

**Міністерство освіти і науки України Національний технічний  
університет України «Київський політехнічний інститут імені  
Ігоря Сікорського» Факультет інформатики та обчислювальної  
техніки Кафедра обчислювальної техніки**

**Лабораторна робота №1**  
з дисципліни  
«Комп'ютерна графіка та мультимедіа»

Виконав:

студент групи ІП-05

Гапій Денис Едуардович

номер у списку групи: 5

Перевірив:

Родіонов П. Ю.


Київ 2022

## Тема: «Вступ до комп'ютерної графіки»

**Мета:** навчитися створюватися та редагувати шейдери, використовуючи платформу ShaderToy та бібліотеку WebGL.

### Завдання 1:

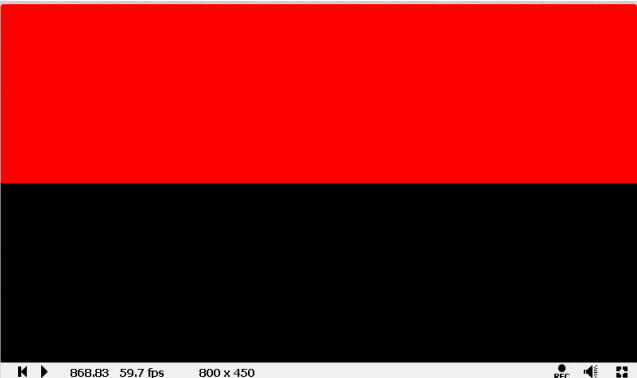
Приклади з методички:

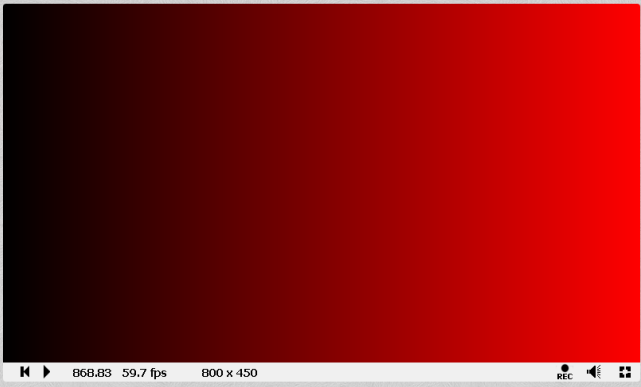
Код	<pre>1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec4 solidRed = vec4(1.0, 0.0, 0.0, 1.0); 4     fragColor = solidRed; 5 }</pre>
Шейдер	
Нотатки	1.0,0.0,0.0 - шейдер суцільного червоного кольору

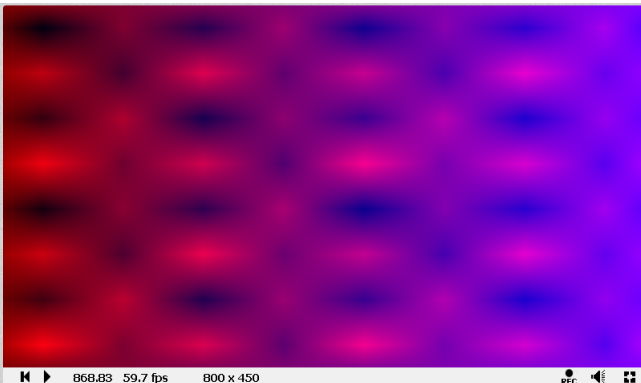
  

Код	<pre>1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec2 xy = fragCoord.xy; // get curr pxl coords 4     vec4 solidColor = vec4(0.0, 0.5, 0.75, 1.0); 5     if (xy.y &lt; 225.0) 6     { 7         solidColor = vec4(1.0, 0.75, 0.0, 1.0); 8     } 9     fragColor = solidColor; 10 }</pre>
-----	--

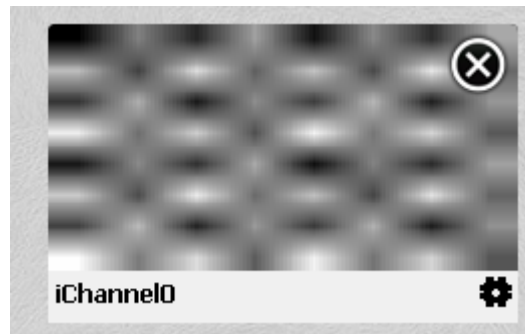
Шейдер	
Нотатки	<p>маємо зміну блакитного кольору на жовтий, якщо поточний піксель по висоті нижче 250 пікселів</p>

