, 15],  
133     242: [15, [11, stroke], 13],  
134     244: [15, 13, 15],  
135     246: [15, [11, stroke], 13],  
136     248: [15, 13, 15],  
137     250: [15, [11, stroke], 13],  
138     252: [15, 13, 15],  
139     254: [15, [11, stroke], 13],  
140     256: [15, 13, 15],  
141     258: [15, [11, stroke], 13],  
142     260: [15, 13, 15],  
143     262: [15, [11, stroke], 13],  
144     264: [15, 13, 15],  
145     266: [15, [11, stroke], 13],  
146     268: [15, 13, 15],  
147     270: [15, [11, stroke], 13],  
148     272: [15, 13, 15],  
149     274: [15, [11, stroke], 13],  
150     276: [15, 13, 15],  
151     278: [15, [11, stroke], 13],  
152     280: [15, 13, 15],  
153     282: [15, [11, stroke], 13],  
154     284: [15, 13, 15],  
155     286: [15, [11, stroke], 13],  
156     288: [15, 13, 15],  
157     290: [15, [11, stroke], 13],  
158     292: [15, 13, 15],  
159     294: [15, [11, stroke], 13],  
160     296: [15, 13, 15],  
161     298: [15, [11, stroke], 13],  
162     300: [15, 13, 15],  
163     302: [15, [11, stroke], 13],  
164     304: [15, 13, 15],  
165     306: [15, [11, stroke], 13],  
166     308: [15, 13, 15],  
167     310: [15, [11, stroke], 13],  
168     312: [15, 13, 15],  
169     314: [15, [11, stroke], 13],  
170     316: [15, 13, 15],  
171     318: [15, [11, stroke], 13],  
172     320: [15, 13, 15],  
173     322: [15, [11, stroke], 13],  
174     324: [15, 13, 15],  
175     326: [15, [11, stroke], 13],  
176     328: [15, 13, 15],  
177     330: [15, [11, stroke], 13],  
178     332: [15, 13, 15],  
179     334: [15, [11, stroke], 13],  
180     336: [15, 13, 15],  
181     338: [15, [11, stroke], 13],  
182     340: [15, 13, 15],  
183     342: [15, [11, stroke], 13],  
184     344: [15, 13, 15],  
185     346: [15, [11, stroke], 13],  
186     348: [15, 13, 15],  
187     350: [15, [11, stroke], 13],  
188     352: [15, 13, 15],  
189     354: [15, [11, stroke], 13],  
190     356: [15, 13, 15],  
191     358: [15, [11, stroke], 13],  
192     360: [15, 13, 15],  
193     362: [15, [11, stroke], 13],  
194     364: [15, 13, 15],  
195     366: [15, [11, stroke], 13],  
196     368: [15, 13, 15],  
197     370: [15, [11, stroke], 13],  
198     372: [15, 13, 15],  
199     374: [15, [11, stroke], 13],  
200     376: [15, 13, 15],  
201     378: [15, [11, stroke], 13],  
202     380: [15, 13, 15],  
203     382: [15, [11, stroke], 13],  
204     384: [15, 13, 15],  
205     386: [15, [11, stroke], 13],  
206     388: [15, 13, 15],  
207     390: [15, [11, stroke], 13],  
208     392: [15, 13, 15],  
209     394: [15, [11, stroke], 13],  
210     396: [15, 13, 15],  
211     398: [15, [11, stroke], 13],  
212     400: [15, 13, 15],  
213     402: [15, [11, stroke], 13],  
214     404: [15, 13, 15],  
215     406: [15, [11, stroke], 13],  
216     408: [15, 13, 15],  
217     410: [15, [11, stroke], 13],  
218     412: [15, 13, 15],  
219     414: [15, [11, stroke], 13],  
220     416: [15, 13, 15],  
221     418: [15, [11, stroke], 13],  
222     420: [15, 13, 15],  
223     422: [15, [11, stroke], 13],  
224     424: [15, 13, 15],  
225     426: [15, [11, stroke], 13],  
226     428: [15, 13, 15],  
227     430: [15, [11, stroke], 13],  
228     432: [15, 13, 15],  
