УДК 004.92

Создания живого узора на языке шейдера

Черкашин Александр Михайлович
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В данной статье описан процесс создания узора. В процессе работы использовался язык шейдера GLSL. В результате было получено изображение живого узора.
Ключевые слова: шейдер, OpenGL Shading Language, GLSL, узор.

Creating a live pattern in a shader language

Cherkashin Alexander Mihailovich
Sholom-Aleichem Priamursky State University
Student

Abstract
This article describes the process of creating a pattern. In the process of work, the GLSL shader language was used. As a result, images of living patterns were displayed.
Keywords: shader, OpenGL Shading Language, GLSL, pattern.

1 Введение

1.1 Актуальность исследования

Данная статья описывает возможности написания программы на языке шейдера для рисования узора.

1.2 Цель исследования

Целью работы является написание программы на языке шейдера для рисования узора.

1.3 Обзор исследований

К. Гдащиец описывает математические формулы для создания абстрактных фрактальных узоров и комплексных плоскостей, комбинации по нахождению корней для создания фрактальных паттернов [1]. В работе С.Густавсон описывает процедурные текстуры, которые вычисляется на лету во время рендеринга с использованием графических процессоров в GLSL [2]. М. Газзиро описывает интерактивное и интуитивное понятное приложение, которое моделирует процесс мраморизации в цифровом виде в режиме реального времени [3].

2. Рабочий процесс

В данной статье использовалась программа, которая написана на языке шейдера OpenGL Shading Language (GLSL) и glslViewer [4].

Листинг 2.1. Исходный код программы для создания эффекта капли.

```
#version 330
#ifdef GL_ES
precision mediump float;
#endif
uniform vec2 u_resolution;
uniform float u_time;
float rand(vec2 n) {
	return fract(sin(dot(n, vec2(12.9898, 4.1414))) * 43758.5453);
}
float noise(vec2 p){
	vec2 ip = floor(p);
	vec2 u = fract(p);
	u = u*u*(3.0-2.0*u);

	float res = mix(
		mix(rand(ip),rand(ip+vec2(1.0,0.0)),u.x),
		mix(rand(ip+vec2(0.0,1.0)),rand(ip+vec2(1.0,1.0)),u.x),u.y);
	return res*res;
}
float fbm(vec2 uv, float frequency, float amplitude, float value, int octaves) {
	for(int i = 0; i < octaves; i++) {
		value += amplitude * noise(frequency * uv);
		amplitude *= 0.5;
		frequency *= 2.0;
	}
	return value;
}
float pattern( in vec2 uv, out vec4 d) {
	uv += sin(vec2(0.12,0.14)*u_time * 0.1 + length(uv)) * 0.3;
	vec2 q = vec2( fbm( uv + vec2(0.0,0.0) , 3.0, 0.5, 0.0, 5 ),
			fbm( uv + vec2(5.2,1.3) , 3.0, 0.5, 0.0, 5 ) );
	q += sin(vec2(0.12,0.14)*u_time + length(uv)) * 0.2;
	vec2 r = vec2( fbm( uv + 4.0*q + vec2(1.7,9.2) , 3.0, 0.5, 0.0, 5 ),
			fbm( uv + 4.0*q + vec2(8.3,2.8) , 3.0, 0.5, 0.0, 5 ) );
	d = vec4(q, r);
	return fbm( uv + 1.0*r + q , 3.0, 0.5, 0.0, 5 );
}
float fn(vec2 uv) {
	vec4 q;
	float a = pattern(uv, q);
	return a;
}
```

```
43 void main (void) {
44         vec2 uv = (gl_FragCoord.xy / u_resolution.xy * 2.0) - 1.0;
45         vec3 col = vec3(1.0);
46         float e = 2.0 / u_resolution.y;
47         vec4 p;
48         float c = pattern(uv, p);
49         col = mix( col, vec3(0.1,0.2,0.9), dot(p.zw, p.zw) );
50         col = mix( col, vec3(0.9,0.5,0.9), 0.2 + 0.5*p.y*p.y );
51         col = mix( col, vec3(0.4,0.2,0.4), 0.5 * smoothstep(1.2,1.3,abs(p.z) +
   abs(p.w)) );
52         col = clamp( col*c*2.0, 0.0, 1.0 );
53         vec3 nor = normalize(vec3( fn(uv + vec2(e, 0.0)) - fn(uv),
54                                    2.0*e,
55                                    fn(uv+vec2(0.0,e)) - fn(uv) ) );
56         vec3 lig = normalize(vec3( 0.2, 0.9, -0.4 ));
57         float dif = clamp(0.6+0.6 * dot( nor, lig ), 0.0, 1.0);
58         vec3 lin = vec3(0.90,0.95,0.95) * (nor.y * 0.5 + 0.23) +
   vec3(0.75,0.95,0.95) * dif;
59         col = col * lin;
60         gl_FragColor = vec4(col, 1.0);
61 }
```

В строке 1 (листинг 2.1) выбрана версия шейдера 3.30.

