МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Институт информатики, математики и робототехники

Отчёт к лабораторной работе по дисциплине «Веб-программирование»

Визуализация спутниковой группировки ГЛОНАСС на WebGL

Выполнили:
Студенты гр. ПРО-341Б
Репин Д.А.
Репин И.А.

Уфа — 2025

Содержание

1 Введение

1.1 Цель работы

Целью работы является разработка сайта с использованием JavaScript и WebGL для визуализации спутниковой группировки ГЛОНАСС, изучение графических возможностей современных фронтенд-технологий и базовых понятий космических путешествий

1.2 Задачи работы

Необходимо разработать веб-сайт с использованием технологии WebGL для визуализации в трехмерном пространстве планеты Земля и спутниковой группировки ГЛОНАСС, в уменьшенном масштабе, но с сохранением верных пропорций с допущением о круговой форме орбит. Для этого необходимо освоить базовые понятия об орбитах, произвести перевод кеплеровских координат в декартовы для отрисовки в WebGL

2 Теоретическая часть

2.1 Навигационная система ГЛОНАСС

ГЛОНАСС — это российская глобальная спутниковая навигационная система, предназначенная для определения координат и скорости объектов в любой точке Земли. Для обеспечения глобального покрытия используется 24 спутника, равномерно распределённые по трём орбитальным плоскостям.

Основными компонентами системы ГЛОНАСС являются космический сегмент, наземный комплекс управления, аппаратура потребителей

Орбитальные параметры и модель

Для описания положения спутников на орбите используют две системы координат:

- **Кеплеровские элементы орбиты** — параметры, которые полностью описывают движение тела в центральном гравитационном поле.
- **Декартовы координаты** — обычные координаты x, y, z в трехмерном пространстве.

В данной работе моделируется движение спутников ГЛОНАСС с рядом упрощающих допущений:

- Орбиты считаются круговыми (эксцентриситет $e = 0$).
- Используется система координат ПЗ-90 (Геоцентрическая, экваториальная).
- Положение спутника определяется с помощью трёх параметров:
    1. **Долгота восходящего узла** (Ω) — угол между направлением на точку весеннего равноденствия и направлением на точку пересечения орбиты с экватором.
    2. **Наклонение орбиты** (i) — угол между плоскостью орбиты и плоскостью экватора Земли. В системе ГЛОНАСС он составляет $64.8°$.

3. **Аргумент широты** (u) — угол между направлением на восходящий узел и текущим положением спутника на орбите. В круговой орбите $u = \nu$ (истинная аномалия).

Остальные элементы классической кеплеровской шестерки (эксцентриситет e, большая полуось a, аргумент перицентра ω) не влияют на модель из-за круговой орбиты и равномерного движения.

Геометрия расположения спутников

В системе ГЛОНАСС используется:

- Три орбитальные плоскости, равномерно смещённые по долготе восходящего узла на $120°$.
- В каждой плоскости по 8 спутников, равномерно распределённых по аргументу широты с шагом $45°$.
- Соседние спутники из разных орбитальных плоскостей сдвинуты друг относительно друга на $13°$ по аргументу широты.
- Движение спутников осуществляется против часовой стрелки в системе координат, связанной с центром Земли.

Преобразование в декартовые координаты

Положение спутника в орбитальной плоскости описывается:

$$\begin{aligned} x' &= r\cos u, \\ y' &= r\sin u, \\ z' &= 0, \end{aligned}$$

где $r = R_E + h$ — радиус круговой орбиты.

Для перехода к декартовой системе координат производится поворот орбитальной плоскости на угол Ω вокруг оси Z и наклон на угол i — с помощью стандартных формул или матриц поворота.

В результате получаем координаты спутника в трёхмерной системе:

$$
\begin{aligned}
x &= r\left(\cos\Omega\cos u - \sin\Omega\sin u\cos i\right), \\
y &= r\left(\sin\Omega\cos u + \cos\Omega\sin u\cos i\right), \\
z &= r\left(\sin u\sin i\right).
\end{aligned}
$$

В программе эти преобразования используются в функциях createOrbits() и computeSatellites(), где вычисляются координаты точек орбит и положений спутников соответственно.

2.2 Технология WebGL

Основные понятия

WebGL (Web Graphics Library) — это JavaScript API для рендеринга интерактивной 3D и 2D графики в веб-браузере без использования плагинов. WebGL использует элемент HTML5 <canvas> и предоставляет аппаратное ускорение для обработки физики и геометрии в графическом процессоре (GPU).

