

ちょ一ムズい講義を
聞くということ

GLSL50連発

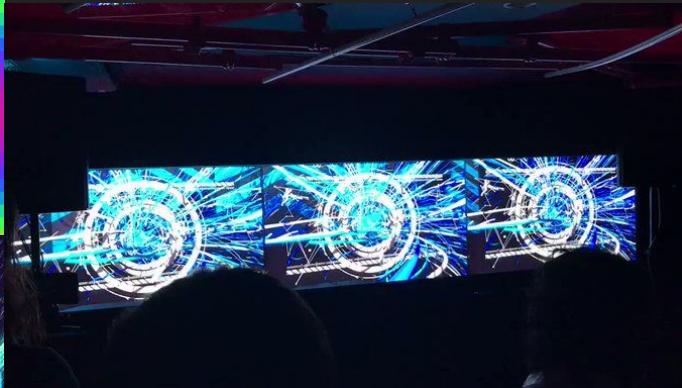
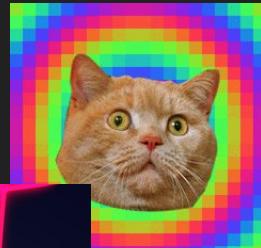
シェーダ味付け101

小渕 豊 (@FMS_Cat)

GLSLスクール 第3回 - 2018年11月13日

こんにちは

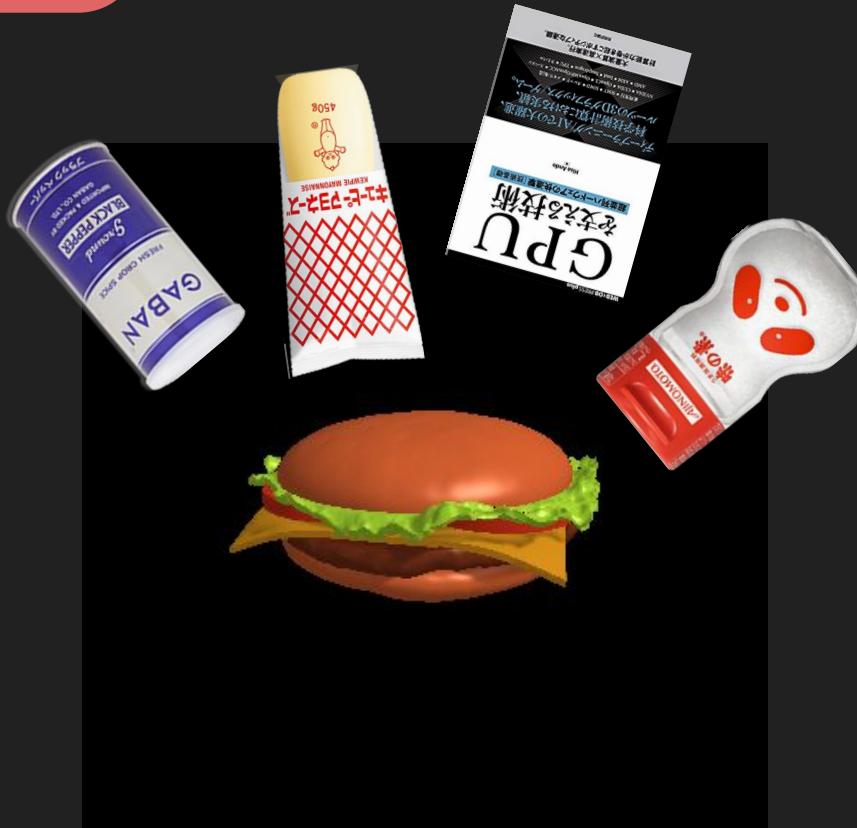
FMS_Catです



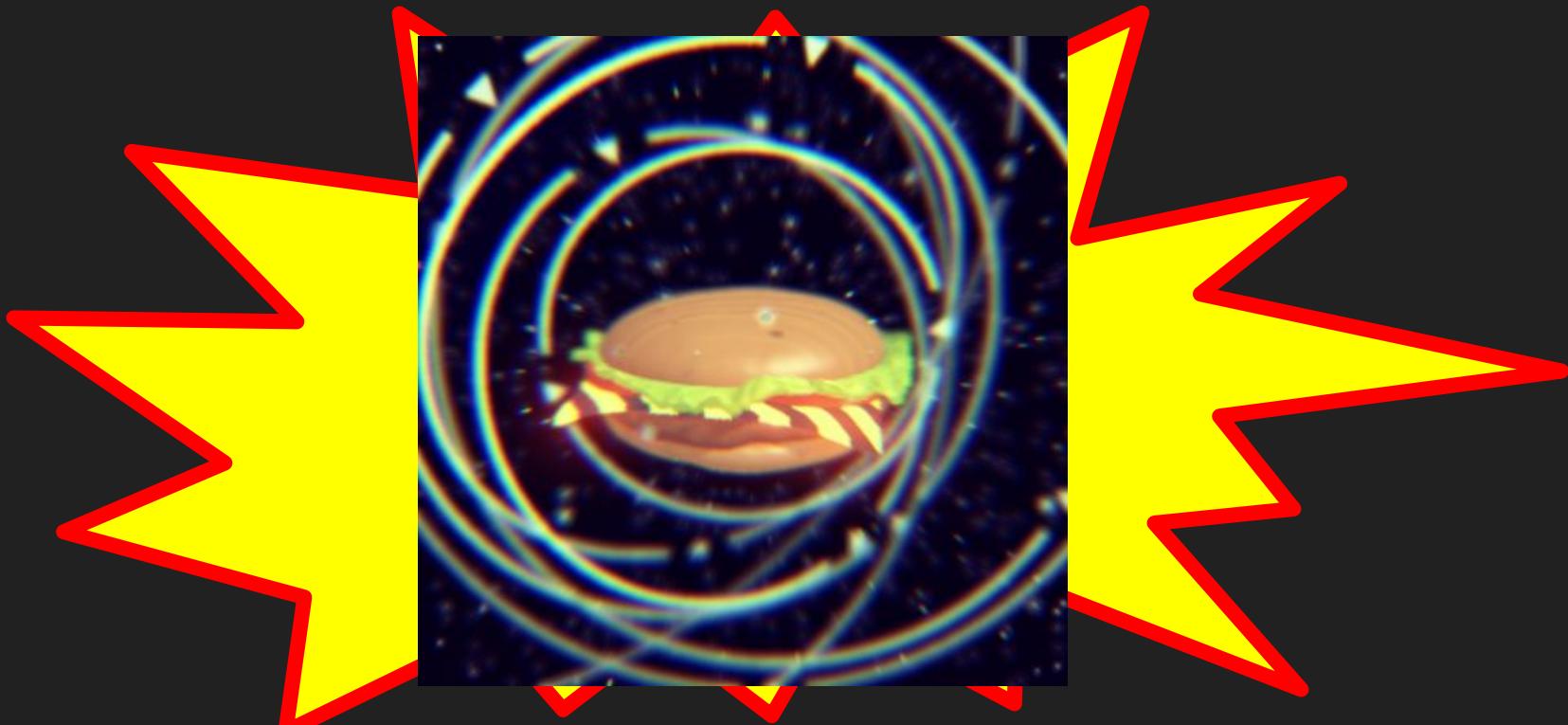
こんにちは



こんにちは



こんにちは



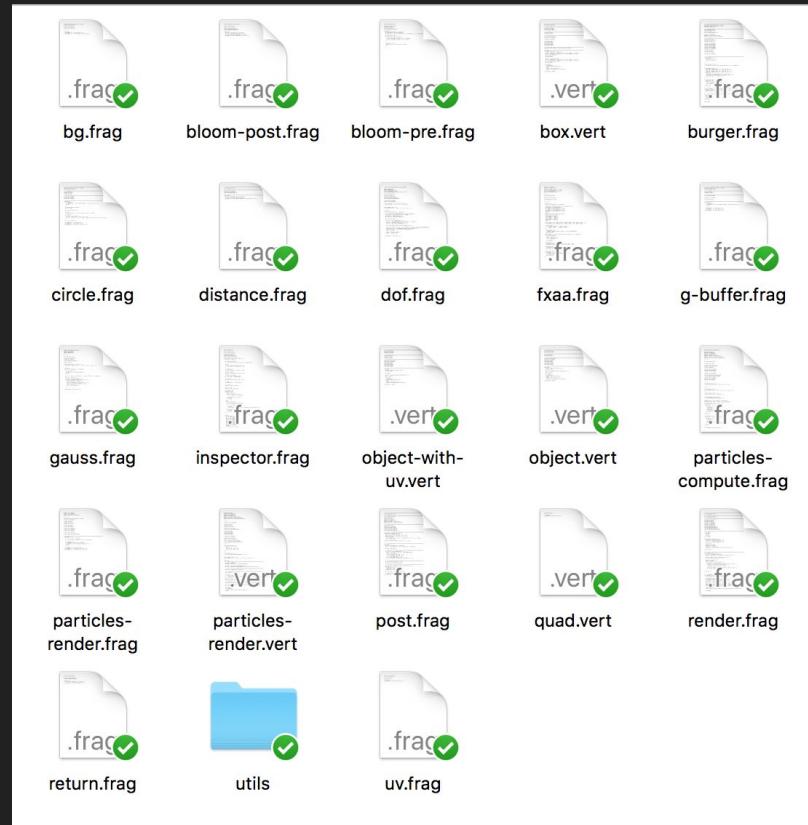
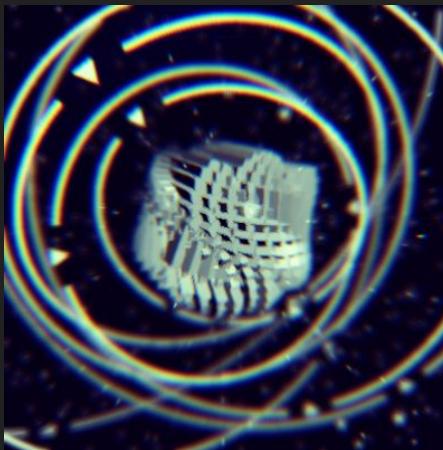
こんにちは

味付けします

- もとの絵があることを前提とする
- 一さじ加えてプロの味
- 調味料 = これ自体が重くなることは望まない
- 今日は味付け例を絵で紹介します

こんにちは

FMS_Catの プロジェクトの様子



シェーダだけで22個！(参照)

