

TouchDesigner GLSL Workshop



amagitakayosi

講義の準備をお願いします

- ・ TouchDesignerのバージョン: **2018.26750**
- ・ サンプルファイル:
[https://github.com/fand/MUTEK2018/raw/
master/glsIEffects.toe](https://github.com/fand/MUTEK2018/raw/master/glsIEffects.toe)

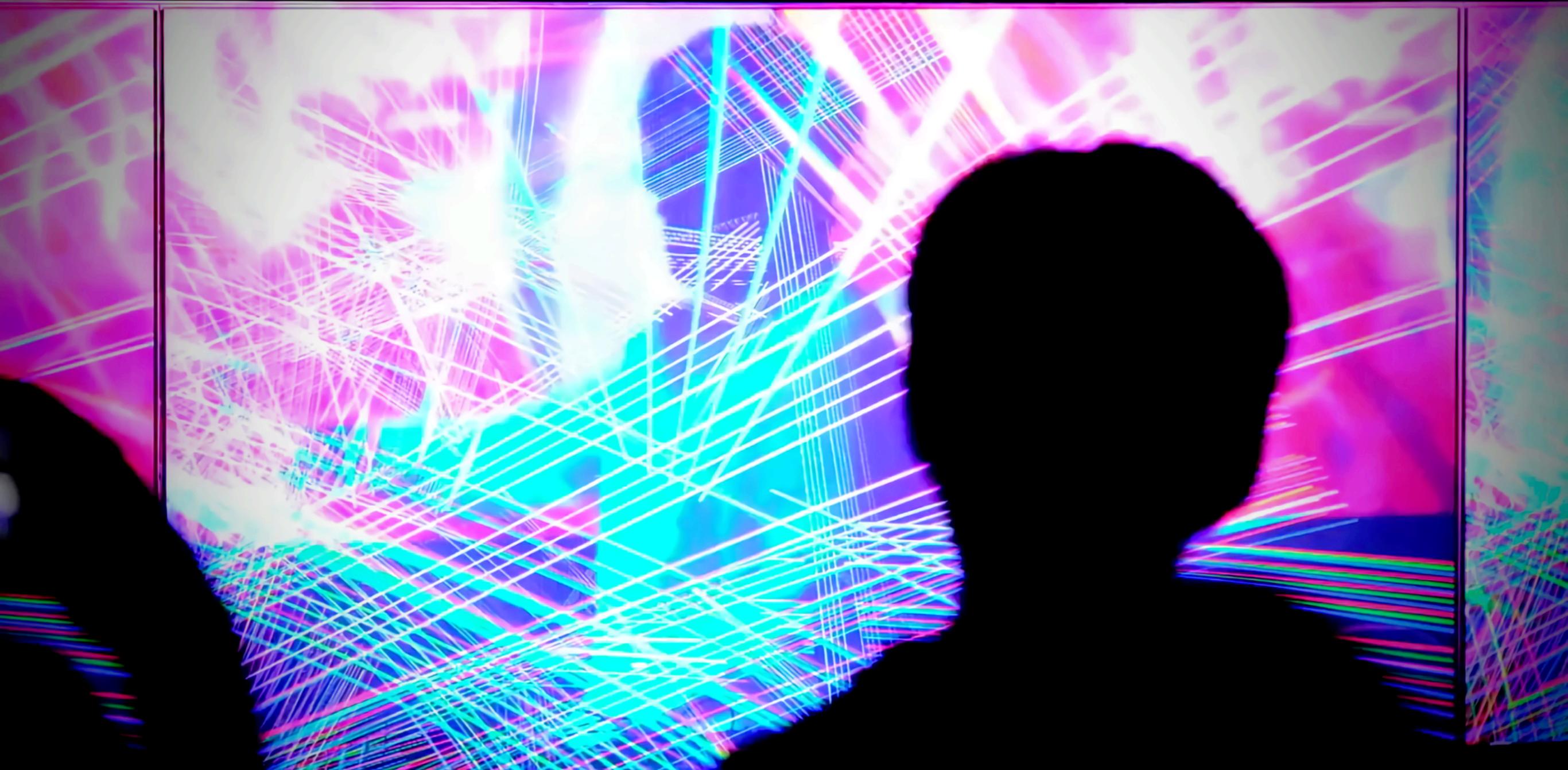


@amagitakayosi

TouchDesignerでVJしたり



UnityでVJしたり



GLSLライブコーディングしています

```
POST_FRAG * XFRAMES
{
    uv = abs(uv - .5);
    uv += volume * .003;
}

if (f < .6)
{
    float l = length(uv - .5);
    uv = rotate(uv - .5, time + l * volume) + .5;
}
else {
    uv = (uv - .5) * (uv - .5) + .5;
    uv = scale(uv, 1. / mod(volume * 2., 1.2));
    uv = rotate(uv - .5, time + volume) + .5;
    uv = abs(uv - .5) + .5;
}

gl_FragColor = texture2D(renderBuffer, uv) * 2.;

if (r > 0.1)
{
    o = renderBuffer, 1.1;
}

if (r < 0.1)
{
    o = renderBuffer, 1.1;
}

upscale(ckbuffer, uv0) * .7;
```

```
aka-171222b.lp aka-171222.lp aka-171222.lp dist-admp.prc m-7002.prc
40 al1Ps "106" >> lowerStableGlobalDensity
41 pf
42 psc
43 lowerGlobalDensity >> core >> setEmsPattern "kitFpPattern7" >> setEmsPattern
"enshrainBlissAtmoPattern1" >> al1Ps "LA8106" >> solo "atmo stab2"
44
45
Type :qal and press <Enter>...will all changes and exit via 43,1 28%
[2] "renicklsnare:/conduc" 22:44 22-Dec-17
```

Player stab2 has been paused.
>>Player perc1 has been paused.

The beat for player atmo to play on is in the future.
"atmo"
currentBeat = 4688.182888785434
playerBeat = 4688.0
diff = 7.8179192145662455
Sleeping and trying again at that time.

今日のメニュー

- ・ GLSLの基礎
- ・ 初級編: RGBグリッチ、色収差
- ・ 発展編: VHS、水彩、アスキーアート
 - ・ 時間みて適当に削ります

スローガン: わからなくとも気にしない

- ・ 細かい計算はあとでサンプル見ればいいです
- ・ 「GLSLでこんなことが出来るんだ～」
という雰囲気を掴んでください

circle.frag

circle2.frag

```
precision mediump float;
uniform float time;
uniform vec2 resolution;
uniform sampler2D backbuffer;
uniform sampler2D spectrum;
// You can use GLSL Sandbox style uniform variables.

vec2 rotate(in vec2 p, in float t) {
    return mat2(sin(t), cos(t), cos(t), -sin(t)) * p;
}

// Hi there!
// This is a demo of GLSL Livecoder!

void main() {
    vec2 p = (gl_FragCoord.xy / resolution) * 2. - .5;
    vec2 uv = (p * .3 + .5);
    p = rotate(p, 2. * time);

    // Audio input is also available!!
    float h = texture2D(spectrum, vec2(abs(p.x * 0.5), .0)).r;
    float m = texture2D(spectrum, vec2(abs(p.x * 0.3), .0)).r;
    float l = texture2D(spectrum, vec2(abs(p.x * 0.1), .0)).r;

    gl_FragColor = vec4(h, m, l, 1.) * (.05 / abs(length(p) - .3));
}
```



GLSLとは

- ・ OpenGLやWebGLのシェーダー言語
- ・ 3DCG、画像処理などで利用される
- ・ VJエフェクトとしてもよく使われる
 - ・ TouchDesigner, VDMX, etc…

GLSLだけでアニメーション作品作る人も.....



<https://www.youtube.com/watch?v=jB0vBmiTr6o>

TouchDesignerで使えるシェーダー

- ・ 頂点シェーダー
- ・ ジオメトリシェーダー
- ・ フラグメントシェーダー
- ・ コンピュートシェーダー

TouchDesignerで使えるシェーダー

- ・ 頂点シェーダー
- ・ ジオメトリシェーダー
- ・ フラグメントシェーダー ← 今日はこれだけ
- ・ コンピュートシェーダー

フラグメントシェーダー

- ・ 面の色を計算するシェーダー
 - ・ ライティング計算したり、テクスチャを貼ったり
- ・ 全ピクセルで毎フレーム実行される
 - ・ 1280x720, 60fpsだと 55296000 回 / 秒!!

**サンプルファイル
開けましたか？**

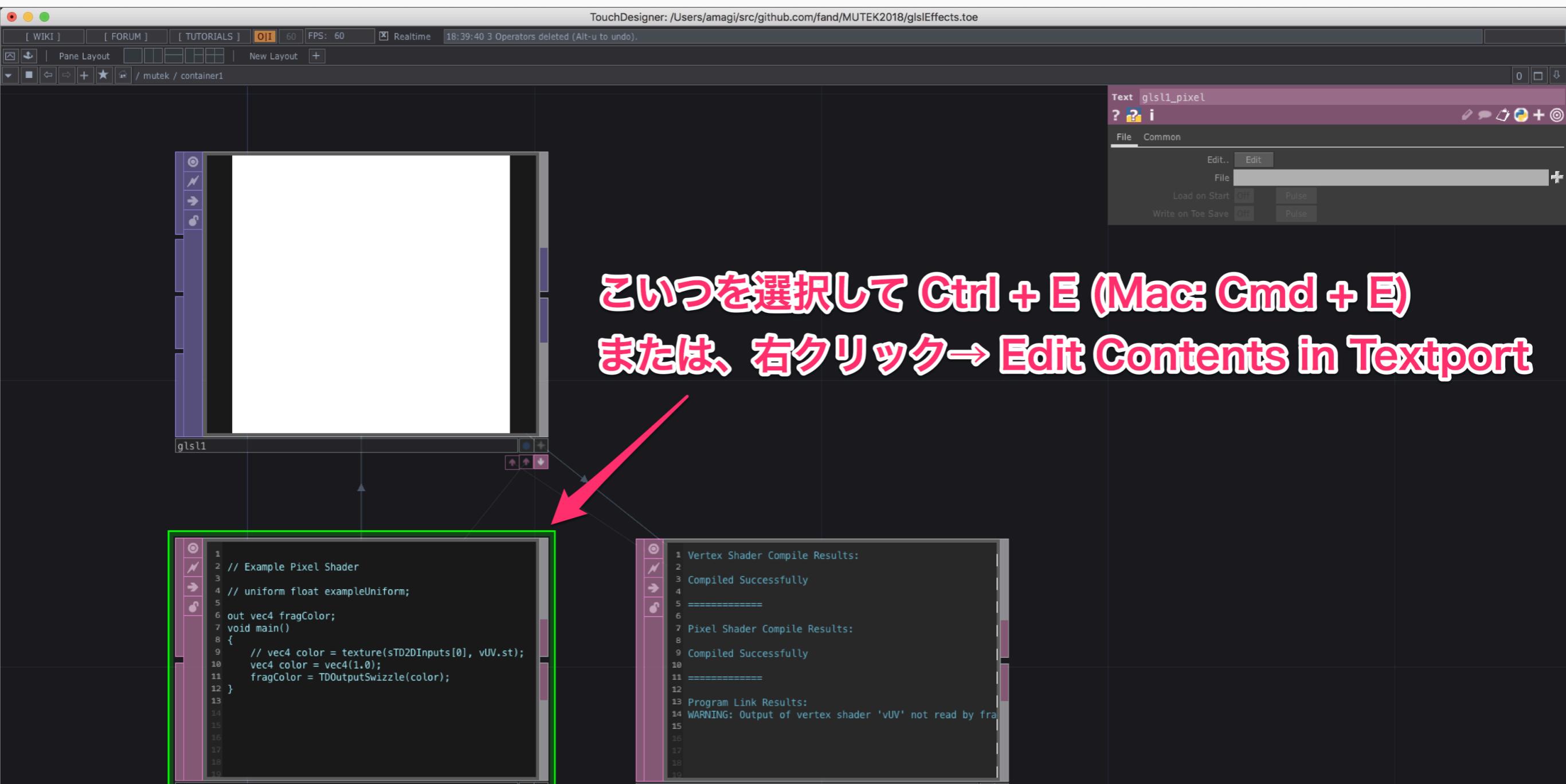
GLSL TOP

introduction

この章の目標

- ・ フラグメントシェーダーだけで図形を描く
- ・ GLSLの基礎文法を学ぶ

適当にGLSL TOPを作りましょう



こいつを選択して **Ctrl + E (Mac: Cmd + E)**
または、右クリック→**Edit Contents in Textport**

GLSL TOPの中身

```
out vec4 fragColor; -  
void main() -  
{ -  
    ... // vec4 color = texture(sTD2DInputs[0], vUV.st); -  
    ... vec4 color = vec4(1.0); -  
    ... fragColor = TDOutputSwizzle(color); -  
}
```

