Министерство просвещения Российской Федерации

Муниципальное автономное общеобразовательное учреждение
средняя общеобразовательная школа № 2
имени Ивана Михайловича Суворова станицы Павловской
Краснодарского края

VIII Международный конкурс исследовательских работ школьников

Исследовательский проект

Программирование алгоритма построения сечения многогранников

Выполнил: Михайлевский Артём Александрович

Ученик 10 Б класса

Руководитель: Черемискина Людмила Павловна

Учитель математики и информатики

2025-2026 г.

Содержание:

МИХАЙЛЕВСКИЙ Артём Александрович
Краснодарский край, Павловский район, станица Павловская
Муниципальное автономное общеобразовательное учреждение
средняя общеобразовательная школа № 2 имени Ивана Михайловича Суворова ст. Павловской, 10 класс

ПРОГРАММИРОВАНИЕ АЛГОРИТМА ПОСТРОЕНИЯ СЕЧЕНИЯ МНОГОГРАННИКОВ

Научный руководитель: Черемискина Людмила Павловна, учитель математики и информатики МАОУ СОШ № 2 им. И.М. Суворова, Павловский район, станица Павловская

Введение

Актуальность**.** В современном школьном курсе стереометрии изучение пространственных фигур и построение их сечений традиционно вызывает значительные трудности у учащихся. Согласно статистике Всероссийского центра изучения общественного мнения (2025), более половины учащихся 10–11 классов испытывают когнитивные затруднения при переходе от плоскостных чертежей к трёхмерным образам. Ученикам сложно представить взаимное расположение точек, прямых и плоскостей в трёхмерном пространстве, оперируя лишь статичными изображениями в учебнике или на доске.

Проблема исследования**.** Как программно реализовать универсальный, математически корректный и производительный алгоритм построения плоскости сечения по трём произвольным точкам для различных типов трёхмерных объектов (призмы, пирамиды, тела вращения) в интерактивном веб-приложении, работающем в браузере без установки дополнительного ПО?

Цель – разработать веб-сайт с использованием библиотеки Three.js, позволяющий визуализировать многогранники и тела вращения, а также строить их сечения по трём произвольно заданным пользователем точкам с высокой точностью позиционирования и наглядностью отображения.

Объект исследования – процесс трёхмерного моделирования сечений геометрических тел в виртуальном пространстве, реализованный средствами Three.js.

Предмет исследования – алгоритмы и методы библиотеки Three.js для интерактивного определения координат точек на поверхности фигуры и последующего построения секущей плоскости.

Гипотеза**.** Возможно создать универсальную программную среду, в которой построение плоскости сечения для любого типа фигур (включая тела вращения, не имеющие рёбер и

плоских граней) будет выполняться по единому принципу: пользователь задаёт три точки с помощью кликов мыши, после чего программно вычисляется и отображается секущая плоскость, ограниченная габаритами фигуры для улучшения визуального восприятия.

Задачи. Изучить HTML, CSS, Javascript и Thee,js; разработать пользовательский интерфейс сайта; запрограммировать алгоритм построения сечений фигур; сделать выводы об проделанной работе.

Методологическая основа**.** При разработке программная реализация базируется на открытой библиотеке Three.js (автор Ricardo Cabello) и официальной документации к ней.

В процессе исследования применялись методы сравнительного анализа, математического моделирования, объектно-ориентированного программирования, модульного тестирования.

Практическая значимость**.** Разработанный веб-сайт может быть непосредственно использован: учителями математики на уроках стереометрии в 10–11 классах при изучении темы «Построение сечений многогранников»; учащимися для самостоятельной подготовки и самопроверки;

1. Теоретическая часть

1.1. История метода сечений и его цифровая трансформация

Метод сечений (или метод следов) является классическим инструментом начертательной геометрии и стереометрии. Суть метода заключается в построении линии пересечения заданной плоскости с поверхностью тела. В традиционном школьном курсе сечение строится последовательно: находятся точки пересечения плоскости с рёбрами многогранника, затем эти точки соединяются отрезками.

С развитием компьютерной техники во второй половине XX века возникла задача автоматизации этого процесса. Первые системы автоматизированного проектирования (CAD), такие как Sketchpad (1963) и позднее AutoCAD (1982), уже содержали инструменты для построения сечений. Переломным моментом стало появление твердотельного моделирования в конце 1980-х годов. Системы, основанные на конструктивной блочной геометрии (CSG), позволили оперировать трёхмерными примитивами и выполнять над ними булевы операции, включая вычитание и пересечение. Сечение в таких системах стало частным случаем булевой операции пересечения тела с бесконечной полуплоскостью.

