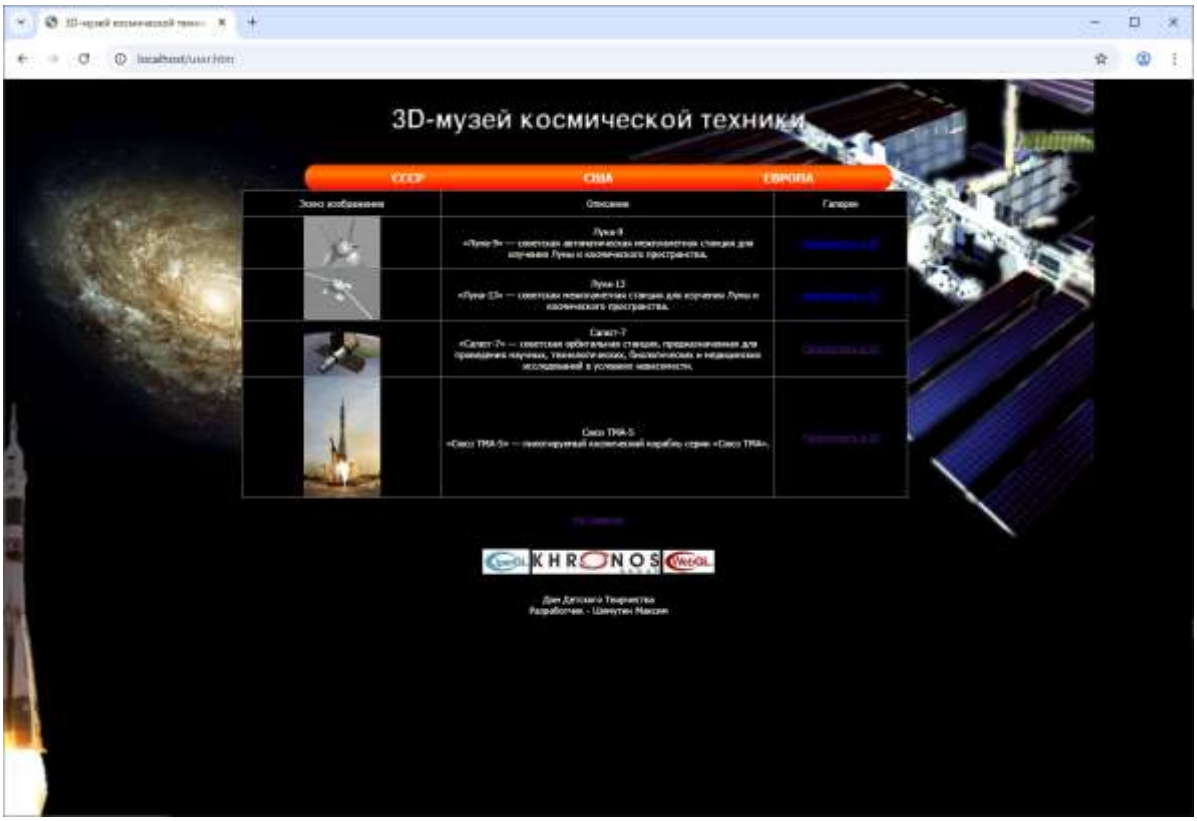
МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ «ДОМ ДЕТСКОГО ТВОРЧЕСТВА»

«ИНТЕРАКТИВНАЯ ВЫСТАВКА КОСМИЧЕСКИХ АППАРАТОВ»

ОБУЧАЮЩИЙ WEB-САЙТ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИИ WEBGL



ПРОЕКТ ПОДГОТОВИЛ:
Шимутин Максим Дмитриевич *08.01.2010 г.р.*

РУКОВОДИТЕЛИ:
педагоги дополнительного образования
Романенко Игорь Николаевич
Романенко Руслана Александровна

КАМЧАТСКИЙ КРАЙ г. ВИЛЮЧИНСК
2025

СОДЕРЖАНИЕ

1. Введение 3

2. Основная часть 5

2.1 Библиотека WebGL 5

2.2 Порядок выполнения работы 5

2.2.1 Создание приложения для просмотра 3D моделей 5

2.2.2 Написание кода Web-страницы 11

2.2.3 Создание графических элементов 12

2.2.4 Тестирование Web-страницы 12

2.2.5 Подготовка 3D моделей 13

3. Заключение 14

4. Список использованной литературы 15

5. Приложение 16

5.1. Видеообзор интерактивной выставки космических аппаратов .. 16

1. ВВЕДЕНИЕ

Космос - загадочное пространство. Ещё издавна человек начал изучать звезды на небе. Для детального изучения, сначала изобрели телескоп, затем человек сам погрузился в объятия звездной пучины - отправился в космический полёт. И чем больше человек узнавал о космосе, тем больше вопросов у него появлялось. За всю историю человечества в космос было запущено не малое количество различных спутников. Их запуски происходили почти в каждом уголке Земли: в Европе, в Азии, в Америке. Каждый из космических аппаратов уникальны по-своему.

Актуальность проекта: Для создания исследовательских работ в области космических технологий, требуется искать информацию по огромному количеству искусственных спутников, различных космических аппаратов, на это расходуется очень много времени, поэтому я решил создать отдельный сайт, на основе библиотеки WEBGL, на котором будут собраны описания моделей космических аппаратов некоторых стран, а также их 3D-модели. Мой проект поможет досконально изучить строение и историю развития космических кораблей и различных искусственных спутников, стран с развитой космической отраслью, а также людей, которые внесли свой вклад в развитие космических технологий, и в целом в историю развития космонавтики.

Гипотеза: С помощью современного уровня развития web-технологий возможно создать сайт, на котором будет собрана подробная информация про космические аппараты, а также будет возможность посмотреть не только их изображения, но и трехмерные модели, что сделает процесс изучения более наглядным и информативным.

Новизна проекта: Новизна моего проекта заключается в том, что в процессе разработки я буду использовать самые современные web-технологии такие как Web-GL, позволяющая с помощью языка программирования JavaScript создавать контекст для вывода трехмерной графики на web-страницах. Это позволит детально рассмотреть модели космических аппаратов в процессе их изучения.

Цель проекта: создание удобного обучающего сайта с использованием технологии WebGL для изучения история развития космических технологий и демонстрации моделей космических аппаратов.

Задачи проекта:

- изучить принципы работы с программой Blender3D;
- установить web-сервер (AppServer 9.3)
- создание 3D-моделей космических аппаратов Европы, Советского Союза и США;
- изучить возможности библиотеку WebGL;
- написать программу на языке JavaScript;
- создать сайт с описанием спутников и их 3D-моделями.