229     434: [15, [11, stroke], 13],  
230     436: [15, 13, 15],  
231     438: [15, [11, stroke], 13],  
232     440: [15, 13, 15],  
233     442: [15, [11, stroke], 13],  
234     444: [15, 13, 15],  
235     446: [15, [11, stroke], 13],  
236     448: [15, 13, 15],  
237     450: [15, [11, stroke], 13],  
238     452: [15, 13, 15],  
239     454: [15, [11, stroke], 13],  
240     456: [15, 13, 15],  
241     458: [15, [11, stroke], 13],  
242     460: [15, 13, 15],  
243     462: [15, [11, stroke], 13],  
244     464: [15, 13, 15],  
245     466: [15, [11, stroke], 13],  
246     468: [15, 13, 15],  
247     470: [15, [11, stroke], 13],  
248     472: [15, 13, 15],  
249     474: [15, [11, stroke], 13],  
250     476: [15, 13, 15],  
251     478: [15, [11, stroke], 13],  
252     480: [15, 13, 15],  
253     482: [15, [11, stroke], 13],  
254     484: [15, 13, 15],  
255     486: [15, [11, stroke], 13],  
256     488: [15, 13, 15],  
257     490: [15, [11, stroke], 13],  
258     492: [15, 13, 15],  
259     494: [15, [11, stroke], 13],  
260     496: [15, 13, 15],  
261     498: [15, [11, stroke], 13],  
262     500: [15, 13, 15],  
263     502: [15, [11, stroke], 13],  
264     504: [15, 13, 15],  
265     506: [15, [11, stroke], 13],  
266     508: [15, 13, 15],  
267     510: [15, [11, stroke], 13],  
268     512: [15, 13, 15],  
269     514: [15, [11, stroke], 13],  
270     516: [15, 13, 15],  
271     518: [15, [11, stroke], 13],  
272     520: [15, 13, 15],  
273     522: [15, [11, stroke], 13],  
274     524: [15, 13, 15],  
275     526: [15, [11, stroke], 13],  
276     528: [15, 13, 15],  
277     530: [15, [11, stroke], 13],  
278     532: [15, 13, 15],  
279     534: [15, [11, stroke], 13],  
280     536: [15, 13, 15],  
281     538: [15, [11, stroke], 13],  
282     540: [15, 13, 15],  
283     542: [15, [11, stroke], 13],  
284     544: [15, 13, 15],  
285     546: [15, [11, stroke], 13],  
286     548: [15, 13, 15],  
287     550: [15, [11, stroke], 13],  
288     552: [15, 13, 15],  
289     554: [15, [11, stroke], 13],  
290     556: [15, 13, 15],  
291     558: [15, [11, stroke], 13],  
292     560: [15, 13, 15],  
293     562: [15, [11, stroke], 13],  
294     564: [15, 13, 15],  
295     566: [15, [11, stroke], 13],  
296     568: [15, 13, 15],  
297     570: [15, [11, stroke], 13],  
298     572: [15, 13, 15],  
299     574: [15, [11, stroke], 13],  
300     576: [15, 13, 15],  
301     578: [15, [11, stroke], 13],  
302     580: [15, 13, 15],  
303     582: [15, [11, stroke], 13],  
304     584: [15, 13, 15],  
305     586: [15, [11, stroke], 13],  
306     588: [15, 13, 15],  
307     590: [15, [11, stroke], 13],  
308     592: [15, 13, 15],  
309     594: [15, [11, stroke], 13],  
310     596: [15, 13, 15],  
311     598: [15, [11, stroke], 13],  
312     600: [15, 13, 15],  
313     602: [15, [11, stroke], 13],  
314     604: [15, 13, 15],  
315     606: [15, [11, stroke], 13],  
316     608: [15, 13, 15],  
317     610: [15, [11, stroke], 13],  
318     612: [15, 13, 15],  
319     614: [15, [11, stroke], 13],  
320     616: [15, 13, 15],  
321     618: [15, [11, stroke], 13],  
322     620: [15, 13, 15],  
323     622: [15, [11, stroke], 13],  
324     624: [15, 13, 15],  
325     626: [15, [11, stroke], 13],  
