МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Кафедра цифрових технологій в енергетиці

Варіант - 10

Звіт з графічно-розрахункової роботи

з дисципліни «Візуалізація графічної та геометричної інформації»

Виконав:

Студент 1-го курсу магістратури Групи ТР-21мп

Кляус Кирило

Перевірив: Демчишин А. А.

Київ – 2022

**Розрахунково-графічна робота (Операції над координатами текстури)**

**Вимоги:**

* Нанести текстуру на поверхню з практичного завдання №2.
* Реалізувати масштабування текстури (координати текстури) обертання навколо визначеної користувачем
* Повинна бути можливість переміщати точку вздовж простору поверхні (i,j) за допомогою клавіатури. наприклад клавіші A і D переміщують точку вздовж параметра u, а клавіші W і S переміщують точку вздовж параметра v.

# Теоретичні відомості

Нанесення текстури — це техніка, яка використовується в комп’ютерній графіці для нанесення зображення текстури на тривимірний об’єкт або поверхню. Зображення текстури містить інформацію про кольори та візерунки, які мають бути відображені на об’єкті чи поверхні. Ця техніка дозволяє створювати більш реалістичну та детальну 3D-графіку шляхом додавання додаткового шару деталей до поверхні об’єкта.

Існує кілька різних способів застосувати накладання текстури на об’єкт. Одним із поширених методів є використання ультрафіолетового відображення, яке передбачає розгортання 3D-об’єкта в плоске 2D-представлення та призначення координат текстури (координати U та V) кожній вершині об’єкта. Потім зображення текстури наноситься на об’єкт за допомогою цих координат текстури. Інший метод полягає у використанні проекційного відображення, яке передбачає проектування зображення текстури на 3D-об’єкт з певного напрямку або точки зору.

Відображення текстур можна використовувати для різних цілей, зокрема для додавання реалістичних деталей до об’єктів, створення спеціальних ефектів і покращення загального візуального вигляду 3D-сцени. Це важливий метод у сфері комп’ютерної графіки, який широко використовується в різноманітних програмах, включаючи відеоігри, 3D-моделювання та комп’ютерно створені зображення (CGI) у фільмах і на телебаченні.

# Опис деталей реалізації

Для того щоб відобразити текстури необхідно спочатку підвантажити зображення, а після чого виконати texImage2d для текстури

Також необхідно задати текстурам параметри:

* TEXTURE\_WRAP\_S, REPEAT
* TEXTURE\_WRAP\_T, REPEAT
* TEXTURE\_MIN\_FILTER, LINEAR
* TEXTURE\_MAG\_FILTER, LINEAR

Підрахунок координат текстури для кожного вертекса реалізовано в основному циклі і здійснюється за допомогою нормалізації параметрів i j, за допомогою яких будується поверхня. Також додані глобальні параметри які зберігають координати точки масштабування та значення масштабування.

# Інструкція користувача

* W S – переміщення точки масштабування вздовж параметра j;
* A D – переміщення точки масштабування вздовж параметра i;
* Для змінення куту створено “бігунок” зі значеннями від -180 до 180

# Опис вихідного коду

Функція для створення фігури

function CreateSurfaceData() {

  let vertexList = [];

  let textureList = [];

  let splines = 100;

  let maxI = 10;

  let maxJ = 10;

  let stepI = step(maxI, splines);

  let stepJ = step(maxJ, splines);

  let getI = (i) => {

    return i / maxI;

  };

  let getJ = (j) => {

    return j / maxJ;

  };

  for (let v = 0; v < 360; v += 20) {

    for (let i = -10; i < maxI; i += stepI) {

      for (let j = -10; j < maxJ; j += stepJ) {

        let first =

          ((a +

            c \* Math.pow(Math.cos(j), 3) \* Math.cos(v) +

            c \* Math.pow(Math.sin(j), 3) \* Math.sin(v)) \*

            Math.cos(i)) /

          2;

        let second =

          ((a +

            c \* Math.pow(Math.cos(j), 3) \* Math.cos(v) +

            c \* Math.pow(Math.sin(j), 3) \* Math.sin(v)) \*

            Math.sin(i)) /

          2;

        let third =

          (b \* i -

            c \* Math.pow(Math.cos(j), 3) \* Math.sin(v) +

            c \* Math.pow(Math.sin(j), 3) \* Math.cos(v)) /

          2;

        vertexList.push(first, second, third);

        textureList.push(getI(i), getJ(j));

        first =

          ((a +

            c \* Math.pow(Math.cos(j + stepJ), 3) \* Math.cos(v) +

            c \* Math.pow(Math.sin(j + stepJ), 3) \* Math.sin(v)) \*

            Math.cos(i + stepI)) /

          2;

        second =

          ((a +

            c \* Math.pow(Math.cos(j + stepJ), 3) \* Math.cos(v) +

            c \* Math.pow(Math.sin(j + stepJ), 3) \* Math.sin(v)) \*

            Math.sin(i + stepI)) /

          2;

        third =

          (b \* (i + stepI) -

            c \* Math.pow(Math.cos(j + stepJ), 3) \* Math.sin(v) +

            c \* Math.pow(Math.sin(j + stepJ), 3) \* Math.cos(v)) /

          2;

        vertexList.push(first, second, third);

        textureList.push(getI(i + stepI), getJ(j + stepJ));

      }

    }

  }

  return { vertexList, textureList };

}

Код вертексного та фрагментного шейдера







