Instrukcja Laboratoryjna: Przetwarzanie Obrazu w Shaderach

Laboratorium Grafiki Komputerowej

5 maja 2026

1 Wstęp Teoretyczny

Większość filtrów obrazu w shaderach realizowana jest za pomocą **splotu (convolution)**. Proces ten polega na obliczeniu nowego koloru piksela na podstawie średniej ważonej kolorów jego sąsiadów. Wagami steruje tzw. **jądro splotu** (*kernel*) – macierz o rozmiarze $N \times N$.

1.1 Filtr Dolnoprzepustowy (Low-Pass)

Przepuszcza sygnały o niskiej częstotliwości, tłumiąc ostre krawędzie i detale. Efektem jest **rozmycie (blur)**. Przykładem jest filtr pudełkowy (*Box Blur*), gdzie każda komórka macierzy ma tę samą wartość:

$$K = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

1.2 Filtr Górnoprzepustowy (High-Pass)

Przepuszcza sygnały o wysokiej częstotliwości. Służy do **wyostrzania** obrazu lub **wykrywania krawędzi**. Suma wag w takim filtrze często wynosi 1 (dla wyostrzania) lub 0 (dla czystej detekcji krawędzi). Przykładowy kernel wyostrzający:

$$K = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

2 Zadania do wykonania

2.1 Zadanie 1: Przygotowanie offsetów próbowania

W shaderze fragmentowym należy zdefiniować tablicę przesunięć, aby pobrać wartości kolorów z otoczenia bieżącego piksela.

```
float offset = 1.0 / 300.0; // Zale ne od rozdzielczo ci
vec2 offsets[9] = vec2[](
    vec2(-offset,  offset), vec2(0.0,  offset), vec2(offset,  offset),
    vec2(-offset,  0.0),    vec2(0.0,  0.0),    vec2(offset,  0.0),
    vec2(-offset, -offset), vec2(0.0, -offset), vec2(offset, -offset)
);
```

Listing 1: Definicja offsetów w GLSL

2.2 Zadanie 2: Implementacja splotu

Zaimplementuj pętlę sumującą kolory z sąsiednich pikseli pomnożone przez wartości wybranego jądra.

```
float kernel[9] = float[](
    1.0/9.0, 1.0/9.0, 1.0/9.0,
    1.0/9.0, 1.0/9.0, 1.0/9.0,
    1.0/9.0, 1.0/9.0, 1.0/9.0
);

void main() {
    vec3 sampleTex[9];
    for(int i = 0; i < 9; i++) {
        sampleTex[i] = vec3(texture(screenTexture, TexCoords.st +
   offsets[i]));
    }
    vec3 col = vec3(0.0);
    for(int i = 0; i < 9; i++)
        col += sampleTex[i] * kernel[i];

    FragColor = vec4(col, 1.0);
}
```

Listing 2: Pętla splotu w shaderze fragmentowym