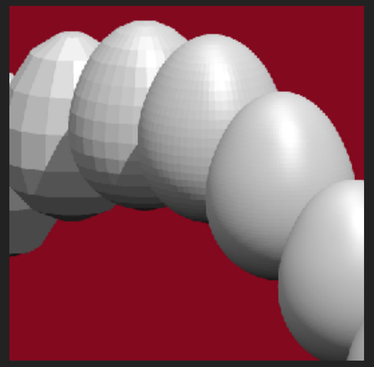
LET'S PLAY WITH FRAGMENT SHADER

フラグメントシェーダで遊ぼう

@ta_ka_tsu

自己紹介



@ta_ka_tsu

- 3D CAD系 プログラマー
- 3D CAM系 プログラマー
- 会社設立
- フリーランス　←今ココ

前回

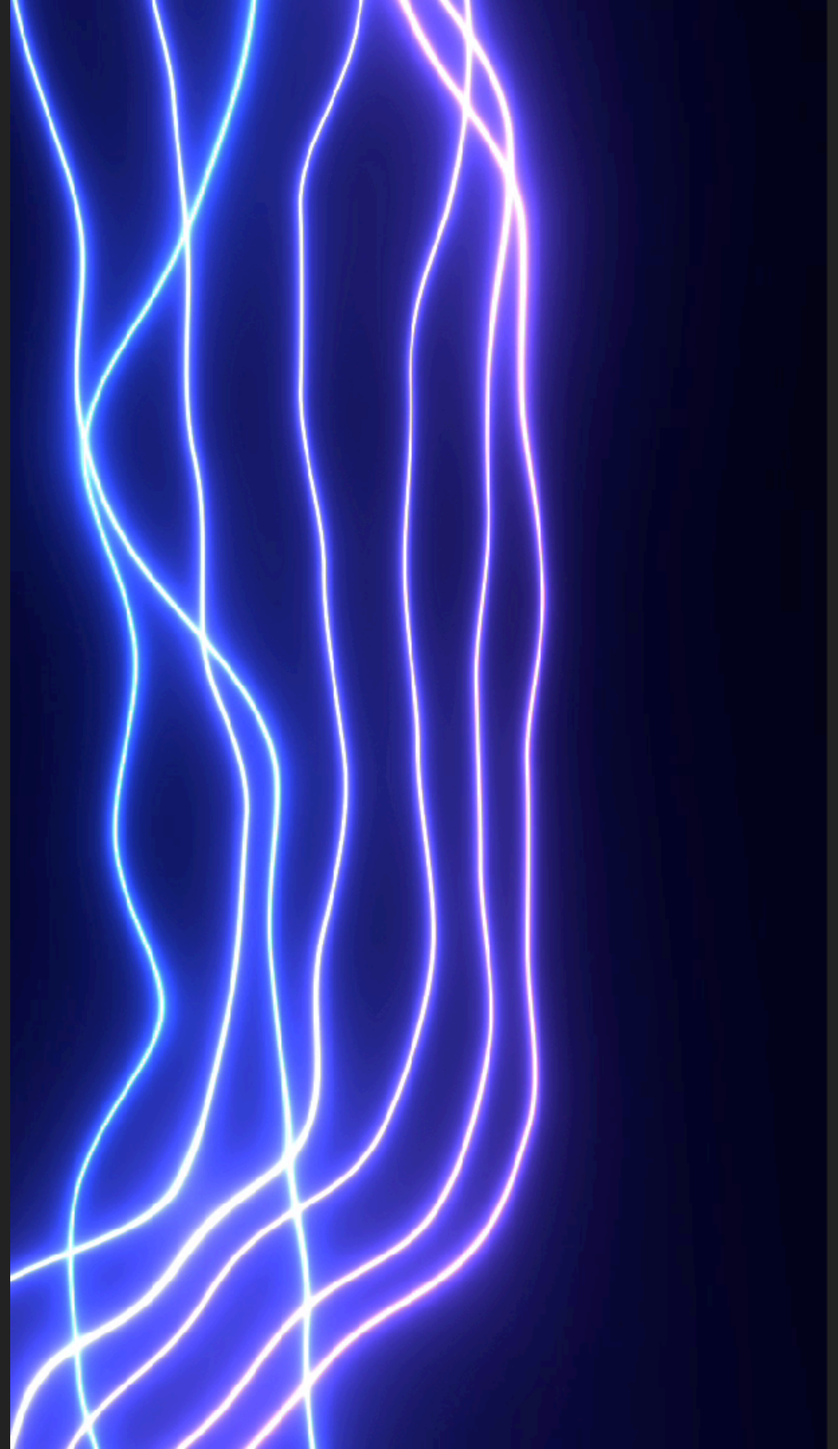
Getting Started with Metal Shading Language