326     628: [15, 13, 15],  
327     630: [15, [11, stroke], 13],  
328     632: [15, 13, 15],  
329     634: [15, [11, stroke], 13],  
330     636: [15, 13, 15],  
331     638: [15, [11, stroke], 13],  
332     640: [15, 13, 15],  
333     642: [15, [11, stroke], 13],  
334     644: [15, 13, 15],  
335     646: [15, [11, stroke], 13],  
336     648: [15, 13, 15],  
337     650: [15, [11, stroke], 13],  
338     652: [15, 13, 15],  
339     654: [15, [11, stroke], 13],  
340     656: [15, 13, 15],  
341     658: [15, [11, stroke], 13],  
342     660: [15, 13, 15],  
343     662: [15, [11, stroke], 13],  
344     664: [15, 13, 15],  
345     666: [15, [11, stroke], 13],  
346     668: [15, 13, 15],  
347     670: [15, [11, stroke], 13],  
348     672: [15, 13, 15],  
349     674: [15, [11, stroke], 13],  
350     676: [15, 13, 15],  
351     678: [15, [11, stroke], 13],  
352     680: [15, 13, 15],  
353     682: [15, [11, stroke], 13],  
354     684: [15, 13, 15],  
355     686: [15, [11, stroke], 13],  
356     688: [15, 13, 15],  
357     690: [15, [11, stroke], 13],  
358     692: [15, 13, 15],  
359     694: [15, [11, stroke], 13],  
360     696: [15, 13, 15],  
361     698: [15, [11, stroke], 13],  
362     700: [15, 13, 15],  
363     702: [15, [11, stroke], 13],  
364     704: [15, 13, 15],  
365     706: [15, [11, stroke], 13],  
366     708: [15, 13, 15],  
367     710: [15, [11, stroke], 13],  
368     712: [15, 13, 15],  
369     714: [15, [11, stroke], 13],  
370     716: [15, 13, 15],  
371     718: [15, [11, stroke], 13],  
372     720: [15, 13, 15],  
373     722: [15, [11, stroke], 13],  
374     724: [15, 13, 15],  
375     726: [15, [11, stroke], 13],  
376     728: [15, 13, 15],  
377     730: [15, [11, stroke], 13],  
378     732: [15, 13, 15],  
379     734: [15, [11, stroke], 13],  
380     736: [15, 13, 15],  
381     738: [15, [11, stroke], 13],  
382     740: [15, 13, 15],  
383     742: [15, [11, stroke], 13],  
384     744: [15, 13, 15],  
385     746: [15, [11, stroke], 13],  
386     748: [15, 13, 15],  
387     750: [15, [11, stroke], 13],  
388     752: [15, 13, 15],  
389     754: [15, [11, stroke], 13],  
390     756: [15, 13, 15],  
391     758: [15, [11, stroke], 13],  
392     760: [15, 13, 15],  
393     762: [15, [11, stroke], 13],  
394     764: [15, 13, 15],  
395     766: [15, [11, stroke], 13],  
396     768: [15, 13, 15],  
397     770: [15, [11, stroke], 13],  
398     772: [15, 13, 15],  
399     774: [15, [11, stroke], 13],  
400     776: [15, 13, 15],  
401     778: [15, [11, stroke], 13],  
402     780: [15, 13, 15],  
403     782: [15, [11, stroke], 13],  
404     784: [15, 13, 15],  
405     786: [15, [11, stroke], 13],  
406     788: [15, 13, 15],  
407     790: [15, [11, stroke], 13],  
408     792: [15, 13, 15],  
409     794: [15, [11, stroke], 13],  
410     796: [15, 13, 15],  
411     798: [15, [11, stroke], 13],  
412     800: [15, 13, 15],  
413     802: [15, [11, stroke], 13],  
414     804: [15, 13, 15],  
415     806: [15, [11, stroke], 13],  
416     808: [15, 13, 15],  
417     810: [15, [11, stroke], 13],  
418     812: [15, 13, 15],  
419     814: [15, [11, stroke], 13],  
420     816: [15, 13, 15],  
421     818: [15, [11, stroke], 13],  
422     820: [15, 13, 15],  