2. ОСНОВНАЯ ЧАСТЬ

2.1. БИБЛИОТЕКА WEBGL.

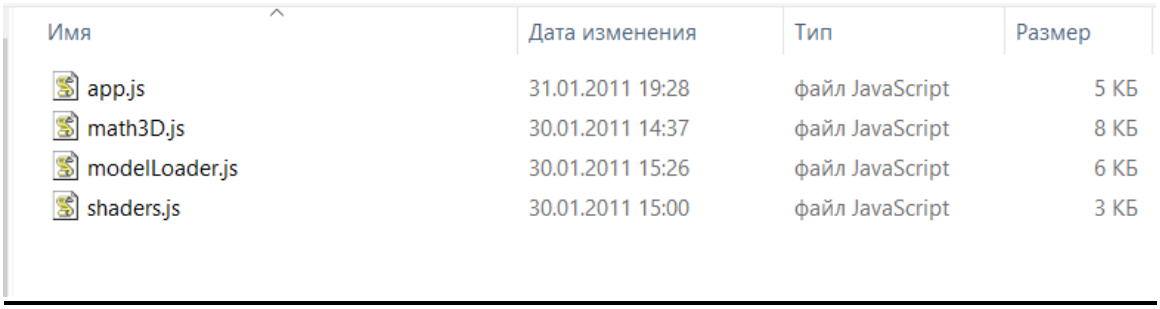
WebGL используется для создания 3D-игр, сайтов с использованием 3D-графики, визуализаций данных, интерактивных 3D-приложений и других элементов сложной графики на web-сайтах. При этом для работы с данной технологией не требуются сторонние плагины или библиотеки. WebGL основана на языке программирования JavaScript, хотя может использоваться с любым языком, который работает с API, за это данная технология и получила свое широкое применение. Библиотека состоит из JavaScript, который управляет рабочим процессом, и специальных кодах шейдеров, которые могут выполняться непосредственно на графических процессорах на видеокартах, благодаря чему разработчики могут получить доступ к дополнительным ресурсам компьютера, увеличить быстродействие. Технология WebGL поддерживается на многих современных браузерах, таких как: Chrome, Opera, Firefox.

По мимо библиотеки. WebGL понадобятся: приложение appServ, включающее в себя веб-сервер apache, интерпретатор языка php, сервер баз данных mySCL. Это нужно для того, чтобы веб-приложение могло функционировать в полном объёме. Так как существуют ограничения по загрузки файлов с локальных дисков, поэтому чтобы модели можно было загрузить требуется разместить приложение на работающем веб_сервере.

2.2. ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

2.2.1. Создание приложения для просмотра 3D моделей в окне браузера

Первым делом был написан программный код с помощью текстового редактора Notepad++. Этот код позволяет открывать сайт в окне браузера (Chrome, Opera, Firefox и другие). Было создано несколько программных файлов на языке программирования JavaScript.



Имя	Дата изменения	Тип	Размер
app.js	31.01.2011 19:28	файл JavaScript	5 КБ
math3D.js	30.01.2011 14:37	файл JavaScript	8 КБ
modelLoader.js	30.01.2011 15:26	файл JavaScript	6 КБ
shaders.js	30.01.2011 15:00	файл JavaScript	3 КБ

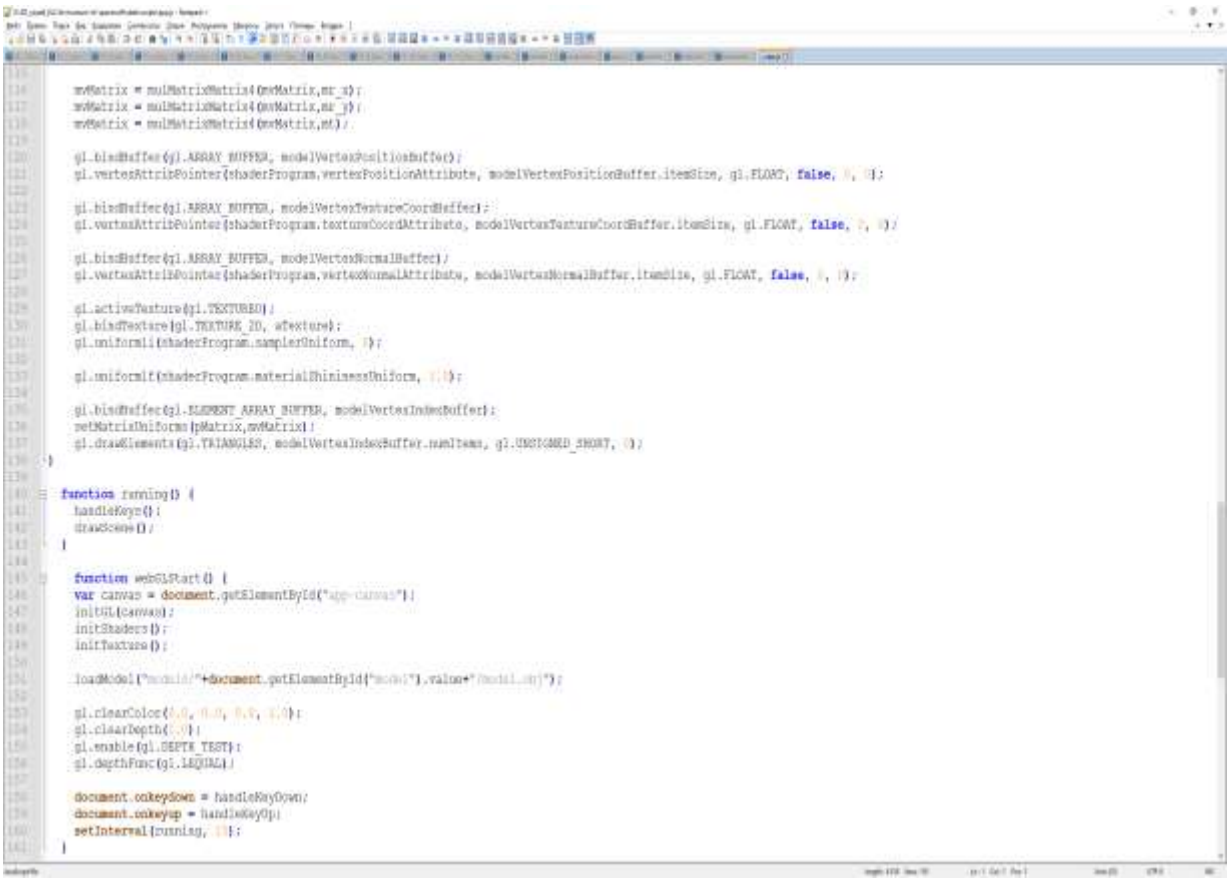
Script app.js – базовое приложение, для инициализации OpenGL и WebGL.

