

Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»
Інститут атомної та теплової енергетики
Кафедра цифрових технологій в енергетиці

Розрахунково-графічна робота

З дисципліни «Візуалізація графічної та геометричної інформації»
Варіант 12

Виконав: Маліков І.С.
Студент групи ТР-21мп

Київ 2022

1. Завдання

Тема роботи: Операції над текстурними координатами

Вимоги:

- Накласти текстуру на поверхню отриману в результаті виконання лабораторної роботи №2.
- Імплементувати масштабування або обертання текстури(текстурних координат) згідно з варіантом: непарні - масштабування, парні - обертання.
- Запровадити можливість переміщення точки відносно якої відбувається трансформація текстури по поверхні за рахунок зміни параметрів в просторі текстури. Наприклад, клавіші A та D для переміщення по осі абсцис, змінюючи параметр u текстури, а клавіші W та S по осі ординат, змінюючи параметр v .

2. Теоретичні відомості

Текстурування є дуже важливою частиною процесу 3D-моделювання. Усі дрібніші візуальні характеристики у 3D-моделюванні, такі як зморшки та окремі нитки килима, є продуктом текстури, нанесеної 3D-художником.

Зазвичай створювані 3D-моделі мають стандартний сірий колір програми. Щоб додати кольори, малюнки та текстури, 2D-фотографії потрібно розмістити на 3D-моделях. Додавання кольорів або властивостей поверхні та матеріалу до 3D-моделі вимагає ще одного кроку вперед у процесі 3D-моделювання, тобто 3D-текстурування. Цей підхід часто призводить до повного кольору та властивостей поверхні 3D-моделі.

Щоб почати процес 3D-текстурування, необхідно спочатку розгорнути модель, що, по суті, те саме, що розгортання 3D-сітки. Коли художники-фактуристи отримують готові моделі від відділу 3D-моделювання, вони створюють UV-карту для кожного 3D-об'єкта. UV-карта — це плоске зображення поверхні 3D-моделі, яке використовується для швидкого накладання текстур. Прямо пов'язуючи 2D-зображення (текстуру) з вершинами багатокутника, UV-відображення може допомогти обернути 2D-зображення (текстуру) навколо 3D-об'єкта, а згенеровану карту можна використовувати безпосередньо в процесі текстурування та затінення.

Більшість програмних систем 3D мають кілька інструментів або підходів для розгортання 3D-моделей. Коли справа доходить до створення UV-карт, це питання особистих уподобань. Якщо ви не збираєтеся використовувати процедурні текстури, майже завжди потрібно розгортати 3D-модель у компоненті текстурування. Це текстури, створені за допомогою математичних методів (процесів), а не безпосередньо записаних даних у 2D або 3D.

3. Виконання завдання

В ході другої лабораторної роботи було створено поверхню під назвою «Поверхня Монжа з циліндричною поверхнею директриси та синусоїдою як меридіаном». Отриману поверхню з освітленням можна побачити на рисунку 3.1.

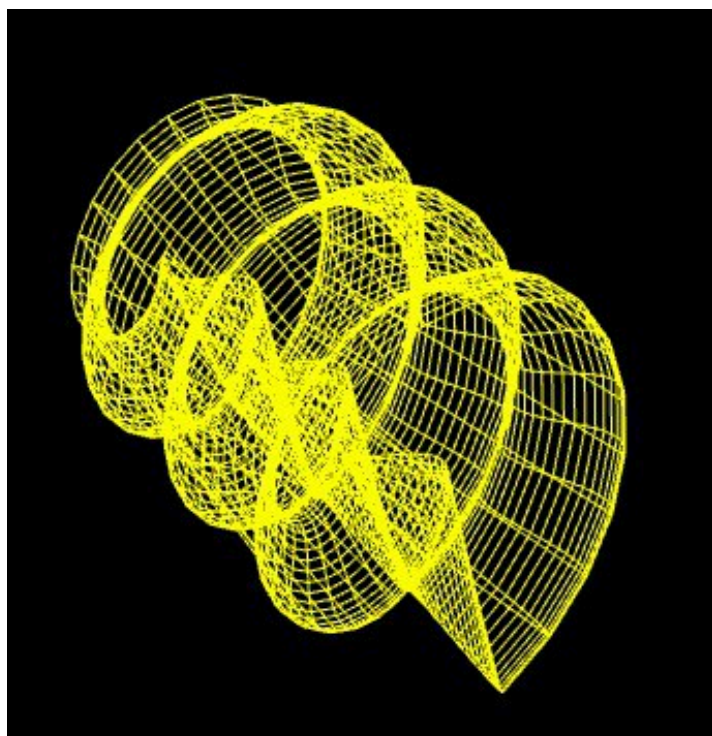


Рис. 3.1 Поверхня Монжа з циліндричною поверхнею директриси та синусоїдою як меридіаном