423     822: [15, [11, stroke], 13],  
424     824: [15, 13, 15],  
425     826: [15, [11, stroke], 13],  
426     828: [15, 13, 15],  
427     830: [15, [11, stroke], 13],  
428     832: [15, 13, 15],  
429     834: [15, [11, stroke], 13],  
430     836: [15, 13, 15],  
431     838: [15, [11, stroke], 13],  
432     840: [15, 13, 15],  
433     842: [15, [11, stroke], 13],  
434     844: [15, 13, 15],  
435     846: [15, [11, stroke], 13],  
436     848: [15, 13, 15],  
437     850: [15, [11, stroke], 13],  
438     852: [15, 13, 15],  
439     854: [15, [11, stroke], 13],  
440     856: [15, 13, 15],  
441     858: [15, [11, stroke], 13],  
442     860: [15, 13, 15],  
443     862: [15, [11, stroke], 13],  
444     864: [15, 13, 15],  
445     866: [15, [11, stroke], 13],  
446     868: [15, 13, 15],  
447     870: [15, [11, stroke], 13],  
448     872: [15, 13, 15],  
449     874: [15, [11, stroke], 13],  
450     876: [15, 13, 15],  
451     878: [15, [11, stroke], 13],  
452     880: [15, 13, 15],  
453     882: [15, [11, stroke], 13],  
454     884: [15, 13, 15],  
455     886: [15, [11, stroke], 13],  
456     888: [15, 13, 15],  
457     890: [15, [11, stroke], 13],  
458     892: [15, 13, 15],  
459     894: [15, [11, stroke], 13],  
460     896: [15, 13, 15],  
461     898: [15, [11, stroke], 13],  
462     900: [15, 13, 15],  
463     902: [15, [11, stroke], 13],  
464     904: [15, 13, 15],  
465     906: [15, [11, stroke], 13],  
466     908: [15, 13, 15],  
467     910: [15, [11, stroke], 13],  
468     912: [15, 13, 15],  
469     914: [15, [11, stroke], 13],  
470     916: [15, 13, 15],  
471     918: [15, [11, stroke], 13],  
472     920: [15, 13, 15],  
473     922: [15, [11, stroke], 13],  
474     924: [15, 13, 15],  
475     926: [15, [11, stroke], 13],  
476     928: [15, 13, 15],  
477     930: [15, [11, stroke], 13],  
478     932: [15, 13, 15],  
479     934: [15, [11, stroke], 13],  
480     936: [15, 13, 15],  
481     938: [15, [11, stroke], 13],  
482     940: [15, 13, 15],  
483     942: [15, [11, stroke], 13],  
484     944: [15, 13, 15],  
485     946: [15, [11, stroke], 13],  
486     948: [15, 13, 15],  
487     950: [15
```

```

16         [7, stroke],
17         [9, stroke],
18         [15, stroke]
19     ],
20     12: [
21         [11, stroke], 13, 15
22     ],
23     14: [9, 13, 15],
24 };
25
26 let par1 = new Parallelepiped({
27     depth: sl(10, 20),
28     height: sl(10, 20),
29     width: sl(10, 20),
30 });
31
32 let par2 = new Parallelepiped({
33     depth: par1.depth,
34     height: sl(5, par1.height - 4),
35     width: slKrome(par1.width, 5, par1.width - 5),
36 });
37
38 let deltaWidth = par1.width - par2.width;
39 let diagonal = (0.25 * deltaWidth.pow(2) +
40 par2.height.pow(2)).sqrt();
41
42 let vertex3D =
43 par1.verticesOfFigure.concat(par2.verticesOfFigure.map((elem) =>
44 shiftCoordinate3D(elem, {
45     x: 0,
46     y: 0,
47     z: -0.5 * (par1.height - par2.height),
48 })));
49
50 let camera = {
51     x: 0,
52     y: 0,
53     z: 0,
54     scale: 1,
55
56     rotationX: -Math.PI / 2 + Math.PI / 14,
57     rotationY: 0,
58     rotationZ: Math.PI / sl(12, 14),
59 };
60
61 let point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D,
62 camera));
63
64 autoScale(vertex3D, camera, point2D, {
