МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

Отчет
по лабораторной работе №4
по дисциплине «КОМПЬЮТЕРНАЯ 3D-ГРАФИКА»
Тема: Текстура изображения

Студентка гр. 5381 _____ Буздина М.А.

Преподаватель _____ Герасимова Т.В.

Санкт-Петербург
2019

Цель работы.

Отредактируйте свою собственную картинку и сделайте ее текстурной картой.

Задание.

1. Разложите текстуру по изображению
2. Закрепите и отцентрируйте свое изображение на объекте

Основные теоретические сведения.

Модуль текстуры, также называемый модулем отображения текстуры (TMU) или модулем обработки текстуры (TPU), является аппаратным компонентом в графическом процессоре, который выполняет выборку. Выборка - это процесс вычисления цвета по текстуре изображения и координатам текстуры. Отображение текстурного изображения на поверхность является довольно сложной операцией, поскольку для нее требуется нечто большее, чем просто возврат цвета текселя, который содержит некоторые заданные координаты текстуры. Это также требует применения соответствующего фильтра уменьшения или увеличения, возможно, используя mipmaps/ мип-карт, если доступно. Быстрая выборка текстур является одним из ключевых требований для хорошей производительности графического процессора [1].

Переменная сэмплера – это переменная в шейдерной программе типа sampler2D или samplerCube. Sampler2D используется, чтобы сделать поиск в стандартном изображении текстуры; samplerCube используется, чтобы сделать поиск в cubemap текстуры Значение переменной сэмплера является ссылкой на единицу текстуры. Значение указывает, какая единица текстуры вызывается, когда переменная сэмплера используется для поиска текстуры. Переменные сэмплера должны быть объявлены как глобальные однородные переменные. едопустимо, чтобы шейдерная программа присваивала значение переменной сэмплера. Значение должно исходить из стороны JavaScrip.

На стороне JavaScript доступные текстурные блоки пронумерованы 0, 1, 2, ..., где максимальное значение зависит от реализации. Количество единиц может быть определено как значение выражения

gl.getParameter (gl.MAX_COMBINED_TEXTURE_IMAGE_UNITS)

Что касается JavaScript, то значение переменной сэмплера является целым числом. Если вы хотите, чтобы в переменной сэмплера использовалась единица текстуры 2, установите значение переменной сэмплера на 2. Это можно сделать с помощью функции *gl.uniform1i.*

Например, предположим, что шейдерная программа объявляет переменную сэмплера

uniform sampler2D u_texture;

Чтобы установить его значение из JavaScript, вам нужно расположение переменной в шейдерной программе. Если *prog* это программа шейдера, местоположение получается вызов

u_texture_location = gl.getUniformLocation (prog, "u_texture");

Затем вы можете указать переменной сэмплера использовать блок текстуры № 2, вызвав gl.uniform1i (u_texture_location, 2);

Обратите внимание, что целочисленное значение не доступно в GLSL целое число указывает сэмплеру, какую текстурную единицу использовать, но у шейдерной программы нет способа узнать номер используемой единицы.

Чтобы использовать текстуру изображения, вам также необходимо создать текстурный объект и загрузить изображение в текстурный объект.

Возможно, вы захотите установить некоторые свойства объекта текстуры, и вы можете создать набор mip-карт для текстуры. И вам придется связать текстурный объект с текстурным блоком. Все это делается на стороне JavaScript.

Функция *gl.texImage2D* используется для загрузки изображения в *текущий* связанный объект текстуры. Но помните, что эта команда и другие команды всегда применяются к текущему связанному объекту текстуры.

Объект текстуры не упоминается в команде; вместо этого объект текстуры должен быть связан до вызова команды.

Вы также должны указать текстурному блоку использовать объект текстуры. Прежде чем вы сможете это сделать, вам нужно сделать текстурный блок «активным», что делается путем вызова функции *gl.activeTexture*. Параметр является одной из констант *gl.TEXTURE0*, *gl.TEXTURE1*, *gl.TEXTURE2*, ..., которые представляют доступные текстурные единицы. (Значения этих констант не равны 0, 1, 2, ...) Первоначально текстурный блок номер 0 активен. Например, чтобы активировать текстурный блок № 2, используйте gl.activeTexture (gl.TEXTURE2);

Чтобы связать объект текстуры, когда активен модуль текстуры 2, тогда объект текстуры *textureObj* связан с модулем текстуры номер 2. Привязка просто сообщает модулю текстуры, какой объект текстуры использовать. То есть блок текстуры 2 будет выполнять обычный поиск текстур с использованием изображения и настроек, которые хранятся в *textureOb*.