フラグメントシェーダーの流れ

- ・ シェーダーはピクセル毎に実行される
- ・ `gl_FragColor`に、そのピクセルの色を
`vec4(r, g, b, a)` 形式で入れてあげる

値の種類

- int: 整数
- float: 浮動小数点数
- vec2 ~ vec4: ベクトル

int: 整数

- ・ C言語などでおなじみ
- ・ `int a = 1;` と書く
- ・ あんま使わへん
 - ・ forループのインデックスくらい

float: 浮動小数点数

- `float a = 1.0;` 又は `float a = 1.;` と書く
 - 小数点を忘れるときエラーになるので注意！
- ファイル先頭の `precision mediump float;` はfloatの精度を指定している
 - `lowp`, `mediump`, `highp` の順に精度が高くなる
 - だいたい`mediump`で十分

vec2, vec3, vec4: ベクター

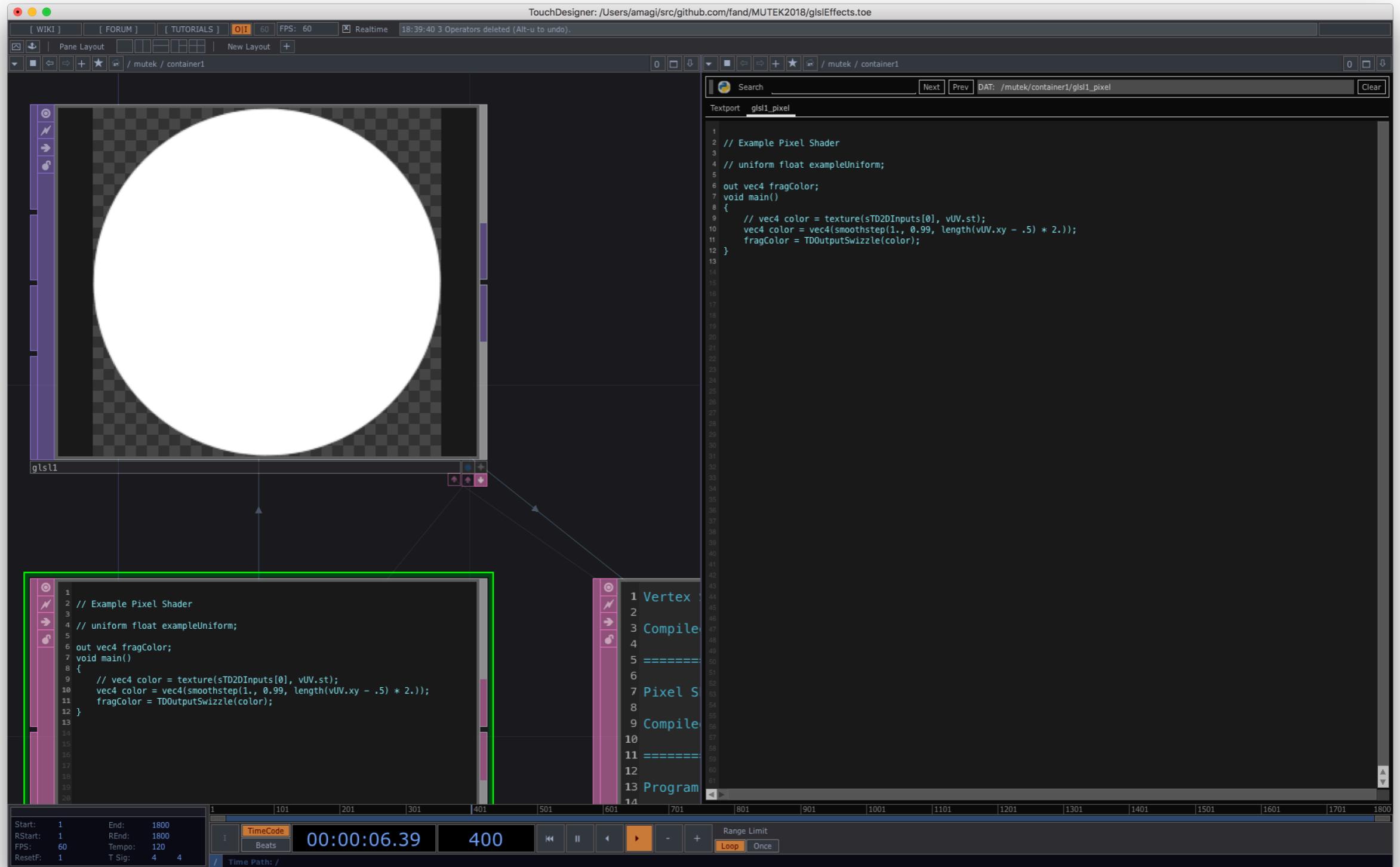
- `vec2 a = vec2(1.0, 2.0);` or `vec2 a = vec2(1.0);`
- `vec2 a = vec2(1);` と書いてもOK
(中身がintになる訳ではない)

```
vec2 a = vec2(1.0, 2.0);  
vec2 b = a.yx;
```

とすると

bは vec2(2.0, 1.0)になる (swizzle)

GLSLで円を描いてみよう

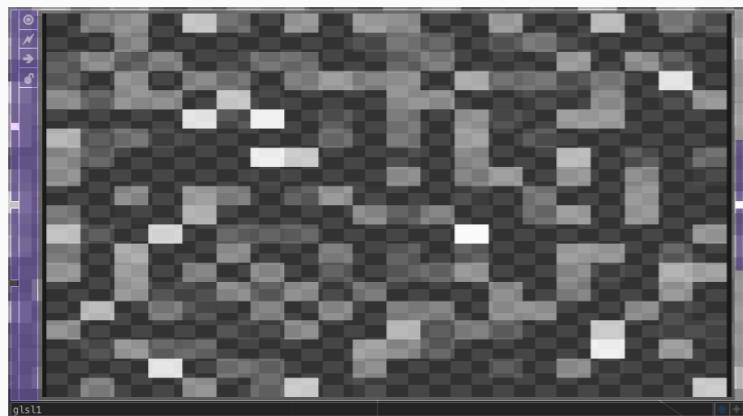




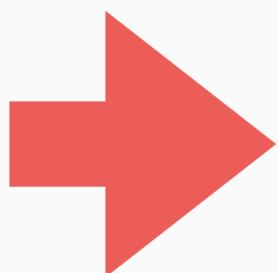
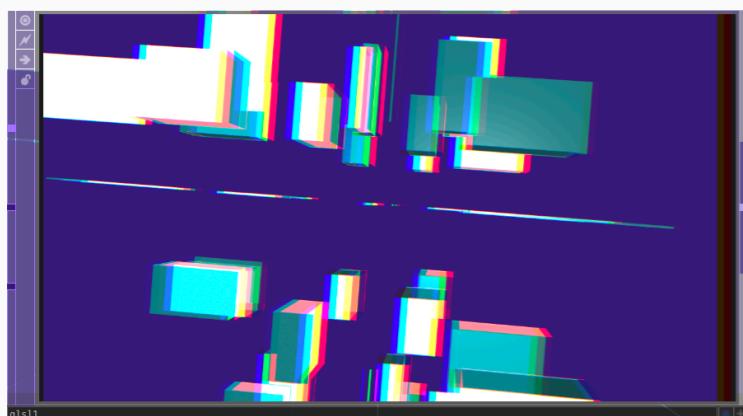
RGB Glitch

処理の流れ

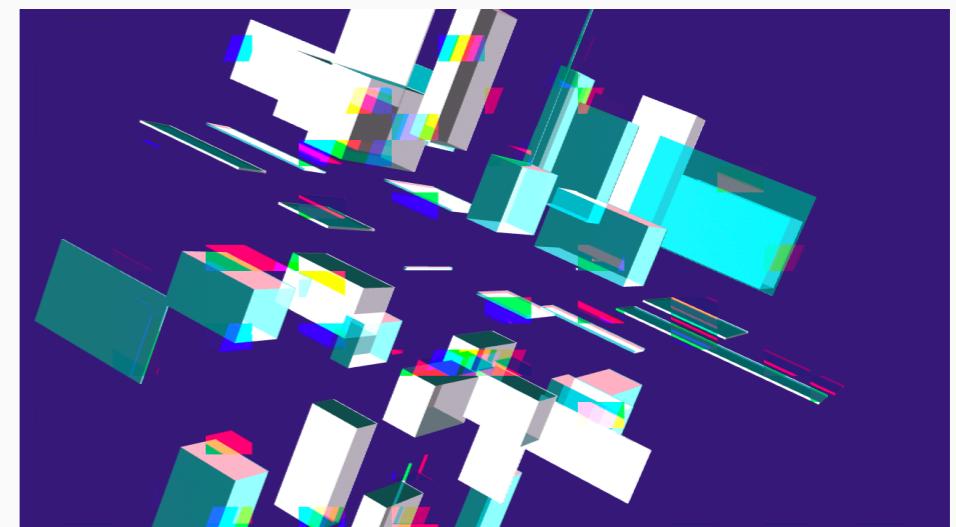
ノイズを作って…



RGBをずらして…



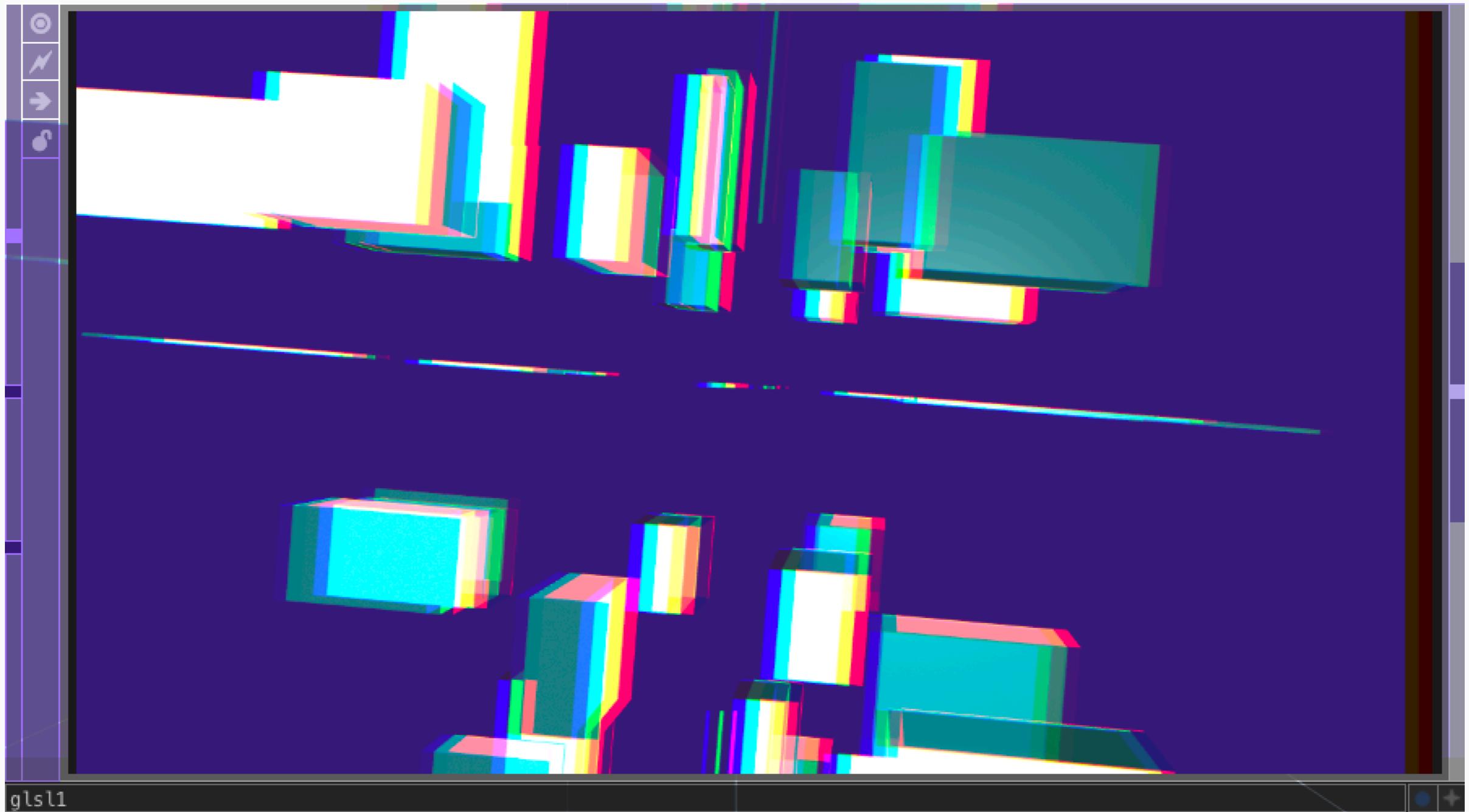
完成！



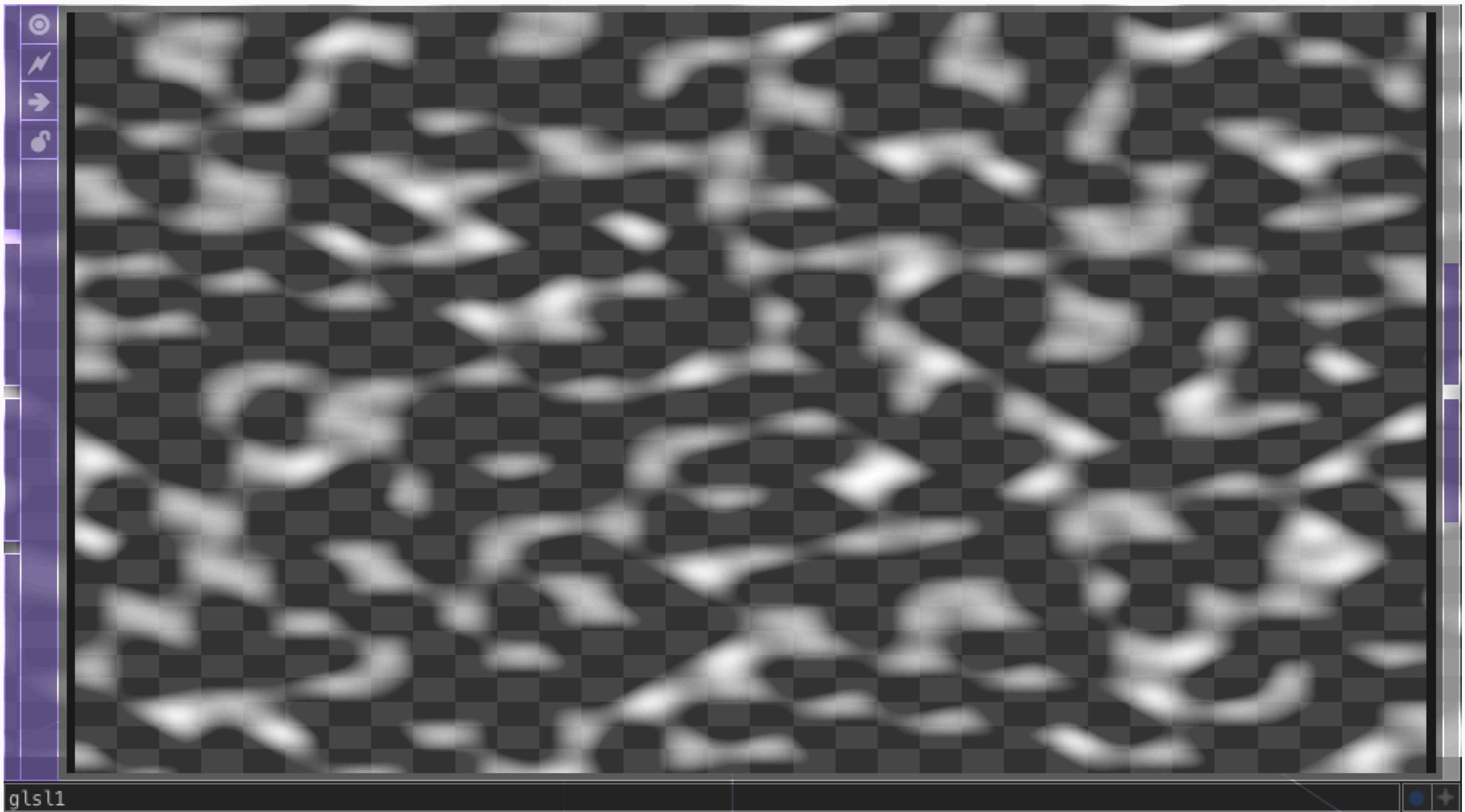
RGBをズらしてみる

```
vec4 color = texture(sTD2DInputs[0], uv);  
color.g = texture(sTD2DInputs[0], uv + vec2(0.01, 0)).g;  
color.b = texture(sTD2DInputs[0], uv + vec2(0.02, 0)).b;
```