WebGL основан на OpenGL ES (OpenGL for Embedded Systems) и позволяет использовать возможности GPU прямо из JavaScript кода. Решение получается кроссплатформенным, эффективным и удобным за счет интеграции с веб-технологиями.

Рендеринг с помощью шейдеров

Одной из ключевых концепций WebGL является использование шейдеров — небольших программ, которые выполняются на GPU. Шейдеры написаны на языке GLSL (OpenGL Shading Language) и используются для определения того, как отображаются вершины и пиксели на экране.

1. **Вершинный шейдер** (Vertex Shader) — обрабатывает каждую вершину геометрических объектов. Он отвечает за преобразование координат вершин из исходного пространства в экранное пространство.
2. **Фрагментный шейдер** (Fragment Shader) — обрабатывает каждый пиксель (фрагмент) внутри геометрических примитивов. Он определяет цвет каждого пикселя с учетом освещения, текстур и других факторов.

В программе вершинный шейдер преобразует координаты вершин и передает в фрагментный шейдер нормали и текстурные координаты. Фрагментный шейдер применяет текстуру Земли и простую модель освещения для создания реалистичного изображения.

Система координат и матричные преобразования

В трехмерной графике широко используются матричные преобразования для перемещения, вращения и масштабирования объектов. В WebGL применяются следующие типы матриц:

1. **Модельная матрица** (Model Matrix) — преобразует координаты из локальной системы координат объекта в мировую систему координат.
2. **Видовая матрица** (View Matrix) — преобразует мировые координаты в систему координат камеры.
3. **Проекционная матрица** — преобразует координаты из пространства камеры в нормализованные координаты устройства.

Итоговая матрица преобразования получается умножением этих трех матриц:

$$MVP = Projection \cdot View \cdot Model$$

В программе матричные преобразования используются для вращения Земли вокруг наклонной оси, размещения спутников на орбитах, управления камерой и изменения точки обзора

3 Практическая часть

Структура программы

Разработанная программа представляет собой веб-приложение, которое визуализирует Землю и орбитальную группировку ГЛОНАСС с помощью WebGL.

Программа включает следующие основные блоки:

- Определение констант и параметров модели
- Инициализация WebGL и создание шейдеров
- Создание геометрии для Земли, орбит и спутников
- Обработчики событий для взаимодействия с пользователем
- Функции для матричных преобразований
- Основной цикл рендеринга

```
const R_EARTH = 6378.137;
const ORBIT_HEIGHT = 19100;
const INCLINATION = 64.8 * Math.PI / 180;
const EARTH_AXIAL_TILT = 23.5 * Math.PI / 180;
const SATS_PER_PLANE = 8;
const NUM_PLANES = 3;
const NUM_SATS = SATS_PER_PLANE * NUM_PLANES;
const ORBIT_RADIUS = R_EARTH + ORBIT_HEIGHT;
const PLANE_OFFSET = 120;
const SAT_OFFSET = 45;
const SCALE_FACTOR = 100;
```

Листинг 1: Константы модели

3.1 Основные функции программы

Создание Земли и орбит

Функция `createSphere` создает сферу для отображения Земли:

```
function createSphere(radius, latBands, longBands) {
  const vertices = [];
  const normals = [];
  const texCoords = [];
  const indices = [];

  for (let lat = 0; lat <= latBands; lat++) {
    const theta = lat * Math.PI / latBands;
    const sinTheta = Math.sin(theta);
    const cosTheta = Math.cos(theta);

    for (let lon = 0; lon <= longBands; lon++) {
      const phi = lon * 2 * Math.PI / longBands;
      const sinPhi = Math.sin(phi);
      const cosPhi = Math.cos(phi);

      const x = cosPhi * sinTheta;
      const y = cosTheta;
      const z = sinPhi * sinTheta;
      const u = 1 - (lon / longBands);
```

```
      const v = lat / latBands;

      normals.push(x, y, z);
      texCoords.push(u, v);
      vertices.push(radius * x / SCALE_FACTOR, radius * y / SCALE_FACTOR,
    radius * z / SCALE_FACTOR);
    }
  }

  for (let lat = 0; lat < latBands; lat++) {
    for (let lon = 0; lon < longBands; lon++) {
      const first = (lat * (longBands + 1)) + lon;
      const second = first + longBands + 1;

      indices.push(first, second, first + 1);
      indices.push(second, second + 1, first + 1);
    }
  }

  return {
    vertices: new Float32Array(vertices),
    normals: new Float32Array(normals),
    texCoords: new Float32Array(texCoords),
    indices: new Uint16Array(indices)
  };
}
```