65     startX: -180,
66     finishX: 160,
67     startY: -160,

```

```

64     finishY: 160,
65     maxScale: 100,
66 });
67
68     point2D = vertex3D.map((coord3D) => project3DTo2D(coord3D, camera));
69     genAssert((point2D[4].x - point2D[8].x).abs() > 20);
70     genAssert((point2D[4].y - point2D[13].y).abs() > 50);
71     genAssert((point2D[12].x - point2D[14].x).abs() > 40);
72     genAssert([point2D[0], point2D[3], point2D[8]].mt_is3ug(), 'Точки
лежат на одной прямой');
73
74     let rand = sl1();
75
76     let paint1 = function(ctx) {
77         let h = 400;
78         let w = 400;
79         ctx.translate(w / 2, h / 2);
80         ctx.lineWidth = 2;
81         ctx.strokeStyle = om.secondaryBrandColors;
82         ctx.drawFigureVer2(point2D, matrixConnections);
83
84         let point = [];
85         ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
86         if (point2D[6].x < point2D[15].x)
87             point = [point2D[10], point2D[15], point2D[6],
point2D[7]].mt_coordinatesOfIntersectionOfTwoSegments();
88         else
89             point = [point2D[10], point2D[15], point2D[6],
point2D[9]].mt_coordinatesOfIntersectionOfTwoSegments();
90         ctx.drawLine(point2D[15].x, point2D[15].y, point.x, point.y);
91
92         point = [point2D[12], point2D[13], point2D[4],
point2D[11]].mt_coordinatesOfIntersectionOfTwoSegments();
93         ctx.drawLine(point2D[4].x, point2D[4].y, point.x, point.y);
94
95         ctx.font = "20px liberation_sans";
96         ctx.signSegmentInMiddle(point2D[2].x, point2D[2].y, point2D[7].x,
point2D[7].y, par1.height, 10, 20);
97         ctx.signSegmentInMiddle(point2D[9].x, point2D[9].y, point2D[14].x,
point2D[14].y, par2.height, -24, 20);
98         ctx.signSegmentInMiddle(point2D[12].x, point2D[12].y,
point2D[15].x, point2D[15].y, par2.width, -10, 20);
99         ctx.signSegmentInMiddle(point2D[0].x, point2D[0].y, point2D[1].x,
point2D[1].y, par1.width, 18, 20);
100        ctx.signSegmentInMiddle(point2D[1].x, point2D[1].y, point2D[2].x,
point2D[2].y, par1.depth, 18, 20);
101    };
102
103    NATask.setTask({
104        text: 'Найдите ' + ['площадь поверхности', 'объём'][rand] + '
многогранника, изображённого на рисунке.',
105        answers: [par1.surfaceArea - deltaWidth * (par2.height +

```

```

106     par2.depth) + 2 * par2.height * par2.depth + 2 *
107         diagonal * par2.depth,
108     par1.volume - par2.height * (par1.width - par2.width) *
109     par2.width
110     ][rand],
111     author: ['Суматохина Александра']
112     });
113     NAtask.modifiers.multiplyAnswerBySqrt(13);
114     NAtask.modifiers.addCanvasIllustration({
115         width: 400,
116         height: 400,
117         paint: paint1,
118     });
119     },
120     1000);
121 })();
122 //144526144

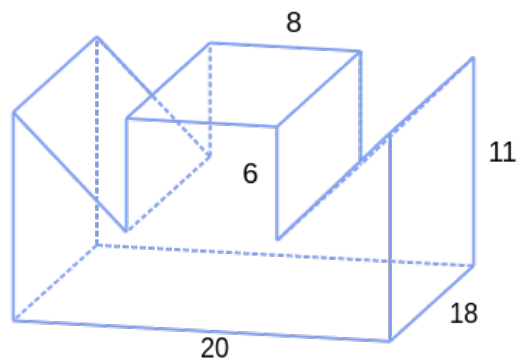
```

Листинг 11: 144526144.js

Примеры генерируемых задач 144526144.js

Найдите площадь поверхности многогранника, изображённого на рисунке.

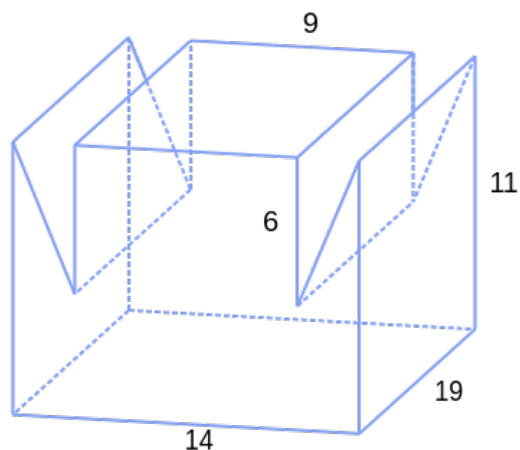
Ответ: 3384



Приложение. 25

Найдите площадь поверхности многогранника, изображённого на рисунке.

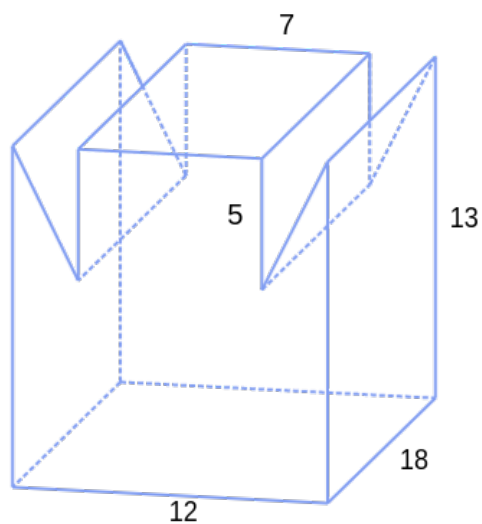
Ответ: 2656



Приложение. 26

Найдите площадь поверхности многогранника, изображённого на рисунке.

Ответ: 2633



Приложение. 27