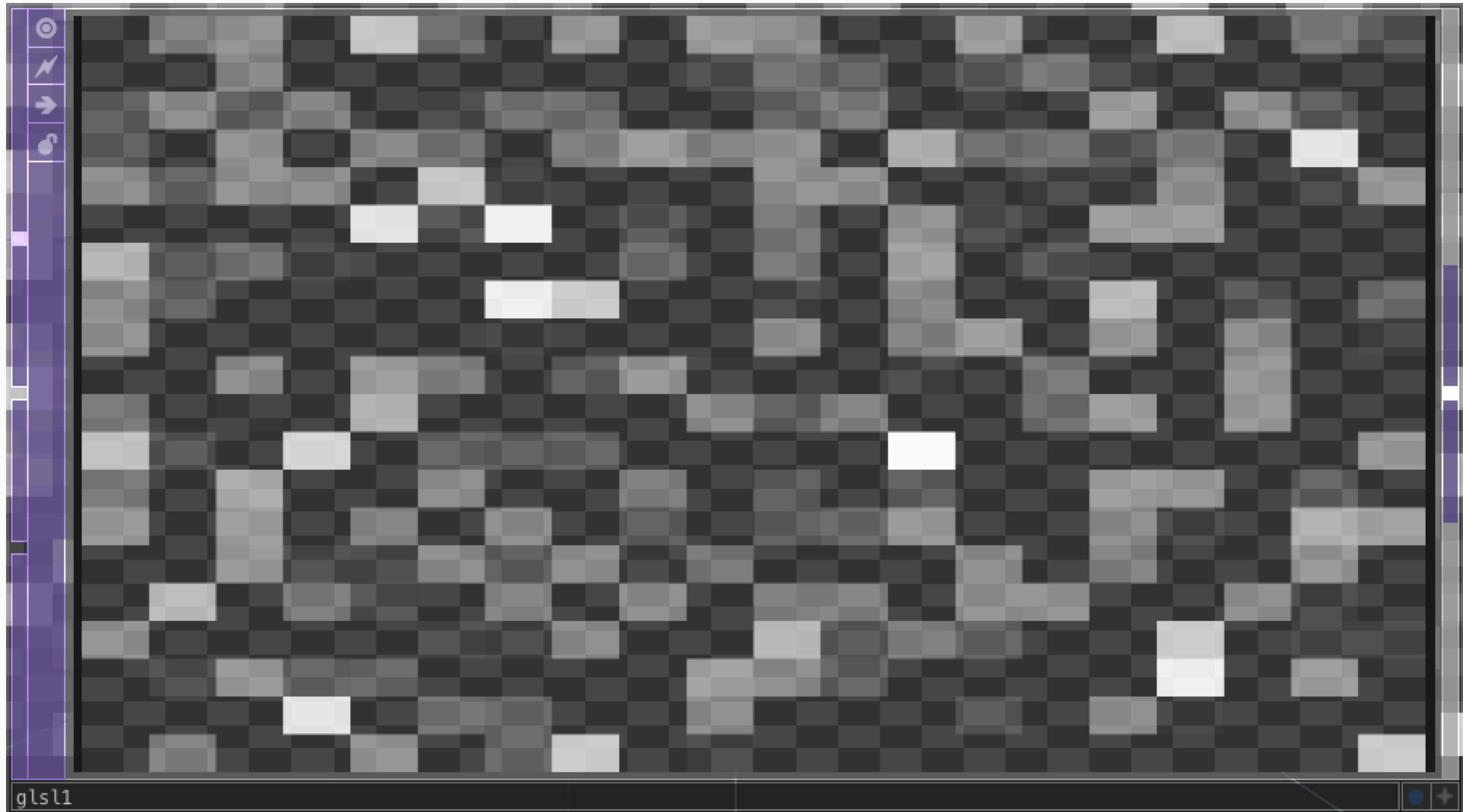
RGBをずらしてみる



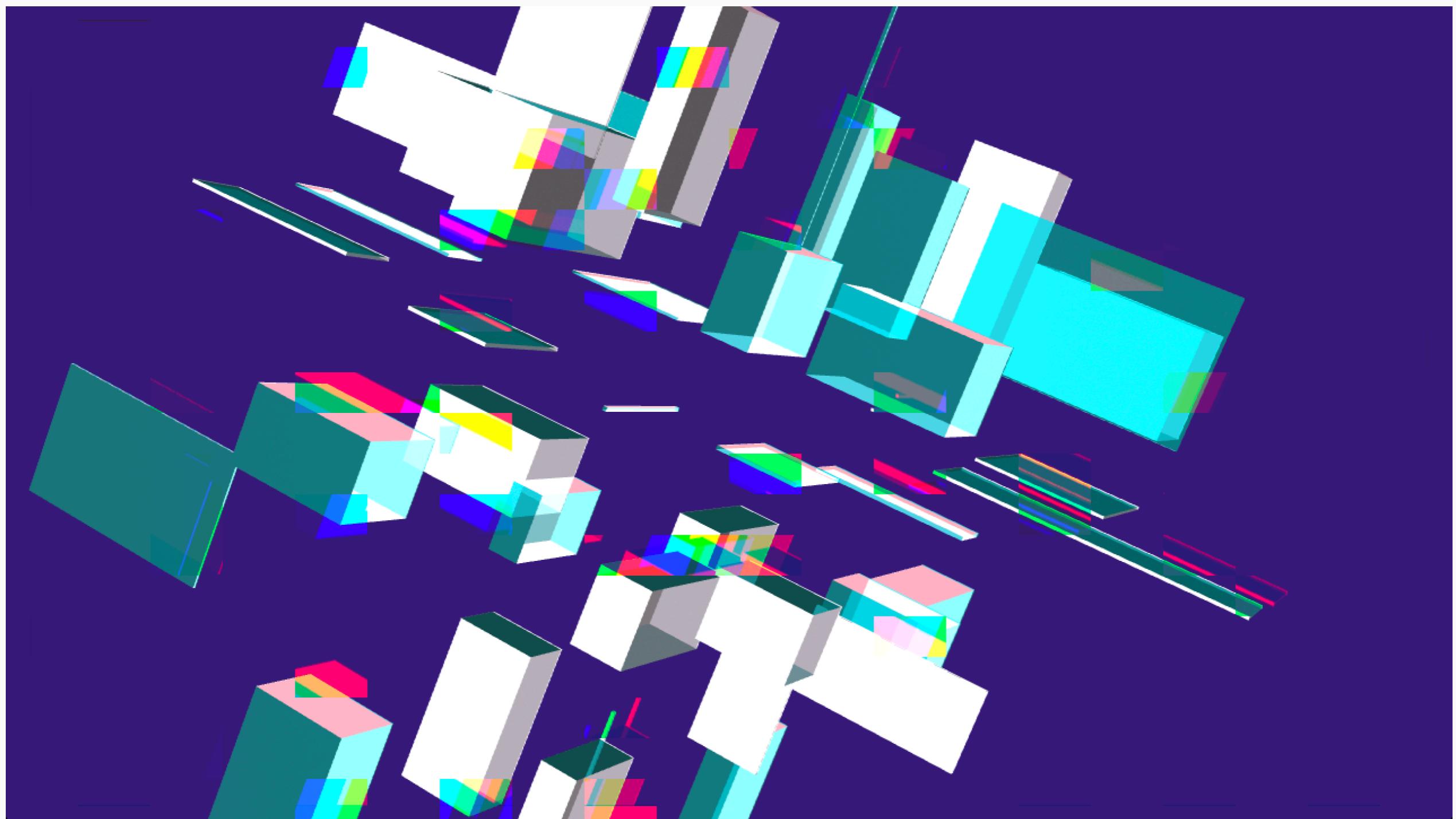
TDSimplexNoiseを使う



ブロックノイズを作る



RGBをブロックノイズでずらす



uniform変数

- ・全ピクセルで共通の変数
- ・TouchDesignerが用意してくれる物もある
(sTD2DInputs等)
- ・自分で設定できる (time, slider等)
 - ・CHOPの値を渡したり、
TOPをテクスチャとして渡したり

The screenshot shows the Houdini interface with several windows open:

- GLSL Editor:** A window titled "glsl1" containing GLSL code for a pixel shader. The code defines uniforms "time" and "slider", and a main function that sets the output color based on UV coordinates and the value of "slider".
- Graph Editor:** A window showing a network of nodes. A node labeled "glsl1" is highlighted with a green border. A red arrow points from the text "CHOPをexportしたりできる" to this node.
- Parameter Editor:** A window titled "glsl1" showing uniform parameters. The "slider" parameter is highlighted with a green background. A red arrow points from the text "エクスプレッションを入れたり" to the "Value" field of the "slider" parameter.
- Compile Results:** A terminal-like window at the bottom showing the compilation status: "Compiled Successfully".

**GLSL TOPのParametersに
エクスプレッションを入れたり
CHOPをexportしたりできる**

```
1 uniform float time;
2 uniform float slider;
3 out vec4 fragColor;
4
5 void main() {
6     vec2 uv = vUV.st;
7     vec4 color = vec4(0);
8 }
```

今回使った組み込み関数

- `floor(x)`: x の小数部分を切り捨てる
- `fract(x)`: x の小数部分だけを抜き出す
- `step(n, x)`: x が n 以上なら1, n 未満なら0

The background consists of numerous 3D cubes of various sizes and colors, including shades of blue, green, yellow, red, and purple, all overlapping each other to create a sense of depth and motion.

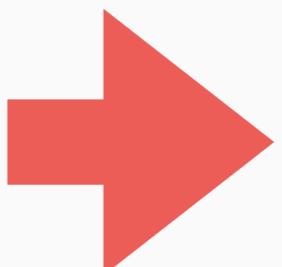
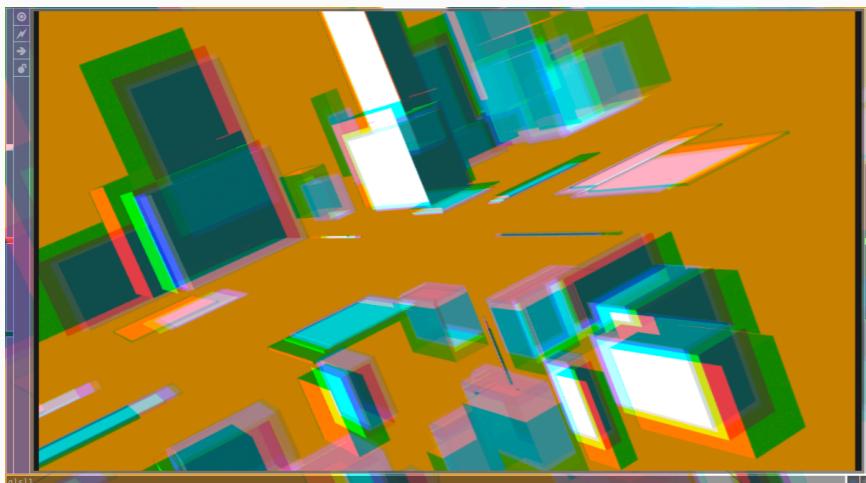
Chroma

色収差エフェクト

- ・画面の端でRGBがずれる効果
 - ・トイカメラとかVRゴーグルで見るあれ
- ・ついでに画面端を暗くするとそれっぽいぞ！
(周辺減光、Vignette)