Script math3D.js – отвечает за работу с 3D графикой для расчёта матриц и векторов.

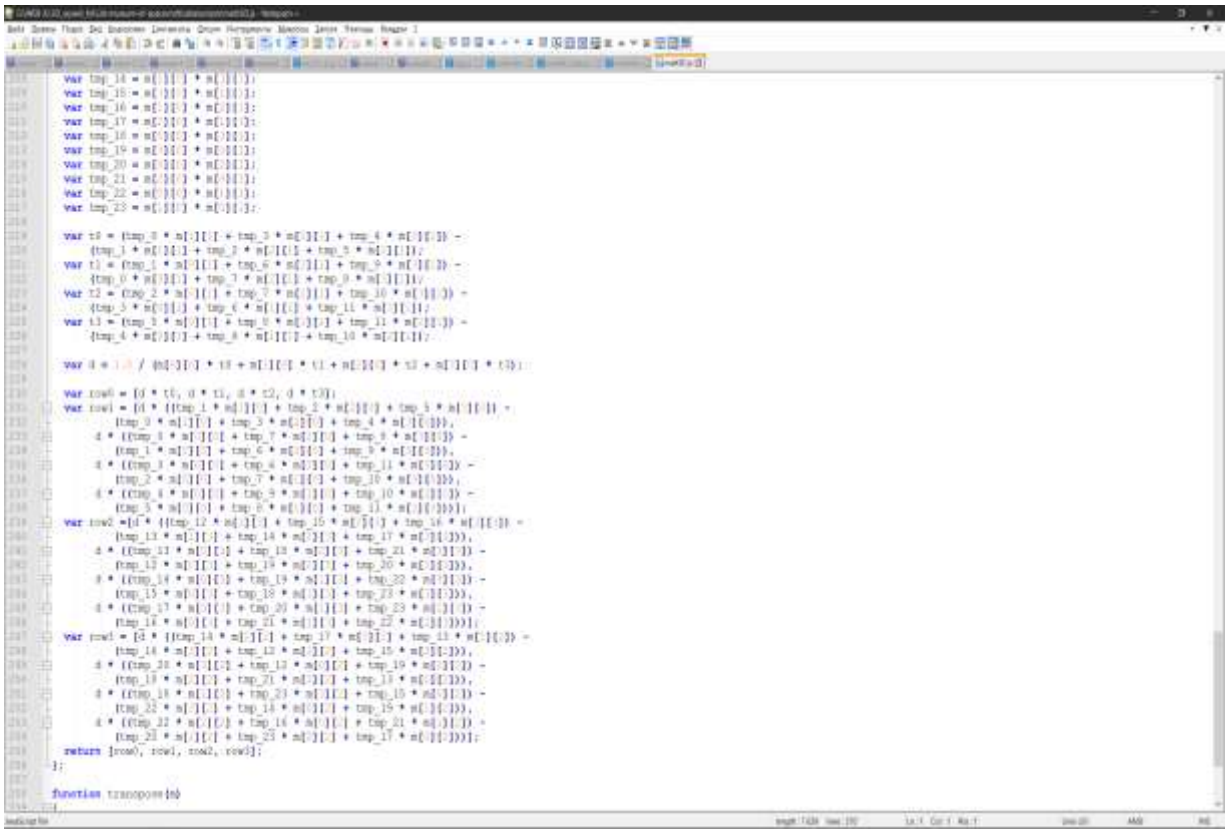
Script modelLoader.js – отвечает за загрузку файлов в формате .obj.

Script shaders.js – отвечает за работу шейдеров.

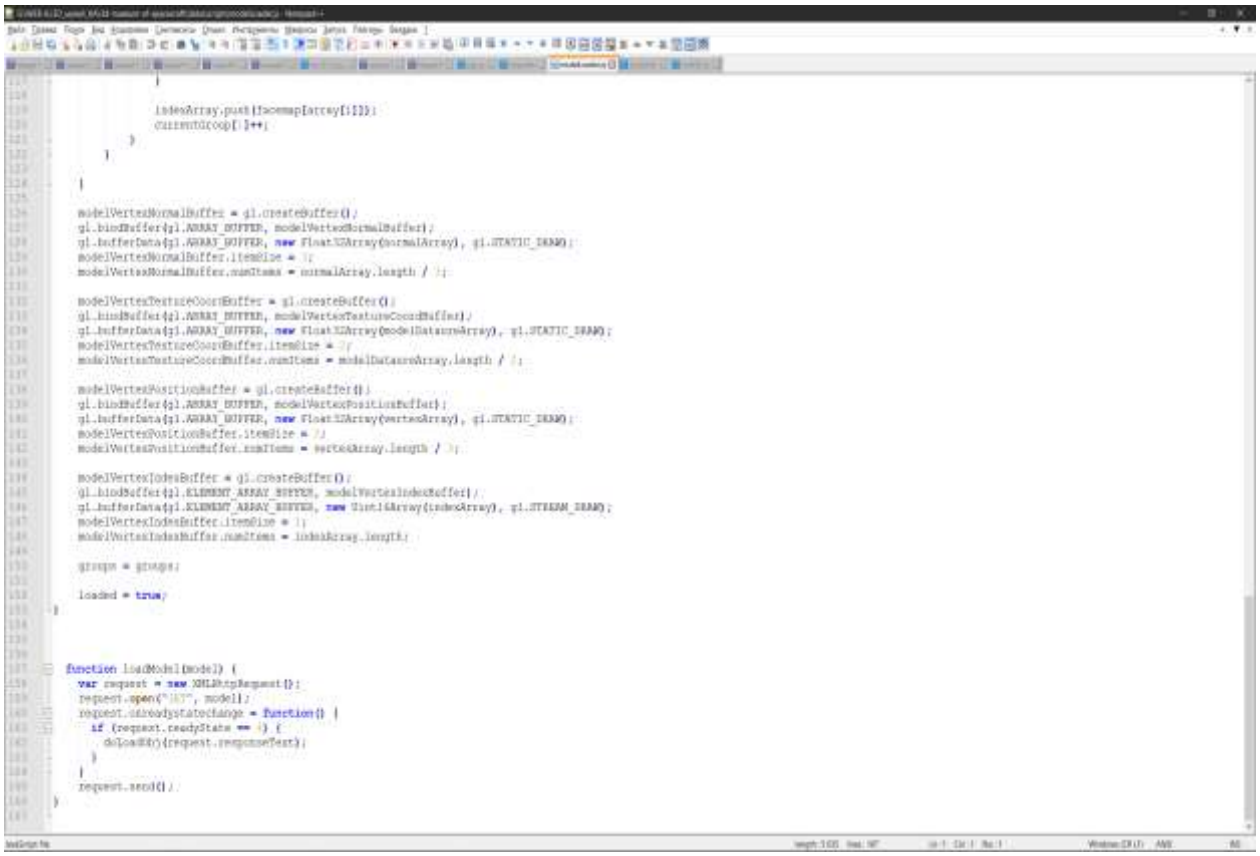
Код для **инициализации OpenGL**, функционирования матрицы и текстур, отрисовка основной сцены, инициализации WebGL.