Код	<pre> 1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec2 xy = fragCoord.xy; // get curr pxl coords 4     xy.x = xy.x / iResolution.x; 5     xy.y = xy.y / iResolution.y; 6     vec4 solidRed = vec4(0.0, 0.0, 0.0, 1.0); 7     if (xy.y &gt; 0.5) 8     { 9         solidRed.r = 1.0; 10    } 11    fragColor = solidRed; 12 }</pre>
Шейдер	
Нотатки	<p>ділення на iResolution дає нам змогу коректно отримати централь координати шейдеру, незалежно від розмірів вікна.</p> <p>маємо зміну чорного кольору на чорний, якщо поточний піксель по висоті вище половини екрану</p>

Код	<pre> 1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec2 xy = fragCoord.xy; // get curr px1 coords 4     xy.x = xy.x / iResolution.x; 5     xy.y = xy.y / iResolution.y; 6     vec4 solidRed = vec4(0.0, 0.0, 0.0, 1.0); 7     solidRed.r = xy.x; 8     fragColor = solidRed; 9 } </pre>
Шейдер	
Нотатки	<p>Так як ділення на iResolution обмежело координати в діапазоні від 0 до 1 (так само як і діапазон для кольорів у vec4).</p> <p>Прирівнявши координати по X та параметр Червоного - маємо зміну чорного кольору на чорний за допомогою градієнту</p>

Код	<pre> 1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec2 xy = fragCoord.xy / iResolution.xy; 4     vec4 texColor = texture(iChannel0,xy); 5     texColor.b = xy.x; 6     fragColor = texColor; 7 } </pre>
Шейдер	

Нотатки

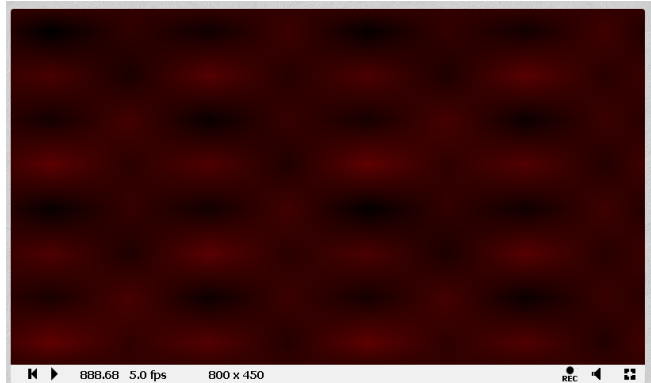


текстура каналу  
+градієнт по Голубому параметру

Код

```
1 void mainImage( out vec4 fragColor, in vec2 fragCoord )
2 {
3     vec2 xy = fragCoord.xy / iResolution.xy;
4     vec4 texColor = texture(iChannel0,xy);
5     texColor.r *= abs(sin(iTime));
6     texColor.g *= abs(cos(iTime));
7     texColor.b *= abs(sin(iTime) * cos(iTime));
8     fragColor = texColor;
9 }
```

Шейдер

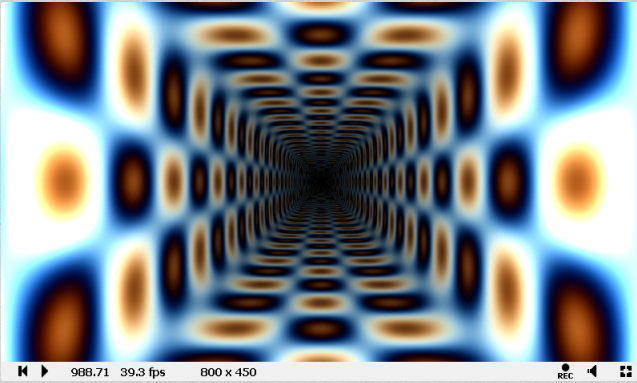


Нотатки

завдяки доданій змінній iTime та модулю від синусної/косинусної функцій ми додаємо нашому шейдеру ефект затемнення протягом часу (і обернено відповідно теж діє)

Код

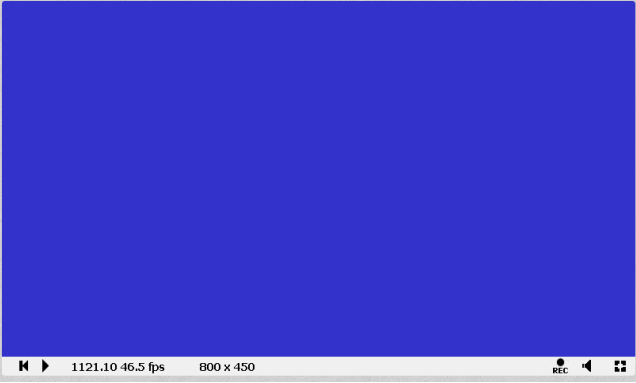
```
1 void mainImage( out vec4 fragColor, in vec2 fragCoord )
2 {
3     vec2 p = (-iResolution.xy + 2.0*fragCoord)/iResolution.y;
4     float a = atan(p.y,p.x);
5     float r = pow(pow(p.x*p.x,4.0) + pow(p.y*p.y,4.0), 1.0/8.0 );
6     vec2 uv = vec2( 1.0/r + 0.2*iTime, a );
7     float f = cos(12.0*uv.x)*cos(6.0*uv.y);
8     vec3 col = 0.5 + 0.5*sin( 3.1416*f + vec3(0.0,0.5,1.0) );
9     col = col*r;
10    fragColor = vec4( col, 1.0 );
11 }
```