処理の流れ

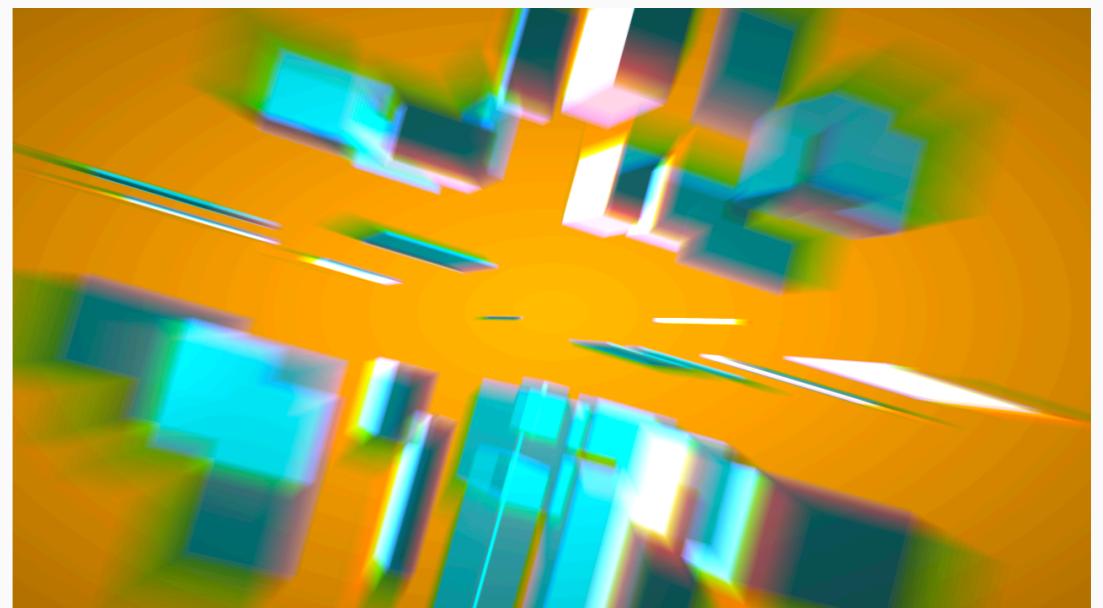
円形にRGBずらして…



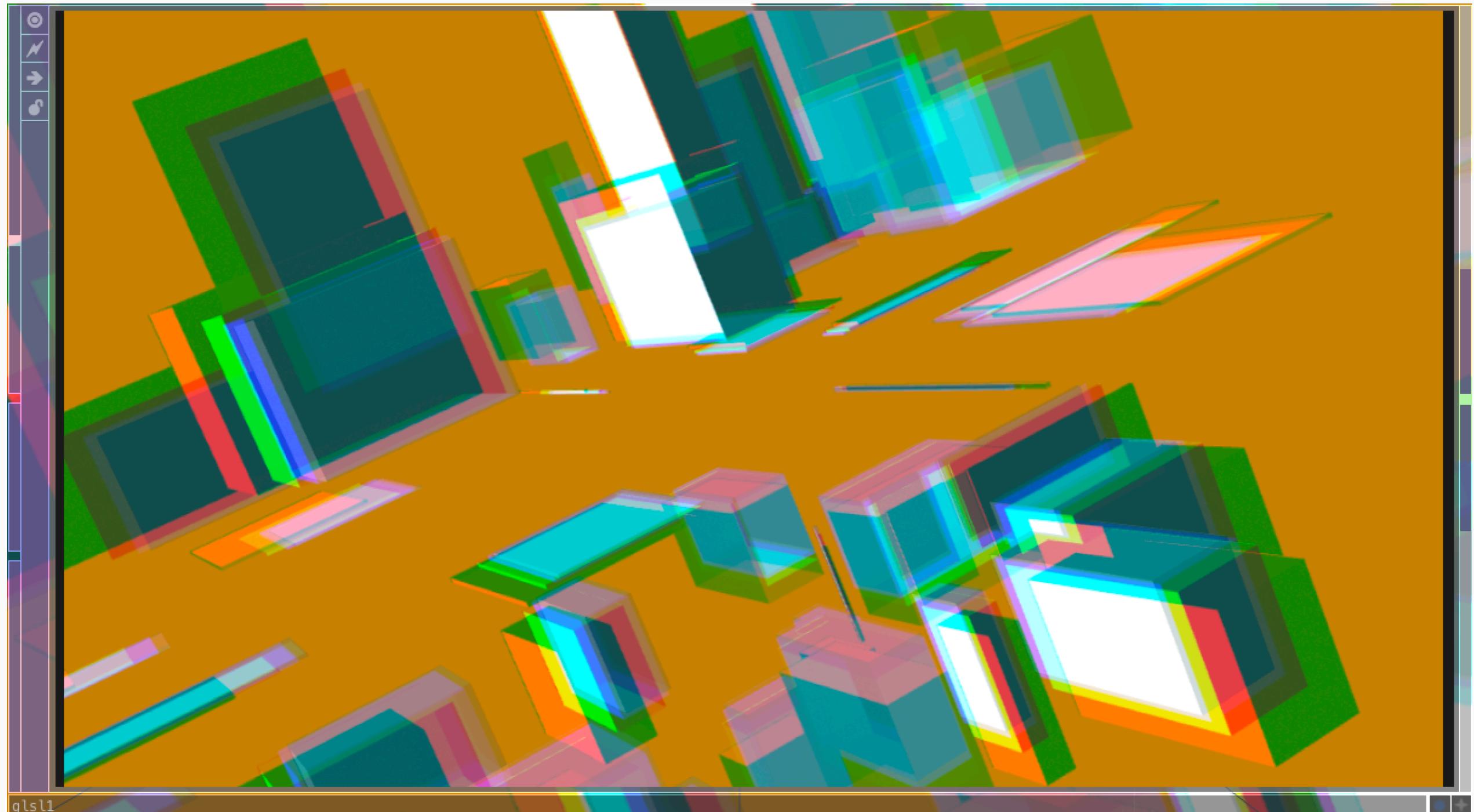
重ねてボカすと…



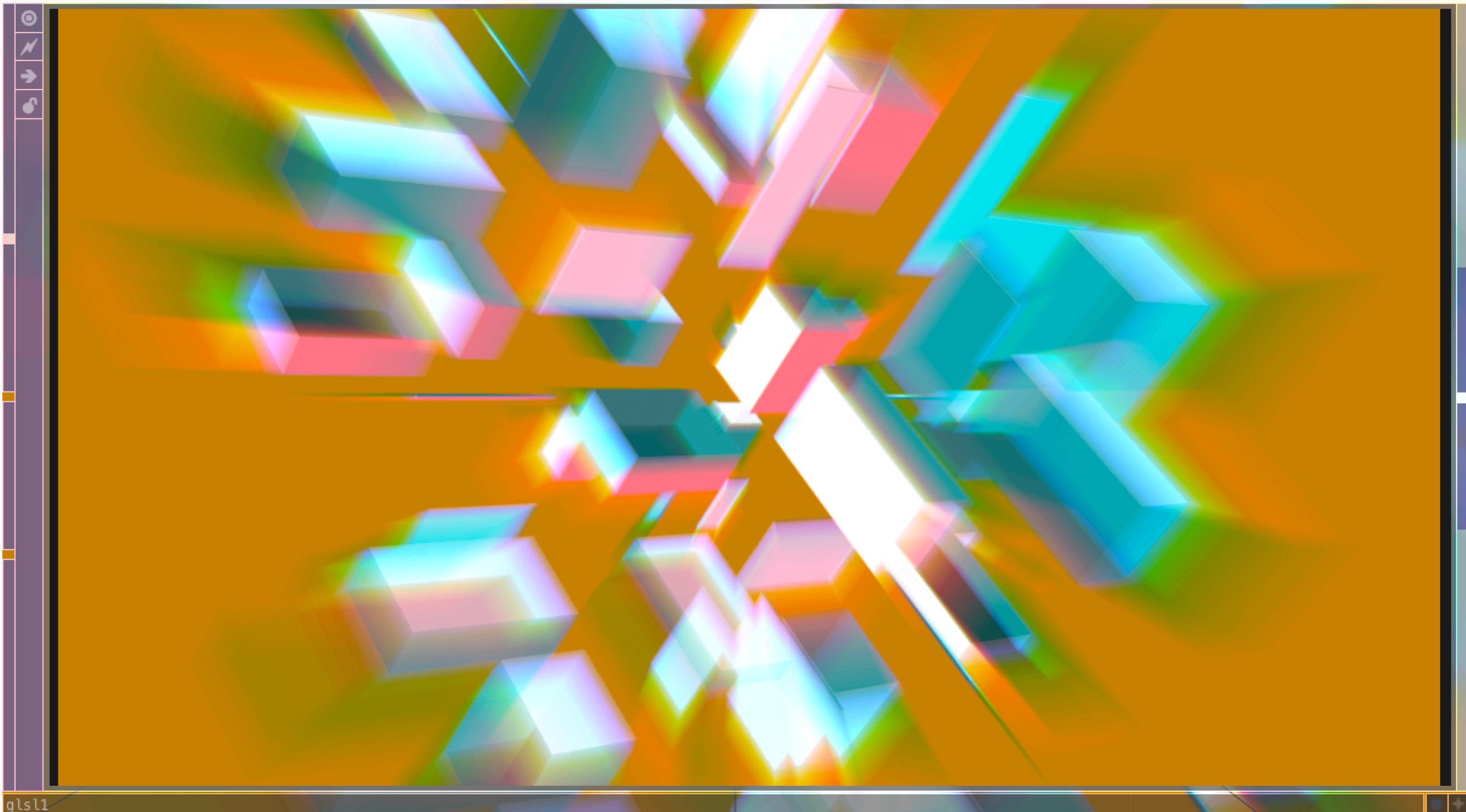
完成！



中心からの距離でズラす



重ねてボカす



重ねてボカす

- ・ ずらし幅を少しづつ増やして色を足していく、
平均をとるといい感じにボケる
- ・ forループを使います

```
// ずらしつつ平均を求める

vec4 blur = vec4(0);

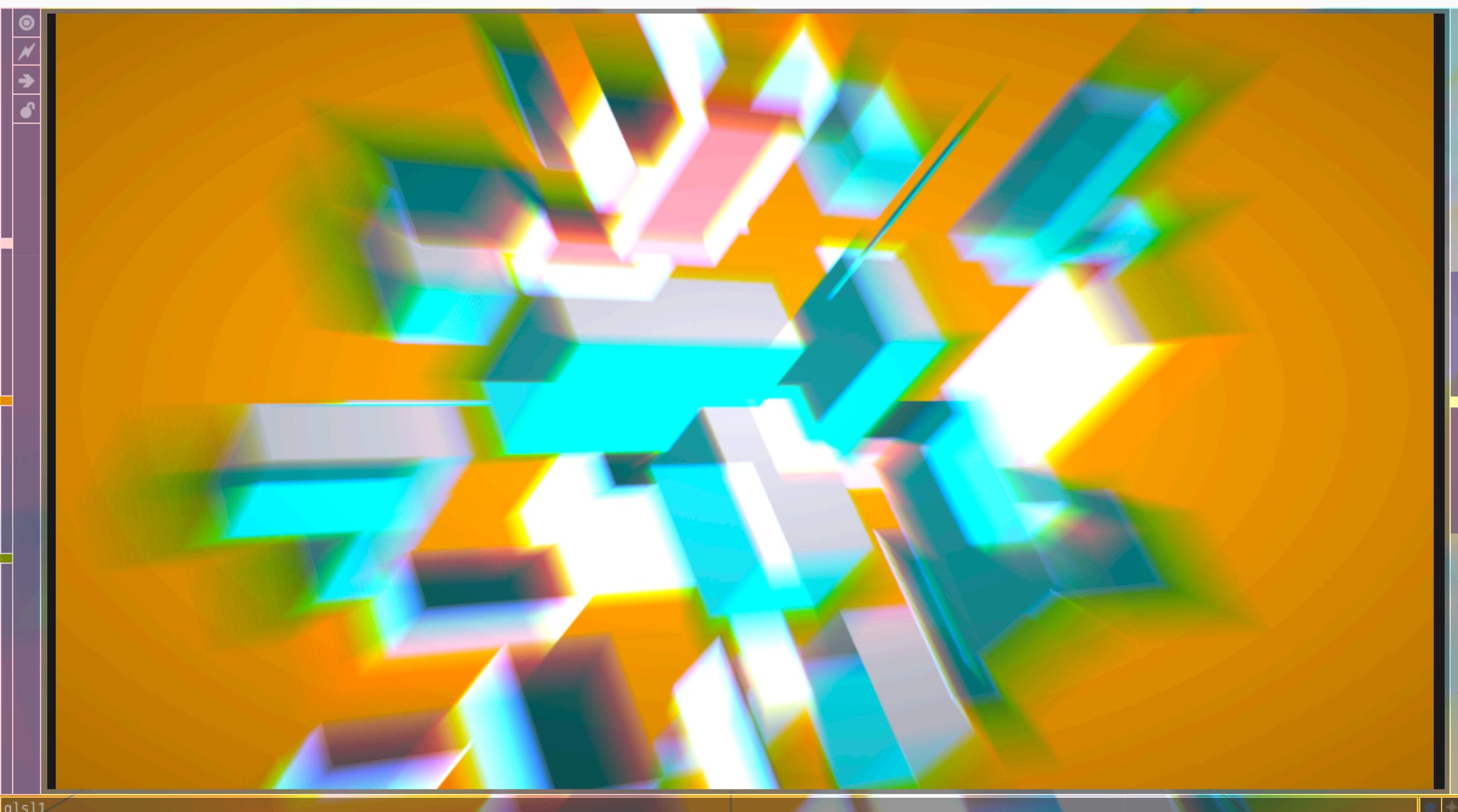
int N = 20;

for (int i = 0; i < N; i++) {
    // ずらし幅の基準
    float d = float(i) * slider * 0.002;

    // Bが内側、Rが外側になるようにズラす
    blur.r += texture(sTD2DInputs[0], scale(uv, d * 7.)).r;
    blur.g += texture(sTD2DInputs[0], scale(uv, d * 3.)).g;
    blur.b += texture(sTD2DInputs[0], scale(uv, d * 1.)).b;
}

blur /= float(N);
blur.a = 1.0;
```

周辺減光を追加



```
// ずらしつつ平均を求める
vec4 blur = vec4(0);
int N = 20;
for (int i = 0; i < N; i++) {
    // ずらし幅の基準
    float d = float(i) * slider * 0.002;
    // Bが内側、Rが外側になるようにズラす
    blur.r += texture(sTD2DInputs[0], scale(uv, d * 7.)).r;
    blur.g += texture(sTD2DInputs[0], scale(uv, d * 3.)).g;
    blur.b += texture(sTD2DInputs[0], scale(uv, d * 1.)).b;
}
blur /= float(N);
blur.a = 1.0;
```

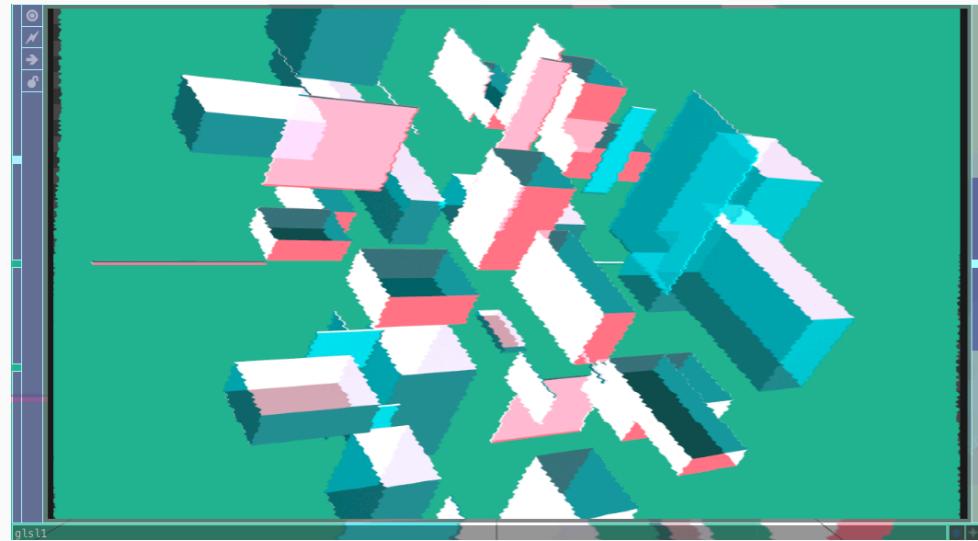
今回使った組み込み関数

- `float(x)`: intをfloatに変換する
- `mix(a, b, t)`: aとbを $t_{(0 \sim 1)}$ の値で混ぜ合わせる
- `for (int i = 0; i < N; i++)`: 処理をN回繰り返す