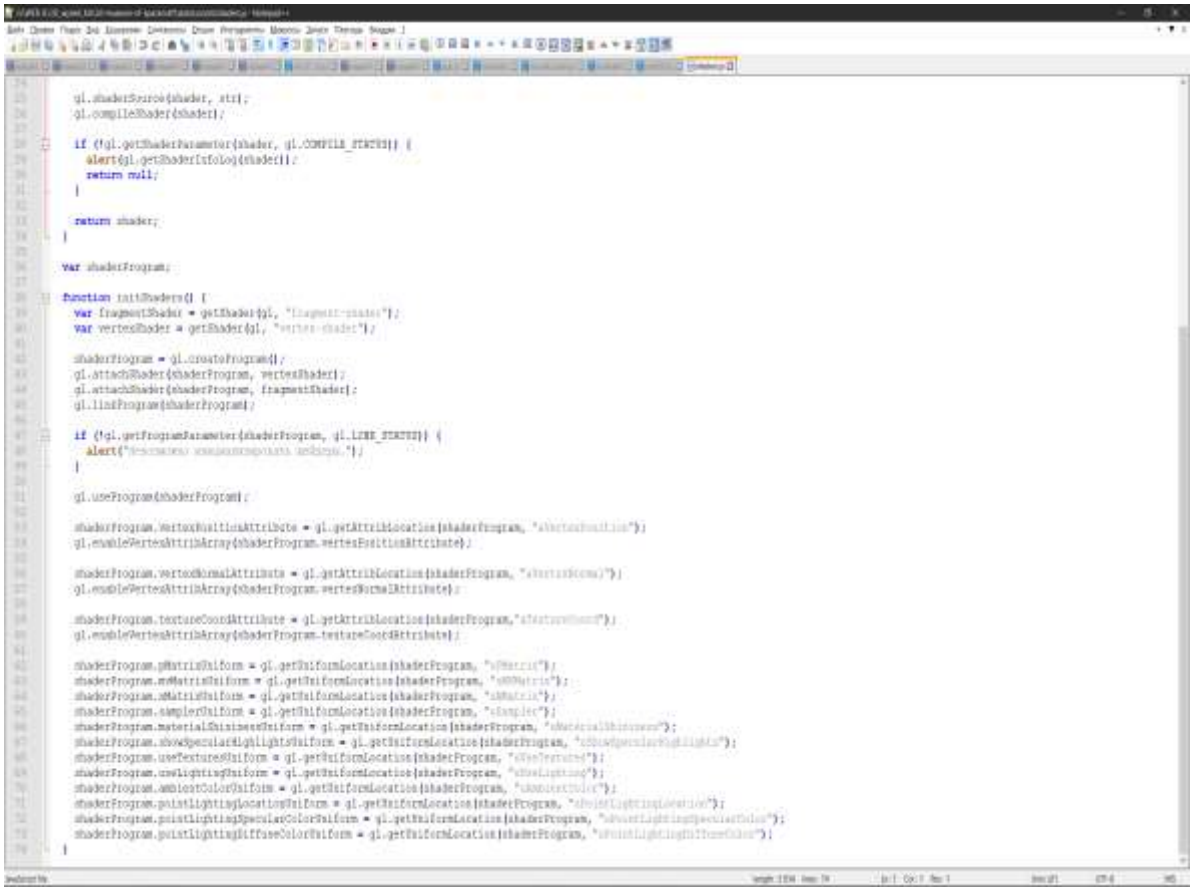
Математическая библиотека для работы с 3D графикой для расчёта матриц и векторов.



Модуль загрузки моделей в формате obj.

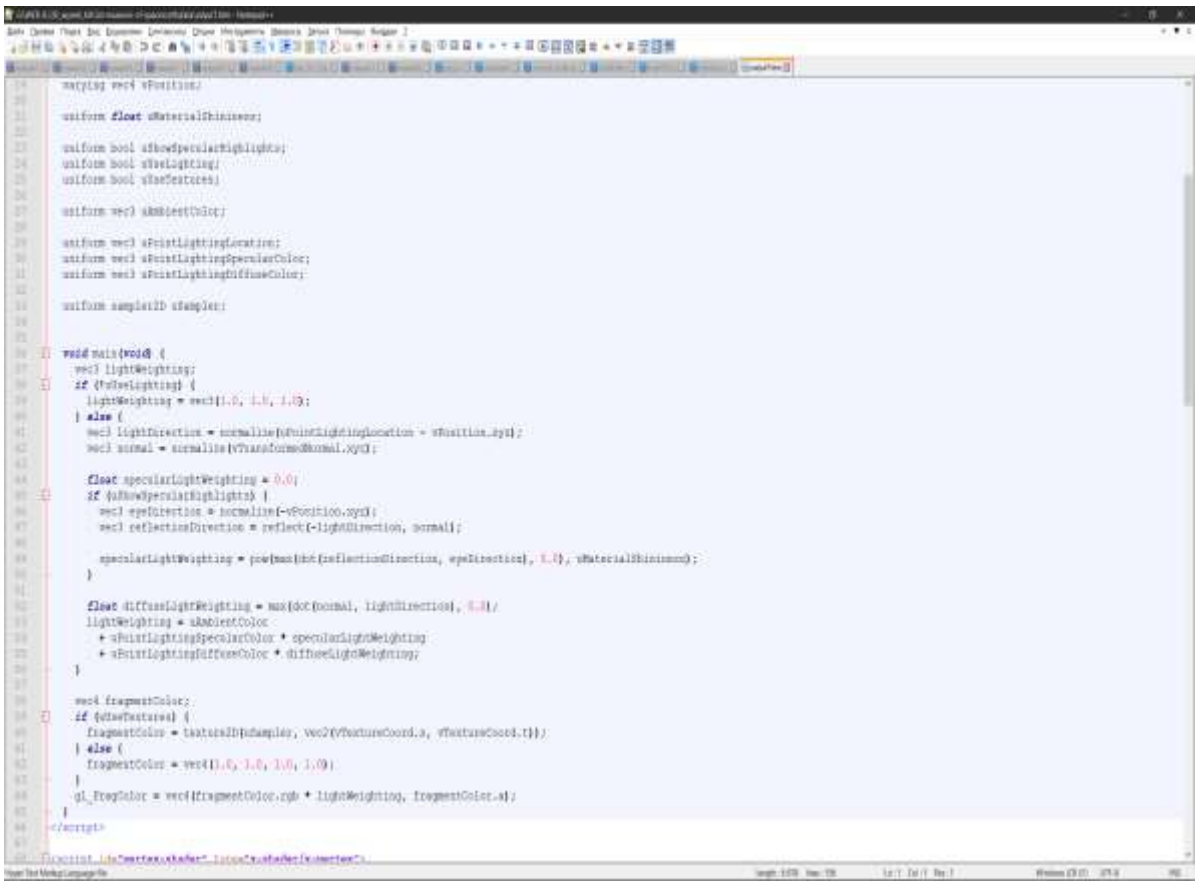


Модуль для работы с шейдерами.