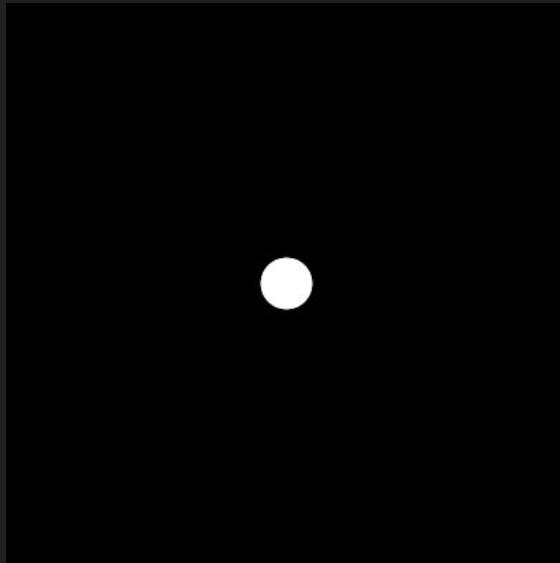
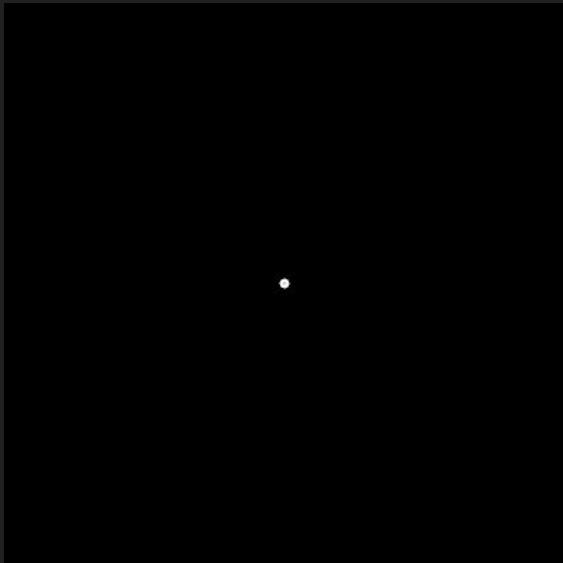
こんにちは

なるべく1パスで
済むことを目指す

(けど2パス以上使うときのセオリーも紹介します)

シェーダだけで22個！

アニメーション

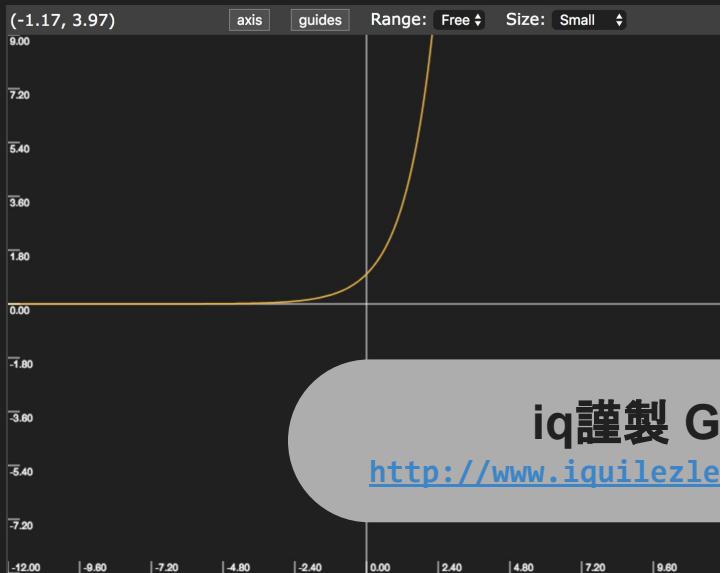


アニメーション

expを使え

アニメーション

$\exp(x) = e^x = e(\text{自然対数の底})\text{のべき乗}$



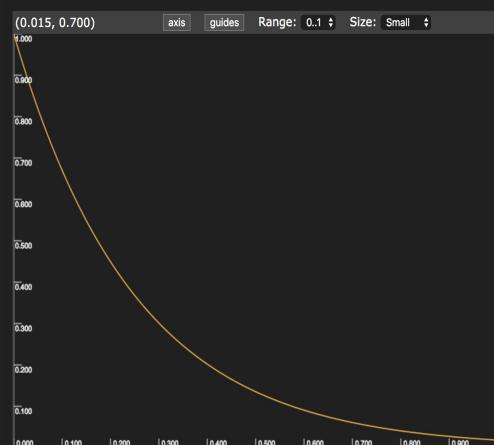
iq謹製 GraphToy

<http://www.iquilezles.org/apps/graphtoy/>

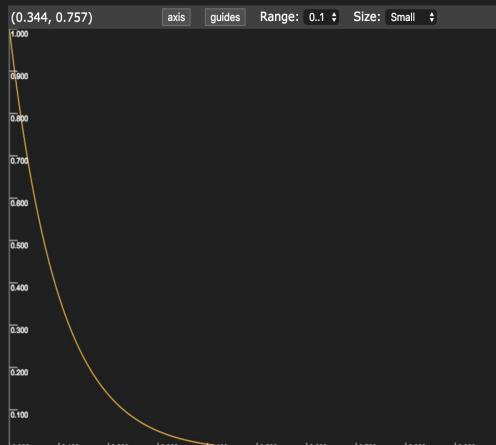
アニメーション



$$\exp(-x)$$



$$\exp(-4*x)$$



$$\exp(-10*x)$$

アニメーション

$\exp(-k*x)$ のここがスゴイ

- x の増加に従い**0に漸近する**(0になることはない)
- **速度変化も同じ形** ($d/dx e^{-kx} = -k e^{-kx}$)
- **あらゆる自然現象でみられる曲線**(指数関数的減衰)

アニメーション

expを使え

アニメーション

```
precision mediump float;
uniform float time;
uniform vec2 resolution;

void main(void) {
    vec2 p = (gl_FragCoord.xy * 2.0 - resolution) / resolution.y;

    // define animation
    float inner = 1.0 - exp(-5.0 * time);
    float outer = 1.0 - exp(-5.0 * (time - 0.2));

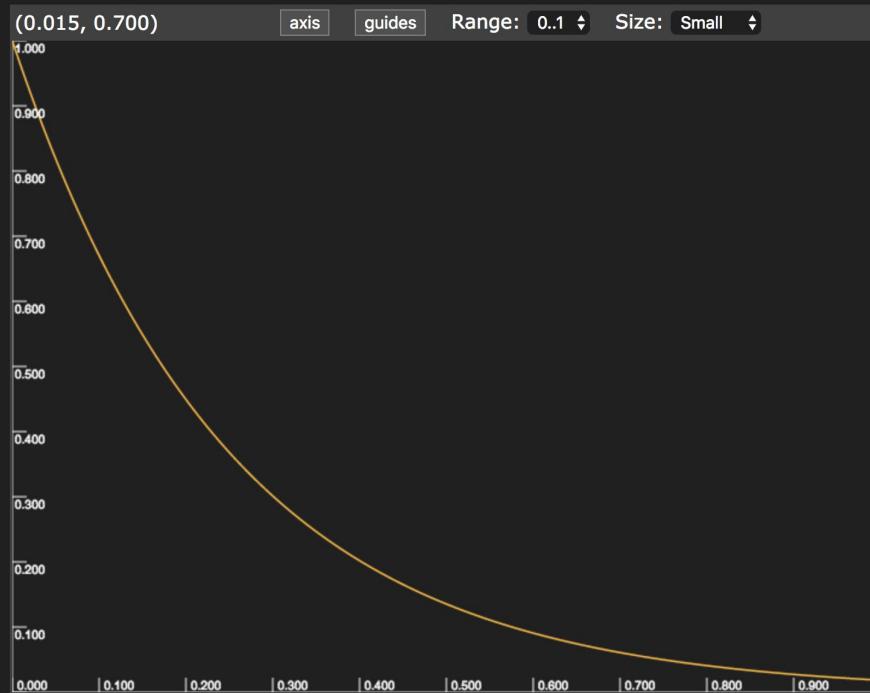
    // shape
    float len = length(p);
    float ssThr = 2.0 / resolution.y;
    float shape = smoothstep(0.0, ssThr, inner - len);
    shape *= smoothstep(0.0, ssThr, len - outer);

    gl_FragColor = vec4(vec3(shape), 1.0);
}
```



アニメーション

速度変化が
急！



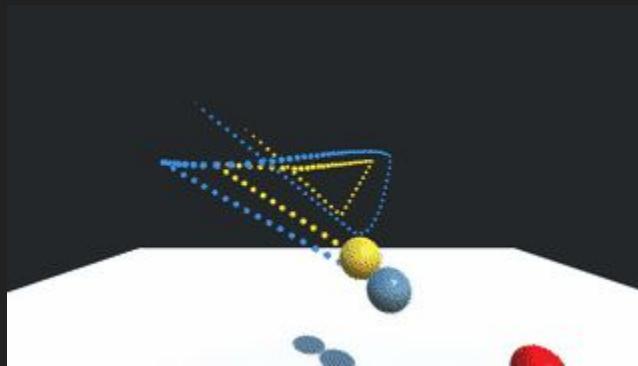
アニメーション

ほかにもいろいろアニメーション

アニメーション

Critically Damped Spring

<https://github.com/keijiro/SmoothingTest>

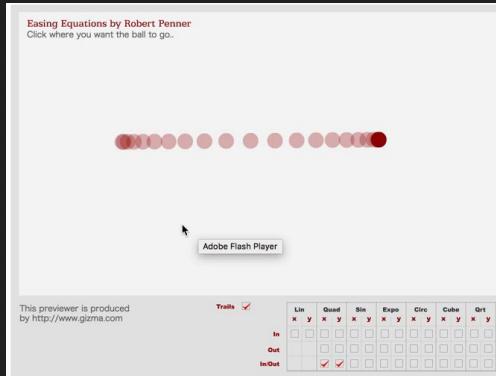


ステート保持が必要(位置・速度)

アニメーション

Easing Equations

<http://gizma.com/easing/>



```
simple linear tweening = no easing, no acceleration

Math.linearIn = function (t, b, c, d) {
    return c*t/d + b;
};

quadratic easing in = accelerating from zero velocity

Math.easeInQuad = function (t, b, c, d) {
    t /= d;
    return c*t*t + b;
};

quadratic easing out = decelerating to zero velocity

Math.easeOutQuad = function (t, b, c, d) {
    t /= d;
    return -c*(t*t-2) + b;
};

quadratic easing in/out = acceleration until halfway, then deceleration

Math.easeInOutQuad = function (t, b, c, d) {
    t /= d/2;
    if (t < 1) return c*(t*t*4) + b;
    t -= 2;
    return -c*(t*t+2) + b;
};
```

