**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**

**"КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО"**

Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота з дисципліни

“Візуалізація графічної та геометричної інформації”

на тему “Операції з текстурними координатами”

Варіант 14

Виконала: студентка групи ТР-32мп

Макаренко Марія Ігорівна

\

Київ - 2023

1. **Завдання**

* Нанести текстуру на поверхню з практичного завдання №2.
* Реалізувати масштабування/обертання текстури (координати текстури) масштабування/обертання навколо визначеної користувачем точки – непарні варіанти реалізують масштабування, парні варіанти реалізують обертання.
* Розробити функціонал для переміщення точки вздовж простору поверхні (u,v) за допомогою клавіатури. наприклад клавіші A і D переміщують точку вздовж параметра u, а клавіші W і S переміщують точку вздовж параметра v.

1. **Теорія**

WebGL (Web Graphics Library) - це технологія для створення графічних 3D-зображень прямо в браузері без використання плагінів. WebGL базується на мові програмування JavaScript та використовує API для доступу до потужностей графічних прискорювачів пристроїв.

Основні риси WebGL:

Гнучкість:

* WebGL дозволяє вбудовувати 3D-графіку в веб-сторінки, що відкриває широкі можливості для візуалізації та інтерактивних додатків.

Відкритий стандарт:

* WebGL є відкритим стандартом, розробленим Консорціумом World Wide Web (W3C) та Корпорацією Khronos Group, що робить його широко сумісним з різними браузерами.

Використання апаратного прискорення:

* WebGL використовує потужності графічних прискорювачів, що дозволяє отримати високу продуктивність та швидку відтворення графіки.

Широкий спектр застосувань:

* Від веб-ігор до візуалізації даних, WebGL знаходить застосування в різних областях веб-розробки.

Мова програмування:

* Програми на WebGL зазвичай пишуться мовою JavaScript, що дозволяє використовувати його разом з іншими технологіями веб-розробки.

Шейдери у WebGL — це програми, написані мовою шейдерної мови (зазвичай GLSL), які використовуються для контролю процесу відтворення графіки на відеокарті. Ось кілька коротких теоретичних відомостей про шейдери у WebGL:

1. Визначення шейдера:

- Шейдер — це програма, яка використовується для обробки графічних даних на відеокарті. У WebGL є два основних типи шейдерів: вершинний (vertex shader) і фрагментний (fragment shader).

2. Вершинний шейдер:

- Вершинний шейдер відповідає за обробку кожного вершини тривимірного об'єкта. Він визначає її позицію, колір, текстурні координати та інші параметри.

3. Фрагментний шейдер:

- Фрагментний шейдер працює на рівні пікселів та визначає кінцевий колір кожного пікселя на екрані. Він відповідає за застосування текстур, визначення освітлення та інших ефектів.

4. Мова GLSL:

- Шейдери у WebGL зазвичай написані мовою GLSL (OpenGL Shading Language). Ця мова є спеціалізованою мовою програмування для шейдерів, яка дозволяє розробникам контролювати графічний конвеєр.

5. Передача даних між шейдерами:

- Вершинний та фрагментний шейдери можуть обмінюватися даними через спеціальні змінні, такі як вектори та текстурні координати, щоб забезпечити спільну роботу обох шейдерів.

6. Компіляція та використання:

- Шейдери компілюються на відеокарті під час виконання програми. Їх потім можна використовувати для обробки графічних об'єктів та створення візуальних ефектів.

Шейдери є потужним інструментом для контролю графічного відображення у WebGL, дозволяючи розробникам створювати складні та ефективні візуальні ефекти.

Накладання текстур у WebGL є важливою частиною візуалізації 3D-графіки. Текстурні координати грають ключову роль у визначенні того, як текстура розташована на поверхні об'єкта. Ось короткі теоретичні відомості:

1. Текстури у WebGL:

- Текстури — це зображення або зображення, яке може бути накладено на поверхню 3D-об'єкта. Вони додають деталі та реалізм графіці.