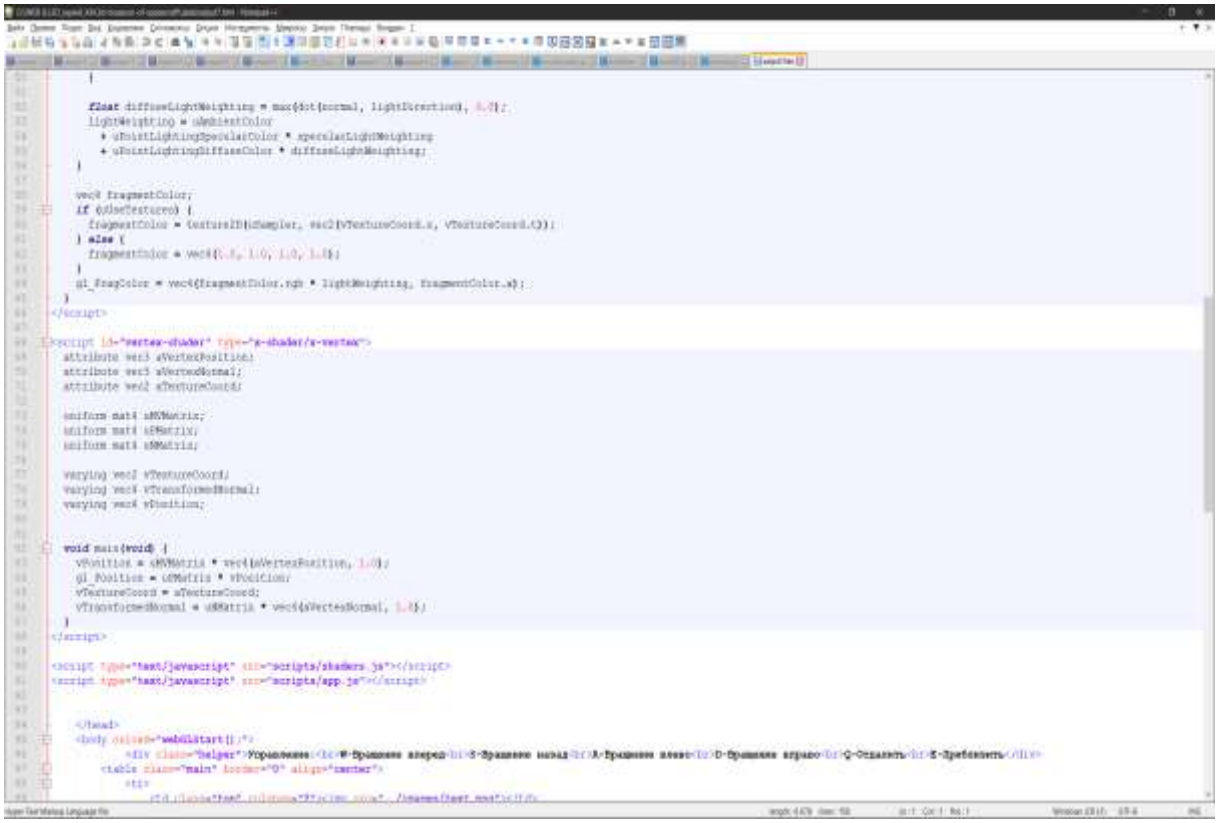


```
    gl.shaderSource(shader, str);
    gl.compileShader(shader);

    if (!gl.getShaderParameter(shader, gl.COMPILE_STATUS)) {
      alert(gl.getShaderInfoLog(shader));
      return null;
    }

    return shader;
  }

  var shaderProgram;

  function initShaders() {
    var fragmentShader = getShader(gl, "fragment-shader");
    var vertexShader = getShader(gl, "vertex-shader");

    shaderProgram = gl.createProgram();
    gl.attachShader(shaderProgram, vertexShader);
    gl.attachShader(shaderProgram, fragmentShader);
    gl.linkProgram(shaderProgram);

    if (!gl.getProgramParameter(shaderProgram, gl.LINK_STATUS)) {
      alert("Невозможно инициализировать шейдеры.");
    }

    gl.useProgram(shaderProgram);

    shaderProgram.vertexPositionAttribute = gl.getAttribLocation(shaderProgram, "aVertexPosition");
    gl.enableVertexAttribArray(shaderProgram.vertexPositionAttribute);

    shaderProgram.vertexNormalAttribute = gl.getAttribLocation(shaderProgram, "aVertexNormal");
    gl.enableVertexAttribArray(shaderProgram.vertexNormalAttribute);

    shaderProgram.textureCoordAttribute = gl.getAttribLocation(shaderProgram,"aTextureCoord");
    gl.enableVertexAttribArray(shaderProgram.textureCoordAttribute);

    shaderProgram.pMatrixUniform = gl.getUniformLocation(shaderProgram, "uPMatrix");
    shaderProgram.mvMatrixUniform = gl.getUniformLocation(shaderProgram, "uMVMatrix");
    shaderProgram.nMatrixUniform = gl.getUniformLocation(shaderProgram, "uNMatrix");
    shaderProgram.samplerUniform = gl.getUniformLocation(shaderProgram, "uSampler");
    shaderProgram.materialShininessUniform = gl.getUniformLocation(shaderProgram, "uMaterialShininess");
    shaderProgram.showSpecularHighlightsUniform = gl.getUniformLocation(shaderProgram, "uShowSpecularHighlights");
    shaderProgram.useTexturesUniform = gl.getUniformLocation(shaderProgram, "uUseTextures");
    shaderProgram.useLightingUniform = gl.getUniformLocation(shaderProgram, "uUseLighting");
    shaderProgram.ambientColorUniform = gl.getUniformLocation(shaderProgram, "uAmbientColor");
    shaderProgram.pointLightingLocationUniform = gl.getUniformLocation(shaderProgram, "uPointLightingLocation");
    shaderProgram.pointLightingSpecularColorUniform = gl.getUniformLocation(shaderProgram, "uPointLightingSpecularColor");
    shaderProgram.pointLightingDiffuseColorUniform = gl.getUniformLocation(shaderProgram, "uPointLightingDiffuseColor");
  }
```