Текстурный объект может быть связан с несколькими текстурными единицами одновременно. Тем не менее, данный текстурный блок может иметь только одну связанную *TEXTURE_2D* за раз.

Работа с текстурными изображениями в WebGL включает в себя работу с текстурными объектами, текстурными блоками и переменными сэмплера. Отношения между тремя иллюстрированы на этой картинке:

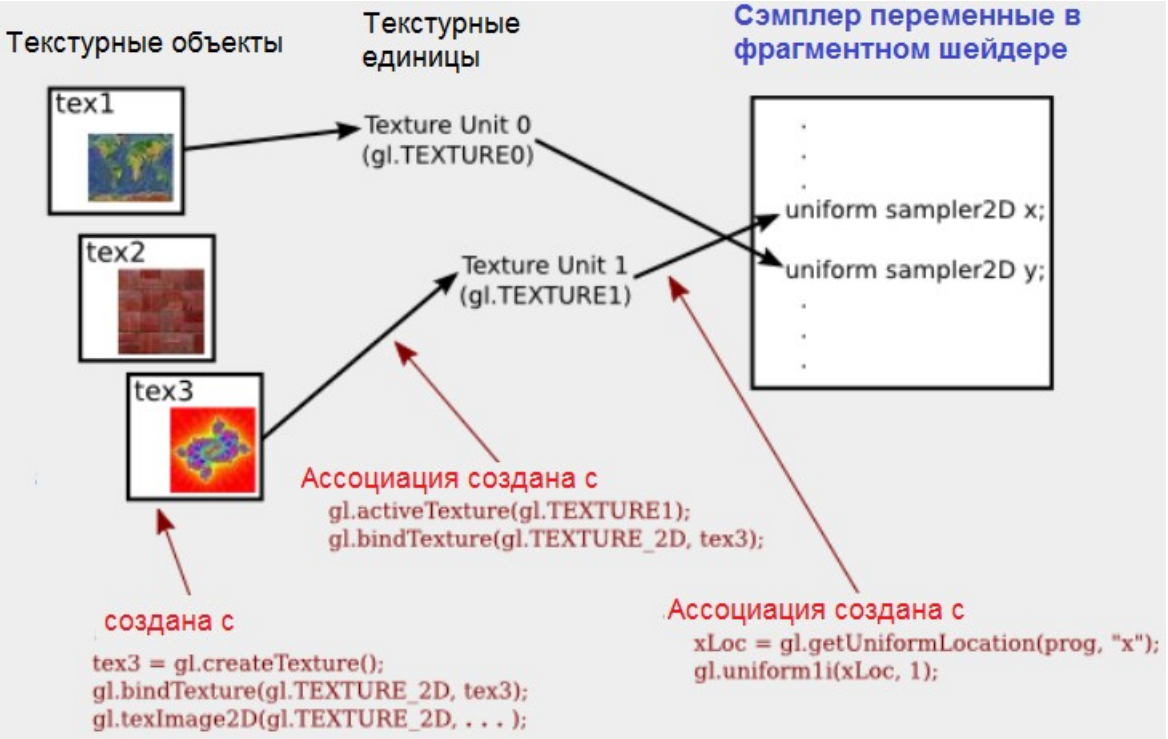

Рисунок 4.1

Переменная сэмплера использует текстурный блок, который использует текстурный объект, который содержит текстурное изображение. Команды JavaScript для настройки этой цепочки показаны на рис 4.1. Чтобы применить текстурное изображение к примитиву, вы должны настроить всю цепочку. Конечно, вы также должны предоставить координаты текстуры для примитива, и вам нужно использовать переменную sampler в программе шейдера для доступа к текстуре.

На этом рис. 4.2. показаны две текстуры, которые объединяются простыми способами для вычисления цветов пикселей в текстурированном квадрате:

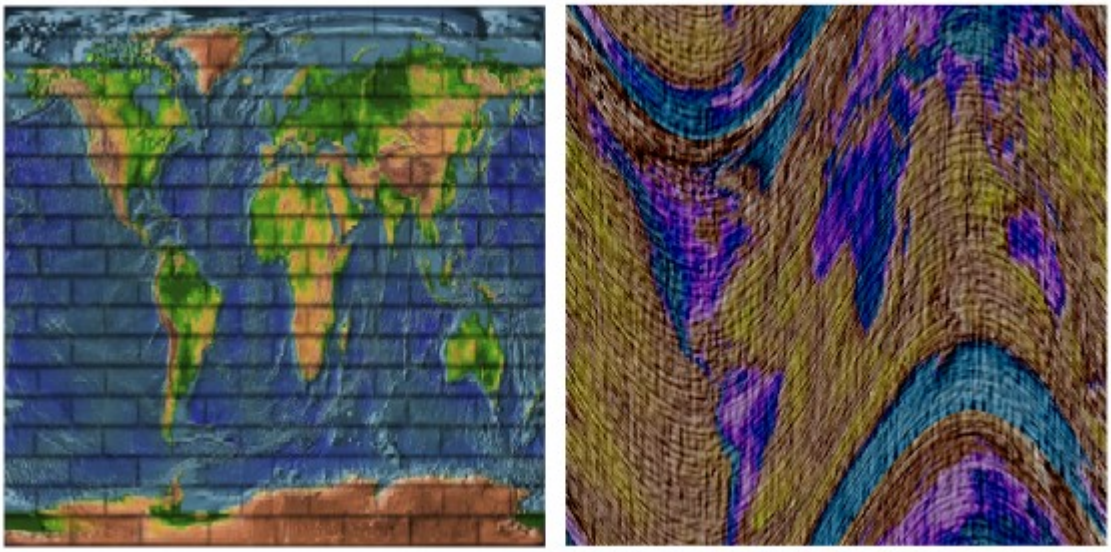

Рисунок 4.2

Используются два текстурных блока. Значения двух одинаковых переменных сэмплера, *u_texture1* и *u_texture2*, задаются во время инициализации с помощью кода

```
u_texture1_location = gl.getUniformLocation (prog, "u_texture1");
u_texture2_location = gl.getUniformLocation (prog, "u_texture2");
gl.uniform1i (u_texture1_location, 0);
gl.uniform1i (u_texture2_location, 1);
```

Значения никогда не меняются. Программа использует несколько текстурных изображений. Для каждого изображения есть текстурный объект.Объекты текстуры хранятся в массиве *textureObjects*. Два всплывающих меню позволяют пользователю выбрать, какие изображения текстуры будут применены к примитиву. Это реализуется в процедуре рисования путем привязки двух выбранных объектов текстуры к блокам текстуры 0 и 1, которые являются единицами, используемыми двумя переменными сэмплера. Код для этого:

```
var tex1Num = Number (document.getElementById ("textureChoice1"). value);
gl.activeTexture (gl.TEXTURE0);
gl.bindTexture (gl.TEXTURE_2D, textureObjects [tex1Num]);

var tex2Num = Number (document.getElementById ("textureChoice2"). value);
```

gl.activeTexture (gl.TEXTURE1);

gl.bindTexture (gl.TEXTURE_2D, textureObjects [tex2Num]);

Вставка изображений в текстурные объекты – это еще один вопрос, к которому мы обратимся.

Ход работы.

Текстурируем объекты кубов и сцены. Также реализуем отрисовку сетки с использованием однопиксельной текстуры.

Код инициализации текстуры изображением

```
function initTexture(url) {
    const texture = gl.createTexture();
    gl.bindTexture(gl.TEXTURE_2D, texture);
    const level = 0;
    const internalFormat = gl.RGBA;
    const width = 1;
    const height = 1;
    const border = 0;
    const srcFormat = gl.RGBA;
    const srcType = gl.UNSIGNED_BYTE;
    const pixel = new Uint8Array([255, 255, 255, 255]);
    gl.texImage2D(gl.TEXTURE_2D, level, internalFormat,
        width, height, border, srcFormat, srcType,
        pixel);
    const image = new Image();
    image.onload = function() {
        gl.bindTexture(gl.TEXTURE_2D, texture);
        gl.texImage2D(gl.TEXTURE_2D, level, internalFormat,
            srcFormat, srcType, image);
        if (isPowerOf2(image.width) && isPowerOf2(image.height)) {
            gl.generateMipmap(gl.TEXTURE_2D);
        } else {
            // Размер не соответствует степени 2.
            // Отключаем MIP'ы и устанавливаем натяжение по краям
            gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S,
gl.CLAMP_TO_EDGE);
            gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T,
gl.CLAMP_TO_EDGE);
            gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
        }
    };
    image.src = url;
    return texture;
}
```

Вызов текстуры с изображением в drawScene

```
gl.activeTexture(gl.TEXTURE0);
gl.bindTexture(gl.TEXTURE_2D, sceneTexture);
gl.uniform1i(shaderProgram.samplerUniform, 0);
```

Результат работы программы представлен на рисунке 1.