CSSのアレ
関数定義すればすぐ使える

アニメーション

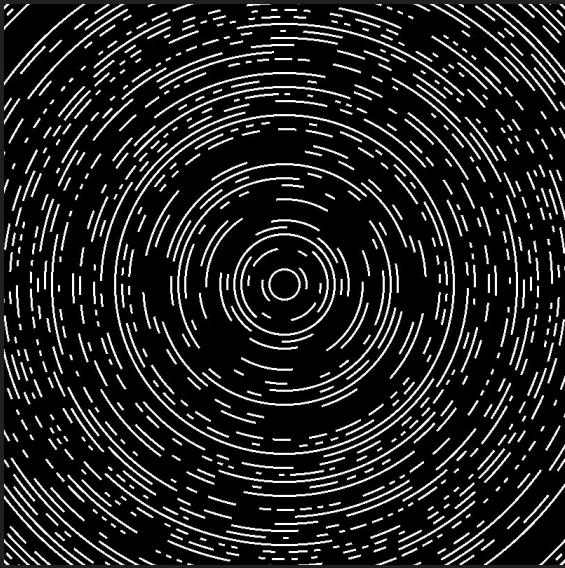
Automaton

<https://github.com/FMS-Cat/automaton>



ベジェ曲線とFXでアニメーションしまくり
JavaScriptが得意なら是非！

色



色



色

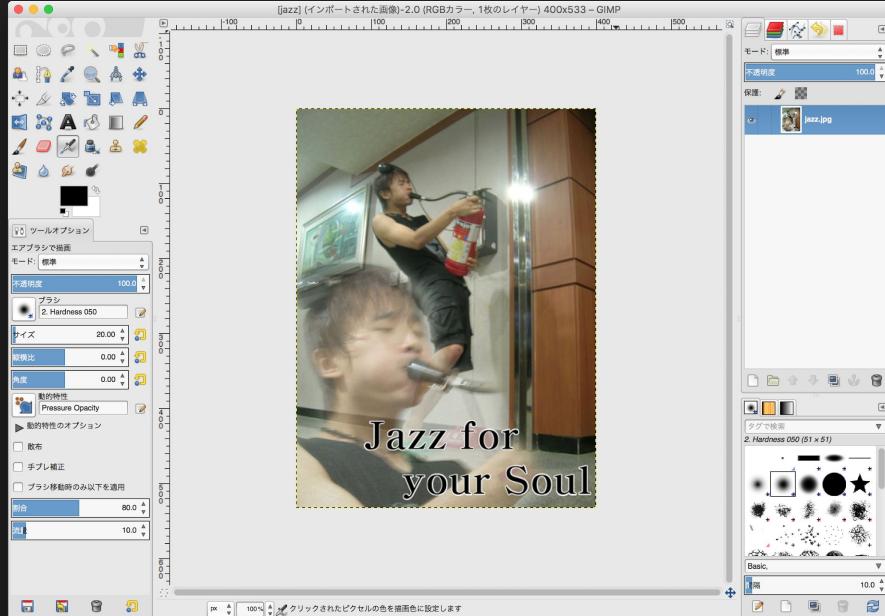


Instagram X-Pro II Photoshop

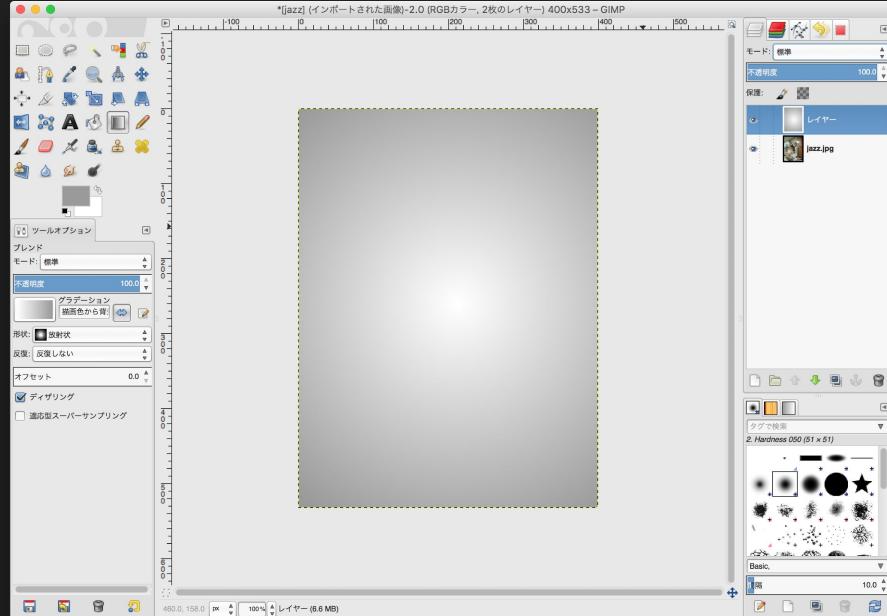
Google 検索

I'm Feeling Lucky

色

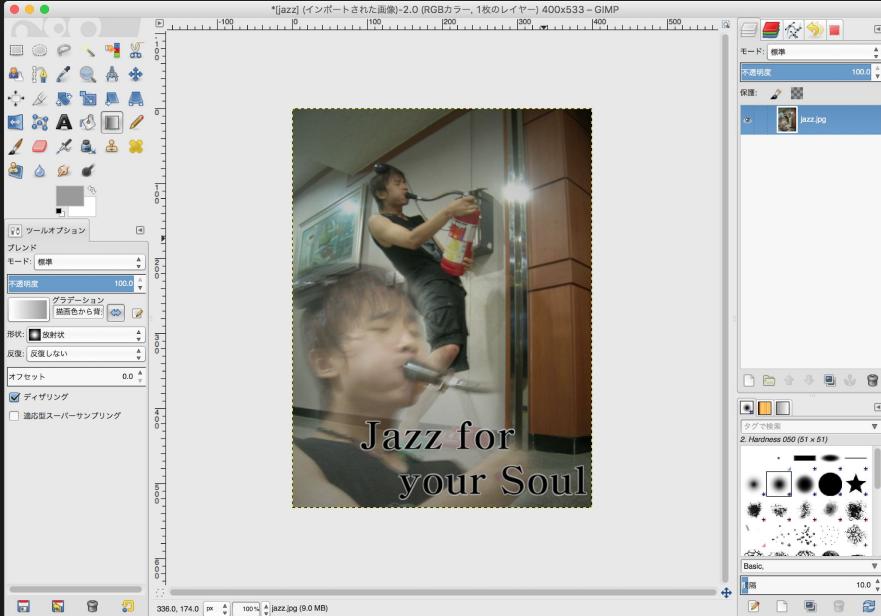


色



グラデーションを作成

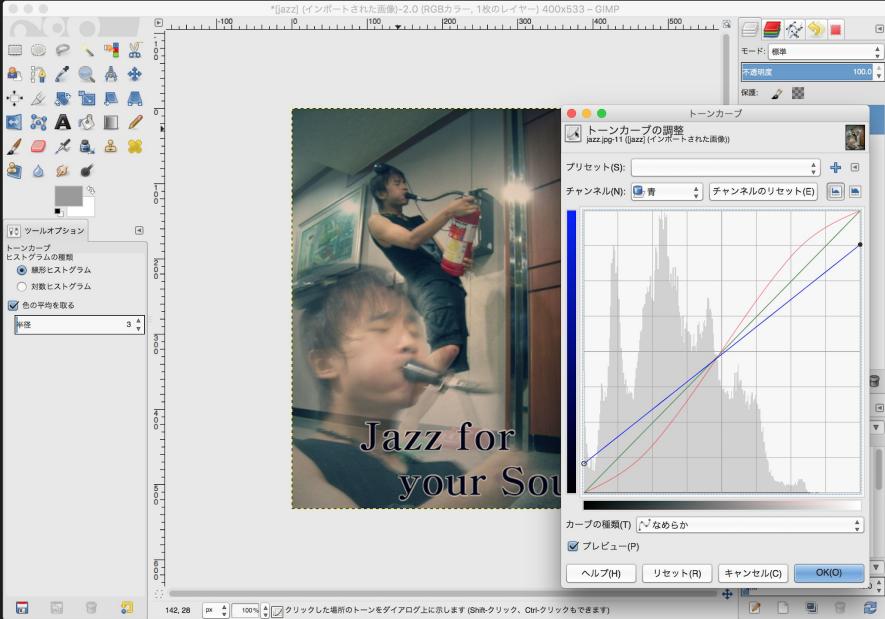
色



グラデーションを作成

乗算合成

色



グラデーションを作成

乗算合成

トーンカーブを適用

色



Jazz for
your Soul



Jazz for
your Soul

グラデーションを作成

乗算合成

トーンカーブを適用

色



グラデーションを作成

乗算合成

トーンカーブを適用

これシェーダができるよね

色

```
col *= 1.0 - length( p ) * 0.2;
```

```
col = vec3(  
    smoothstep( 0.0, 1.0, col.x ),  
    col.y,  
    0.1 + 0.8 * col.z  
);
```

グラデーションを作成

乗算合成

トーンカーブを適用

色

おたくはすぐ 色収差

色



```
col.x = texture2D( samplerLenna, uv + vec2( 0.01, 0.0 ) ).x;  
col.y = texture2D( samplerLenna, uv ).y;  
col.z = texture2D( samplerLenna, uv - vec2( 0.01, 0.0 ) ).z;
```

色



```
col.x = texture2D( samplerLenna, 0.5 + 1.03 * ( uv - 0.5 ) ).x;  
col.y = texture2D( samplerLenna, uv ).y;  
col.z = texture2D( samplerLenna, 0.5 + 0.97 * ( uv - 0.5 ) ).z;
```

色



```
col.x = texture2D( samplerLenna, 0.5 + 1.03 * ( uv - 0.5 ) ).x;  
col.y = texture2D( samplerLenna, uv ).y;  
col.z = texture2D( samplerLenna, 0.5 + 0.97 * ( uv - 0.5 ) ).z;
```

色

おたく臭さが若干抜けた
色収差

色



色

拡大方向にブラー

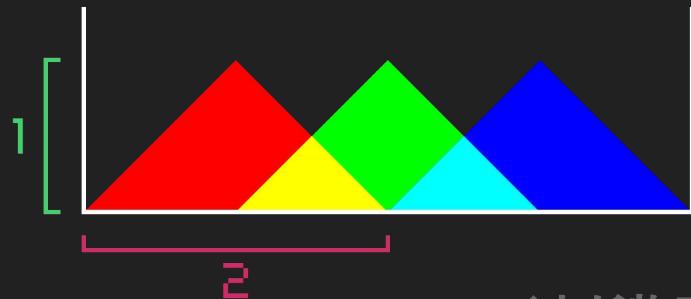


Lenna (迫
真)

色

拡大方向にブラー

R, G, Bそれぞれの総和が1になるようにグラデーション



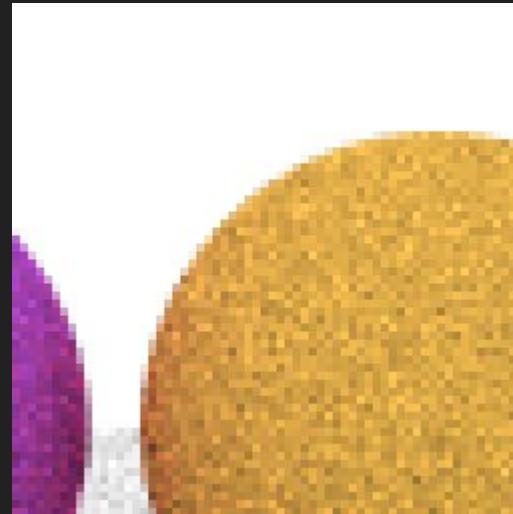
(まだ物理的に正しくない)