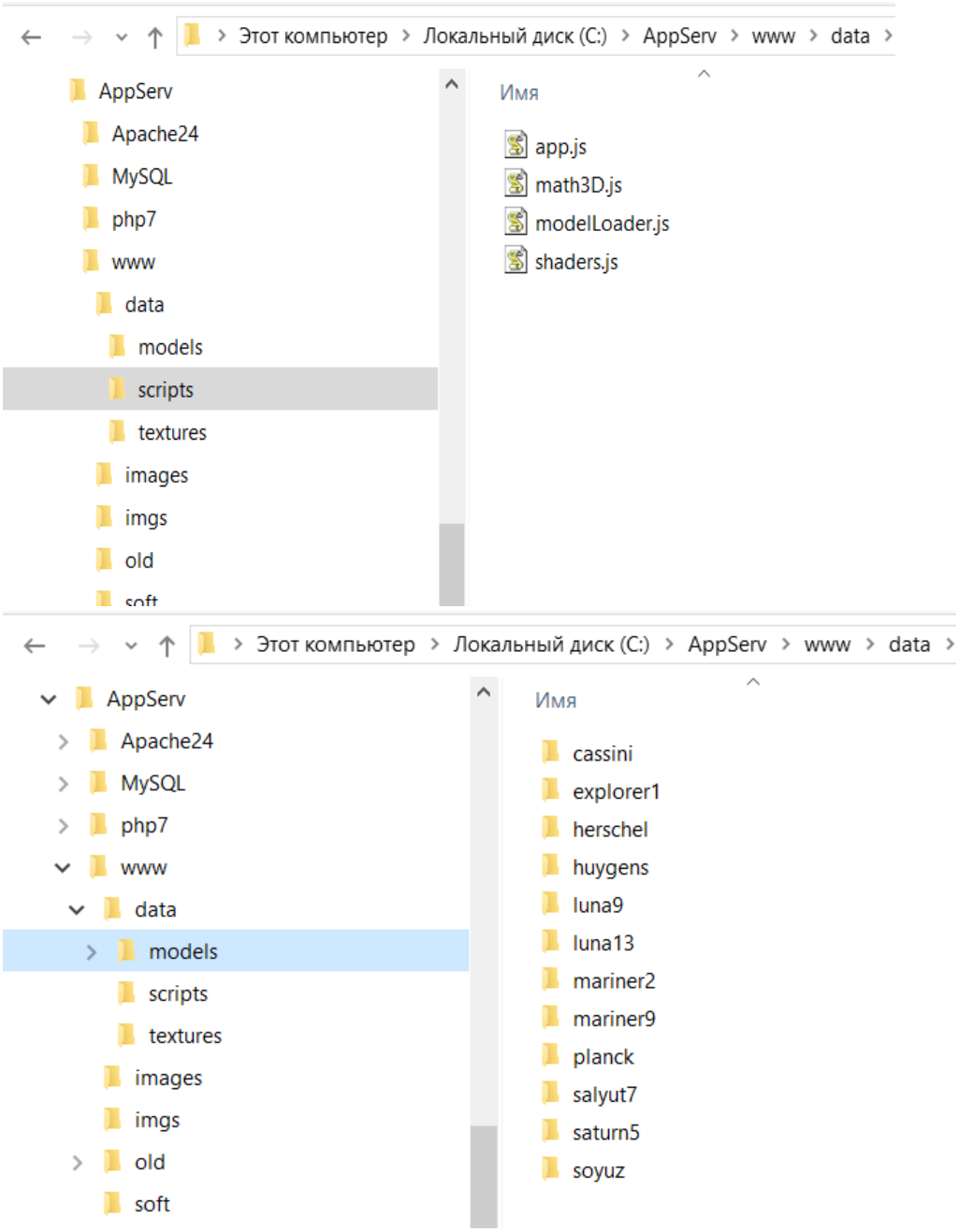
Вершинный и фрагментный шейдеры.



```
  varying vec4 vPosition;

  uniform float uMaterialShininess;

  uniform bool uShowSpecularHighlights;
  uniform bool uUseLighting;
  uniform bool uUseTextures;

  uniform vec3 uAmbientColor;

  uniform vec3 uPointLightingLocation;
  uniform vec3 uPointLightingSpecularColor;
  uniform vec3 uPointLightingDiffuseColor;

  uniform sampler2D uSampler;


  void main(void) {
    vec3 lightWeighting;
    if (!uUseLighting) {
      lightWeighting = vec3(1.0, 1.0, 1.0);
    } else {
      vec3 lightDirection = normalize(uPointLightingLocation - vPosition.xyz);
      vec3 normal = normalize(vTransformedNormal.xyz);

      float specularLightWeighting = 0.0;
      if (uShowSpecularHighlights) {
        vec3 eyeDirection = normalize(-vPosition.xyz);
        vec3 reflectionDirection = reflect(-lightDirection, normal);

        specularLightWeighting = pow(max(dot(reflectionDirection, eyeDirection), 0.0), uMaterialShininess);
      }

      float diffuseLightWeighting = max(dot(normal, lightDirection), 0.0);
      lightWeighting = uAmbientColor
        + uPointLightingSpecularColor * specularLightWeighting
        + uPointLightingDiffuseColor * diffuseLightWeighting;
    }

    vec4 fragmentColor;
    if (uUseTextures) {
      fragmentColor = texture2D(uSampler, vec2(vTextureCoord.s, vTextureCoord.t));
    } else {
      fragmentColor = vec4(1.0, 1.0, 1.0, 1.0);
    }
    gl_FragColor = vec4(fragmentColor.rgb * lightWeighting, fragmentColor.a);
  }
</script>
```