by kazuhiro4949

Published **March 16, 2018** in Technology

Nagoya iOS meetup vol. 2

https://speakerdeck.com/kazuhiro4949/getting-started-with-metal-shading-language



やってみた

http://glslsandbox.com/e#45831.0

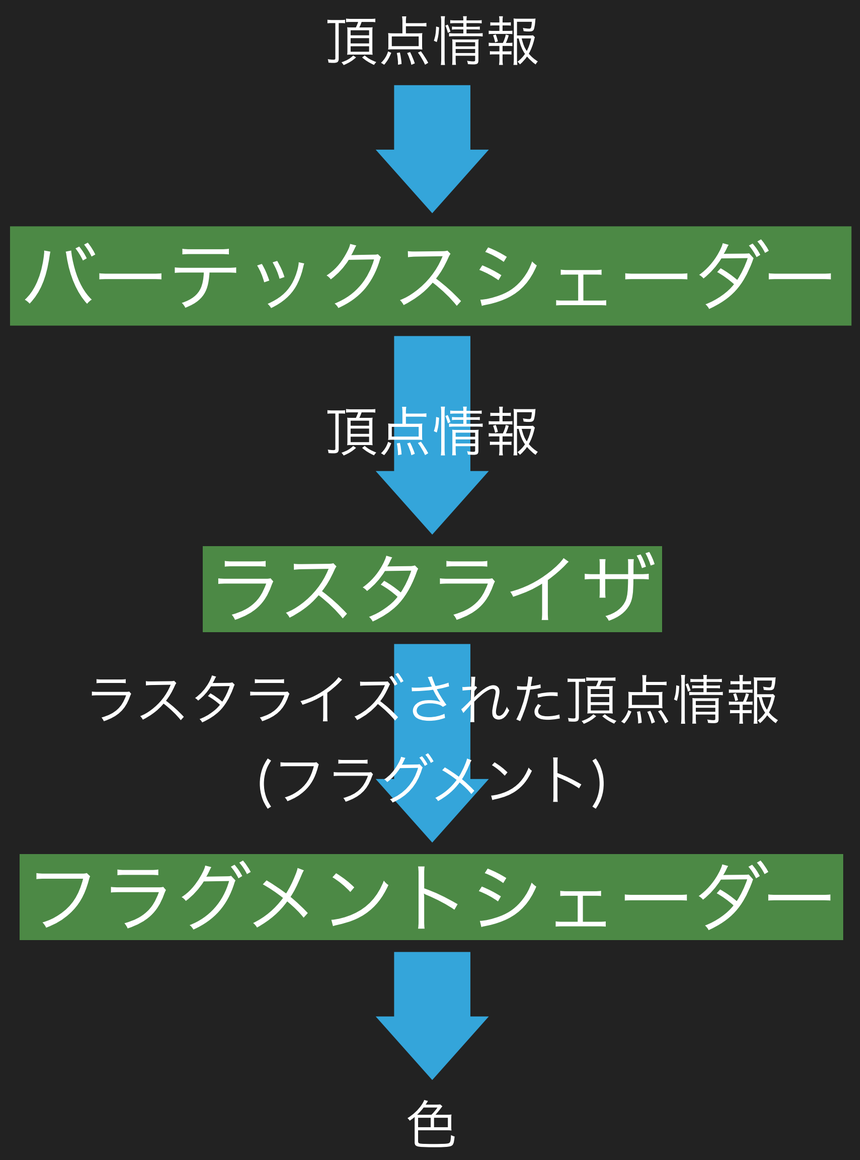
からMSLに移植

参考：GLSLを作成＆実行＆投稿できるサイト

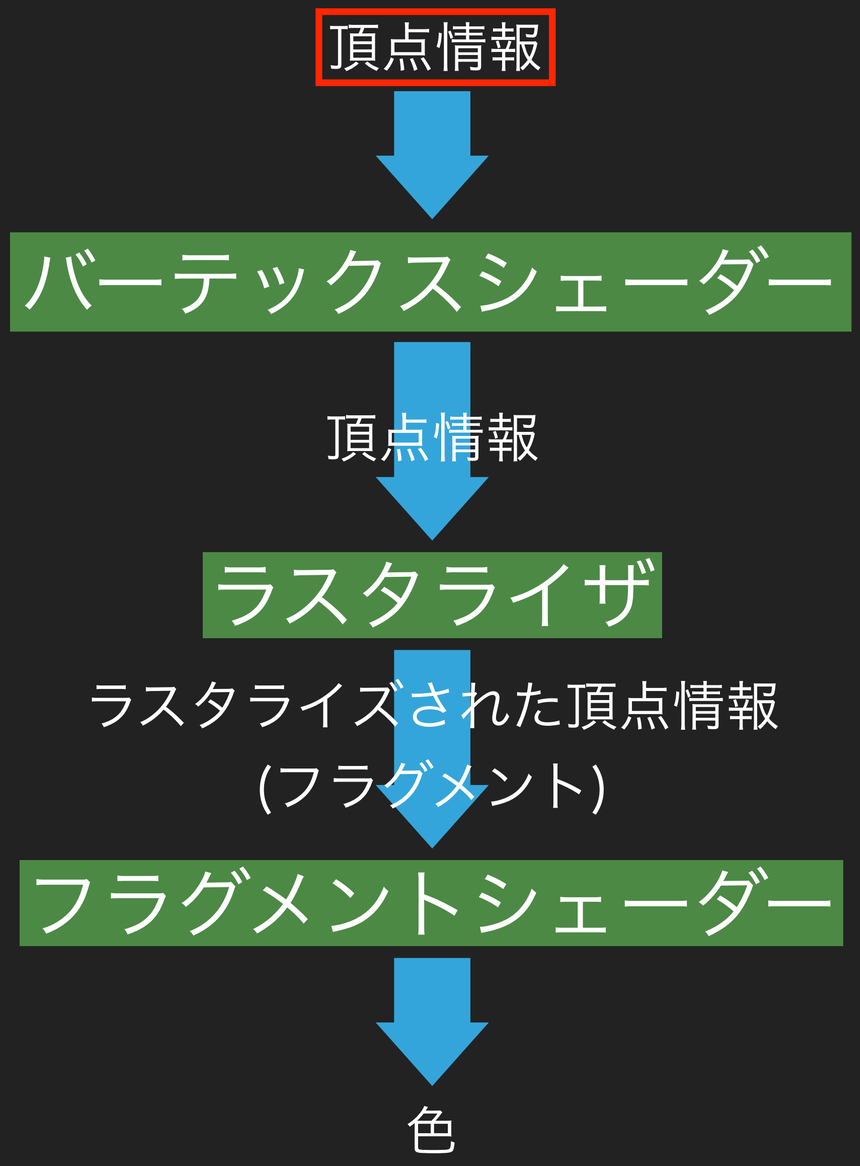
GLSL Sandbox	http://glslsandbox.com/
Shadertoy	https://www.shadertoy.com/
glslfan	https://glslfan.com/

Metal描画の仕組み
(超簡易版)

パイプラインの流れ


頂点情報
バーテックスシェーダー
頂点情報
ラスタライザ
ラスタライズされた頂点情報
(フラグメント)
フラグメントシェーダー
色

パイプラインの流れ



入力：

頂点情報 = 頂点の座標＋α

パイプラインの流れ

頂点情報

↓

バーテックスシェーダー

↓

頂点情報

↓

ラスタライザ

↓

ラスタライズされた頂点情報
(フラグメント)