色

アンチエイリアス

色

アンチエイリアス
物体の縁に注目！



色

アンチエイリアス
物体の縁に注目！



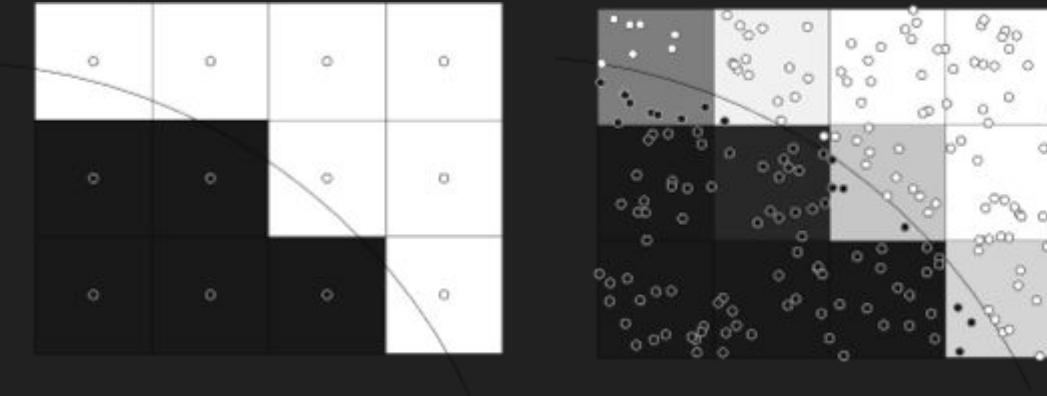
描画手法ごとに作戦が変わってくる



色

Supersampling AA (SSAA)

Oldschool, Bruteforce

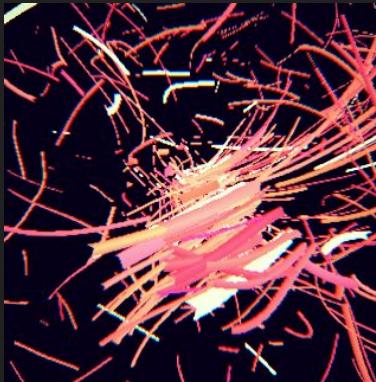


1ピクセル内でランダムサンプリング
当然めっちゃ重い！

色

Fast Approximate AA (FXAA)

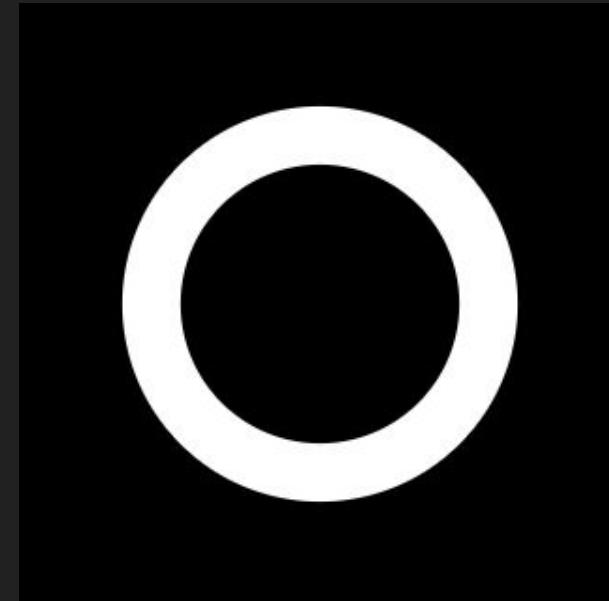
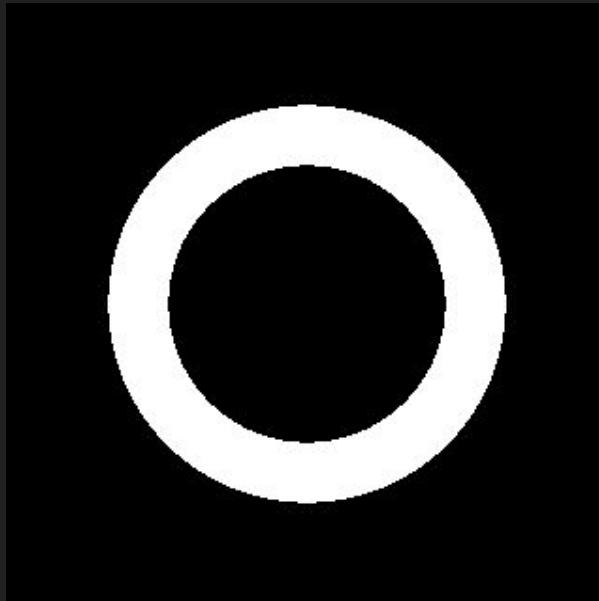
NVIDIA's Post-Processing Anti-Aliasing



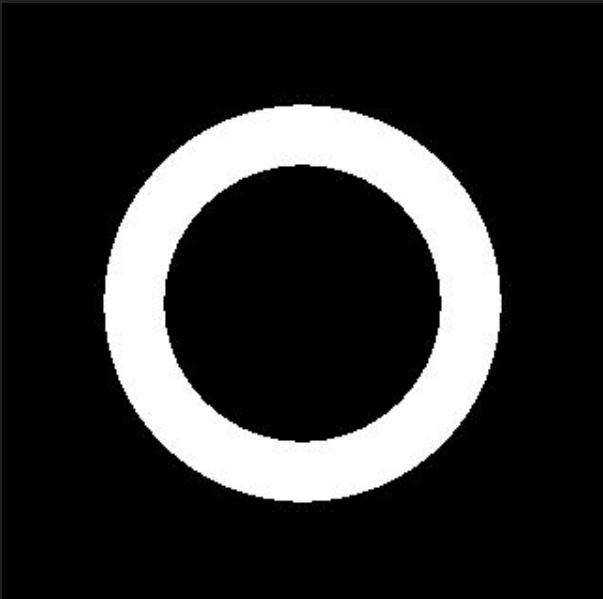
エッジ検出 → ぼかし
ポストエフェクトとして実装できる

色

境界を線形補間



色



```
precision mediump float;
uniform float time;
uniform vec2 resolution;

void main(void) {
    vec2 p = (gl_FragCoord.xy * 2.0 - resolution) / resolution.y;

    // define animation
    float inner = 1.0 - exp(-5.0 * time);
    float outer = 1.0 - exp(-5.0 * (time - 0.2));

    // shape
    float len = length(p);

    float shape = step(len, inner);
    shape *= step(outer, len);

    gl_FragColor = vec4(vec3(shape), 1.0);
}
```

色



```
precision mediump float;
uniform float time;
uniform vec2 resolution;

#define saturate(x) clamp(x,0.,1.)
#define linearstep(a,b,x) saturate((x-a)/(b-a))

void main(void) {
    vec2 p = (gl_FragCoord.xy * 2.0 - resolution) / resolution.y;

    // define animation
    float inner = 1.0 - exp(-5.0 * time);
    float outer = 1.0 - exp(-5.0 * (time - 0.2));

    // shape
    float len = length(p);
    float ssThr = 2.0 / resolution.y;
    float shape = linearstep(0.0, ssThr, inner - len);
    shape *= linearstep(0.0, ssThr, len - outer);

    gl_FragColor = vec4(vec3(shape), 1.0);
}
```

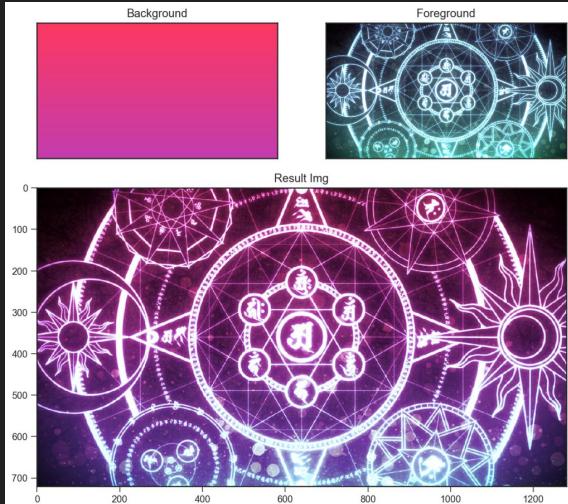
色

その他の色トピックス

色

描画モード

<http://optie.hatenablog.com/entry/2018/03/15/212107>



描画モードまとめ表 A₃

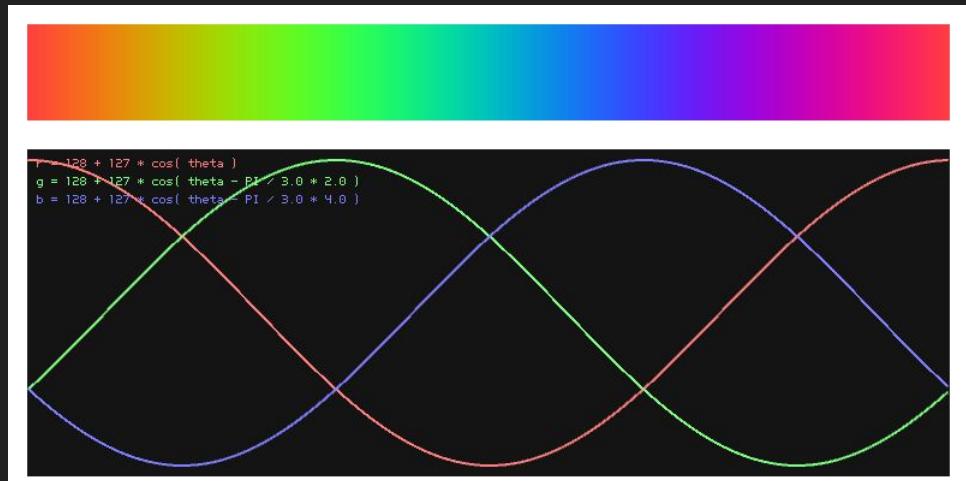
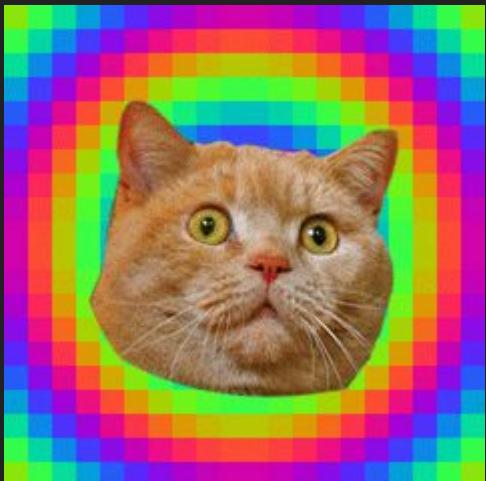
@Optie_f