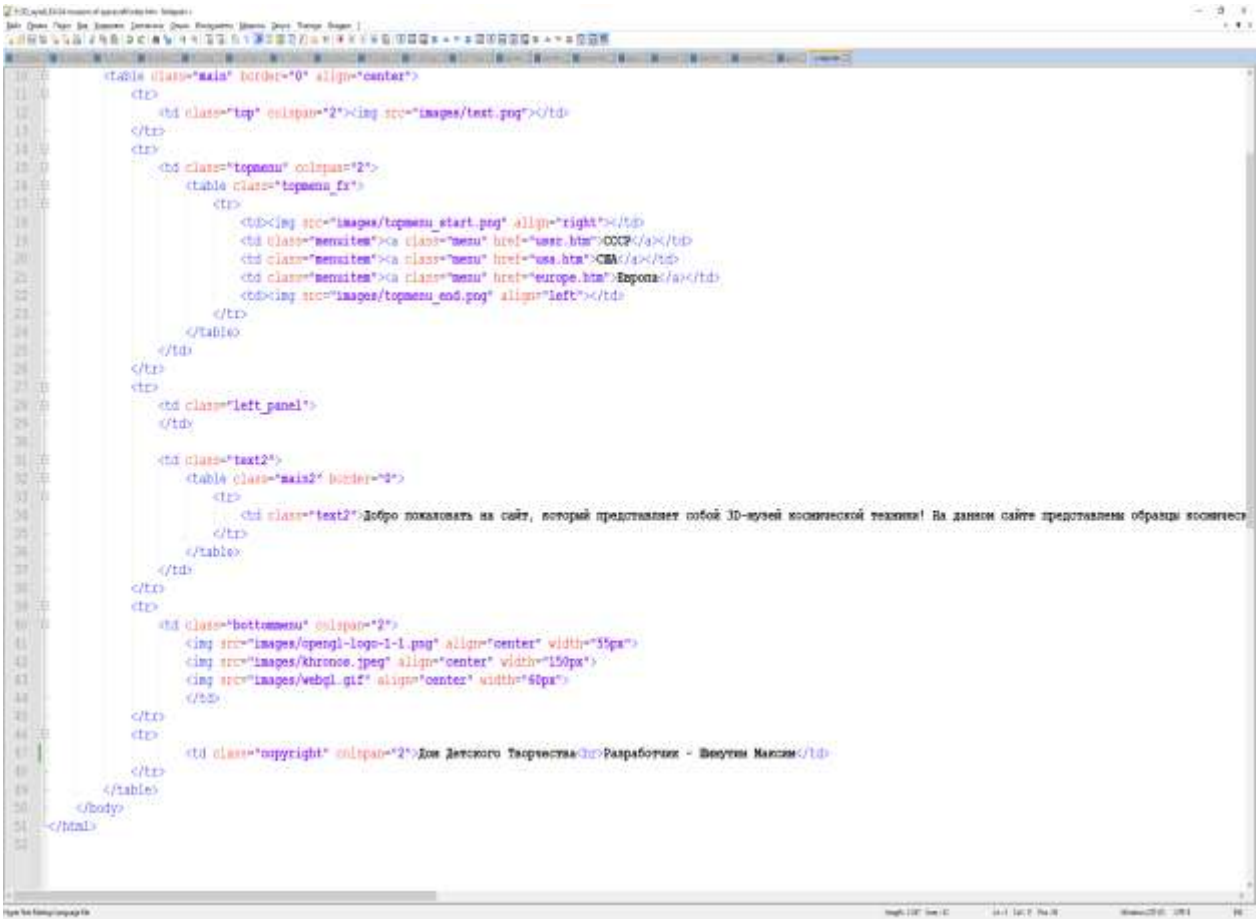


Структура файлов проекта.



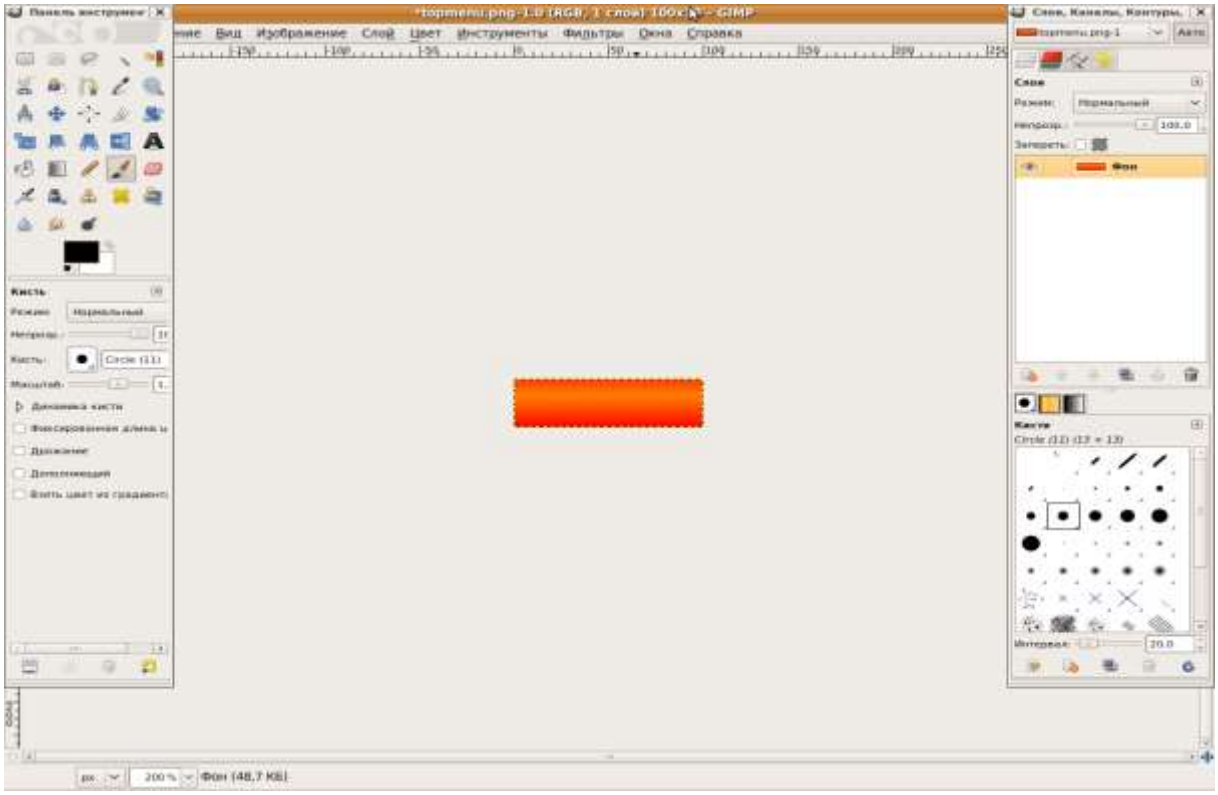
2.2.2. Написание кода Web-страницы.

Lkz ntcnbПосле создания кода для самого приложения я приступил к созданию кода Web-страницы, а также код для каждого из 12 космических аппаратов: по 4 для каждого представителя (СССР, США, Европа). Данный программный код был точно также написан на основе текстового редактора Notepad++.



