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 математике 2024

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

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 10 и 11 класс 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 номер 2 по теме "вектора", количество которых, для прорешивания было очень мало. А по теме "Производная и первообразная" банк заданий 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 приводят 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 соответствуют условиям задачи, а иногда отвлекают 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 подготовиться 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 B»). [4] Планируется с течением времени включить в проект тесты по другим предметам школьной программы.

«Мини-интеграция» — это форма сотрудничества с образовательными интернет-ресурсами, при которой учебно-методический материал на странице ресурса дополняется виджетами тренажера с заданиями, соответствующими теме статьи, для возможности практического применения полученных знаний. В настоящее время достигнуто сотрудничество с двумя образовательными ресурсами: ege-ok.ru и matematikalegko.ru.

1. Глава первая

1.1.

2. Глава вторая

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. Возможности библиотеки во много раз превосходят требуемую функциональность для нашего проекта.

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

Стоит отметить, что задач по теме "Стереометрия" огромное множество. 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.

Каноническим положением будем называть такое расположение многогранника, когда его высота, проходящая через центр масс его основания, совпадает с осью аппликат и начало координат делится пополам.

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

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 области, иначе `false`.

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

Увеличивает свойство объекта `camera.scale`, пока все двухмерные координаты `vertex2D` вида `{x,y}` находится в некоторой 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. Работа с canvas

`CanvasRenderingContext2D.prototype.drawFigure = function(vertex, matrixC`

Соединяет линиями точки массива `vertex` с элементами `{x,y}` в соответствии с матрицей связей `matrixConnections`, которая является массивом,который может содержать в себе 0, 1 и массив step, указывающий на отрисовку пунктиром.

Пример матрицы связей:

```
let matrixConnections = [
    [1],
    [strok, strok],
    [0, 0, strok],
    [1, 0, 0, 1],
    [0, 1, 0, 1, 1]
  ];
```

`CanvasRenderingContext2D.prototype.drawFigureVer2 = function(vertex, mat`

Соединяет линиями точки массива `vertex` с элементами `{x,y}` в соответствии с матрицей связей `matrixConnections`, которая является объектом с числовыми полями (номерами вершин), которые содержат в себе массив с номерами вершин для связи с ними.

Пример матрицы связей:

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

2.4. Этапы разработки шаблоны с вспомогательным чертежом

- Создание объекта нужного класса (фигуры)
- Преобразование канонических координат на двумерную плоскость при помощи функции `project3DTo2D`
- Масштабирование координат функцией `autoScale`
- Корректирование матрицы связей (добавление диагоналей или сечений)
- Отрисовка фигуры `drawFigure`

Для примера возьмём задание 27074

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let paint1 = function (ctx) {
    };

    NAtask.setTask({
      text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем треугольной пирамиды
   ABCA_1.',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 100000);
})();
```

1. Создадим объект класса `Parallelepiped`.

```
(function () {
  retryWhileError(function () {
    NAinfo.requireApiVersion(0, 2);

    let par = new Parallelepiped({
      depth: sl(10, 50),
      height: sl(10, 50),
      width: sl(10, 50),
    });

    let paint1 = function (ctx) {
    };

    NAtask.setTask({
      text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем треугольной
   пирамиды ABCA_1.',
      answers: 0,
      author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 100000);
})();
```

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

```
(function () {
    retryWhileError(function () {
        NAinfo.requireApiVersion(0, 2);

        let par = new Parallelepiped({
            depth: sl(10, 50),
            height: sl(10, 50),
            width: sl(10, 50),
        });

        let camera = {
            x: 0,
            y: 0,
            z: 0,
            scale: 5,

            rotationX: -Math.PI / 2 + Math.PI / 14,
            rotationY: 0,
            rotationZ: 2 * Math.PI / 3,
        };

        let point2DPar = par.verticesOfFigure.map((coord3D) =>
   project3DTo2D(coord3D, camera));

        autoScale(par.verticesOfFigure, camera, point2DPar, {
            startX: -180,
            finishX: 160,
            startY: -160,
            finishY: 160,
            maxScale: 50,
        });

        point2DPar = par.verticesOfFigure.map((coord3D) => project3DTo2D(coord3D,
   camera));

        let paint1 = function (ctx) {
        };

        NAtask.setTask({
            text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем
   треугольной пирамиды ABCA_1.',
            answers: 0,
            author: ['Суматохина Александра']
        });
        NAtask.modifiers.addCanvasIllustration({
            width: 400,
            height: 400,
            paint: paint1,
        });
    }, 100000);
})();
```

3. Перемещаемся в середину спрайта. Отрисовываем фигуру функцией `drawFigure`, отдав в неё матрицу связей для параллелепипеда.