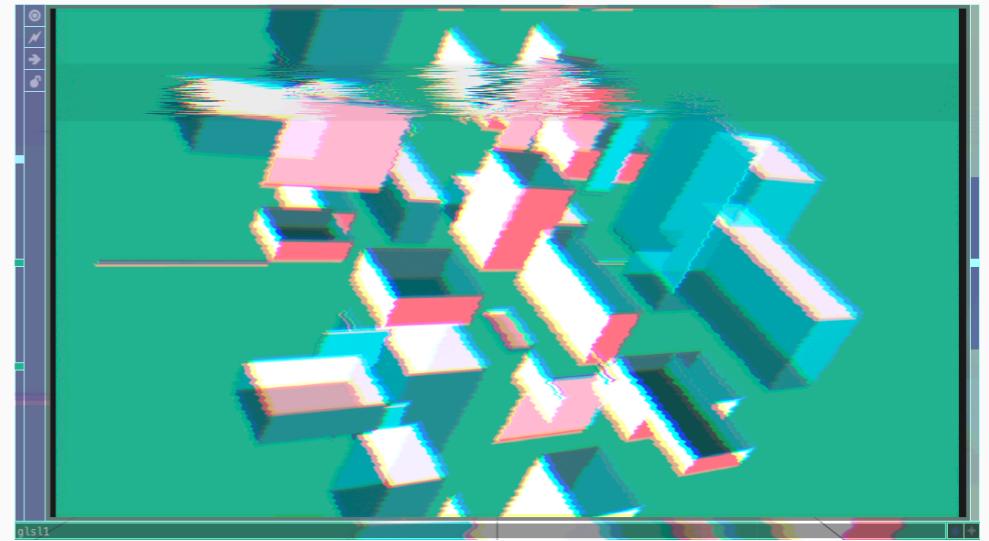
VHS

処理の流れ

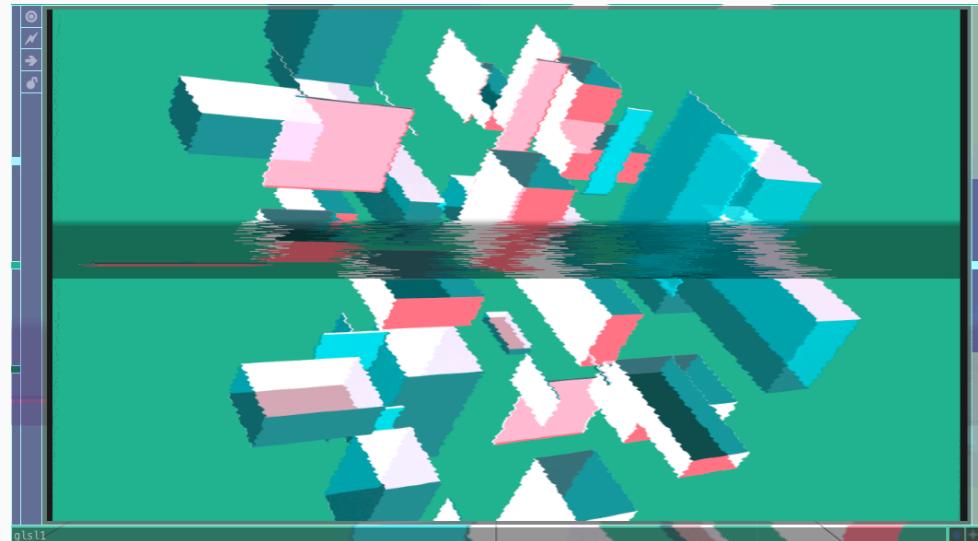
横にゲジゲジさせて…



RGBずらして…



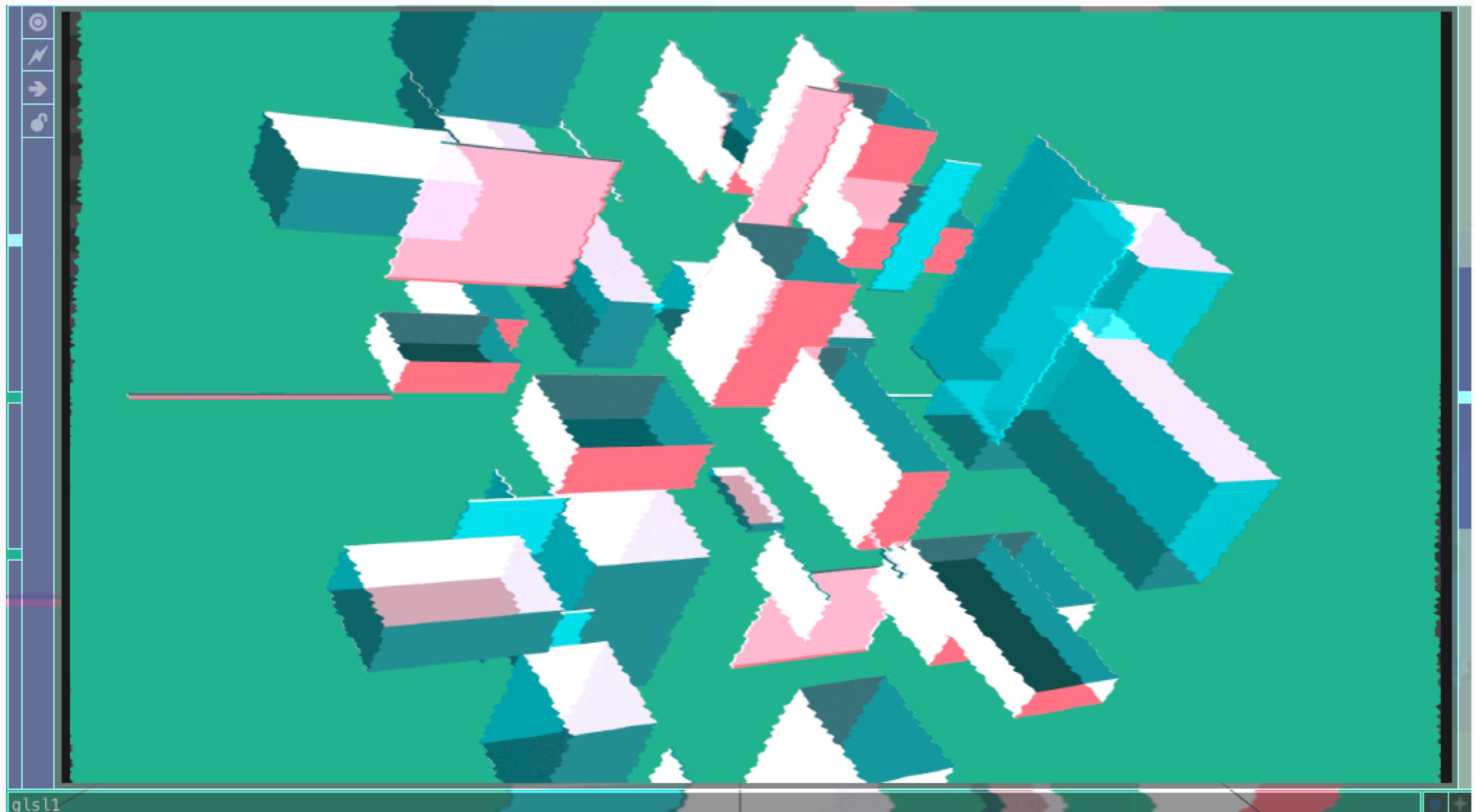
ノイズの帯を追加して…



ザラザラさせたら完成



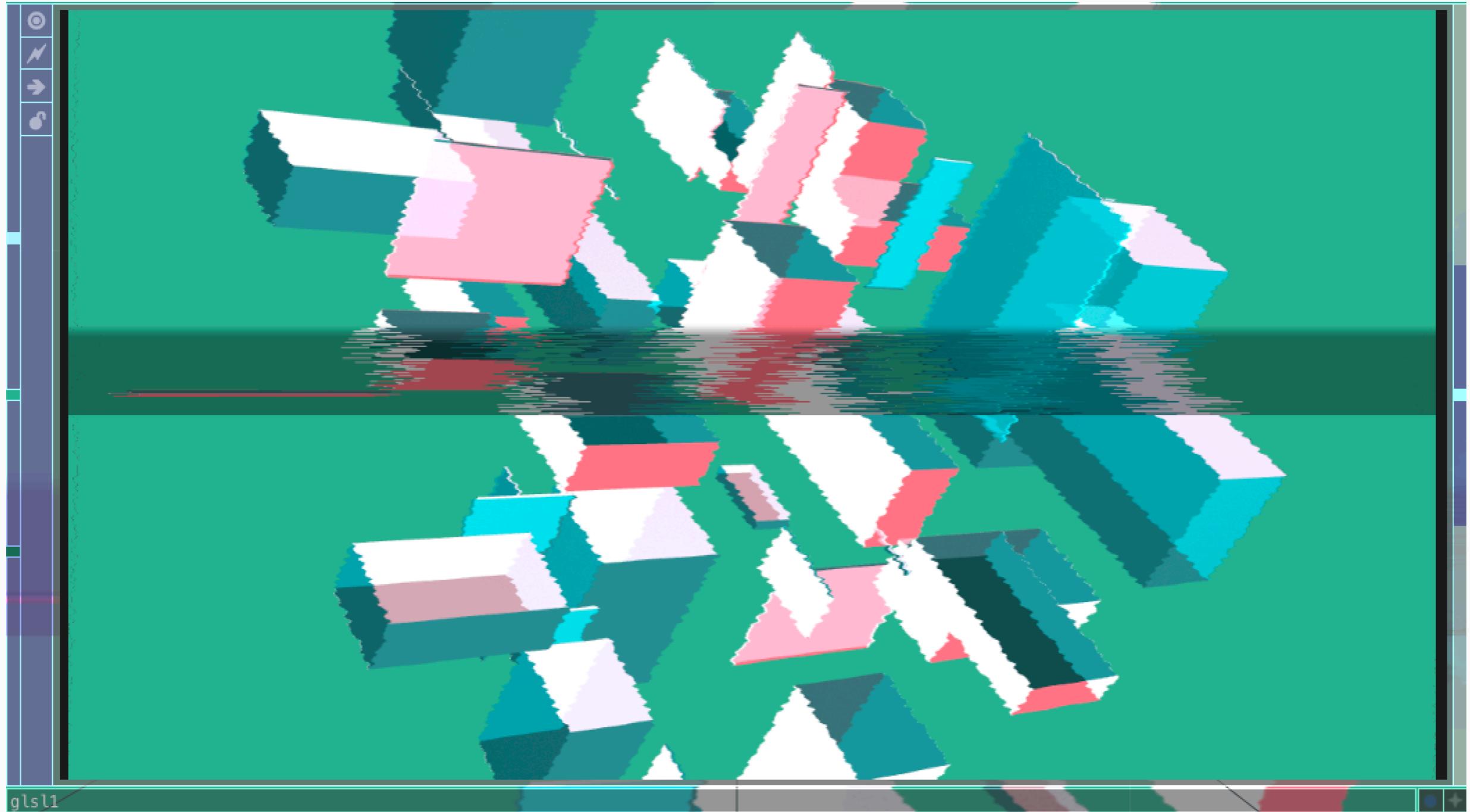
横にゲジゲジさせる



y座標でx座標をズラす

```
uv.x += (-  
....TDSimplexNoise(vec2(uv.y, time)) * 0.01 +  
....TDSimplexNoise(vec2(uv.y * 100., time * 20.)) * 0.003 -  
) * slider;
```