明るくなる			
明/暗強化	A+Bの50%クレーを塘に処理が変わる。 明部は左の演算、暗部は右の演算	比較（明）	暗くなる
R,G,Bごとの比較： 高い(低い)方の値	乗算系： 穏やか、白飛び(黒濁れ)しない 乗算	スクリーン $1 - (1 - A) \times (1 - B)$	オーバーレイ (A,B)入力されたビヘイドライト
加算系： 強い、白飛び(黒濁れ)やすい 加算	覆い焼きリニア (=加算)	$A + B$	焼き込みリニア $1 - \{ (1 - A) + (1 - B) \}$
除算系： 前景の明部(暗部)だけ強くかかる 除算	覆い焼きカラー $B + (1 - A)$	ビビッドライト (二値化するとハードミックス)	焼き込みカラー $1 - \{ (1 - B) * A \}$
RGB合計値の比較： 高い(低い)方のピクセル	カラー比較（明）	ソフトライト*	カラー比較（暗）
HSL系 (色相、彩度、輝度)定義で、上のレイヤーのXXだけ 下のレイヤーにそのまま移す			
色相	カラー (色相&彩度)	差 $ B - A $	減算 $B - A$
彩度	輝度	除外 $2 \times \{ (A+B) \times 2 - A \times B \}$	除算 B / A
その他			
*ソフトライトはアブリケーションによって 処理が異なり、独自の計算が行われている。 Photoshopでは、以下の計算式			
$\begin{cases} 2AB + 20.5 - A\beta^2 & (A < 0.5) \\ 2(1 - A)\beta + 2(1 - 0.5)\beta^2 & (\text{otherwise}) \end{cases}$			
これは、 β と「Gamma補正したB」の値を Aの画面値で割り算した、つまり、以下の通り。			
Aが暗： $\gamma=1$ (黒) と $\gamma=1$ (白) の間で A の部分 Aが明： $\gamma=1$ (黒) と $\gamma=0.5$ (白) の間で A の部分 結論：変化の幅が限定的なので、穏やかな結果になる。			

色

Sine Curve Gradient

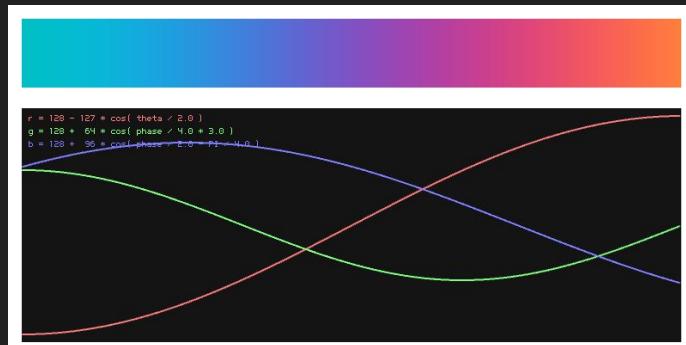
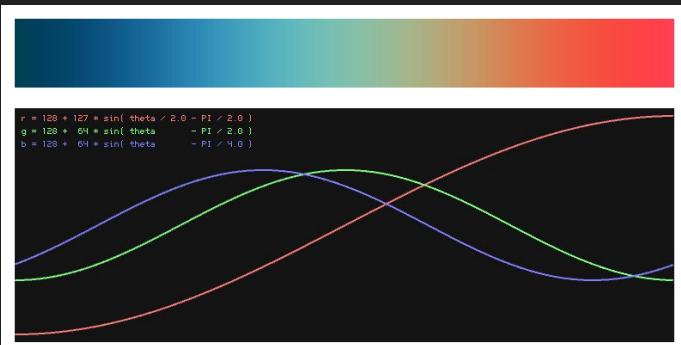
`catColor(float t)`



色

Sine Curve Gradient

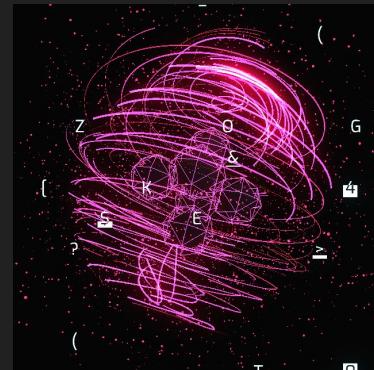
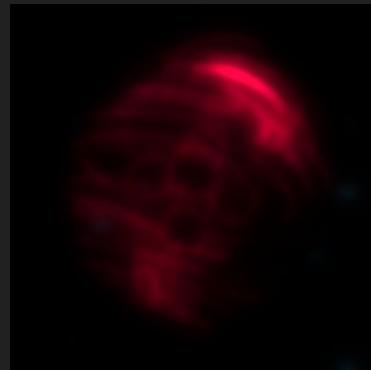
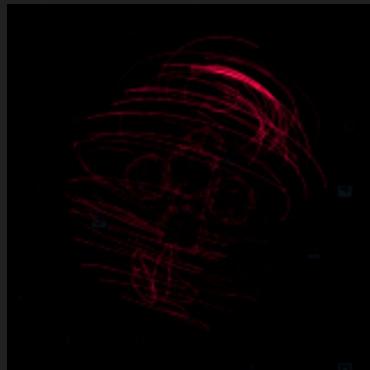
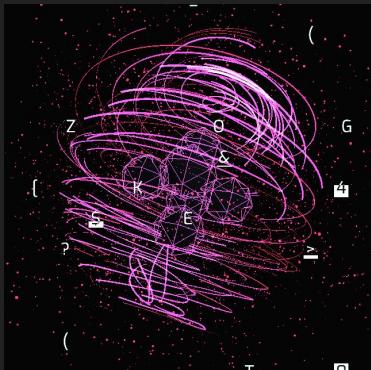
other cases



色

Bloom

まぶしい



明るいスポット抽出



ぼかす

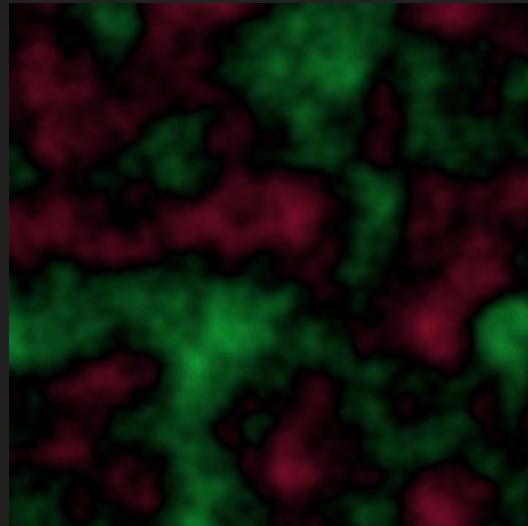


元の絵に足す

ノイズ

ノイズと一言で言うと
一般にはフラクタルノイズ(たぶん)

- 位相に応じてランダム
- でも連続的
- 有機的な表現に有用



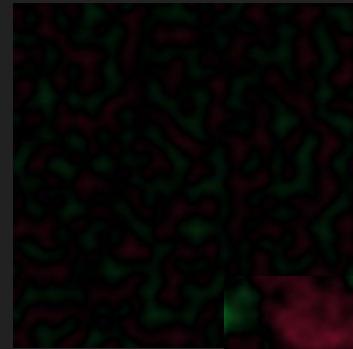
ノイズ



+



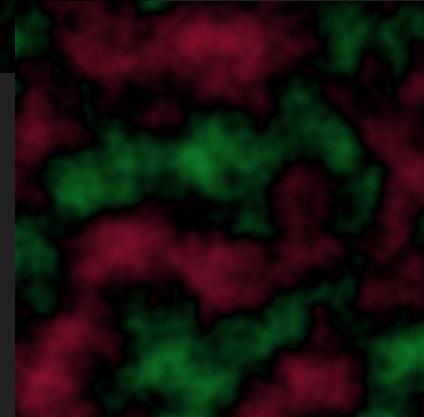
+



+

...

=



ノイズ

実装してみよう

```
float hash2D( vec2 v ) {
    return fract( sin( dot( v, vec2( 4.2787, 3.7915 ) ) ) * 2851.6893 );
}

float hash3D( vec3 v ) {
    return fract( sin( dot( v, vec3( 4.2787, 3.7915, 4.8911 ) ) ) * 2851.6893 );
}

float fade( float x ) {
    return ( ( 6.0 * x - 15.0 ) * x + 10.0 ) * x * x * x;
}

float perlin3D( vec3 v ) {
    vec3 vI = floor( v );
    vec3 vF = fract( v );

    float sum = 0.0;
    for ( int iz = 0; iz < 2; iz ++ ) {
        for ( int iy = 0; iy < 2; iy ++ ) {
            for ( int ix = 0; ix < 2; ix ++ ) {
                vec3 contF = vec3( ix, iy, iz );
                vec3 cont = vI + vec3( ix, iy, iz );
                vec3 cont2v = vF - contF;
                vec3 h = normalize( 2.0 * vec3(
                    hash3D( cont ),
```

ノイズ

<https://github.com/ashima/webgl-noise>

The screenshot shows the GitHub repository page for 'ashima / webgl-noise'. The repository title is 'Procedural Noise Shader Routines compatible with WebGL'. It has 87 commits, 3 branches, 0 releases, and 3 contributors. The latest commit was on 16 May 2016. The repository is licensed under MIT. The README file is present. The footer notes that the wiki contains more information and that the code is released under the MIT license.

ashima / webgl-noise

Code Issues Pull requests Projects Wiki Insights

87 commits 3 branches 0 releases 3 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

stegu Merge pull request #17 from stegu/master ... Latest commit b7d1861 on 16 May 2016

benchmark Created warning about old code 3 years ago

demo Created warning about old code 3 years ago

src Created file 3 years ago

LICENSE Update LICENSE 3 years ago

README Added link to clone repo 3 years ago

README

The wiki for this repository contains more information.

Simplex noise functions are (C) Ashima Arts and Stefan Gustavson
Classic noise functions are (C) Stefan Gustavson
Cellular noise functions are (C) Stefan Gustavson
The "psrnoise" functions are (C) Stefan Gustavson

Sources code for the noise functions is released under the MIT License.

ノイズ

さて

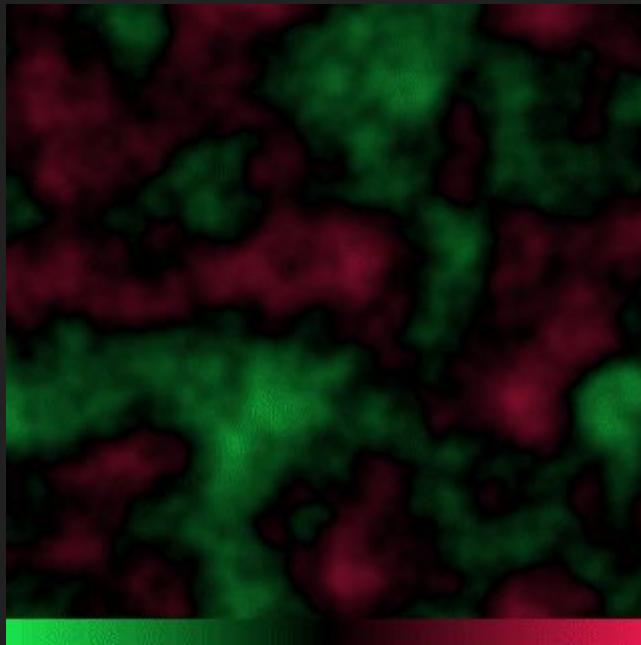
ノイズ

持っているのは.....