2. Текстурні координати:

- Текстурні координати — це пари чисел (зазвичай у вигляді (s, t)), які вказують положення пікселя на текстурі, що відповідає конкретній точці на поверхні об'єкта.

3. Діапазон текстурних координат:

- Текстурні координати лежать в діапазоні від 0 до 1. Координата (0, 0) вказує на верхній лівий край текстури, а (1, 1) — на нижній правий край.

4. Лінійна фільтрація:

- При великих текстурах або великих об'єктах може виникнути необхідність у масштабуванні текстури. Це можна виконати за допомогою лінійної фільтрації, яка враховує середнє значення пікселів.

5. Адресація текстур:

- WebGL надає можливість вибору методу адресації текстурних координат поза межами діапазону [0, 1]. Можна використовувати методи повторення, обтинання або інший.

6. Множення текстур:

- При змішуванні текстур з матеріалами або освітленням, можна використовувати операцію множення текстур. Це дозволяє створювати складні ефекти та додавати деталі до поверхні об'єкта.

Використання текстур та правильна робота з текстурними координатами дозволяють створювати реалістичні та деталізовані зображення у веб-графіці за допомогою WebGL.

1. **Деталі розробки**

Мені було надано завдання "Drop". У процесі виконання другого практичного завдання я створила програму, яка відображає поверхню у формі неперервних трикутників.

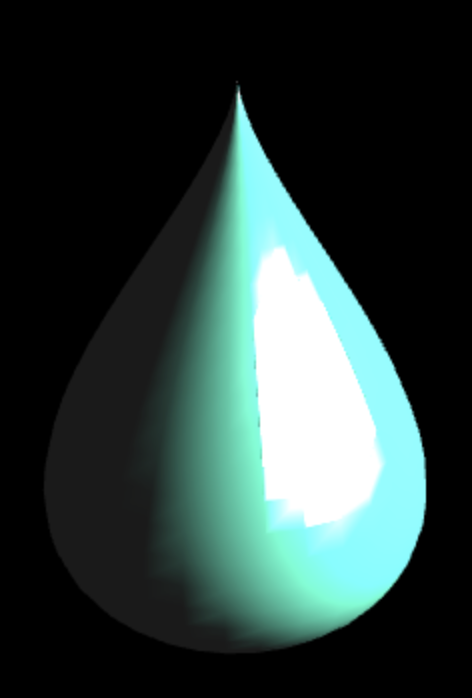


Рисунок 1. Вигляд поверхні

Я вибрала наступне зображення для розрахунково-графічної роботи. Для забезпечення сумісності з більшою кількістю операційних систем і браузерів, я вибрала зображення розміром 512х512 пікселів у форматі .jpeg.



Рисунок 2. Зображення текстури

Була застосована текстура на поверхню, попередньо підготовивши буфер текстурних координат. Кожна з цих координат відповідає елементу масиву вершинного буфера. Згідно з варіантом, текстура повинна обертатися, тому був створений відповідний параметр uniform, який визначатиме кут обертання. У програмі шейдеру для визначення кольору пікселя фігури, на яку застосовано текстуру, необхідно використовувати функцію texture2D(). Ця функція приймає об'єкт класу sampler2D як перший аргумент, який зберігає дані про зображення, та другу текстурну координату.



Рисунок 3. Текстура накладена на поверхню

Створено новий об'єкт класу Model для відображення центральної точки, щодо якої буде проводитись трансформація текстури. Графічне представлення цього об'єкта виглядає як сфера, яка знаходиться на поверхні.



Рисунок 4. Точка на текстурованій поверхні

Відповідно до вказаних параметрів, була реалізована можливість обертання текстури навколо точки на поверхні.

1. **Інструкції користувача**

Для того щоб обертати фігуру, утримуйте ліву кнопку миші та тягніть в напрямку, який вам потрібен, відносно центру.