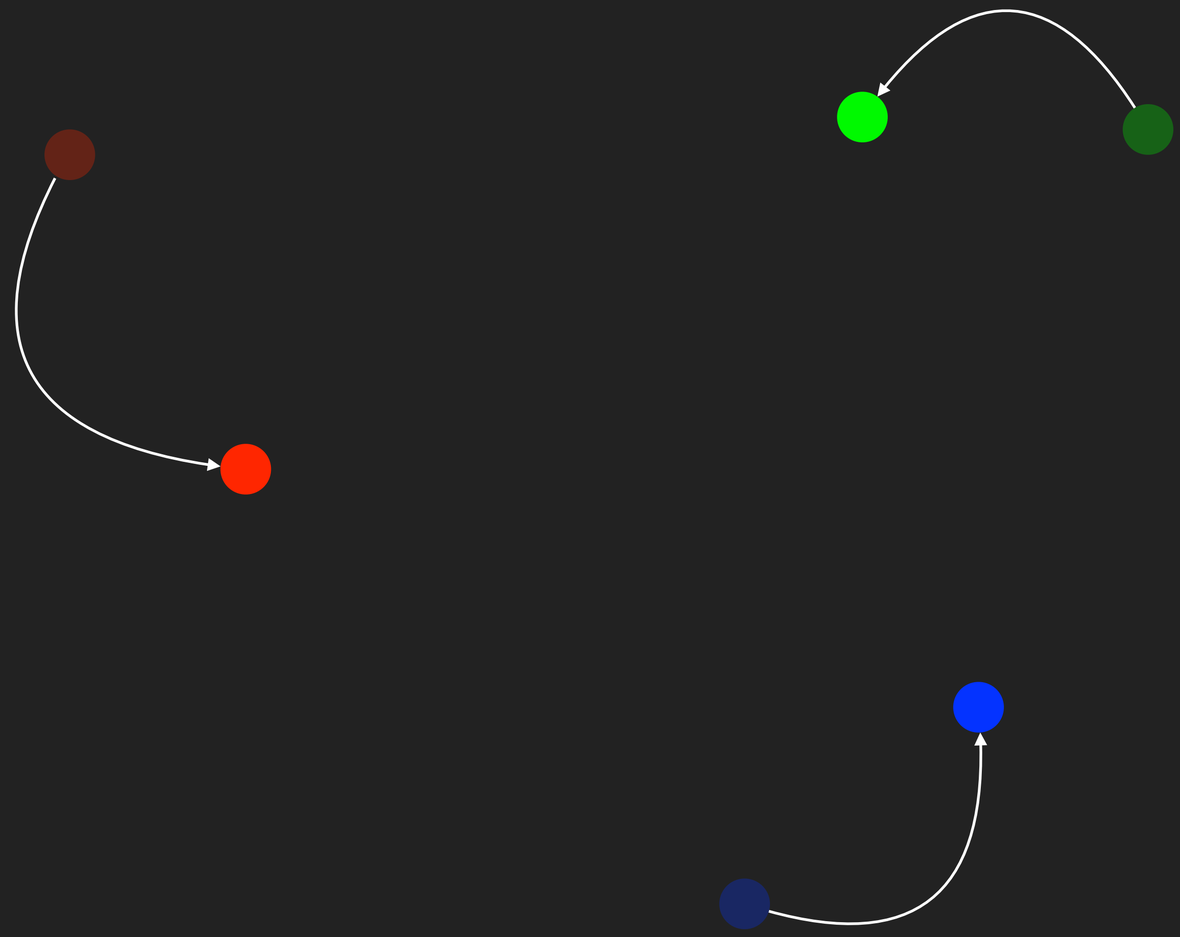
↓

フラグメントシェーダー

↓

色

バーテックスシェーダー：
頂点情報を加工する



パイプラインの流れ

頂点情報

↓

バーテックスシェーダー

↓

頂点情報

↓

ラスタライザ

↓

ラスタライズされた頂点情報
(フラグメント)