Шейдер	
Нотатки	<p>віддзеркаливши / продублювавши координати, потім 'нахиливши' пікселі до центру екрана (замінивши параметр <math>x</math> (який в подальшому застосовується для значення '<math>y</math>' у нашому векторі) до прикладу на 1, то вийде коридор де квадрати будуть розміщуватися поступово наближаючись до них); за допомогою певного патерну та множення на час досягли візуалізації квадратів за цим паттерном, відповідно, за певною палітрою налаштували наш колір та затемнили його до умовного 'центру'</p>

## Завдання 2:

Створіть шейдер, що заливає довільним кольором екран.

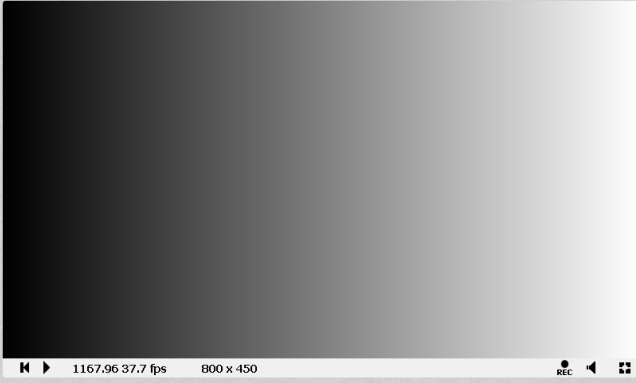
Код	<pre> 1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     fragColor = vec4( 0.2, 0.2, 0.8, 1.0 ); 4 } </pre>
-----	--

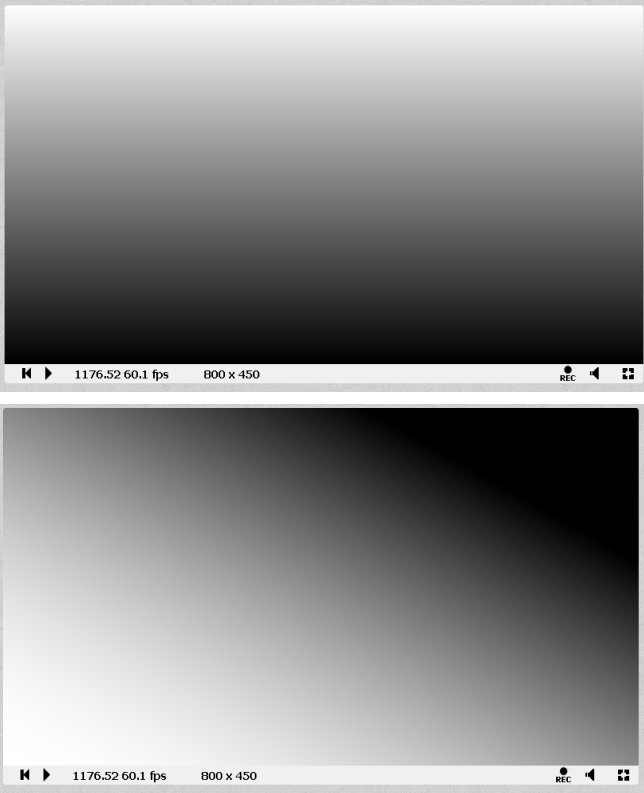
Шейдер	
Нотатки	-

**Завдання 3:**

**В тексті роботи змініть приклад з горизонтальним градієнтом на вертикальний**

**або діагональний. Також можна додати нові кольори.**

Код	<div>► Shader Inputs</div> <pre>1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     { 4         vec2 xy = fragCoord.xy / iResolution.xy; 5         vec4 solidColor = vec4(xy.x, xy.x, xy.x, 1.0); 6         fragColor = solidColor; 7     } 8 }  void mainImage( out vec4 fragColor, in vec2 fragCoord ) {     vec2 xy = fragCoord.xy / iResolution.xy;     vec4 solidColor = vec4(xy.y, xy.y, xy.y, 1.0);     fragColor = solidColor; }  void mainImage( out vec4 fragColor, in vec2 fragCoord ) {     vec2 xy = fragCoord.xy / iResolution.xy;     vec4 solidColor = vec4(cos(xy.x + xy.y), cos(xy.y+xy.x), cos(xy.x+xy.y), 1.0);     fragColor = solidColor; }</pre>
Шейдер	