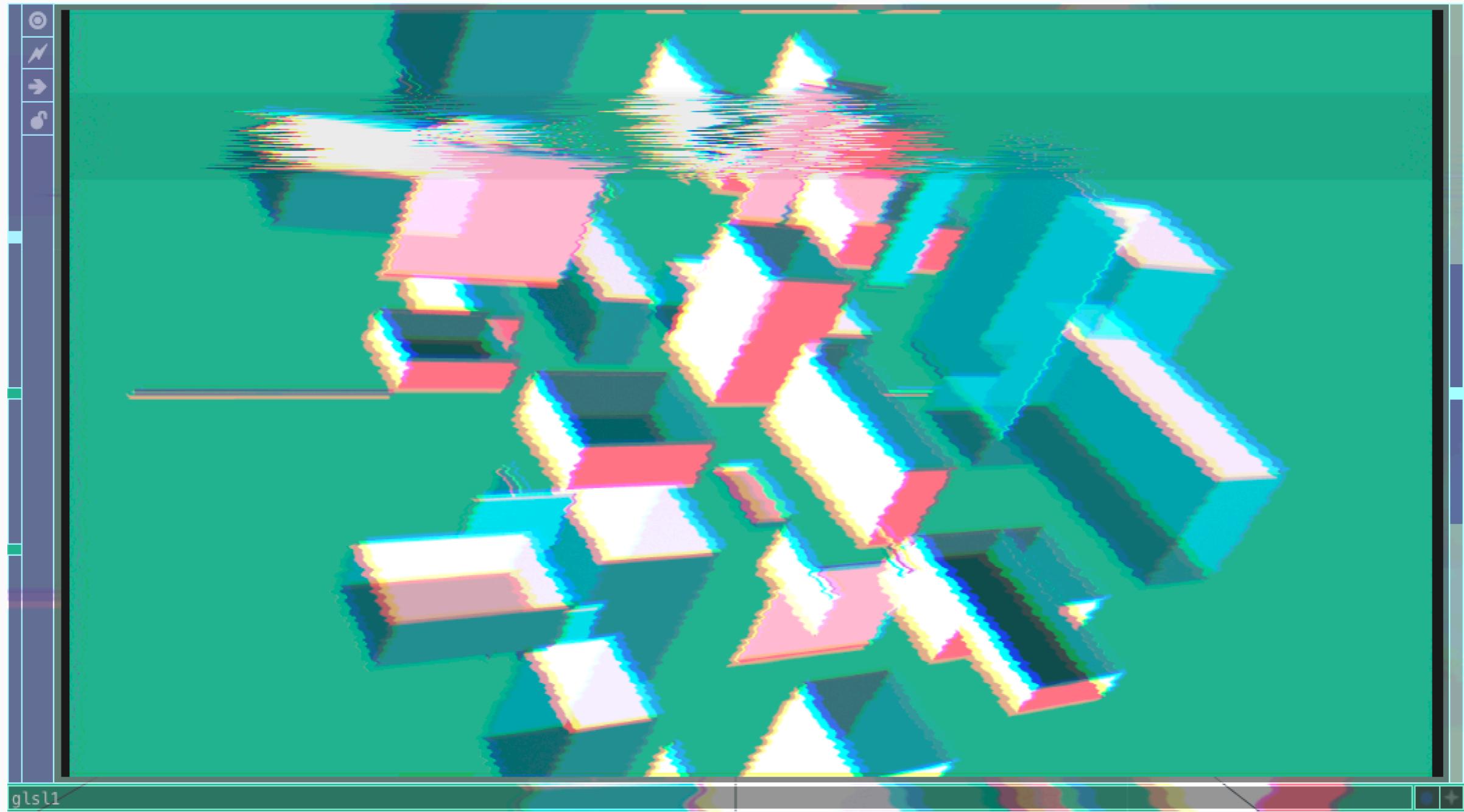
ノイズの帯を追加



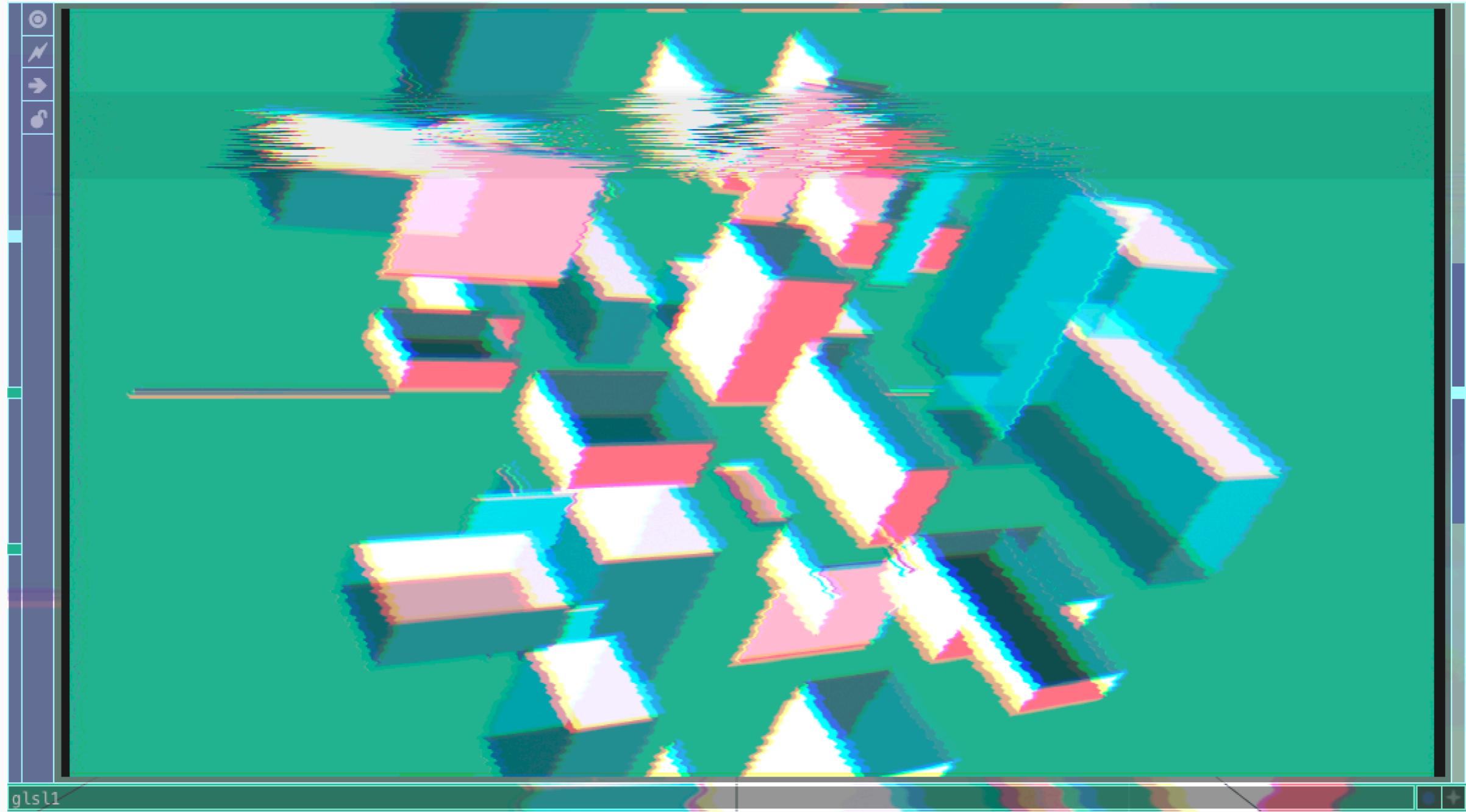
一部分だけノイズを追加

```
// ノイズの帯を追加
float band = 1. - smoothstep(0.1, 0.12, mod(uv.y + time * 0.4, 1.));
uv.x += TDSSimplexNoise(vec2(uv.y * 300., time * 20.)) * band * 0.03 * slider;
uv.x = fract(uv.x);
color.rgb *= 1. - band * 0.4 * slider;
```

x方向にRGBずらす



全体的にザラザラさせる

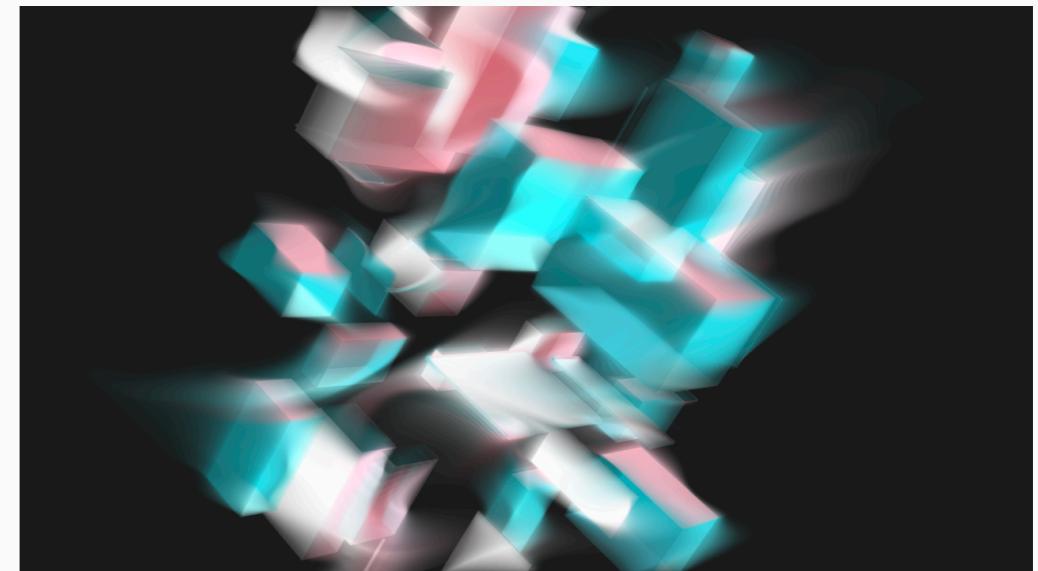
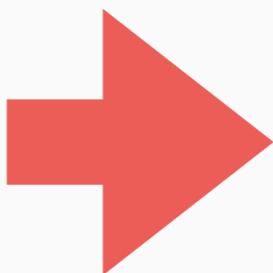
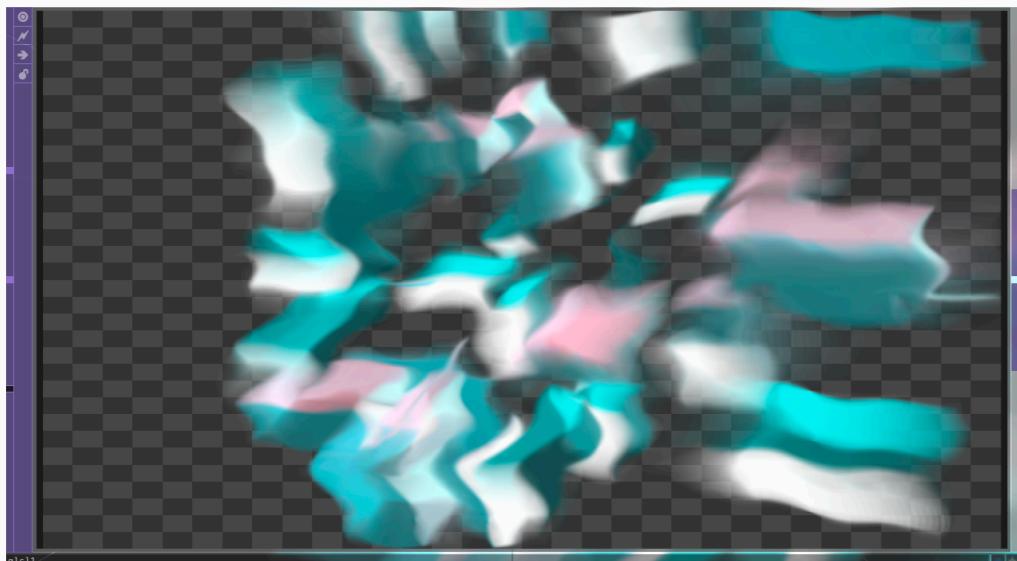


The background consists of a dark, almost black, surface covered with numerous overlapping, translucent polygons. These polygons are primarily colored in shades of teal, red, and grey. They are arranged in a way that creates a sense of depth and movement, resembling a liquid or melting state.

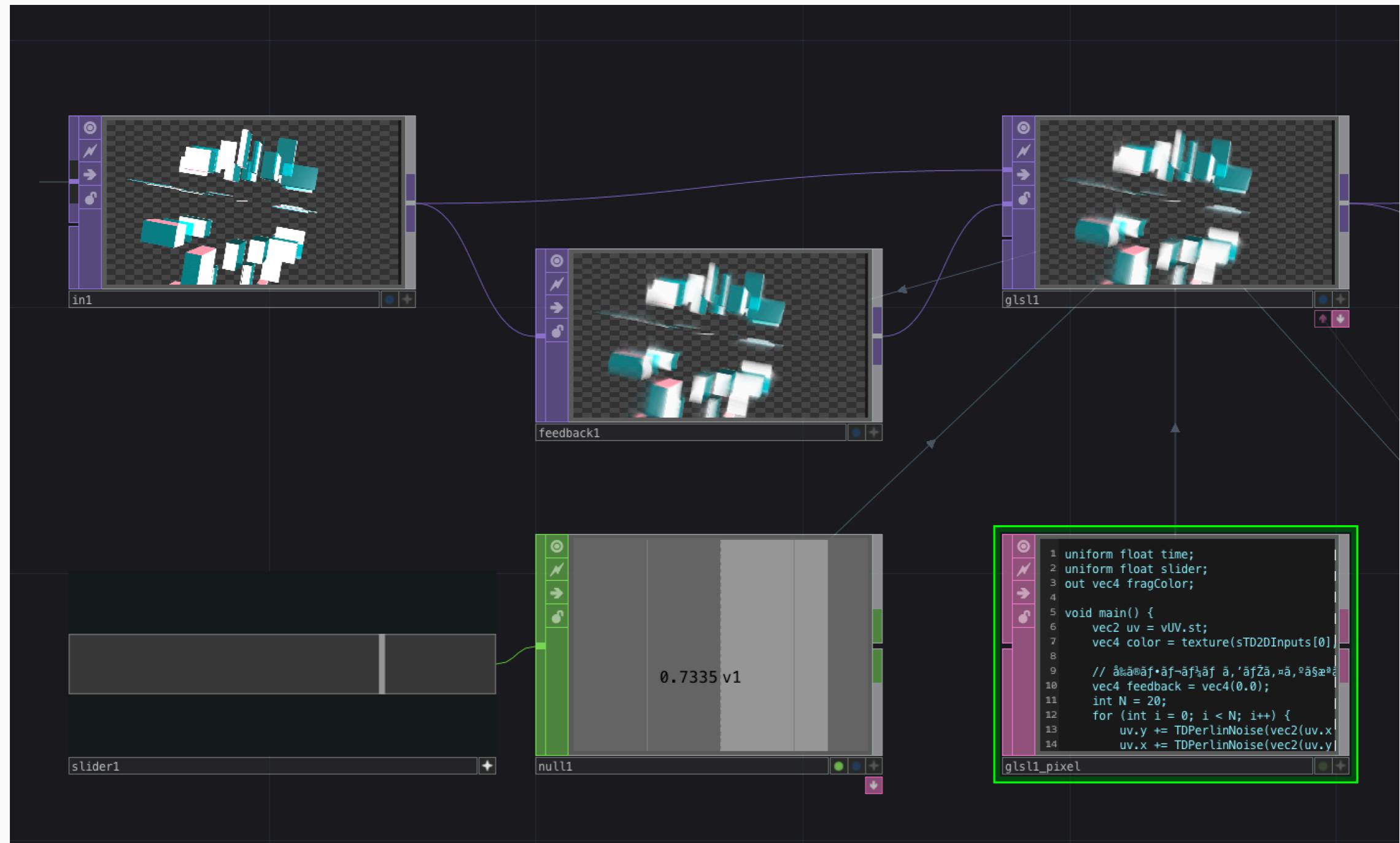
Melt

処理の流れ

フィードバックを歪ませて… 重ねてボカすと完成！



Feedback TOPで前のフレームを取得



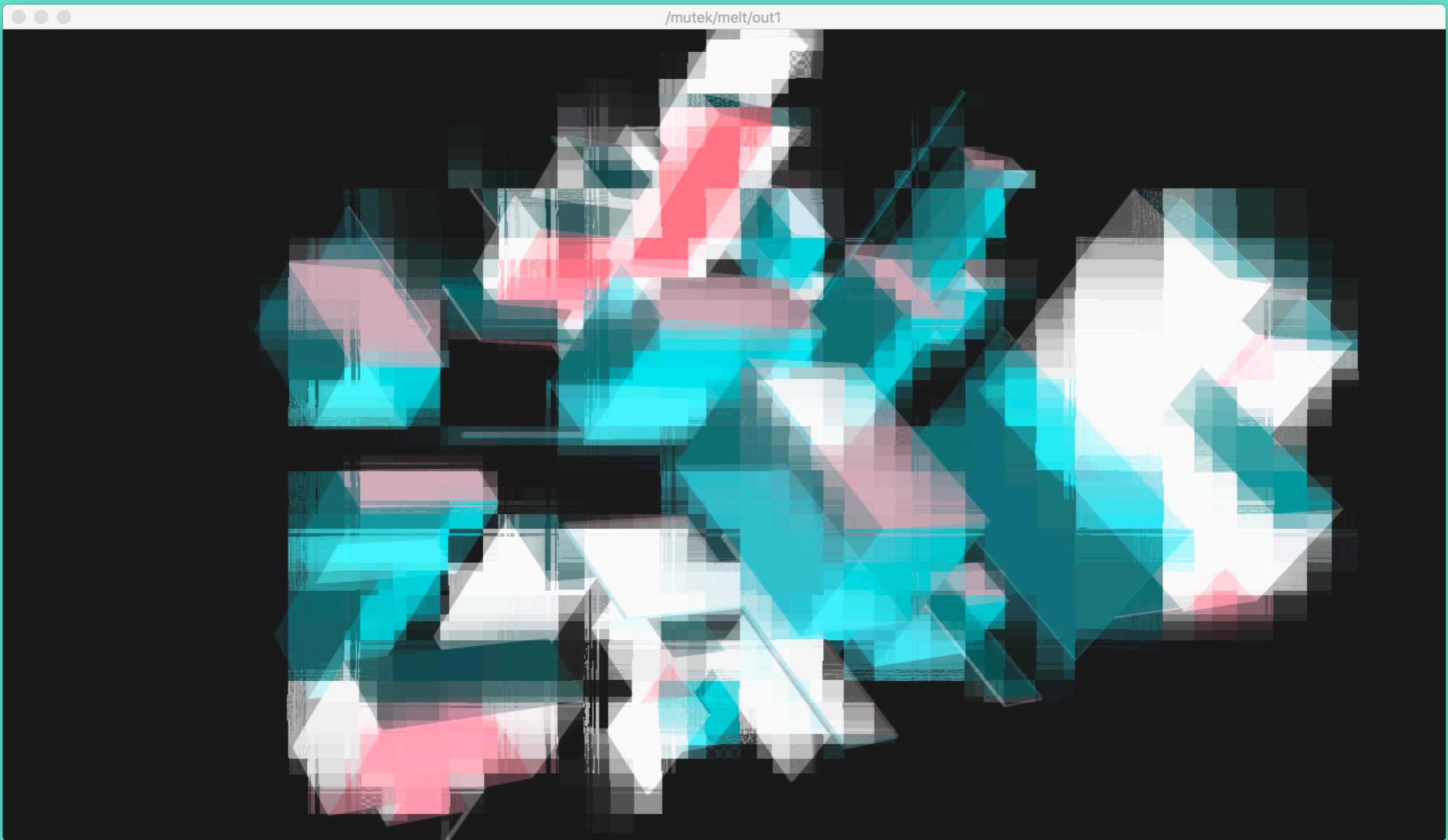
前のフレームを歪ませて

```
uv.y += TDPerlinNoise(vec2(uv.x * 10., time * 0.2)) * 0.001;  
uv.x += TDPerlinNoise(vec2(uv.y * 10., time * 0.2)) * 0.001;  
feedback += texture(sTD2DInputs[1], uv);
```