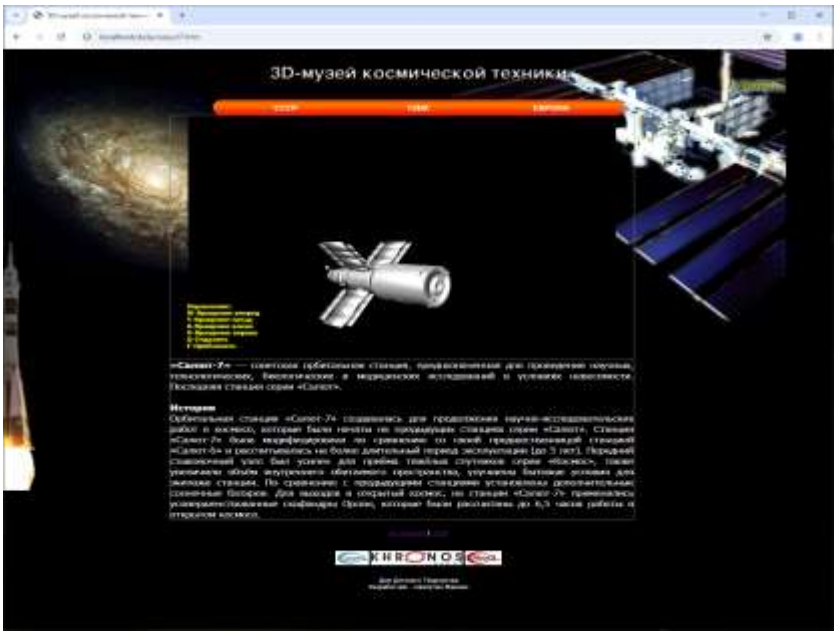
2.2.3. Создание графических элементов.

Вся графика сайта была создана через графический редактор GIMP.



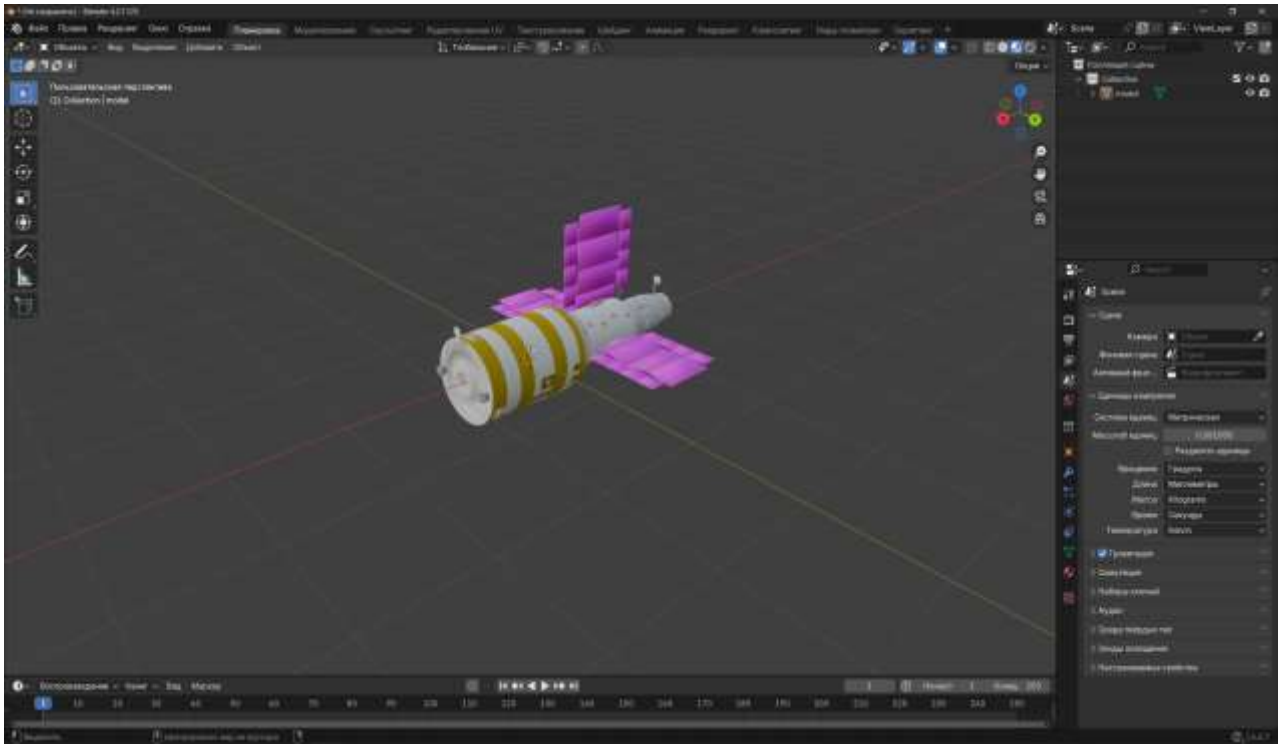
2.2.4. Тестирование Web-страницы.

Для начала необходимо установить приложение AppServer, сочетающее в себе web-сервер, сервер баз данных. Тестирование сайта происходило с использованием браузера с поддержкой WebGL (использовался браузер Mozilla Firefox и Google Chrome). Тестирование показало положительный результат в работе сайта.



2.2.5. Подготовка 3D моделей.

Для начала нужно было произвести конвертирование в подходящий для приложения просмотра 3D моделей формат. Я выбрал формат (.obj), сделав конвертирование через пакет для работы с 3D графикой Blender. Так же приходилось самому изменять и моделировать так как многих моделей не было в свободном доступе и их приходилось создавать и дорабатывать.



3. **ЗАКЛЮЧЕНИЕ.**

В заключении можно сказать, что, выдвинутая гипотеза о том, что на современном уровне развития web-технологий можно создать сайт, на котором будет собрана подробная информация про космические аппараты, полностью подтверждена. В процессе работы над проектом удалось создать интерактивную web выставку 3D моделей космических аппаратов мира, что упростит процесс обучения астрономии и истории развития космических технологий, а также облегчит в последующем создание новых исследовательских работ и проектов в области космонавтики.

4. СПИСОК ЛИТЕРАТУРЫ

1. «CoderNet: «Что такое WebGL и как его включить. Подробная инструкция для чайников». https://codernet.ru/articles/web/chto_takoe_webgl_i_kak_ego_vklyuchit_podrobnaya_instrukcziya_dlya_chajnikov/ (дата обращения: 09.11.2025)»

2. «Metanit.com: «Онлайн-книга по WebGL» https://metanit.com/web/webgl/ (дата обращения: 13.11.2025)»

3. «Media.contented: «Интерактивная 2D/3D-графика для сайтов» https://media.contented.ru/znaniya/instrumenty/webgl-interaktivnaya-2d-3d-grafika-dlya-saitov/ (дата обращения: 14.11.25)»

4. ПРИЛОЖЕНИЕ

Видеообзор интерактивной выставки космических аппаратов.