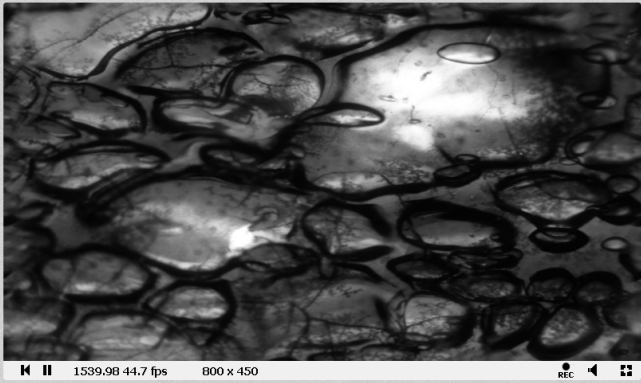
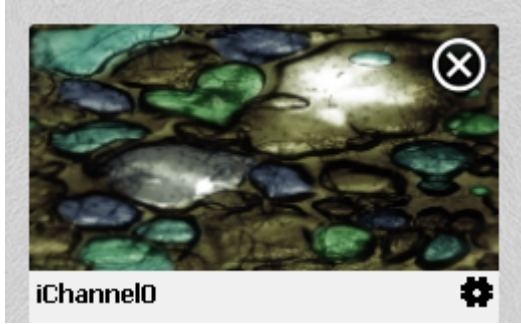
	
<p>Нотатки</p>	<p>горизонтальний / вертикальний / діагональний</p>

## Завдання 4:

Спробуйте написати шейдер, що робить зображення чорно-білим.

<p>Код</p>	<pre> 1  void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2  { 3      vec2 xy = fragCoord.xy / iResolution.xy; 4      vec4 texColor = texture(iChannel0.xy); 5      float ftiB = sqrt((texColor.r * texColor.r + texColor.g * texColor.g + texColor.b * texColor.b) / 3.0); 6      texColor.r = ftiB; 7      texColor.g = ftiB; 8      texColor.b = ftiB; 9 10     fragColor = texColor; 11 }</pre>
------------	--

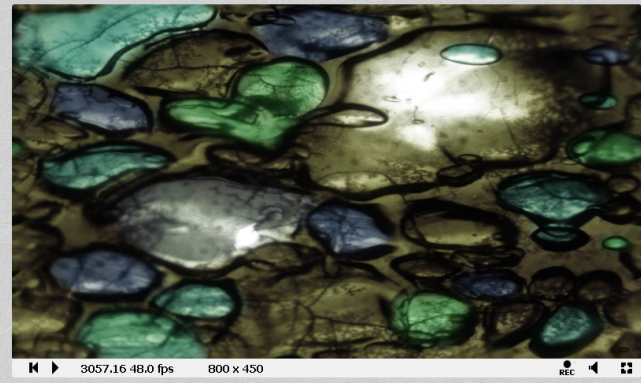


Шейдер	
Нотатки	 <p>iChannel0</p> <p>colorized channel texture</p>

# Завдання 5:

Спробуйте написати шейдер, який міняє кольорове зображення на чорно-біле

та знову робить його кольоровим.

Код	<pre> 1 void mainImage( out vec4 fragColor, in vec2 fragCoord ) 2 { 3     vec2 xy = fragCoord.xy / iResolution.xy; 4     vec4 texColor = texture(iChannel0, xy); 5     float tAurg = sqrt(texColor.r * texColor.r + texColor.g * texColor.g + texColor.b * texColor.b); 6     float fAIB = (tAurg / 3.0); 7     vec4 newColor = texColor; 8     if (sin(iTime) &gt; 0.0) { 9         newColor = vec4(fAIB, fAIB, fAIB, 1.0); 10    } 11 12    fragColor = newColor; 13 } 14 </pre>
Шейдер	

Нотатки	Кольорове зображення змінюється на Чорно-Біле відносно часу (по синусоїді)
---------	--

## Висновок:

Виконуючи цю лабораторну роботу, я дізнався основні теоретичні поняття, по типу: Шейдер, GLSL, Графіка, Програмовані шейдери і тд. Опанував базовий функціонал платформи [ShaderToy](#) та основні методи / синтаксис мови GLSL.

Також використовуючи наведені викладачем приклади та трохи експериментуючи зі зміною значень координат / кольорів, бавлячись з додаванням у шейдер залежності від часу - вдалось реалізувати чимало різноманітних моделей забарвлення шейдеру.

Дізнався про нові ресурси: веб-платформу <https://gamedevelopment.tutsplus.com/> та цікавий ютуб канал (The Art of Code). Дякую ;)

Не менш корисним було почути, а потім і прочитати про алгоритм Cosine Wave Pattern.