- vec2 位置
- float 現在時刻
- vec3を渡すと模様が出る謎の関数
- (その他どうぐいろいろ)

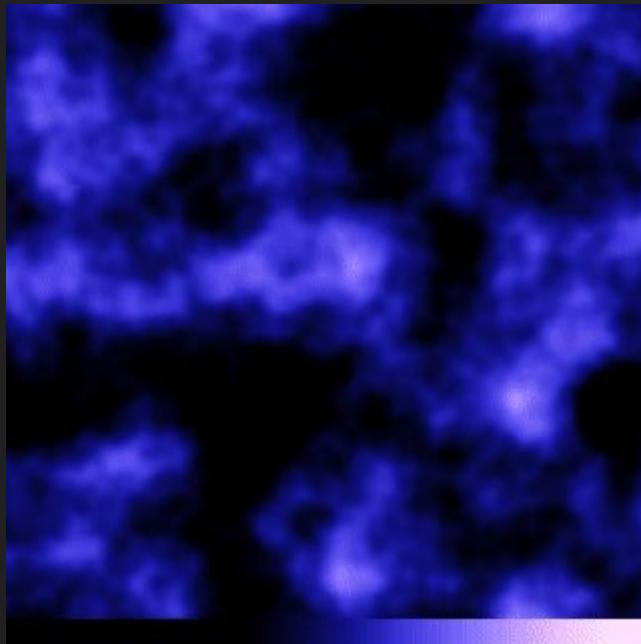
この3つを最大限活用して何ができるか？

ノイズ



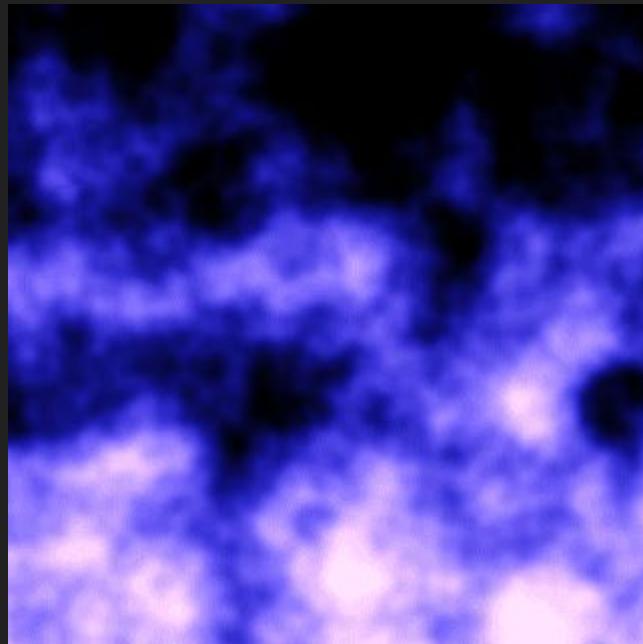
```
float sum = noise( uv, time );  
  
vec3 col = vec3( 0.0 );  
col += vec3( 0.9, 0.1, 0.3 ) * saturate( sum );  
col += vec3( 0.1, 0.9, 0.3 ) * saturate( -sum );
```

ノイズ

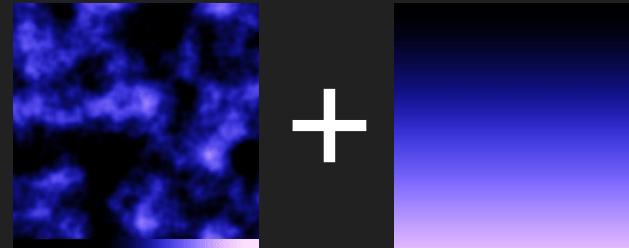


```
// purple gradient!
col = mix(
    vec3( 0.0, 0.0, 0.0 ),
    mix(
        vec3( 0.1, 0.1, 0.8 ),
        vec3( 1.1, 0.9, 1.3 ),
        smoothstep( 0.0, 1.0, sum )
    ),
    smoothstep( -0.3, 0.3, sum )
);
```

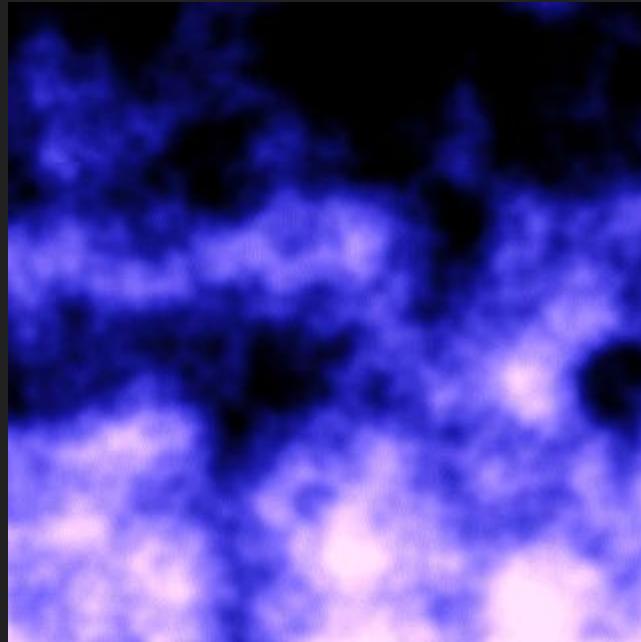
ノイズ



```
sum += 0.7 - uv.y;
```

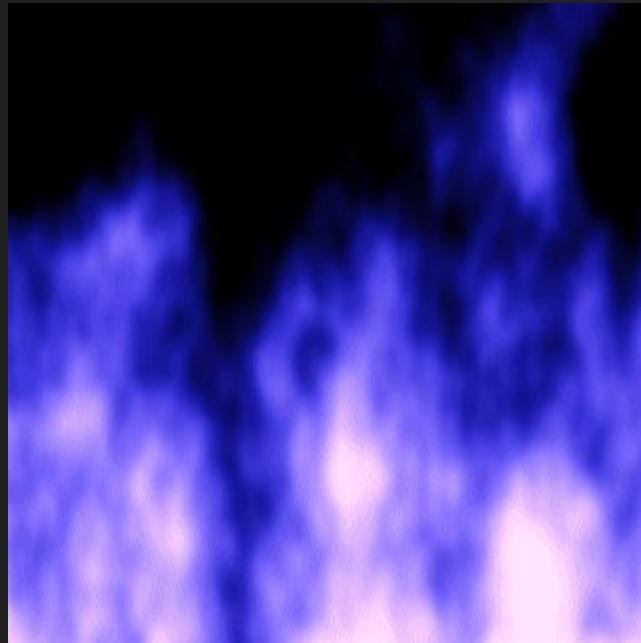


ノイズ



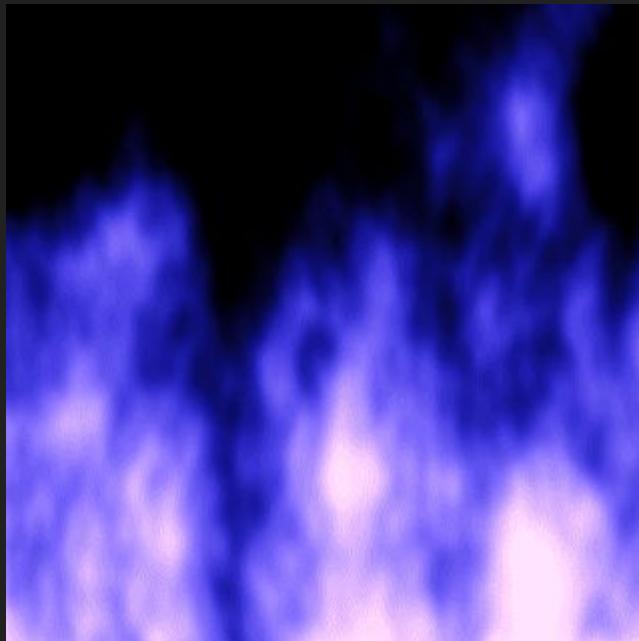
```
sum += 0.7 - uv.y;  
uv.y -= 1.5 * time;
```

ノイズ



```
sum += 0.7 - uv.y;  
uv.y *= 0.5;  
uv.y -= 1.5 * time;
```

ノイズ

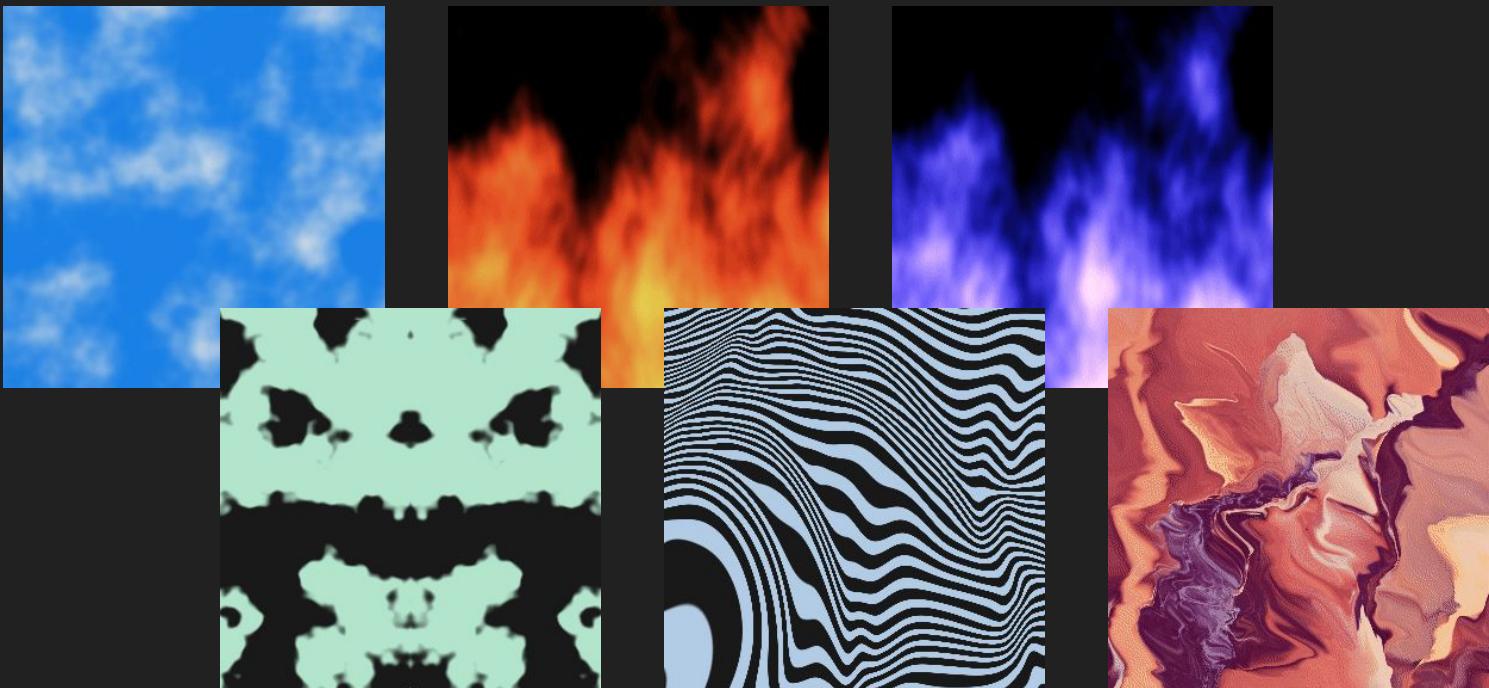


```
sum += 0.7 - uv.y;  
uv.x += 0.01 * sin( 10.0 * uv.y ) * sin( 5.0 *  
uv.x );  
uv.y *= 0.5;  
uv.y -= 1.5 * time;
```