В начале XXI века с ростом производительности GPU появилась возможность выполнять подобные вычисления в реальном времени. Технологии OpenGL и WebGL предоставили механизм плоскостей отсечения (clipping planes), позволяющих «срезать» геометрию аппаратно. Однако этот метод не даёт в явном виде многоугольник сечения, что ограничивает его применение в образовательных задачах. Параллельно развивались векторные методы вычисления пересечения луча с поверхностью, широко применяемые в трассировке лучей. Именно этот подход был выбран в данной работе для реализации позиционирования точек на цилиндре и конусе, поскольку он обеспечивает субпиксельную точность и полный контроль над процессом.

1.2. Классификация трёхмерных фигур

Для разработки корректного и расширяемого программного обеспечения необходимо чётко классифицировать типы фигур. В школьном курсе стереометрии можно выделить две крупные категории:

1. Многогранники – тела, поверхность которых состоит из конечного числа плоских многоугольников:
   - призмы (треугольная, шестиугольная, прямоугольный параллелепипед/куб);
   - пирамиды (четырёхугольная, шестиугольная);
   - правильные многогранники (тетраэдр).
2. Тела вращения – тела, полученные вращением плоской фигуры вокруг оси:

- цилиндр (вращение прямоугольника);
- конус (вращение прямоугольного треугольника).

С точки зрения компьютерной графики, все эти фигуры могут быть представлены двумя основными способами [1, с. 78]:

А. Полигональные сетки (Polygon Meshes). Фигура описывается множеством вершин и треугольников (или четырёхугольников), которые аппроксимируют её поверхность. Этот способ универсален и поддерживается всеми графическими API. Именно он используется для куба, призм, пирамид и тетраэдра в данном проекте. Преимущества: простота рендеринга, возможность использования стандартного рейкастинга. Недостатки: неточная передача криволинейных поверхностей без большого количества полигонов.

Б. Параметрические поверхности. Фигура описывается математическим уравнением. В чистом виде такой подход редко используется для рендеринга, но идеально подходит для точных пересечений с лучом.

В разрабатываемом приложении используется гибридный подход: для визуализации цилиндра и конуса применяются встроенные примитивы Three.js, а для определения координат точки клика используется собственный математический решатель, вычисляющий пересечение с идеальной математической поверхностью.

2. Практическая часть

2.1. Обоснование выбора инструментов и архитектуры приложения

Для реализации проекта была выбрана библиотека Three.js — наиболее популярная и документированная надстройка над WebGL. [2] Она позволяет абстрагироваться от низкоуровневых вызовов графического API и быстро создавать сцены с освещением, тенями и камерой.

Архитектура приложения модульная: под каждую фигуру создана отдельная HTML-страница и соответствующий JavaScript-файл (main.js). Это сделано для упрощения навигации и независимости компонентов. Визуально страницы объединены общим стилем (фон #0f0f1f, синие тона фигур, красные маркеры). Основные модули включают:

- инициализацию сцены, камеры, освещения и элементов управления (OrbitControls);
- функцию создания конкретной фигуры;
- массив для хранения сфер-маркеров;
- функции createSphere, removeSphere;
- алгоритм raycaster для обработки кликов;
- функцию createPlaneThroughSpheres.

В главном меню сайта (index.html) реализован каталог фигур с превью-изображениями, что обеспечивает удобную навигацию.

2.2. Процесс программирования сечений для различных классов фигур

Общий принцип**.** На сцене присутствует одна фигура. Пользователь кликает левой кнопкой мыши по её поверхности. Поскольку фигура является мешем (Mesh), Three.js позволяет получить точку пересечения луча из камеры с геометрией объекта (Raycaster.intersectObject). В этой точке создаётся сфера красного цвета с подсветкой. Всего можно создать не более трёх сфер.

Особенности реализации для многогранников (куб, призмы, пирамиды, тетраэдр). Для этих фигур используется стандартный механизм рейкастинга по мешу. У куба (BoxGeometry) и тетраэдра (собран вручную из вершин) точно определяются координаты пересечения с гранью. При создании сферы её позиция копирует точку пересечения (intersect.point). Для предотвращения случайных кликов по рёбрам (которые тоже являются объектами LineSegments) в коде добавлена проверка userData.type. Рёбра помечены как edge и игнорируются.