```
(function () {
    retryWhileError(function () {
        NAinfo.requireApiVersion(0, 2);
```

```
    let par = new Parallelepiped({
        depth: sl(10, 50),
        height: sl(10, 50),
        width: sl(10, 50),
    });

    let camera = {
        x: 0,
        y: 0,
        z: 0,
        scale: 5,

        rotationX: -Math.PI / 2 + Math.PI / 14,
        rotationY: 0,
        rotationZ: 2 * Math.PI / 3,
    };

    let point2DPar = par.verticesOfFigure.map((coord3D) =>
project3DTo2D(coord3D, camera));

    autoScale(par.verticesOfFigure, camera, point2DPar, {
        startX: -180,
        finishX: 160,
        startY: -160,
        finishY: 160,
        maxScale: 50,
    });

    point2DPar = par.verticesOfFigure.map((coord3D) => project3DTo2D(coord3D,
camera));

    let paint1 = function (ctx) {
        let h = 400;
        let w = 400;
        ctx.translate(h / 2, w / 2);
        ctx.lineWidth = 2;
        ctx.strokeStyle = om.secondaryBrandColors;
        let strok = [5, 4];
        ctx.drawFigure(point2DPar, [
            [strok],
            [0, 1],
            [strok, 0, 1],
            [0, 0, 0, 1],
            [strok, 0, 0, 0, 1],
            [0, 1, 0, 0, 0, 1],
            [0, 0, 1, 0, 1, 0, 1],
        ]);
    };

    NAtask.setTask({
        text: 'Объем параллелепипеда ABCDA_1B_1C_1D_1 равен 9. Найдите объем
треугольной пирамиды ABCA_1.',
        answers: 0,
        author: ['Суматохина Александра']
    });
    NAtask.modifiers.addCanvasIllustration({
        width: 400,
        height: 400,
        paint: paint1,
    });
```

```
    }, 100000);
})();
```

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.assertSaneDecimals` (исключает нецелый ответ) и `NAtask.modifiers.variativeABC(letter)` (заменяет все буквы в задании на случайные).

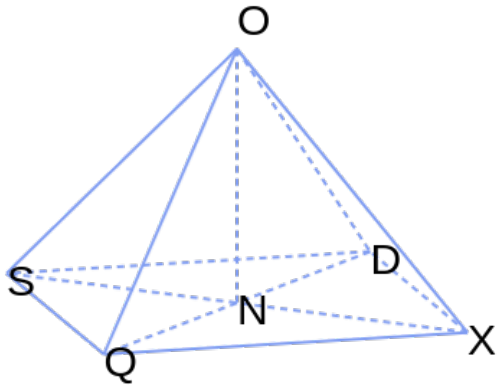
```
(function() {
  retryWhileError(function() {
    NAinfo.requireApiVersion(0, 2);

    let par = new Parallelepiped({
      depth: sl(10, 50),
      height: sl(10, 50),
      width: sl(10, 50),
    });

    let pyr = new Pyramid({
      height: par.height,
      baseArea: 0.5 * par.baseArea,
    });

    let camera = {
      x: 0,
      y: 0,
      z: 0,
      scale: 5,

      rotationX: -Math.PI / 2 + Math.PI / 14,
      rotationY: 0,
      rotationZ: 2 * Math.PI / 3,
    };

    let point2DPar = par.verticesOfFigure.map((coord3D) => project3DTo2D(coord3D,
   camera));

    autoScale(par.verticesOfFigure, camera, point2DPar, {
      startX: -180,
      finishX: 160,
      startY: -160,
      finishY: 160,
      maxScale: 50,
    });

    point2DPar = par.verticesOfFigure.map((coord3D) => project3DTo2D(coord3D,
   camera));

    let letter = ['A', 'B', 'C', 'D', '⊠D', '⊠A', '⊠B', '⊠C',];

    let paint1 = function(ctx) {
      let h = 400;
      let w = 400;
      ctx.translate(h / 2, w / 2);
      ctx.lineWidth = 2;
      ctx.strokeStyle = om.secondaryBrandColors;
      let strok = [5, 4];
```