Листинг 2: Функция createSphere

```
function createOrbits() {
  const orbitPoints = [];
  const numPointsPerOrbit = 200;

  for (let plane = 0; plane < NUM_PLANES; plane++) {
    const U = (plane * PLANE_OFFSET) * Math.PI / 180;

    for (let i = 0; i <= numPointsPerOrbit; i++) {
      const angle = (i / numPointsPerOrbit) * 2 * Math.PI;
      const r = ORBIT_RADIUS / SCALE_FACTOR;
      const cosO = Math.cos(U), sinO = Math.sin(U);
      const cosu = Math.cos(angle), sinu = Math.sin(angle);
      const cosi = Math.cos(INCLINATION), sini = Math.sin(INCLINATION);

      const x = r * (cosO * cosu - sinO * sinu * cosi);
      const y = r * (sinu * sini);
      const z = r * (sinO * cosu + cosO * sinu * cosi);

      orbitPoints.push(x, y, z);
    }
  }

  return new Float32Array(orbitPoints);
}
```

Листинг 3: Функция createOrbits

Расчет положения спутников

```
function computeSatellites(time) {
  const satPositions = [];

  for (let plane = 0; plane < NUM_PLANES; plane++) {
    const U = (plane * PLANE_OFFSET) * Math.PI / 180;

    for (let i = 0; i < SATS_PER_PLANE; i++) {
      const arg_lat = ((i * SAT_OFFSET + time * rotationSpeed) % 360) *
   Math.PI / 180;
      const r = ORBIT_RADIUS / SCALE_FACTOR;
      const cosO = Math.cos(U), sinO = Math.sin(U);
      const cosu = Math.cos(arg_lat), sinu = Math.sin(arg_lat);
      const cosi = Math.cos(INCLINATION), sini = Math.sin(INCLINATION);

      const x = r * (cosO * cosu - sinO * sinu * cosi);
      const y = r * (sinu * sini);
      const z = r * (sinO * cosu + cosO * sinu * cosi);

      satPositions.push(x, y, z);
    }
  }
  return new Float32Array(satPositions);
}
```

Листинг 4: Функция computeSatellites

Рендеринг сцены

```
function render(time) {
  time *= 0.001;

  gl.viewport(0, 0, gl.canvas.width, gl.canvas.height);
  gl.clearColor(0, 0, 0, 1);
  gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
  gl.enable(gl.DEPTH_TEST);

  earthRotation += rotationSpeed / 50;

  const aspect = canvas.width / canvas.height;
  const proj = perspective(Math.PI / 4, aspect, 1, 2000);
  const view = lookAt(0, 0, cameraDistance, cameraAngleX, cameraAngleY);
  const viewProj = multiply(proj, view);

  if (showEarth) {
    // ...
  }
  if (showEarthAxis) {
    // ...
  }
  if (showOrbits) {
    // ...
  }
  const positions = computeSatellites(time * 30);
}
```