やりすぎた場合

ノイズ



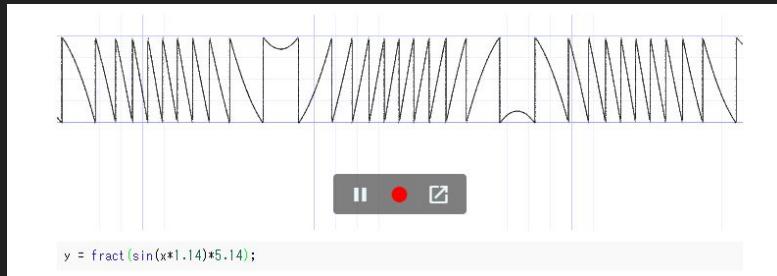
ノイズ

余談



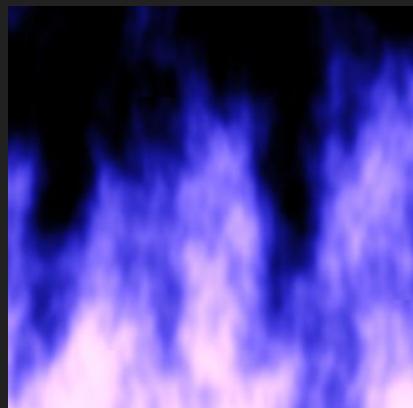
```
float hash3D( vec3 v ) {  
    return fract( sin(  
        dot( v, vec3( 1.14, 5.14, 1.919 ) )  
    ) * 810.893 );  
}
```

いわゆる“FractSin” [\[Ref.\]](#)

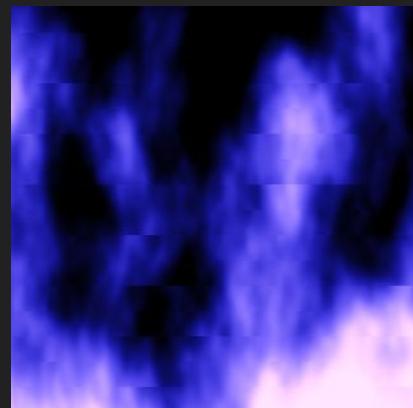


ノイズ

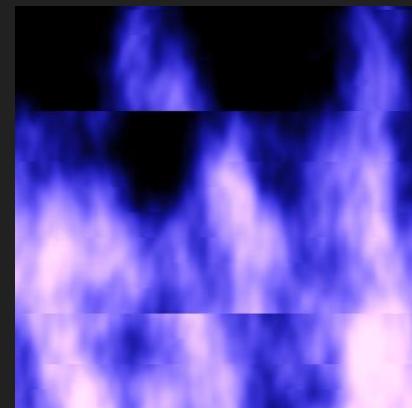
余談



time = 2.0



time = 20.0



time = 200.0

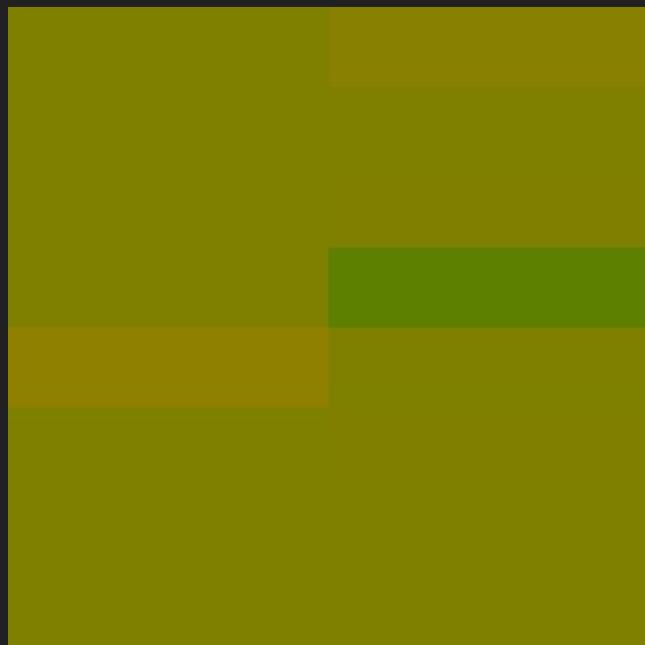
ノイズ



ノイズ



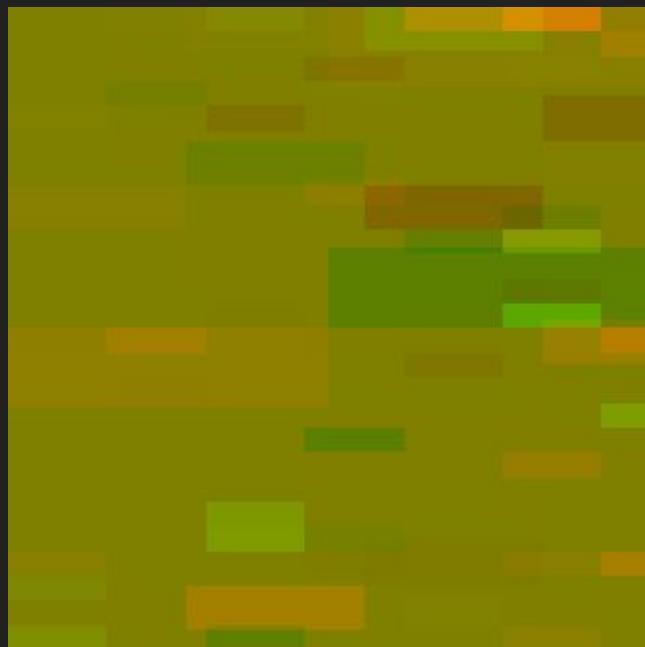
ノイズ



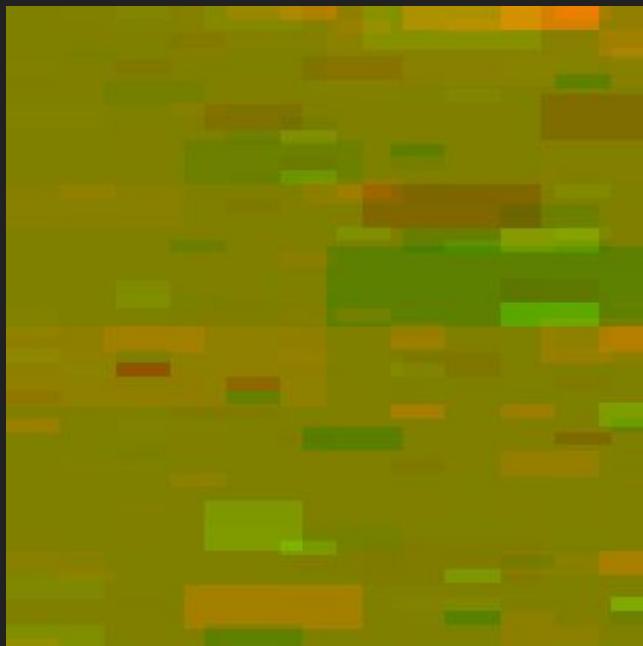
ノイズ



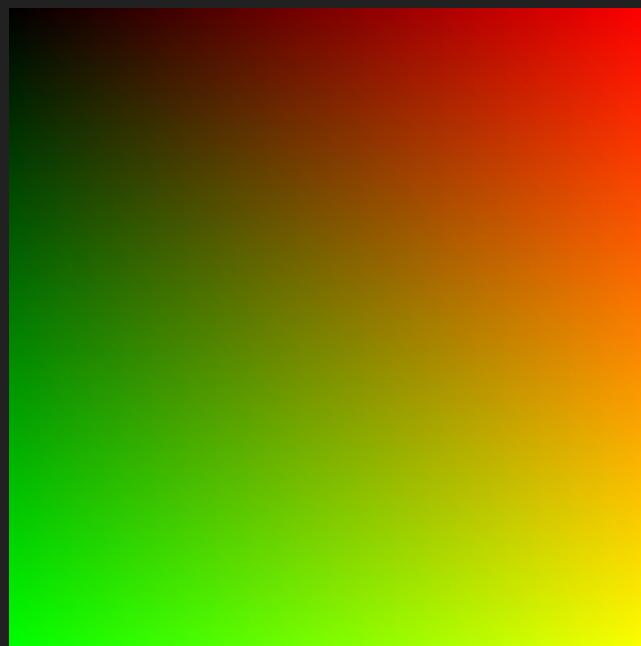
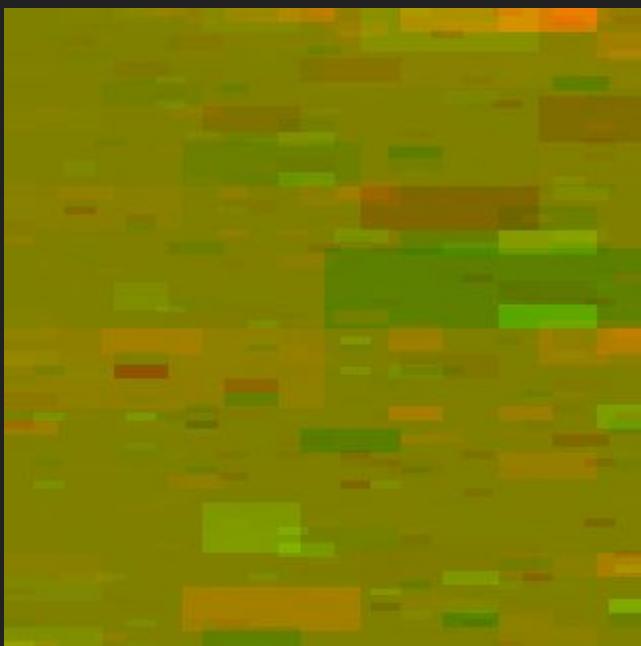
ノイズ