Особенности реализации для тел вращения (цилиндр, конус). Стандартный рейкастинг Three.js для CylinderGeometry и ConeGeometry корректно работает, однако было принято решение реализовать собственный математический алгоритм пересечения луча с

поверхностью. Это сделано для демонстрации глубокого понимания геометрии. Функции getIntersectionPointWithCylinder и getIntersectionPointWithCone решают квадратное уравнение поверхности:

- Для цилиндра: $x^2 + z^2 = R^2$, ограничение по y от 0 до высоты.
- Для конуса: $x^2 + z^2 = (R/H)^2 * (H - y)^2$.

Данные функции также обрабатывают пересечения с основаниями (для цилиндра — круги сверху и снизу, для конуса — только основание). Это позволяет ставить точки строго на поверхности фигуры, включая торцы.

Построение плоскости**.** Как только массив spheres достигает длины 3, вызывается функция createPlaneThroughSpheres(). Она:

1. Удаляет предыдущую плоскость (если она была).
2. Вычисляет два вектора v1 = p2 - p1, v2 = p3 - p1.
3. Находит нормаль n = v1 × v2, нормализует её.
4. Создаёт геометрию PlaneGeometry размером 20x20.
5. Помещает центр плоскости в центр тяжести трёх точек.
6. Ориентирует плоскость по нормали с помощью lookAt().
7. Добавляет зелёный контур.

2.3. Аналитическая таблица

В ходе разработки были зафиксированы ключевые параметры для каждого типа фигур. Результаты представлены в таблице 1.

Сравнение методов реализации для разных типов фигур

Тип фигуры	Способ создания геометрии	Метод определения точки клика	Наличие артефактов
Куб / Параллелепипед	BoxGeometry	Raycaster (штатный)	Нет
Треугольная призма	BufferGeometry по вершинам	Raycaster (штатный)	Нет
Шестиугольная призма	BufferGeometry (14 вершин)	Raycaster (штатный)	Нет
Тетраэдр	BufferGeometry (4 вершины)	Raycaster (штатный)	Нет
4-уг. пирамида	BufferGeometry (1 вершина + квадрат)	Raycaster (штатный)	Нет

6-уг. пирамида	BufferGeometry (1 вершина + 6 углов)	Raycaster (штатный)	Нет
Цилиндр	CylinderGeometry	Математический решатель	Нет
Конус	ConeGeometry	Математический решатель	Нет

Как видно из таблицы 1, все 8 типов фигур успешно реализованы. Для многогранников использован стандартный рейкастинг, для тел вращения — математический решатель. Артефактов визуализации не обнаружено.

Заключение

В рамках данной исследовательской работы была достигнута поставленная цель: разработан веб-сайт для интерактивного построения сечений пространственных фигур. Продукт полностью функционирует.

В соответствии с задачами работы:

1. Изучена история вопроса: от метода следов в начертательной геометрии до современных алгоритмов компьютерной графики.

2. Рассмотрены три подхода к реализации сечений, выбран оптимальный для образовательного веб-проекта.

3. Спроектирована структура сайта, состоящая из главной страницы-каталога и страниц фигур.

4. Реализованы модели 8 геометрических тел с использованием различных техник Three.js.

5. Разработан механизм установки точек на основе Raycaster и собственных математических вычислений.

6. Реализована динамическая визуализация секущей плоскости, изменяющей своё положение и ориентацию по трём точкам.

Гипотеза, выдвинутая во введении, подтвердилась полностью: универсальный подход (плоскость как маркер) действительно позволил в едином стиле реализовать сечение и для куба, и для конуса, несмотря на разницу в их геометрических свойствах.

Ценность работы. Для автора работа стала возможностью применить знания по векторной алгебре и программированию на JavaScript для создания реально работающего продукта. Для образовательного сообщества сайт даёт бесплатный и наглядный инструмент для изучения стереометрии. В дальнейшем планируется добавить возможность автоматического построения точного многоугольника сечения для многогранников (вычисление вершин и соединение их линиями красного цвета), а также реализовать сечение сферы.

Литература

1. Dirksen, J. Learn Three.js: Program 3D animations and visualizations for the web with JavaScript. – Бирмингем: Packt Publishing, 2023. – 554.
2. Three.js Documentation / Official Three.js website. – URL: https://threejs.org/docs/ (дата обращения: 15.10.2025).