Для текстури було обрано картинку з інтернету формату «jpg». Після чого завантажив її на github, щоб в подальшому використовувати посилання на неї і не стикатися з проблемою Cross-Origin Resource Sharing policy.

В графічному редакторі було налаштовано розмір картинки так, щоб ширина і висота були рівні, а також, аби сторона мала розмір 2^n в пікселях.

З метою накладання текстури на поверхню, в першу чергу було створено декілька змінних в коді шейдера. Після чого були створення посилання на них в коді програми. Були також створені функції для генерації бUVера даних текстури.

Обрану картинку можна побачити на рисунку 3.2.



Рис. 3.2 Обрана текстура

Поверхню з накладеною текстурою можна побачити на рисунку 3.3.



Рис. 3.3 Поверхня Монжа з циліндричною поверхнею директриси та синусоїдою як меридіаном з накладеною текстурою

Для відображення умовної точки відносно якої буде виконватися трансформація текстури, в класі моделі було створено відповідну функцію. Замість відображення точки було прийнято рішення відображати сферу, адже працюємо в 3д-просторі. Для відображення сфери необхідно було створити функцію, яка б створювала геометрію для неї. Модель з умовною точкою зображено на рисунку 3.4.



Рис. 3.4 Поверхня з умовною точкою

Для роботи з текстурою було створено ще кілька змінних в коді шейдера: обертання текстури, розташування умовної точки в (u,v) координатах, змінну для розташування сфери на відповідне місце поверхні в 3д-просторі.

Для реалізації переміщення точки по поверхні та обертання текстури було додано відповідні функції на відповідні вхідні дані від користувача.

4. Вказівки користувачу

Користувач може керувати переміщенням умовної точки по поверхні, обертанням текстури відносно умовної точки, а також орієнтацією поверхні в просторі. При чому останні два пункти здійснюються в один і той же спосіб.

Переміщення умовної точки реалізовано за допомогою введення з клавіатури (рис. 4.1): клавіші W та S здійснюють переміщення точки за параметром v в додатньому та від'ємному напрямках відповідно, клавіші A та D здійснюють переміщення точки за параметром u у від'ємному та додатньому напрямках відповідно.

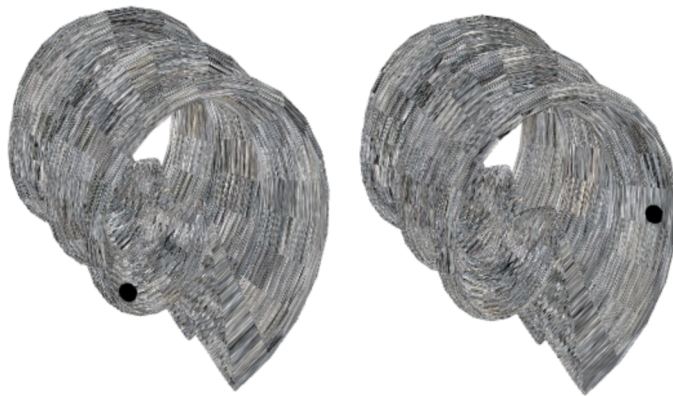


Рис. 4.1. Переміщення умовної точки

Орієнтація поверхні в просторі, а також трансформація текстури(рисунок 4.2) здійснюється за допомогою введення з миші: необхідно затиснути лівою клавiшею миші у області відображення поверхні та потягнути в будь-яку сторону. Варто зауважити, що обертання текстури залежить від переміщення миші по горизонталі, тобто, при зміщенні затиснутої мишу тільки вгору або тільки вниз буде здійснюватись лише зміна орієнтації поверхні в просторі(рисунок 4.3)