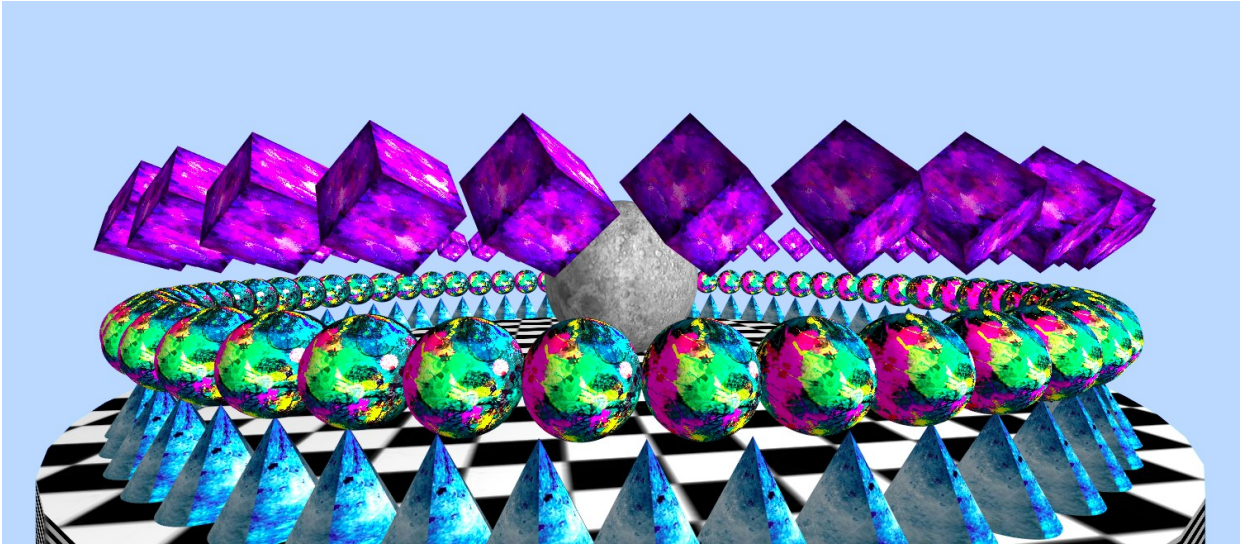

Рисунок 1 — Текстурированная сцена

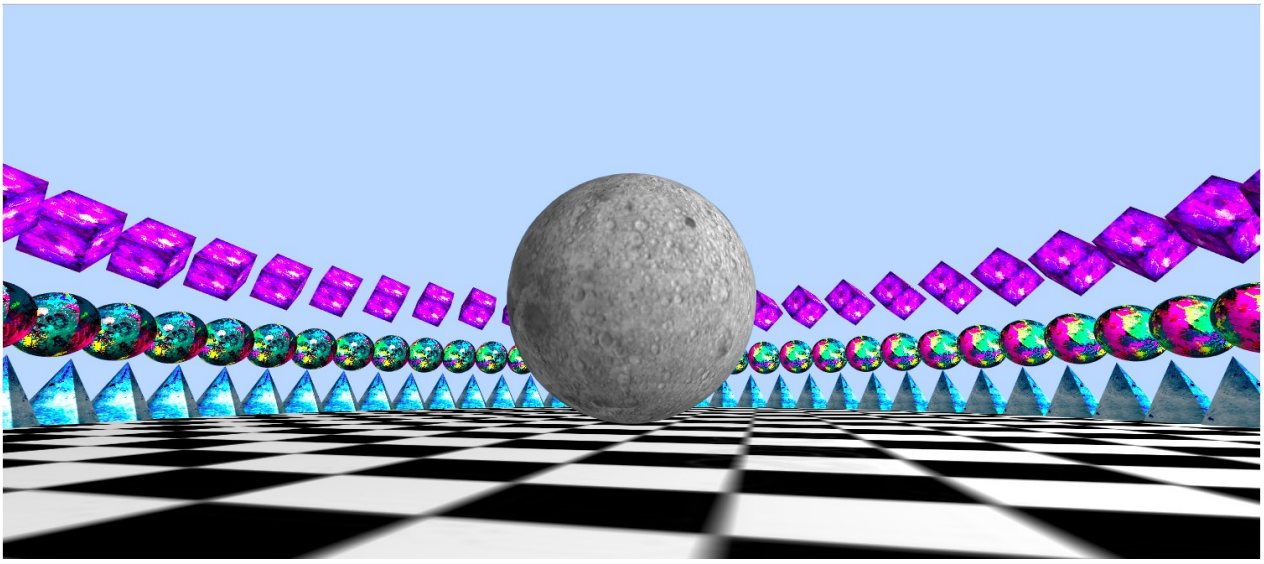

Рисунок 2 — Текстурированная сцена

Выводы.

В ходе данной работы была отредактирована сцена и сделана текстурной картой.