Приложение 1

Листинг функции построения плоскости по трём точкам (файл main.js для куба)

javascript

```
// Создание плоскости через три сферы
function createPlaneThroughSpheres() {
    if (spheres.length < 3) return;

    // Удаляем старую плоскость
    if (plane) {
        scene.remove(plane);
        plane.geometry.dispose();
        plane.material.dispose();
    }

    // Получаем позиции сфер
    const p1 = spheres[0].position;
    const p2 = spheres[1].position;
    const p3 = spheres[2].position;

    // Вычисляем нормаль плоскости
    const v1 = new THREE.Vector3().subVectors(p2, p1);
    const v2 = new THREE.Vector3().subVectors(p3, p1);
    const normal = new THREE.Vector3().crossVectors(v1, v2).normalize();

    // Создаем геометрию плоскости
    const planeGeometry = new THREE.PlaneGeometry(20, 20);
    // Центр плоскости
    const center = new THREE.Vector3()
        .add(p1)
        .add(p2)
        .add(p3)
        .multiplyScalar(1/3);

    // Создаем и настраиваем плоскость
    plane = new THREE.Mesh(planeGeometry, planeMaterial);
```

```
    plane.position.copy(center);
    plane.lookAt(center.clone().add(normal));

    // Добавляем контур
    const edges = new THREE.EdgesGeometry(planeGeometry);
    const lineMaterial = new THREE.LineBasicMaterial({ color: 0x27ae60 });
    const edgesMesh = new THREE.LineSegments(edges, lineMaterial);
    plane.add(edgesMesh);

    scene.add(plane);
}
```

Приложение 2

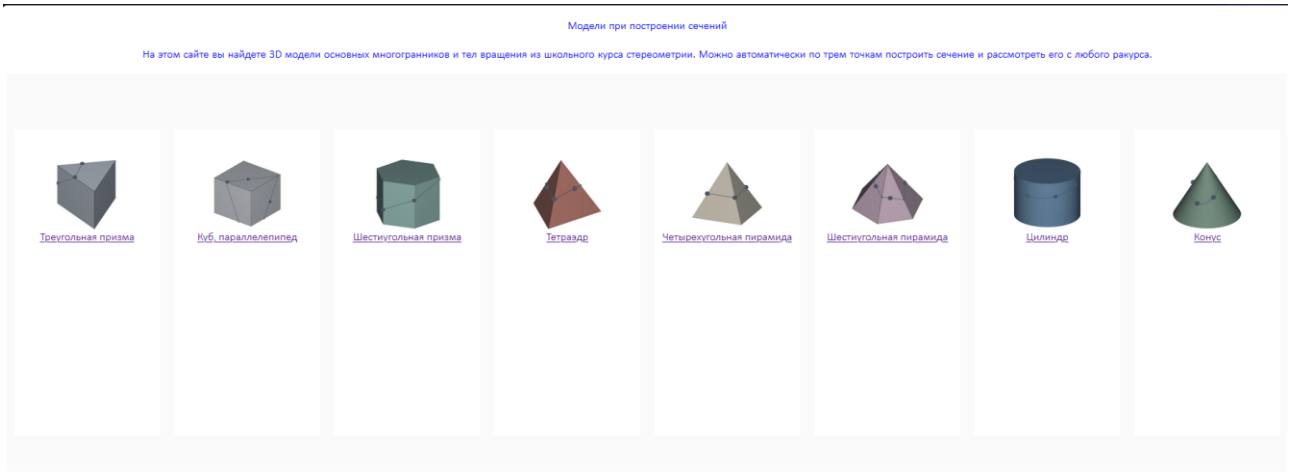
Листинг функции вычисления пересечения с конусом (файл main.js для конуса)

javascript

```
function getIntersectionPointWithCone(ray) {
    const height = 4;
    const radius = 2.5;
    const slope = radius / height;

    const ox = ray.origin.x, oy = ray.origin.y, oz = ray.origin.z;
    const dx = ray.direction.x, dy = ray.direction.y, dz = ray.direction.z;

    const A = dx*dx + dz*dz - slope*slope*dy*dy;
    const B = 2*(ox*dx + oz*dz + slope*slope*dy*(height - oy));
    const C = ox*ox + oz*oz - slope*slope*(height - oy)*(height - oy);

    const discriminant = B*B - 4*A*C;
    if (discriminant < 0) return null;

    const sqrtDisc = Math.sqrt(discriminant);
    const t1 = (-B + sqrtDisc) / (2*A);
    const t2 = (-B - sqrtDisc) / (2*A);

    let t = null;
    if (t1 > 0 && t2 > 0) t = Math.min(t1, t2);
    else if (t1 > 0) t = t1;
    else if (t2 > 0) t = t2;
    else return null;
    const point = new THREE.Vector3(
        ox + t * dx,
        oy + t * dy,
        oz + t * dz
    );
    if (point.y < 0 || point.y > height) return null;
    return point;
}
```