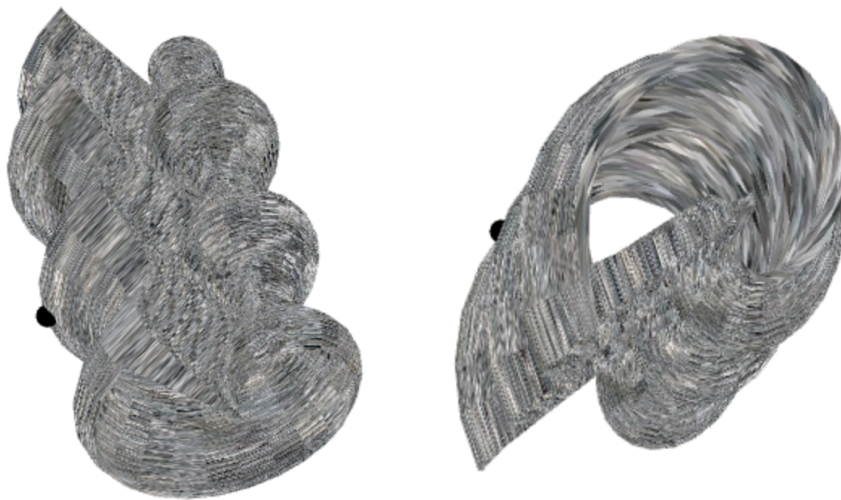


Рис. 4.2. Трансформація текстури

На рисунку 4.3 можна помітити що точка та текстура залишились на одному і тому самому місці відносно поверхні. Змінилась лише орієнтація поверхні в просторі.

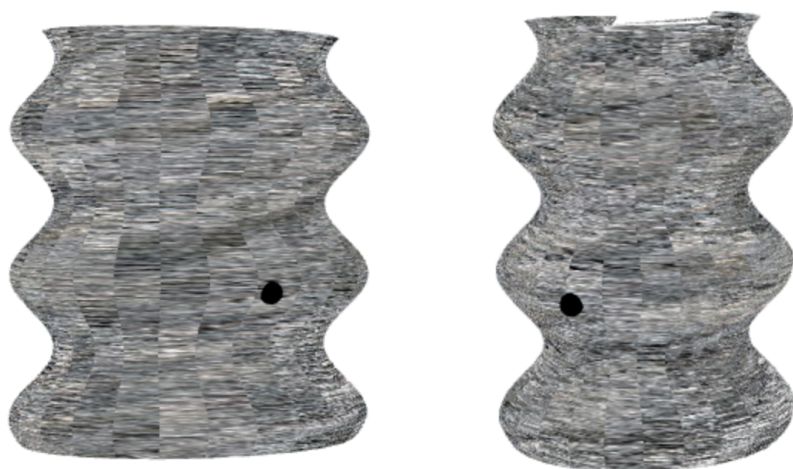


Рис. 4.3. Лише орієнтація поверхні в просторі

5. Код

```
function LoadTexture() {
  let texture = gl.createTexture();
  gl.bindTexture(gl.TEXTURE_2D, texture);
  gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
  gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MAG_FILTER, gl.LINEAR);

  const image = new Image();
  image.crossOrigin = 'anonymus';

  image.src =
    "https://raw.githubusercontent.com/IvanMalikov/vggi/CGW/CGWv12/textu
    re.jpeg";
  image.onload = () => {
    gl.bindTexture(gl.TEXTURE_2D, texture);
    gl.texImage2D(
      gl.TEXTURE_2D,
      0,
      gl.RGBA,
      gl.RGBA,
      gl.UNSIGNED_BYTE,
      image
    );
    draw()
  }
}

onmousemove = (e) => {
  angle = map(e.clientX, 0, window.outerWidth, 0, Math.PI)
  draw()
};
window.onkeydown = (e) => {
  switch (e.keyCode) {
    case 87:
```

```
        userPoint.y -= 0.01;
        break;
    case 83:
        userPoint.y += 0.01;
        break;
    case 65:
        userPoint.x += 0.01;
        break;
    case 68:
        userPoint.x -= 0.01;
        break;
}
userPoint.x = Math.max(0.51, Math.min(userPoint.x, 0.999))
userPoint.y = Math.max(0.01, Math.min(userPoint.y, 0.999))
draw();
}
```