ノイズ



ノイズ



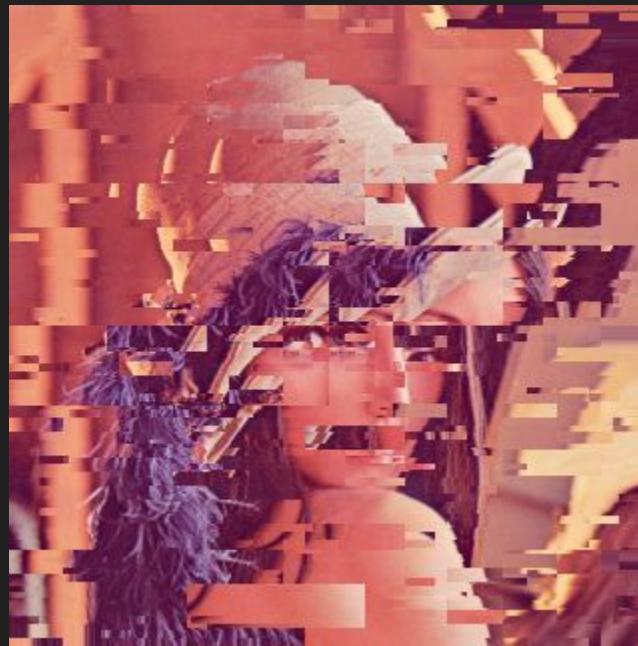
ノイズ



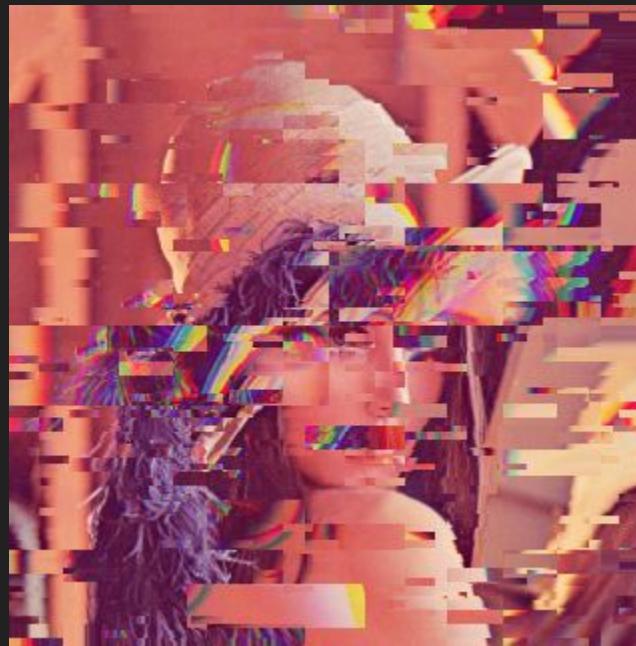
Sample



ノイズ



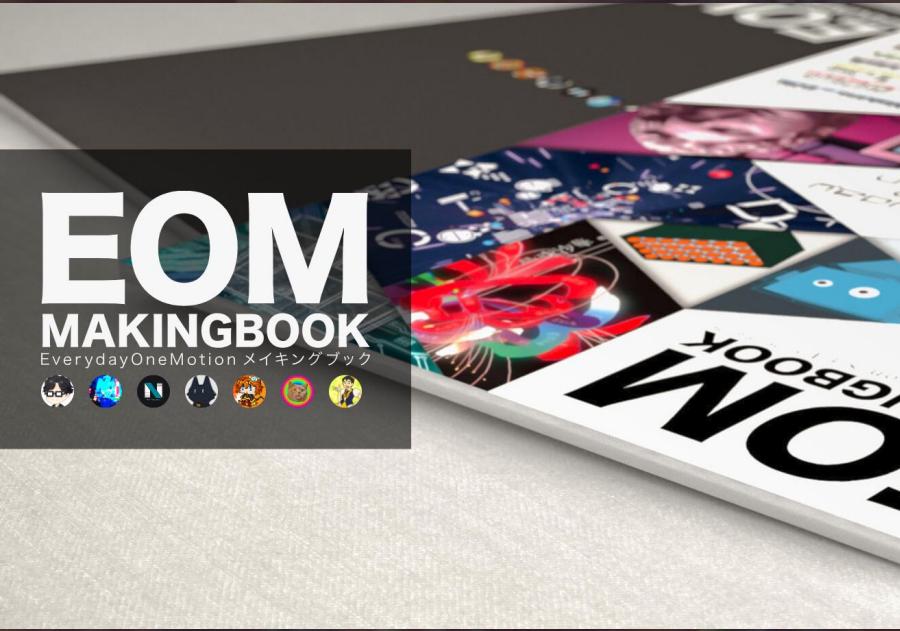
ノイズ



ノイズ

おたく

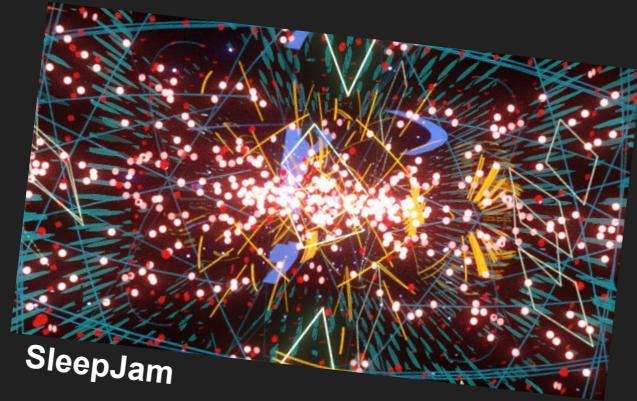
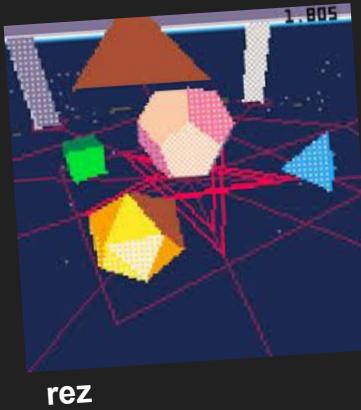
ノイズ



割と本当に
伝えたかったこと

ほかの畠からパクれ

割と本当に
伝えたかったこと



ほかの畠からパクれ



Keijiro Takahashi



beepie

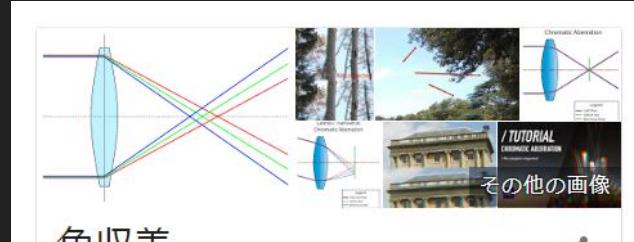
割と本当に
伝えたかったこと

技術は違えど
表現手法・マインド
はパクれる

割と本当に
伝えたかったこと



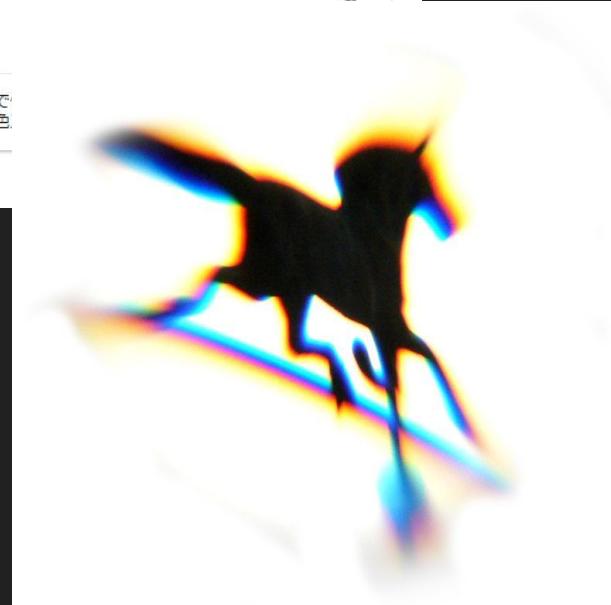
割と本当に 伝えたかったこと



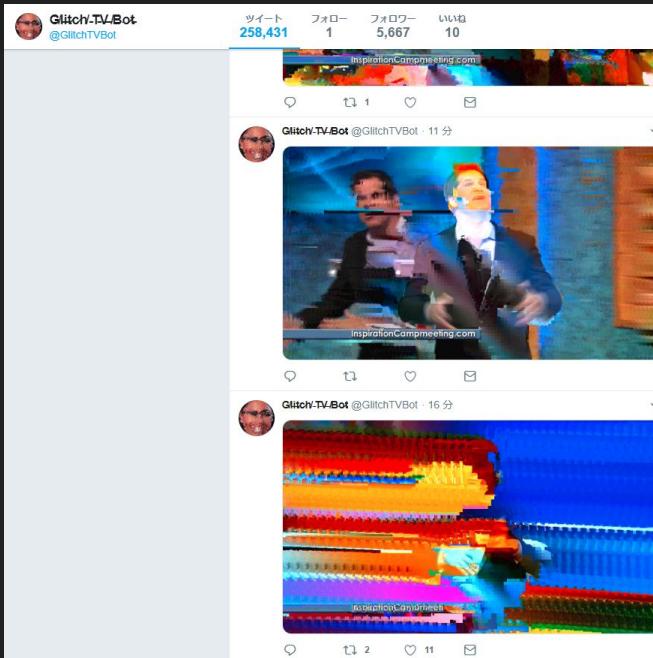
色収差

いろしゅうさ

色収差とは、レンズ類で
発生する収差で、像の色



割と本当に 伝えたかったこと



Datamosh

Author: Plugin Play

License Type

Price: \$39.99 Qty: 1

License terms [TRY](#) [ADD TO CART](#)

Break your videos and find the pixel wonderland :) The only way to true Datamosh inside of After Effects

VIDEOS

TUTORIALS, VIDEO

HOW TO DATAMOSH VIDEOS

▶ VIDEO ⏪ STICKY 38 COMMENTS 42

Video datamoshed with Avidem... 後で見る 共有

Datamoshing is the process of manipulating the data of media files in order to achieve visual or auditory effects when the file is decoded. In some cases the term **datamoshing** is used to describe this process applied to any type of media file – I like to think it applies solely to video since it results in moving images being moshed together. Regardless of the application of the term, **datamoshing** videos can be done quite easily with free, cross-platform tools.