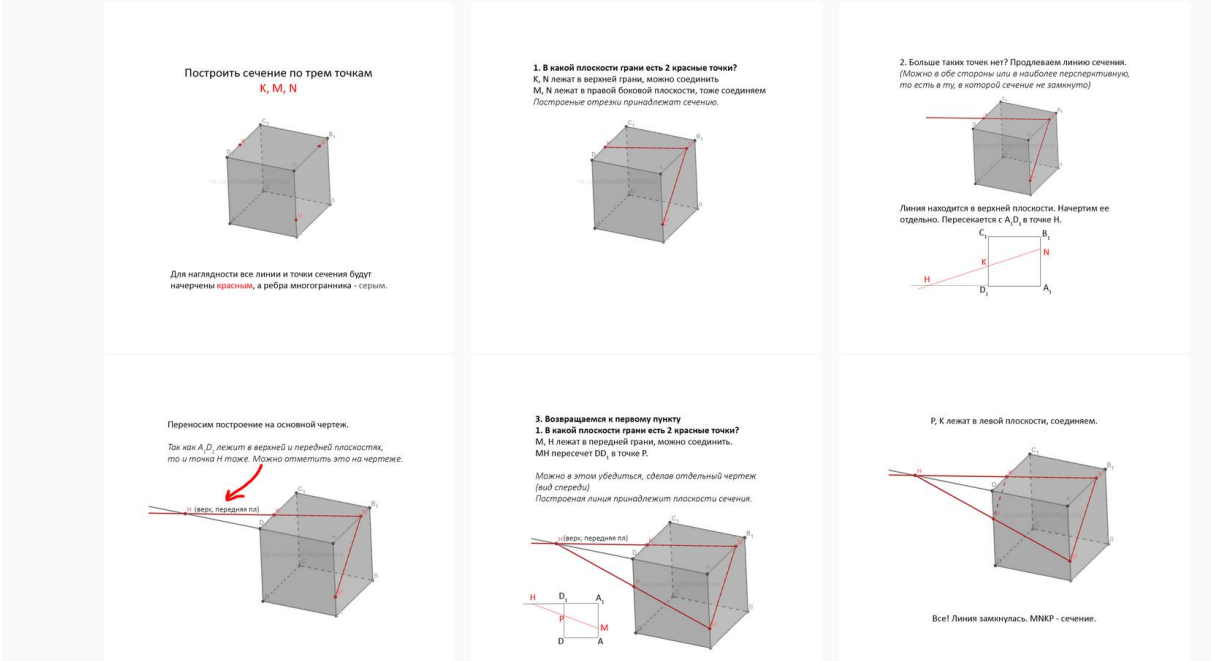
Приложение 3

Скриншот главной страницы сайта



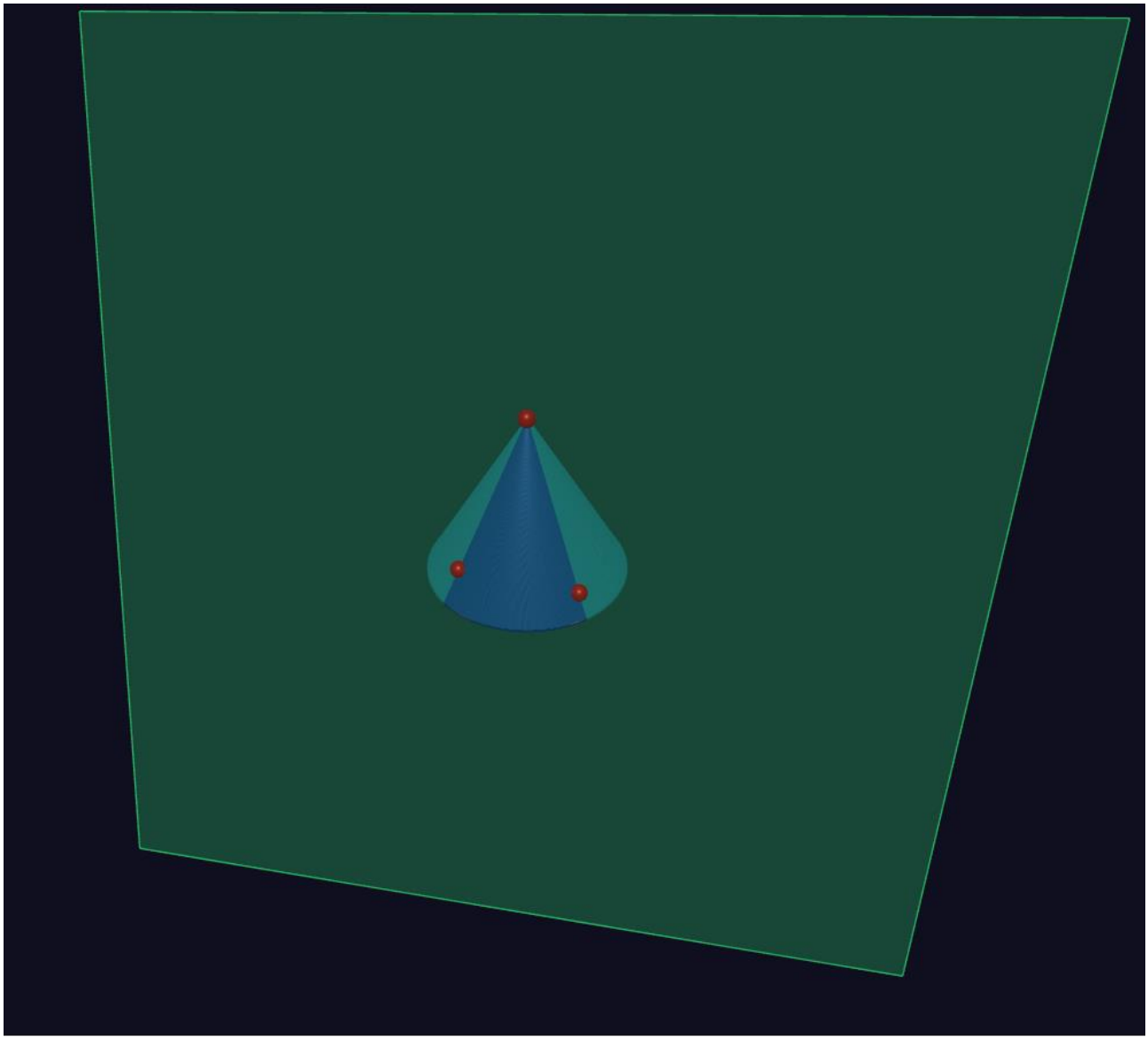
Приложение 4