```
      ctx.drawFigure(point2DPar, [
        [strok],
        [0, 1],
        [strok, 0, 1],
        [0, 0, 0, 1],
        [strok, 0, 0, 0, 1],
        [0, 1, 0, 0, 0, 1],
        [0, 0, 1, 0, 1, 0, 1],
      ]);

      ctx.font = "25px liberation_sans";
            point2DPar.forEach((elem, i) => ctx.fillText(letter[i], elem.x, elem.y
   + ((i < point2DPar.length / 2) ? 15 : -10)));
    };

    NAtask.setTask({
      text: 'Объем параллелепипеда $ABCDA_1B_1C_1D_1$ равен $'+par.volume+'$. Найдите
   объем треугольной пирамиды $ABCA_1$.',
      answers: par.volume/6,
      author: ['Суматохина Александра']
    });

    NAtask.modifiers.assertSaneDecimals();
    NAtask.modifiers.variativeABC(letter);

    NAtask.modifiers.addCanvasIllustration({
      width: 400,
      height: 400,
      paint: paint1,
    });
  }, 100000);
})();
```

2.5. Шаблоны по теме Стереометрия

Задание №3011

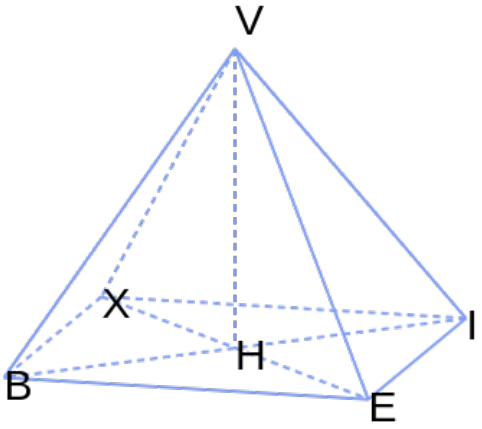
В правильной четырёхугольной пирамиде $OQSDX$ с основанием $QSDX$ боковое ребро равно $\sqrt{1489{,}5}$, сторона основания равна 39. Найдите объём пирамиды.



Ответ: 13689

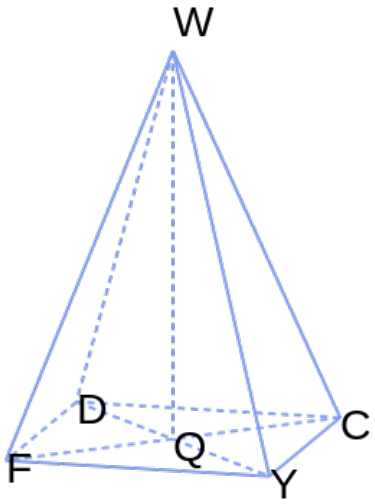
В правильной четырёхугольной пирамиде $VEBXI$ с основанием $EBXI$ боковое ребро равно $\sqrt{848{,}5}$, сторона основания равна 27. Найдите объём пирамиды.

Ответ: 5346



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