Листинг 5: Функция render

4 Примеры работы

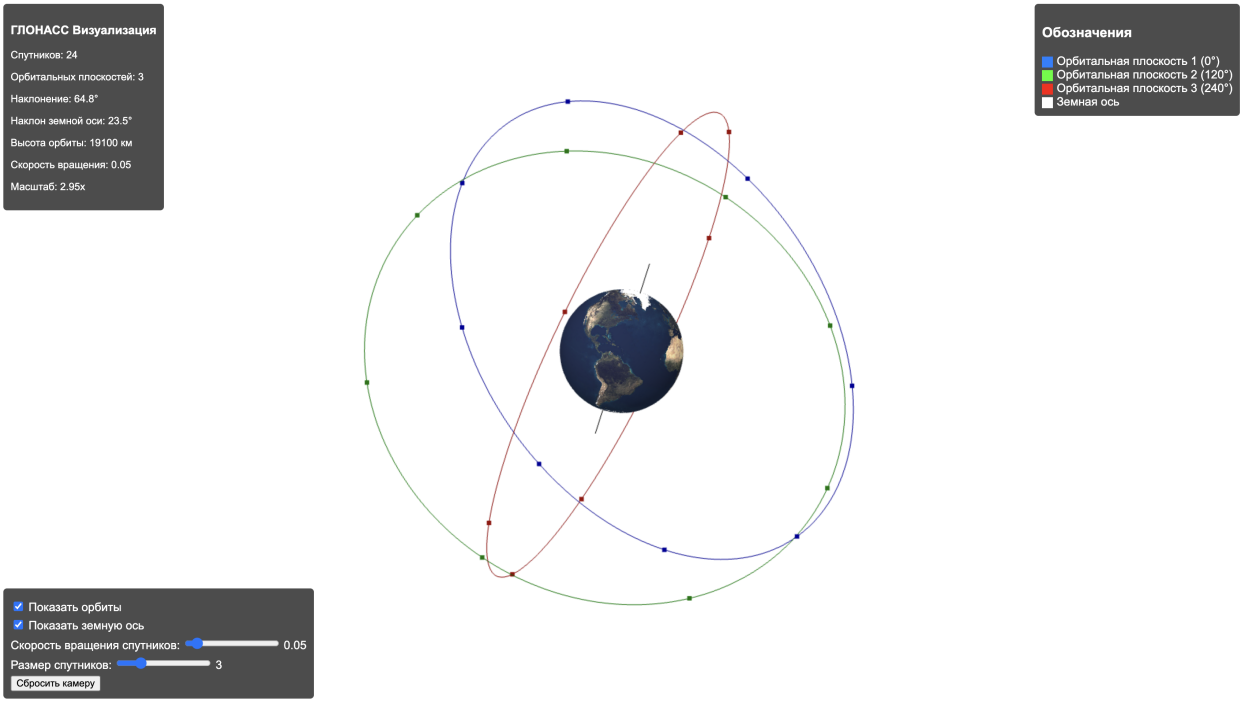

Рис. 1: Интерфейс программы

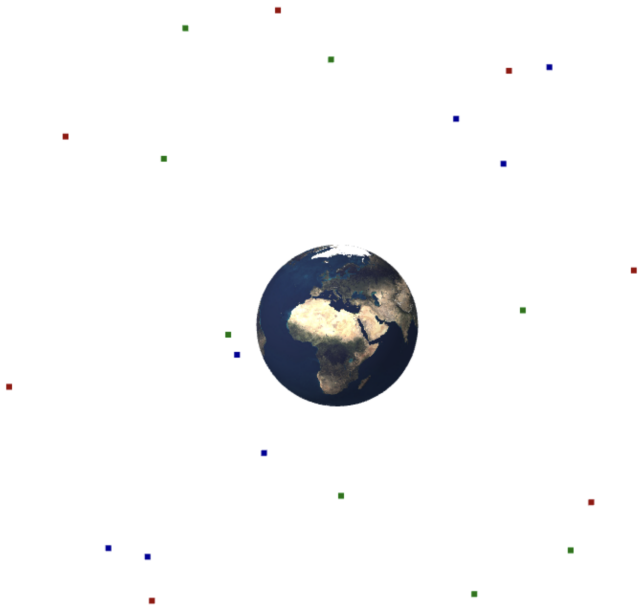

Рис. 2: С выключенной визуализацией

5 Выводы

В ходе выполнения лабораторной работы была успешно разработана веб-система для визуализации спутниковой группировки ГЛОНАСС с использованием технологии WebGL.

Программа позволяет наблюдать:

1. Трехмерную модель Земли с текстурой, вращающуюся с наклоном оси в 23,5°
2. Три орбитальные плоскости, повернутые относительно друг друга на 120°
3. 24 спутника ГЛОНАСС (по 8 на каждой орбите), движущиеся по круговым орбитам
4. Земную ось, показывающую наклон вращения планеты

Пользовательский интерфейс предоставляет следующие возможности:

- Вращение сцены с помощью мыши
- Масштабирование сцены колесом мыши
- Изменение скорости движения спутников
- Изменение размера отображения спутников
- Включение/выключение отображения орбит и земной оси
- Отображение информации о параметрах визуализации

6 Список источников

1. ГЛОНАСС Вики — ru.wikipedia.org/wiki/ГЛОНАСС
2. Статья на Хабре — https://habr.com/ru/articles/164185/
3. Лекции по дисциплине «Веб-программирование», УУНиТ, 2025