Скриншот страницы сайта



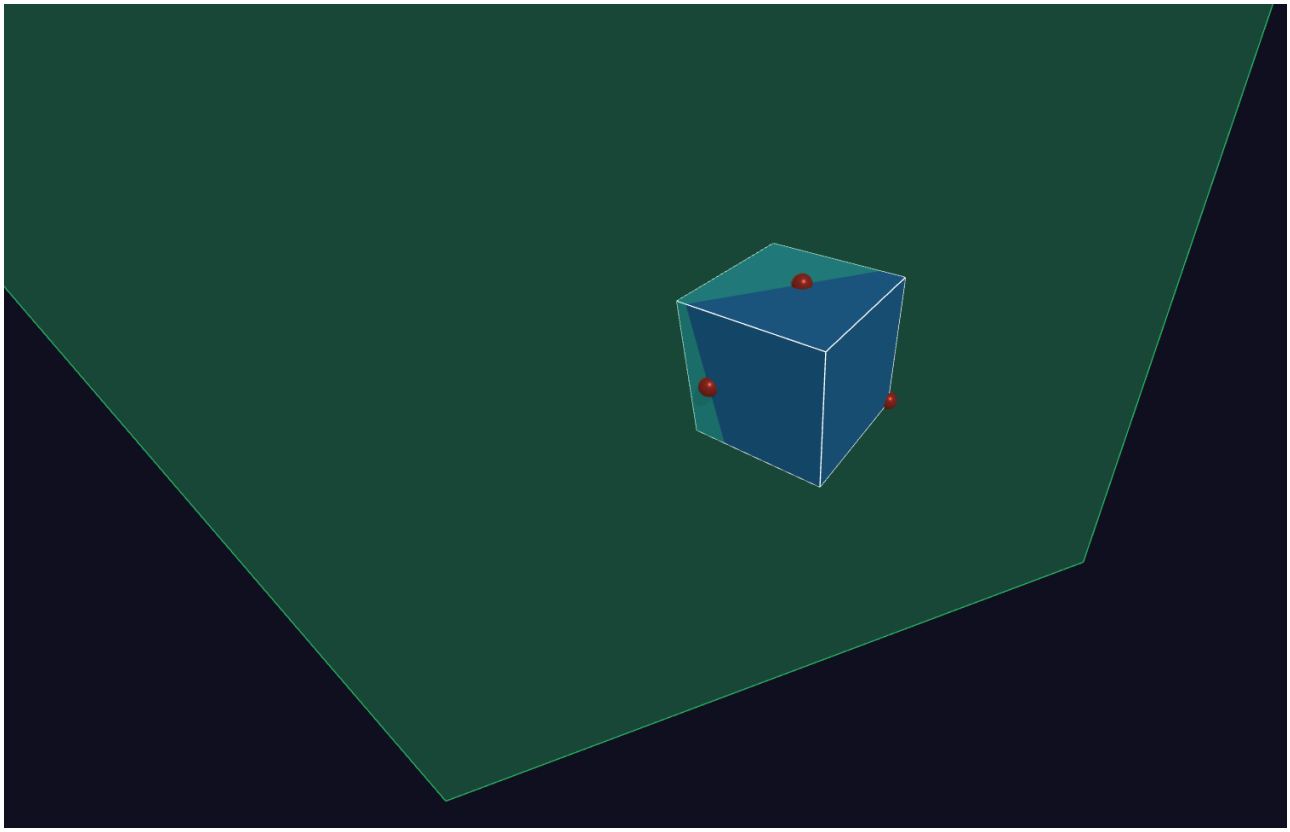
Приложение 5

Построение сечения конуса



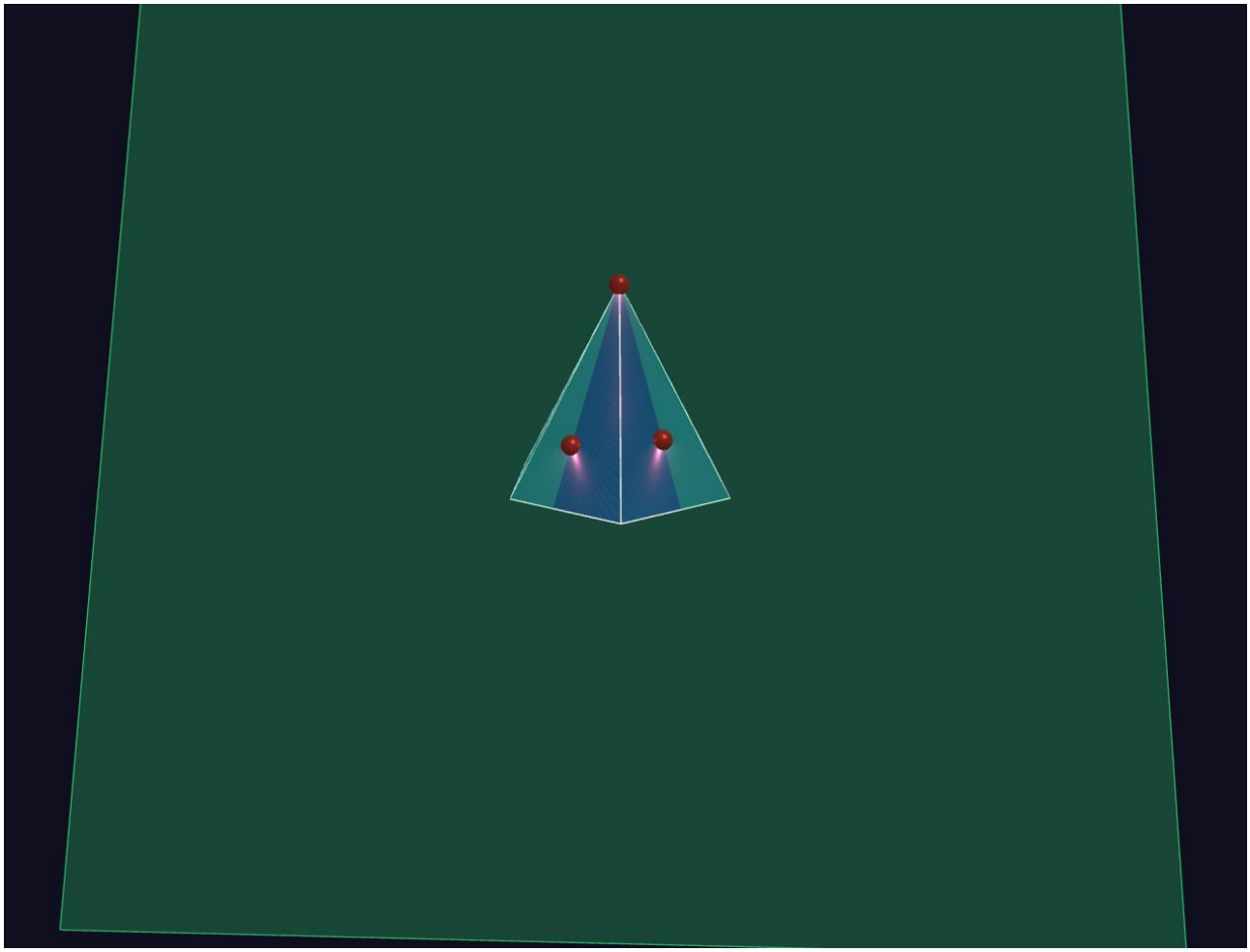