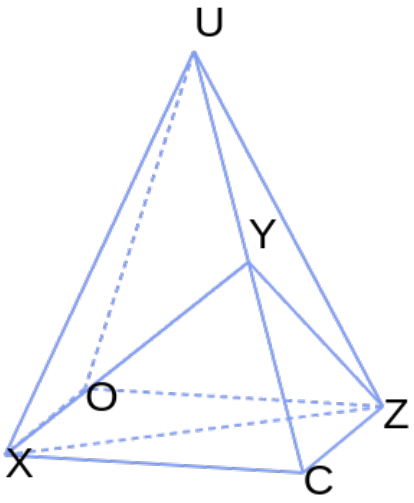
Ответ: 6720



Задание №27114

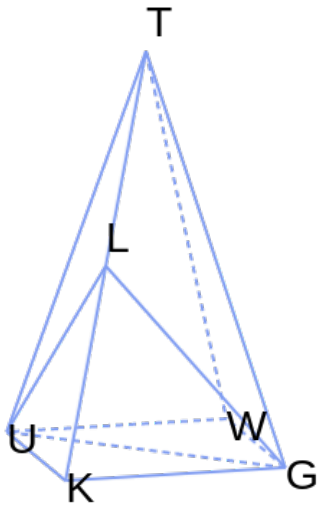
Объём правильной четырёхугольной пирамиды $UCXOZ$ равен 31164. Точка Y – середина ребра UC. Найдите объём треугольной пирамиды $YCXZ$.

Ответ: 7791



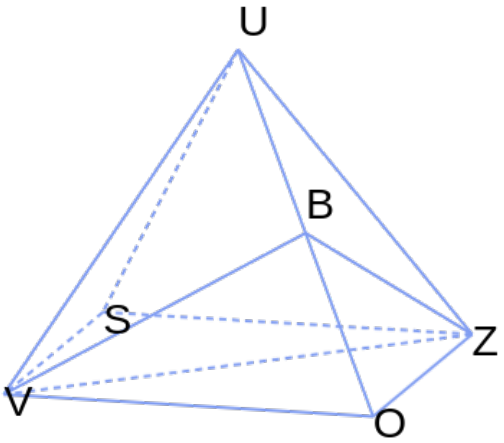
Объём правильной четырёхугольной пирамиды $TKUWG$ равен 4800. Точка L – середина ребра TK. Найдите объём треугольной пирамиды $LKUG$.

Ответ: 1200



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

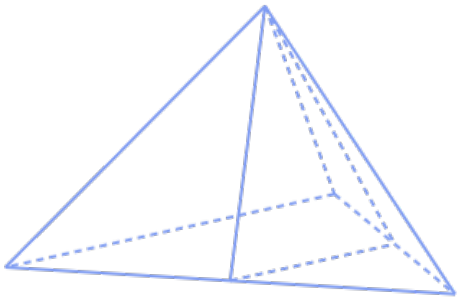
Ответ: 6412,5



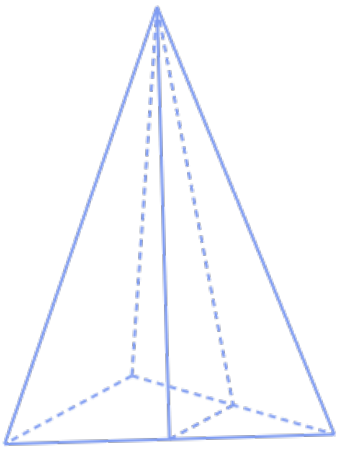
Задание №27115

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

Ответ: 640

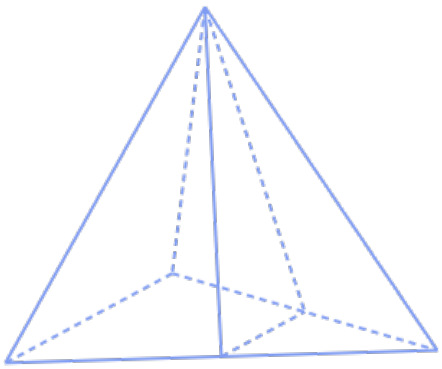


Объём треугольной пирамиды равен 4950. Через вершину пирамиды и среднюю линию её основания проведена плоскость (см. рисунок). Найдите объём отсечённой треугольной пирамиды.



Ответ: 1237,5

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



Ответ: 3024

Заключение

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] Пошаговая инструкция по созданию элементарных шаблонов. – URL: https://math.vsu.ru/chas-ege/doc/shabl-b1-po-shagam.html

[4] Федеральный институт педагогических измерений. – URL: https://fipi.ru/ege/otkrytyy-bank-zadaniy-ege

[5] Единый государственный экзамен. – URL: https://ru.wikipedia.org/wiki/Единый_государственный_экзамен