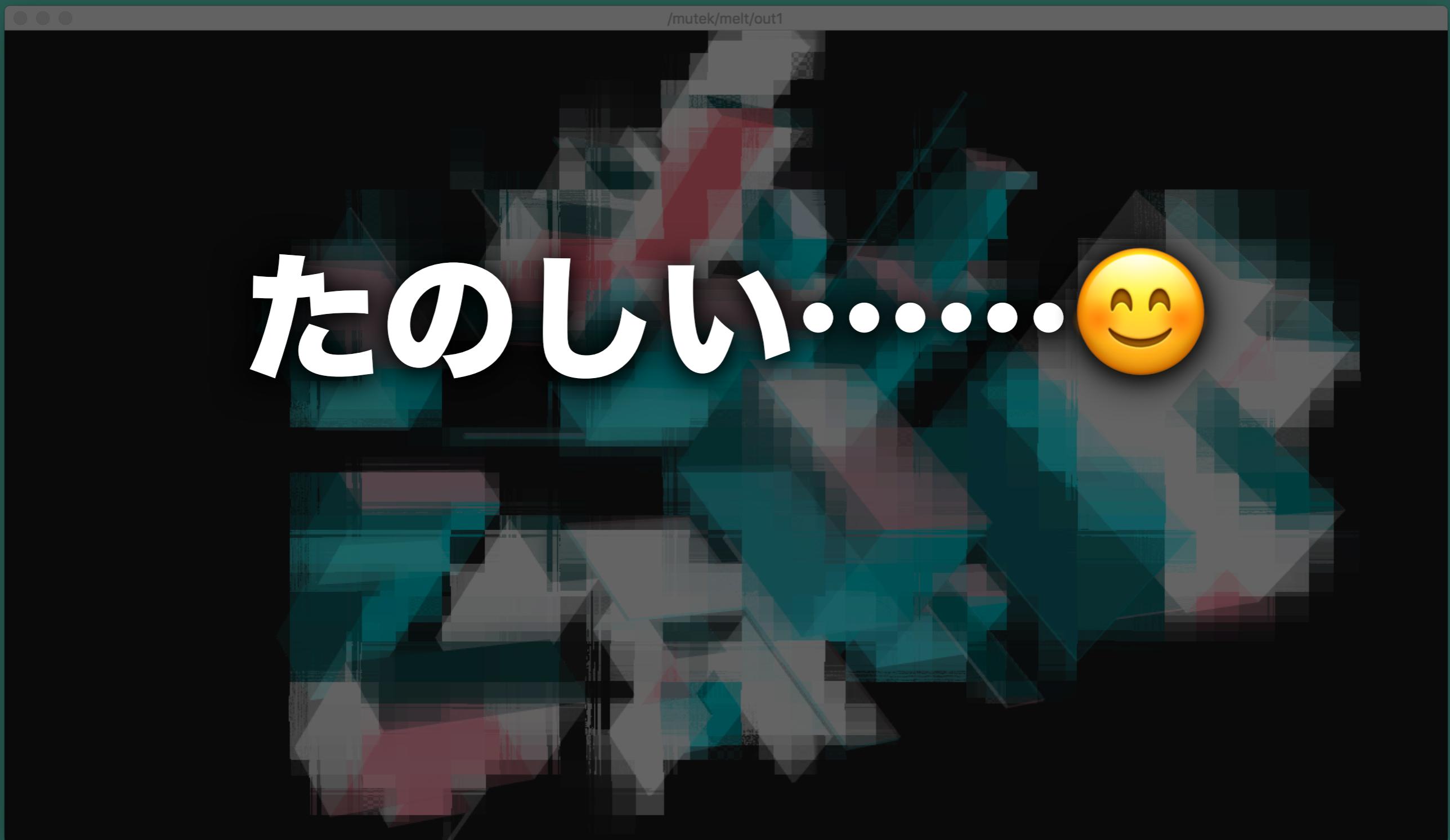
重ねてボカす

```
vec4 feedback = vec4(0.0);  
int N = 20;  
for (int i = 0; i < N; i++) {  
    ... uv.y += TDPerlinNoise(vec2(uv.x * 10., time * 0.2)) * 0.001;  
    ... uv.x += TDPerlinNoise(vec2(uv.y * 10., time * 0.2)) * 0.001;  
    ... feedback += texture(sTD2DInputs[1], uv);  
}  
feedback /= float(N);
```

ノイズ弄るとこうなります



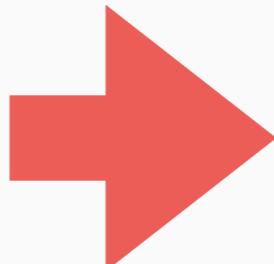
ノイズ弄るとこうなります



ASCI Art

処理の流れ

ブロックの明るさを計算



完成！



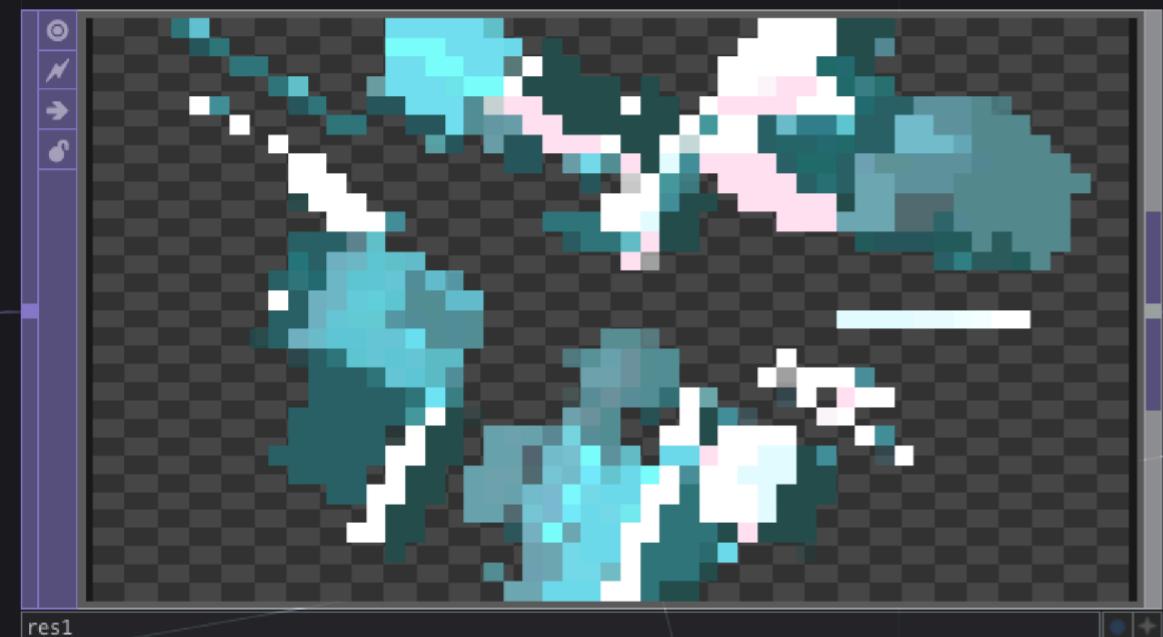
文字テクスチャを参照して



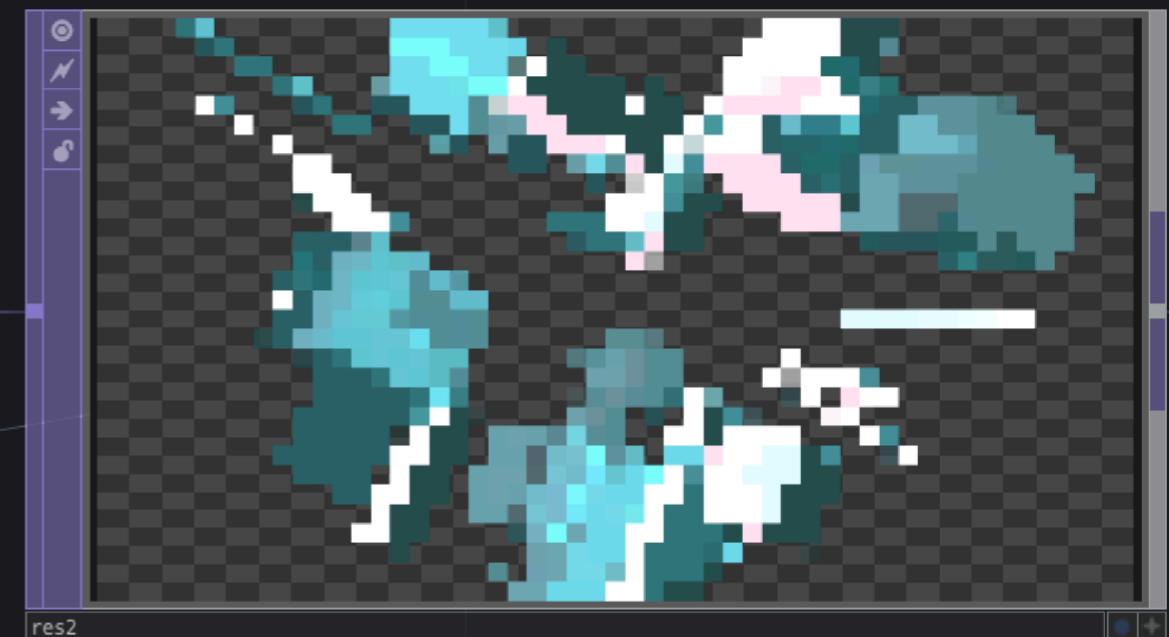
[FORUM] [TUTORIALS] OI 60 FPS: 60 Realtime 3:05:45 Save Project .toe: glsLEffects.toe saved successfully.

