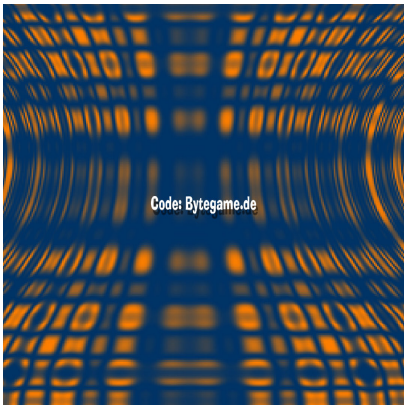

Shader-Effekt: Wellen für den Hintergrund

Description

Du bist auf der Suche nach einem Hintergrundeffekt für Credits oder Hauptmenü? Dann ist dieser Shader-Effekt womöglich perfekt für dein Retro-Game.

Das Titelbild sieht womöglich etwas seltsam aus, aber in Bewegung macht der Effekt echt etwas her:

https://www.bytegame.de/wp-content/uploads/2024/02/Wave-Credits.mp4

Du kannst den Effekt gut an Deine Bedürfnisse anpassen. So lässt sich bspw. der Zoom verändern, die Bewegung und natürlich die Farben und Geschwindigkeit. Im Video habe ich über *GUI-Draw* einen Text drüber gelegt, damit du sehen kannst, wie es als Credit-Hintergrund wirken würde. Doch kommen wir gleich zum Code.

Shader-Code

```
varying vec2 v_vTexcoord;

uniform float time;          // Zeitvariable
uniform vec2 resolution;     // Bildschirmauflösung

void main()
{
    vec2 uv = ceil((v_vTexcoord / resolution.xy - 0.5) * 255.0) / 255.0 + cos(
    uv.x *= 1.0 + sin(time + uv.y * 2.0) * sin(time + uv.y * 2.0) * sin(time +
    uv *= sin(uv * 45.0);

    // Zusätzliche Verzerrung
    uv.x = uv.x + sin(time * 0.5 + uv.y * 2.0) * 0.1;
    uv.y = uv.y + sin(time * 0.5 + uv.x * 2.0) * 0.1;

    // Zwei Farben, dunkler für bessere Sichtbarkeit des Textes
    vec3 baseColor = vec3(0.0, 0.2, 0.4);  // Dunkelblau
    vec3 highlightColor = vec3(1.0, 0.5, 0.0);  // Orange
```

```
    vec3 finalColor = mix(baseColor, highlightColor, (sin(uv.x * 10.0) + 1.0)

    vec4 fragColor = vec4(finalColor, 1.0);

    gl_FragColor = fragColor;
}
```

Der Shader erzeugt ein sich bewegendes, wellenförmiges Muster auf dem Bildschirm, das dynamisch zwischen zwei Farben wechselt.

Die Bildkoordinaten werden normalisiert und diskretisiert, um einen Rastereffekt zu erzeugen. Dies wird durch die Verwendung von `v_vTexcoord` und `resolution` erreicht. Wie wir gleich sehen werden, geben wir dieses mal nicht die Bildschirmauflösung an den Shader weiter, sondern eine Zahl. Im Beispiel ist es die 1. Je größer diese Zahl ist, umso mehr zoomen wir rein. Umgekehrt können wir natürlich auch rauszoomen.

Durch die Anwendung von Sinusfunktionen auf die normalisierten Koordinaten (uv) in Abhängigkeit von der Zeit entsteht eine komplexe, wellenartige Bewegung. Du kannst selbst mit den Zahlen experimentieren und so die Auswirkung besser verstehen.

Es wird eine weitere, sanftere Bewegung hinzugefügt, um das Muster organisch erscheinen zu lassen. Dies wird durch die Anwendung von Sinusfunktionen auf die Koordinaten erreicht. Außerdem gibt es zwei Grundfarben: Dunkelblau und Orange. Die Mischung dieser Farben erfolgt dynamisch basierend auf den Sinuswerten der Koordinaten, was zu einem harmonischen Farbübergang führt. Das Muster selbst ist dabei ein verzerrtes, weiches Schachbrettmuster.

Die Bewegung wird – wie in allen Shader-Tutorials hier – durch die Zeitvariable gesteuert.

Objekt

Create-Event

```
time = 0;
time_add = 0.02;
```

Wir starten die Variable `time` mit 0 und bestimmen mit `time_add`, wie schnell diese gesteigert wird. Je größer die Zahl, umso schneller bewegt sich der Hintergrund.

Draw-Event

```
time += time_add;

shader_set(sh_wave01);
shader_set_uniform_f(shader_get_uniform(sh_wave01,"resolution"), 1, 1);
shader_set_uniform_f(shader_get_uniform(sh_wave01,"time"),time);
draw_surface_ext(application_surface, 0, 0, 1, 1, 0, c_white, 0);
shader_reset();
```

In Zeile 4 sehen wir den Unterschied zu anderen Shader-Effekten. Indem wir die 1 für x und y verändern, können wir den Effekt zoomen.

Hier noch ein Beispiel, wo dieser Effekt – und viele weitere – verwendet wurde:

Weiterführende Links

Shader-Programmierung 1: Grundlagen und Sprites
Shader-Effekt: Warping
Shader-Effekt: Interferenz-Effekt

Date Created
2. Februar 2024
Author
sven