Приложение 6

Построение сечения куба



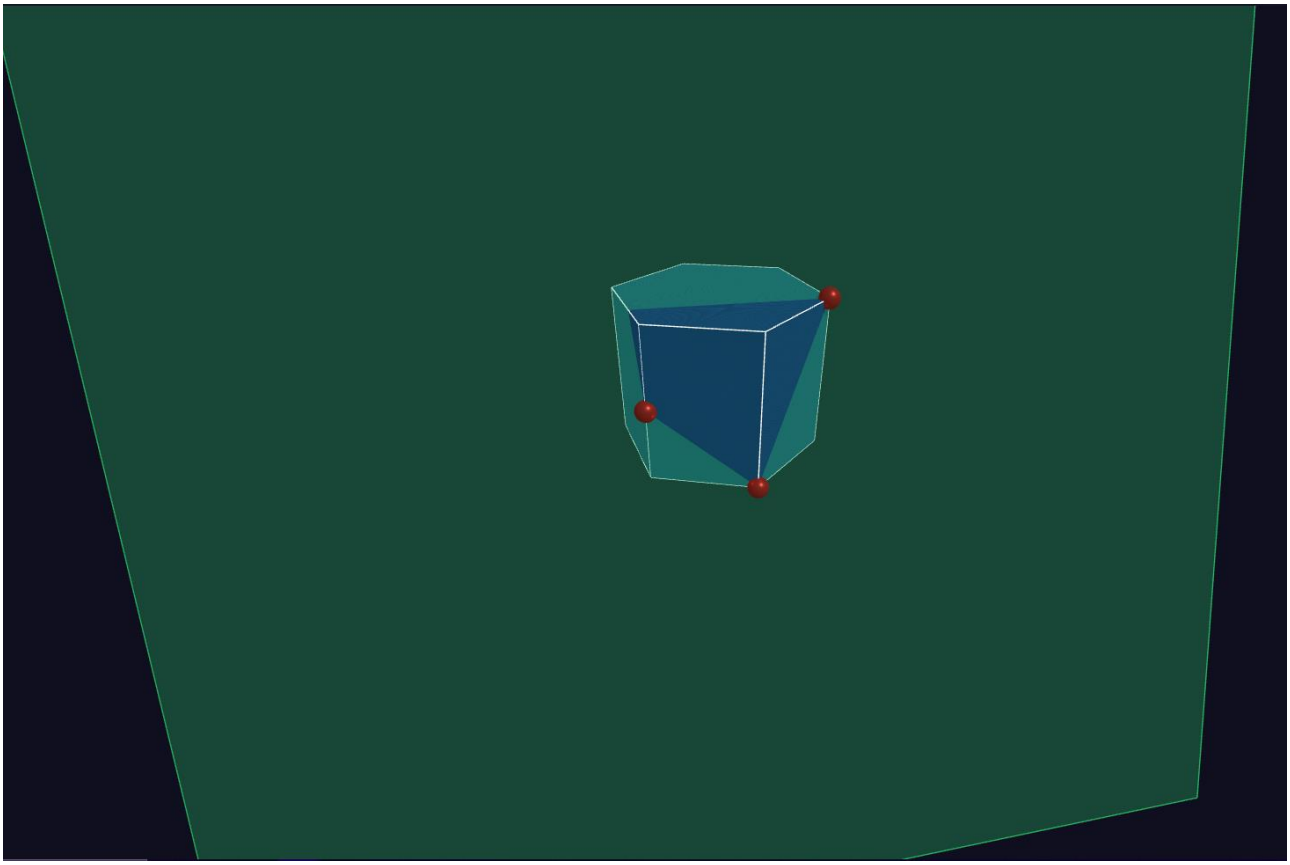
Приложение 7

Построение сечения тетраэдра



Приложение 8

Построение сечения шестиугольной призмы



Приложение 9

Построение сечения треугольной призмы

Приложение 10

Построение сечения цилиндра