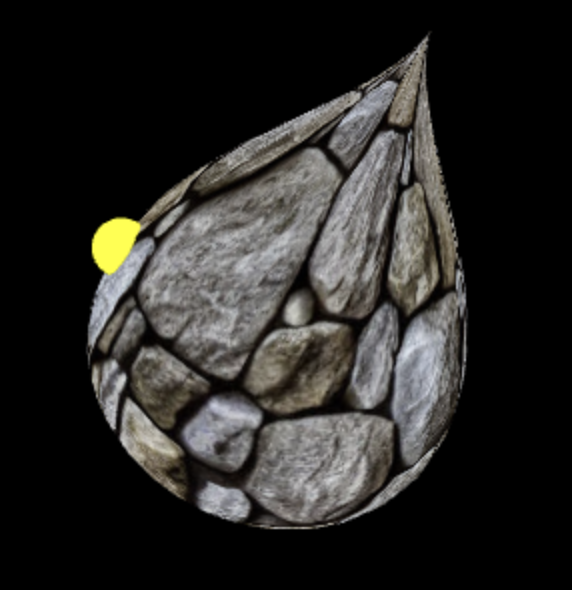
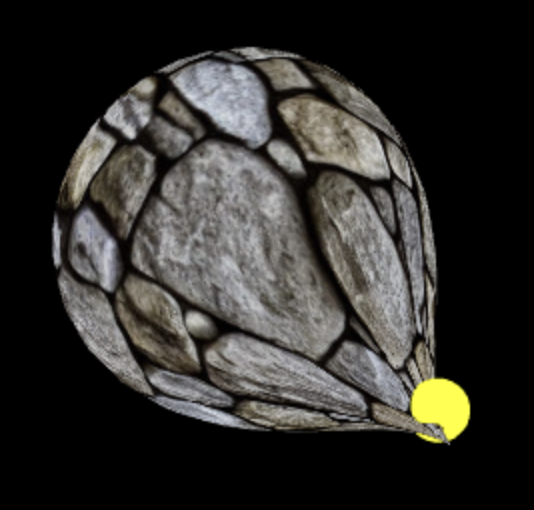


Рисунок 5. Вигляд фігури до та після обертання

Можна переміщати центр обертання за допомогою клавіш WASD. Кожне натискання зсуває цю точку по поверхні на фіксований крок. Рух відбувається до досягнення встановленого обмеження.

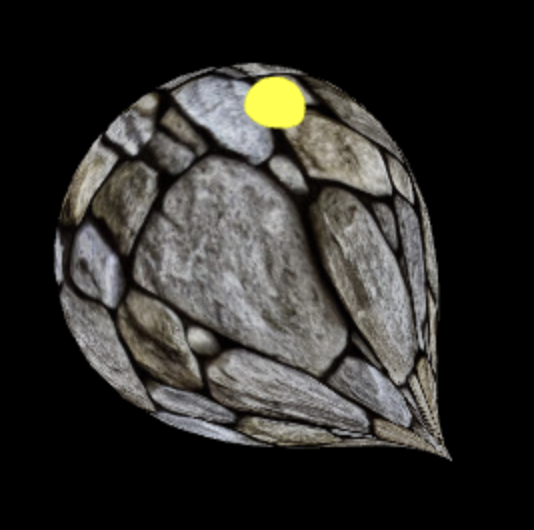
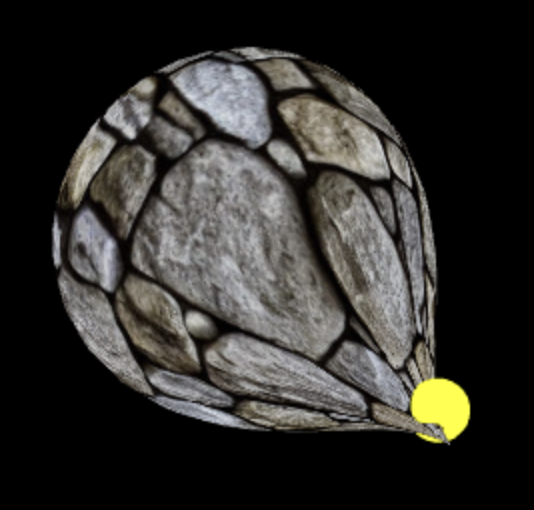