New Layout +

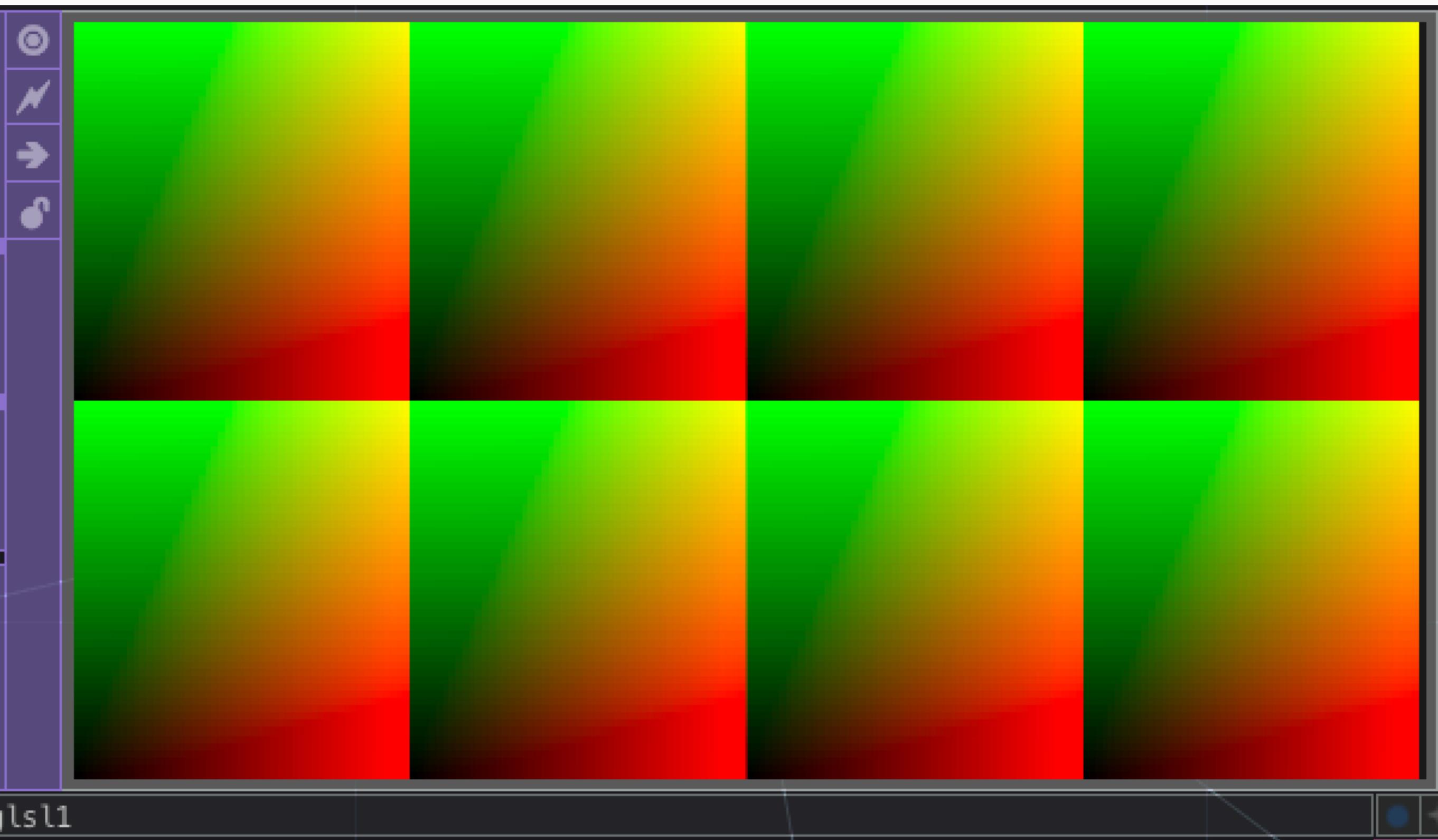
resolution TOPで
解像度を下げる



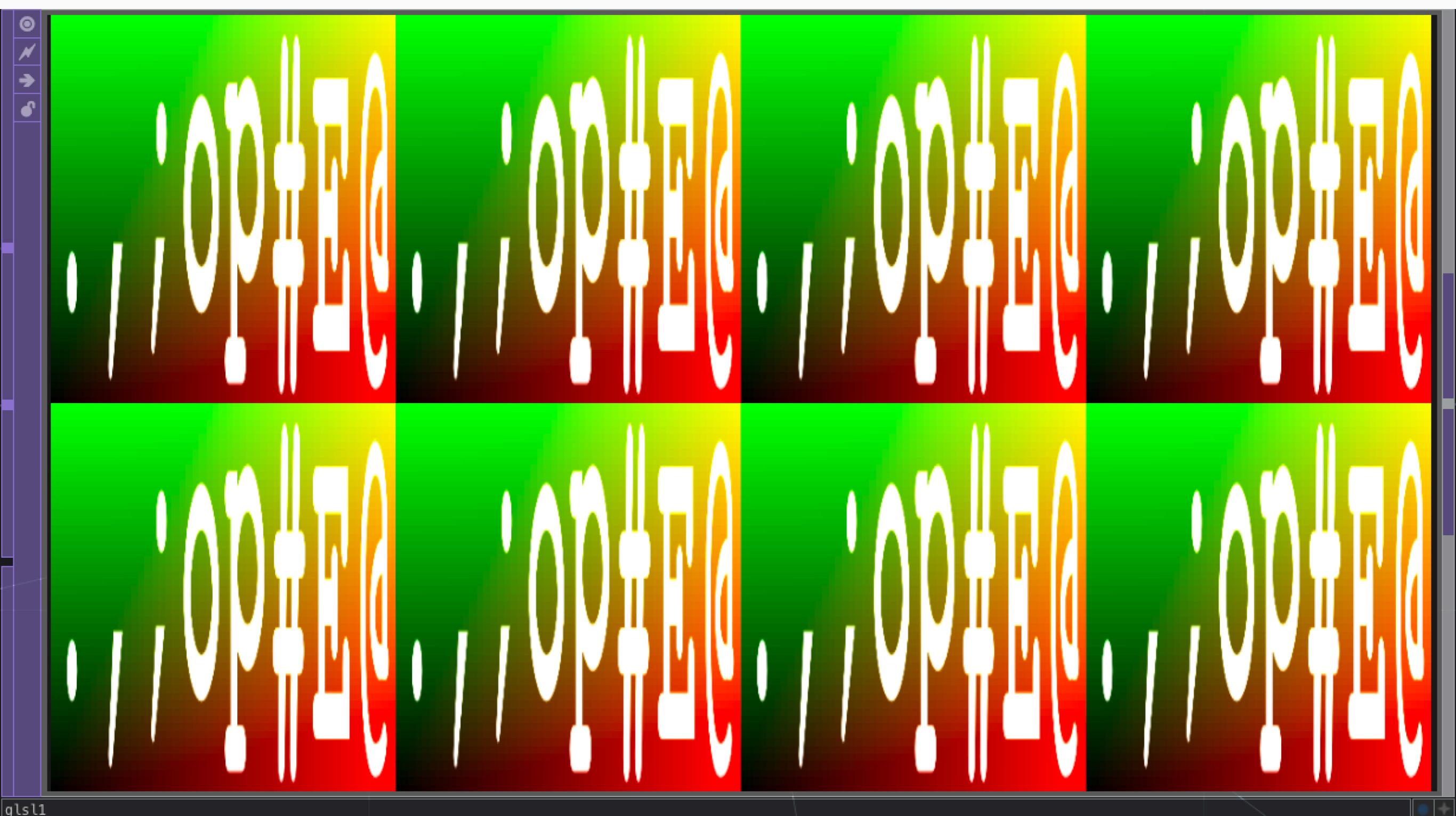
再びresolution TOPで
解像度を戻している



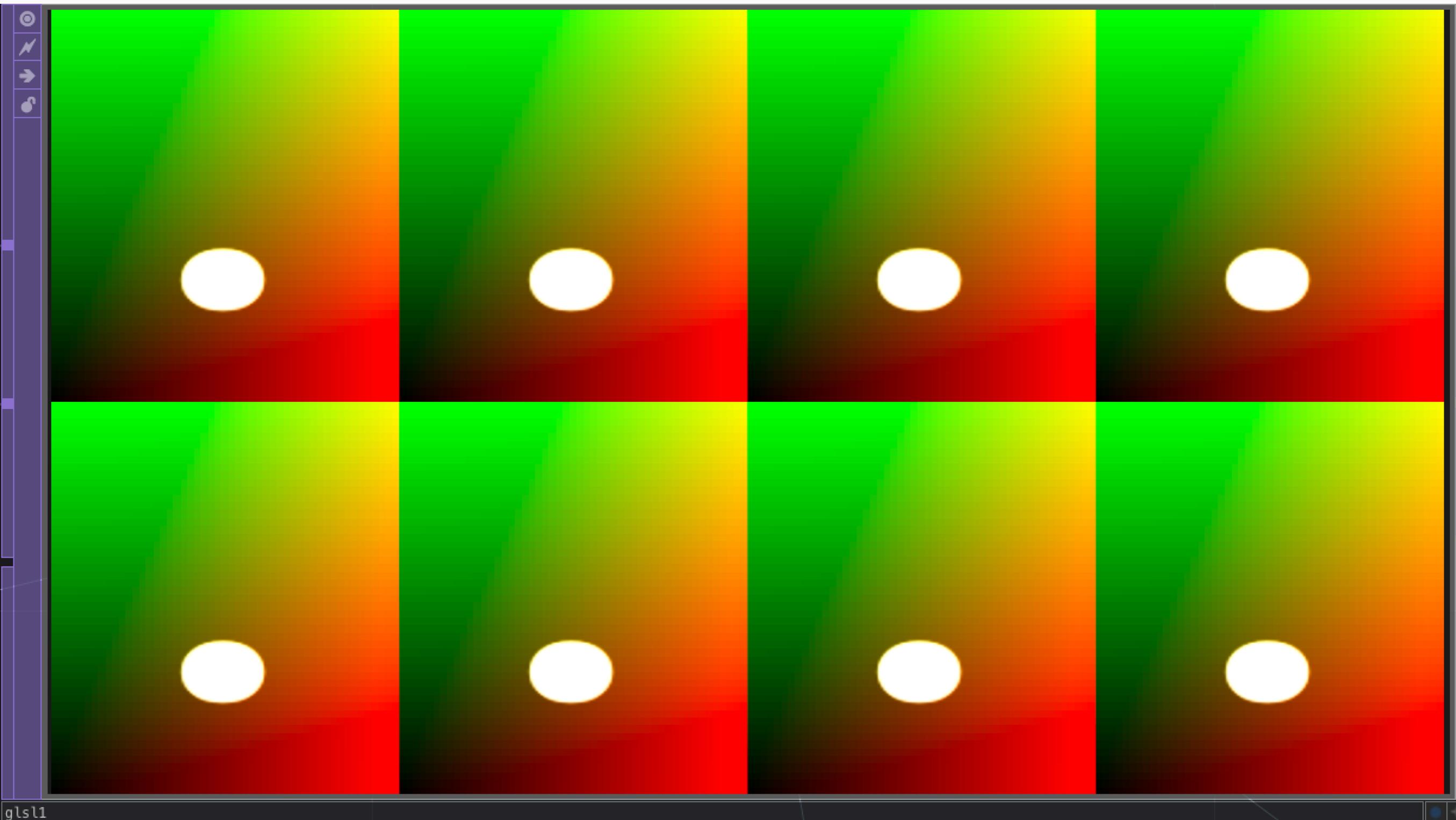
uv座標をブロックに分割



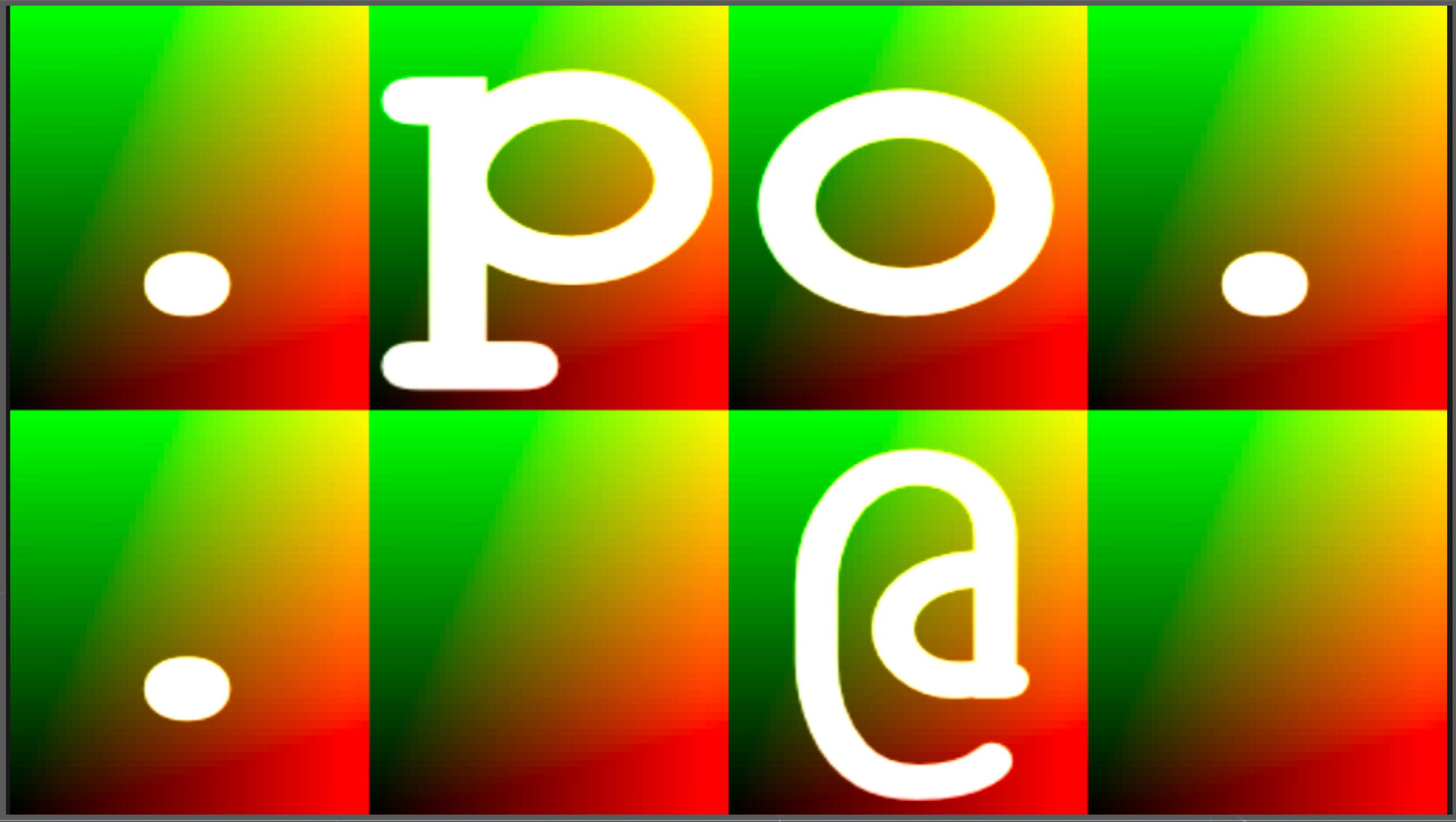
ブロック毎に文字を表示したい…



uv.x = uv.x /8.0;



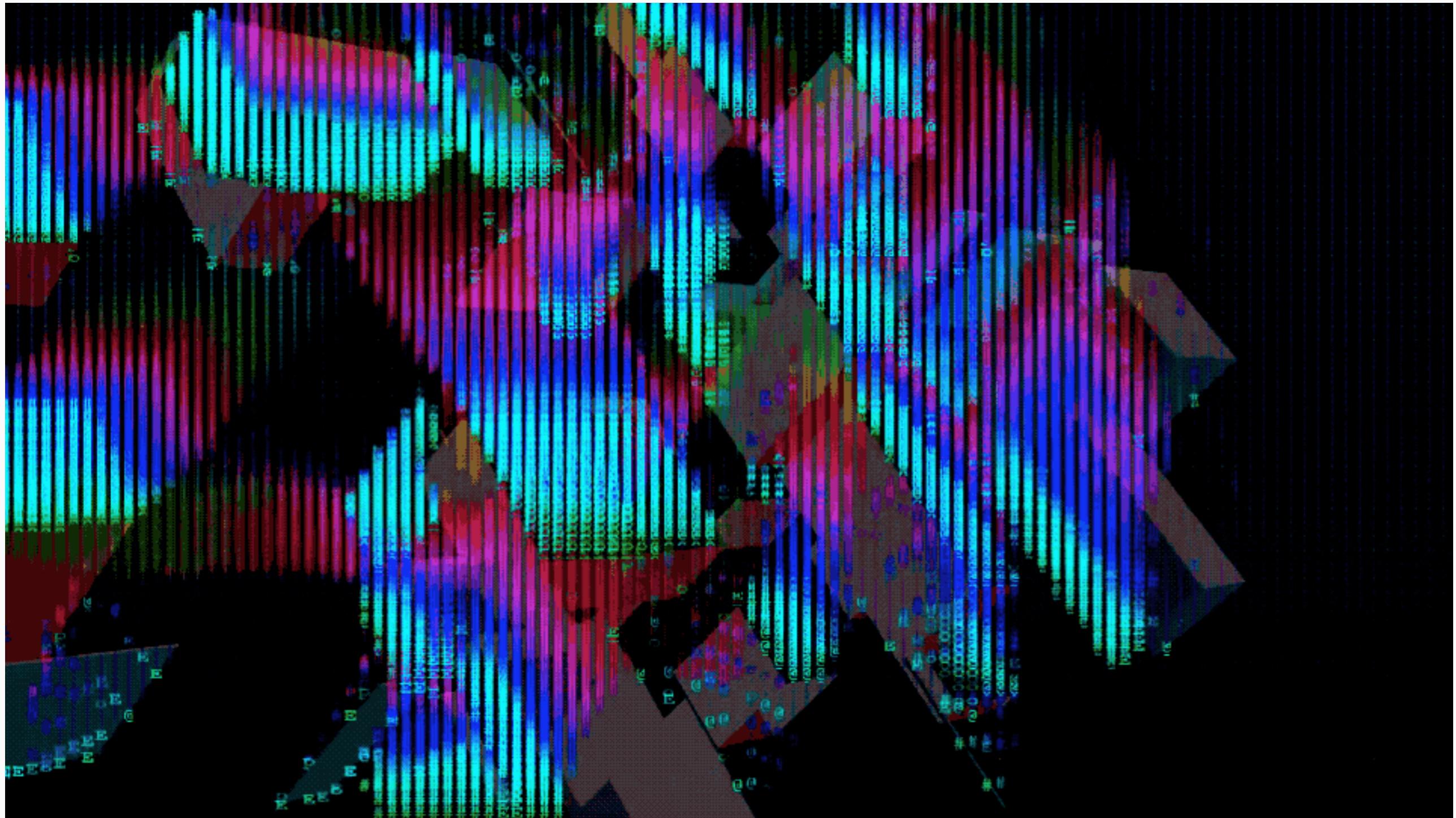
uv.x += gray * (1.0/8.0);



最終的にこうなる

```
// x座標を文字に合わせる
float gray = floor(length(color.rgb) * NUM_CHARS);
uv.x = (uv.x + gray) / NUM_CHARS;
```

完成~~



最後に

- ・ GLSLのパワフルさ、楽しさが伝わればいいな～
- ・ 質問や感想は、このあとの空き時間やTwitter等でお気軽にどうぞ！

GLSL on TouchDesigner参考リンク

- Write a GLSL TOP - Derivative

https://docs.derivative.ca/Write_a_GLSL_TOP

- TouchDesignerでGLSLを扱うには - Qiita

<https://qiita.com/ToyoshiMorioka/items/4543531738f99ac9eaef>

GLSL自体の参考リンク

- The Book Of Shaders

<https://thebookofshaders.com/?lan=jp>

- wgl.org | GLSL

<https://wgl.org/d/glsl/>

- VEDA

<https://veda.gl/>



@amagitakayosi