В строках 5 — 6:

u_resolution - разрешение экрана,

u_time — время в секундах,

В строках 7 — 19:

Функция rand генерирует шум, a noise создает клеточный шум, код взят из [5].

В строках 20 — 27, функция fBM генерирует ландшафт [6].

В строках 28 — 37, функция рисует узор [7].

В строках 38 — 42, функция выводит однокомпонентный цвет узора.

В строках 44 — 46, переменная uv — экранное пространство от -1 до 1. col — основной цвет. e — размер (высота) в пикселях.

В строке 47. p — дополнительный узор, полученный из функции pattern.

В строке 48 получаем узор.

В строках 49 — 52 задаем цвет узора.

В строках 53 - 55 создаем рельеф.

В строке 56 задаем направления источника света.

В строках 57 — 58 задаем цвет рельефа.

В строке 59 умножаем цвета узора и рельефа.

В строке 60 выводим изображение.

glslViewer main.frag -w 800 -h 600

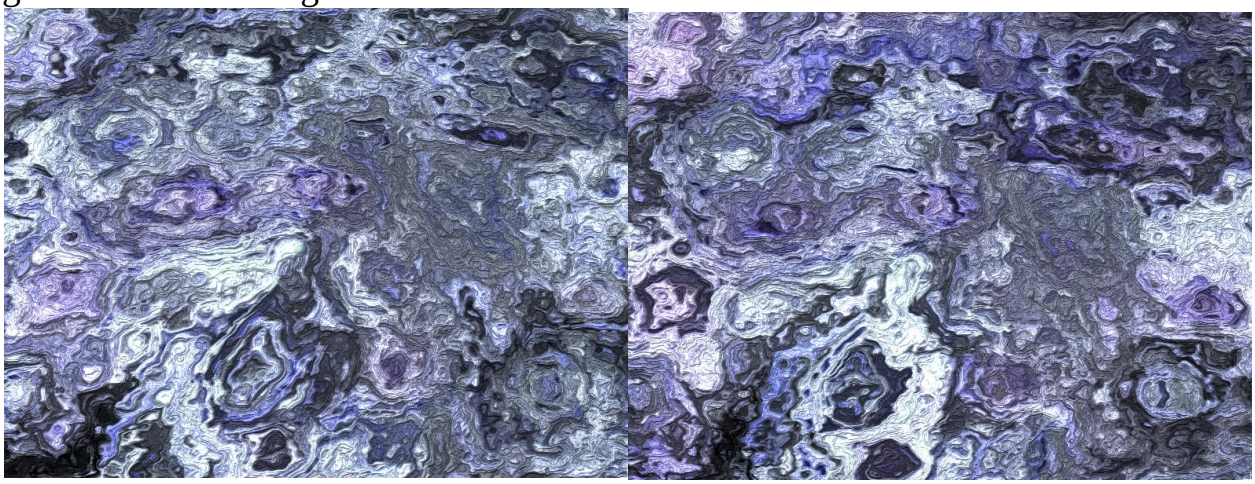

Рисунок 2.1. Результаты работы программы

3 Выводы

В результате работы была написана программа на языке шейдера для получения узора с использования функция fBM.

Библиографический список

1. Gdawiec K. Fractal patterns from the dynamics of combined polynomial root finding methods //Nonlinear Dynamics. 2017. Т. 90. №. 4. С. 2457-2479.
2. Gustavson S. Procedural textures in GLSL // OpenGL Insights: OpenGL, OpenGL ES and WebGL community experiences. CRC Press, 2012. С.105-119.
3. Gazziro M. et al. A Computational Method for Interactive Design of Marbling Patterns //2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames). IEEE, 2018. С. 1-109.
4. patriciogonzalezvivo/glslViewer: Console-based GLSL Sandbox for 2D/3D shaders shaders. // GitHub URL: https://github.com/patriciogonzalezvivo/glslViewer (дата обращения: 2022-09-03).
5. GLSL Noise Algorithms GitHub // GitHub Gist URL: https://gist.github.com/patriciogonzalezvivo/670c22f3966e662d2f83 (дата обращения: 2022-09-03).
6. Fractal Brownian Motion (fBM) - Godot Shaders // Godot Shaders URL: https://godotshaders.com/snippet/fractal-brownian-motion-fbm/ (дата обращения: 2022-09-03).
7. Inigo Quilez :: computer graphics, mathematics, shaders, fractals, demoscene and more // Inigo Quilez URL: https://iquilezles.org/articles/warp/ (дата обращения: 2022-09-03).