Рисунок 6. Демонстрація переміщення точки відносно нерухомої фігури

Використовуючи слайдер з підписом "Встановити кут обертання текстури", можна регулювати кут обертання відносно заданої точки на поверхні. Якщо значення кута обертання не дорівнює нулю, помітно, що при переміщенні точки текстура також зсувається. Це відбувається тому, що обертання відбувається відносно іншої точки на поверхні, яка в свою чергу взаємодіє з іншою текстурною координатою.

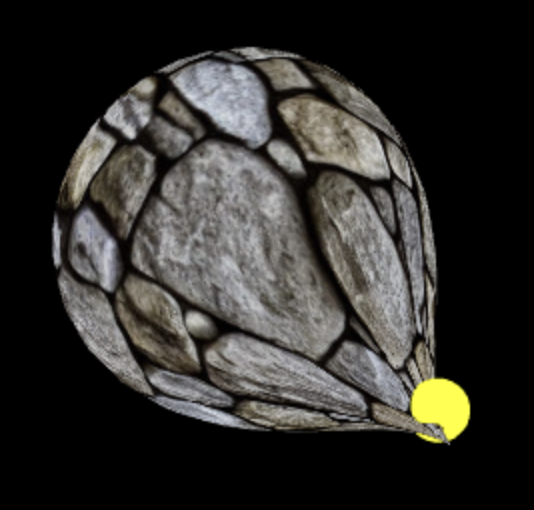


Рисунок 7. Демонстрація зміни кута обертання текстури

При перезавантаженні сторінки значення кута обертання фігури та положення точки на поверхні буде відновлено до значень за замовчуванням.

1. **Код програми**

**Код функції частини програми на javascript**

function draw() {...

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false, modelViewProjection);

gl.uniformMatrix4fv(shProgram.iNormalMatrix, false, normalMatrix);

/\* Draw the six faces of a cube, with different colors. \*/

gl.uniform4fv(shProgram.iColor, [1, 1, 0, 1]);

gl.uniform3fv(shProgram.iLightDir, [1, Math.sin(Date.now() \* 0.001), 1]);

gl.uniform1f(shProgram.iAngle,

document.getElementById('angle').value);

gl.uniform2fv(shProgram.iTexTr, ppoint);

lightModel.BufferData([0, 0, 0, 1, Math.sin(Date.now() \* 0.001), 1])

surface.Draw();

gl.uniform3fv(shProgram.iLightDir, [1, Math.sin(Date.now() \* 0.001), 180]);

// lightModel.DrawLine();

gl.uniformMatrix4fv(shProgram.iModelViewProjectionMatrix, false,

m4.multiply(modelViewProjection,

m4.translation(...getVertex(ppoint[0] \* Math.PI \* 2, ppoint[1] \* Math.PI, 0.5, 1))));

point.Draw(); }

function drawAnim() {

draw()

window.requestAnimationFrame(drawAnim)

}

**Код функції частини програми на GLSL**

**Частина коду вершинного шейдера**

vec4 t = translation(-texTr)\*vec4(texCoord,0.0,1.0);

vec4 r = rotation(angle)\*t;

vec4 tt = translation(texTr)\*r;

varTexCoord = tt.xy;

**Частина коду фрагментного шейдера**

gl\_FragColor = texture2D(tmu,varTexCoord);

if(lightDir.z>90.0){

gl\_FragColor = color; }