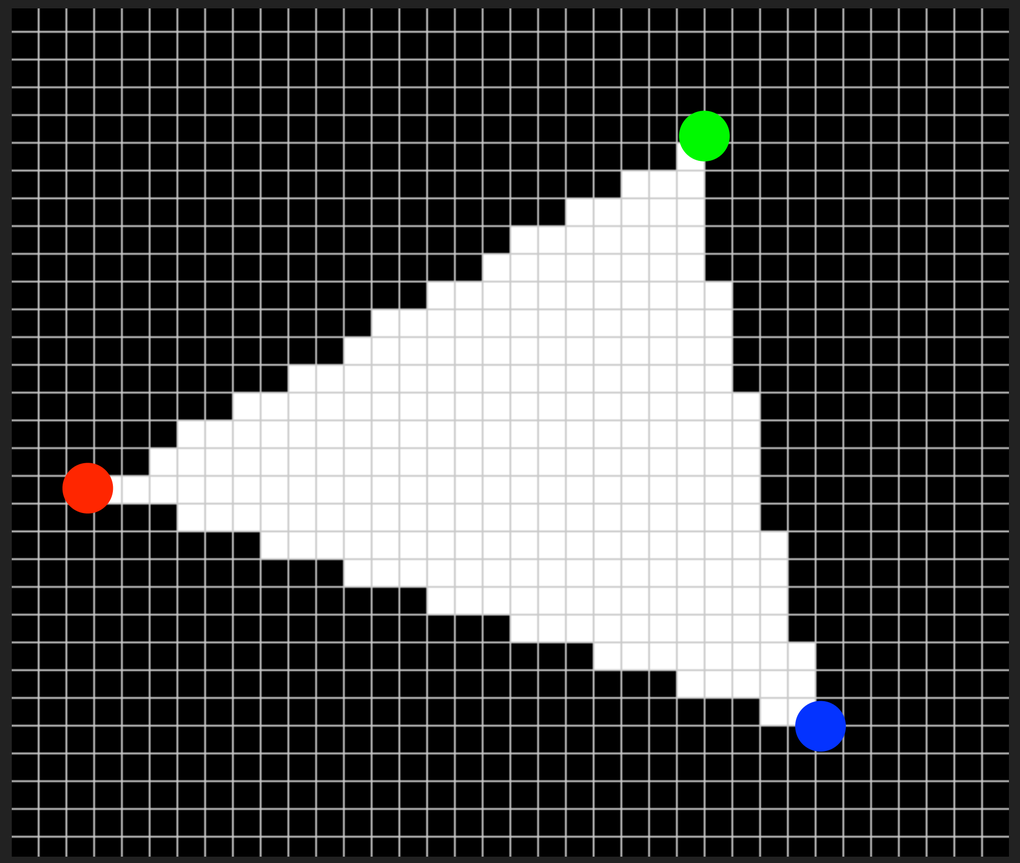
↓

フラグメントシェーダー

↓

色

ラスタライザ：
格子状の情報に変換する



パイプラインの流れ

頂点情報

↓

バーテックスシェーダー

↓

頂点情報

↓

ラスタライザ

↓

ラスタライズされた頂点情報
(フラグメント)

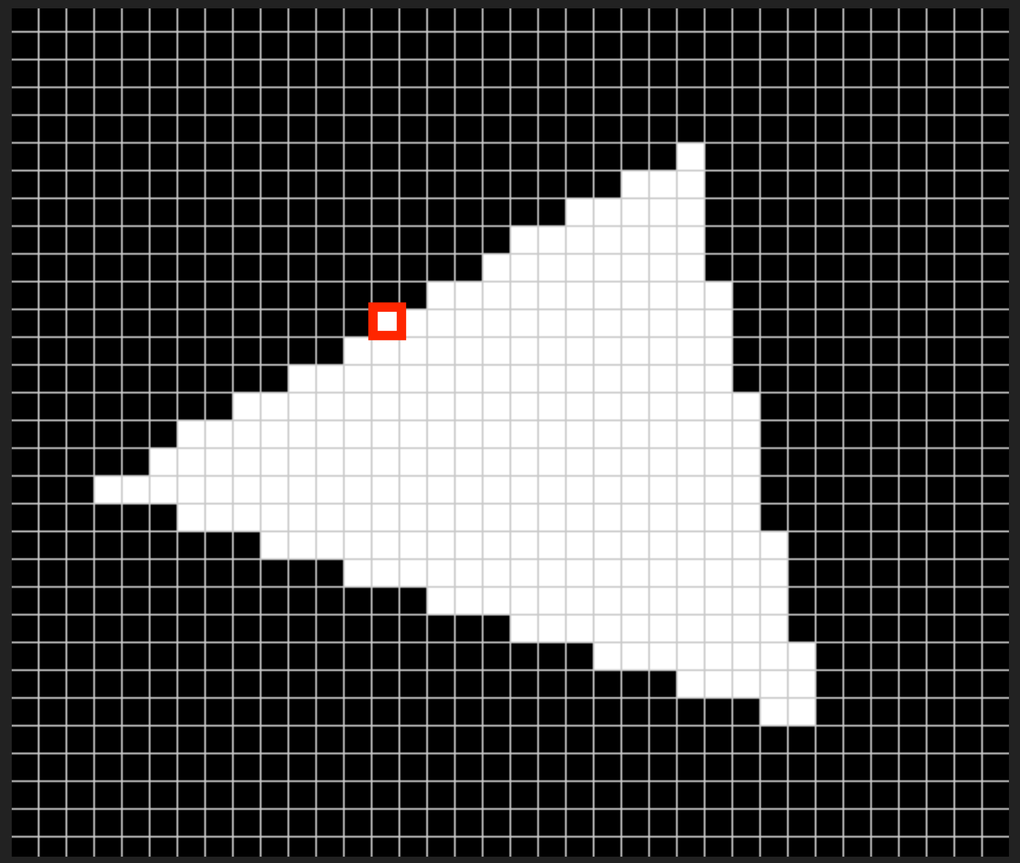
↓

フラグメントシェーダー

↓

色

フラグメント：
格子状に格納された情報



パイプラインの流れ

頂点情報

↓

バーテックスシェーダー

↓

頂点情報

↓

ラスタライザ

↓

ラスタライズされた頂点情報
(フラグメント)

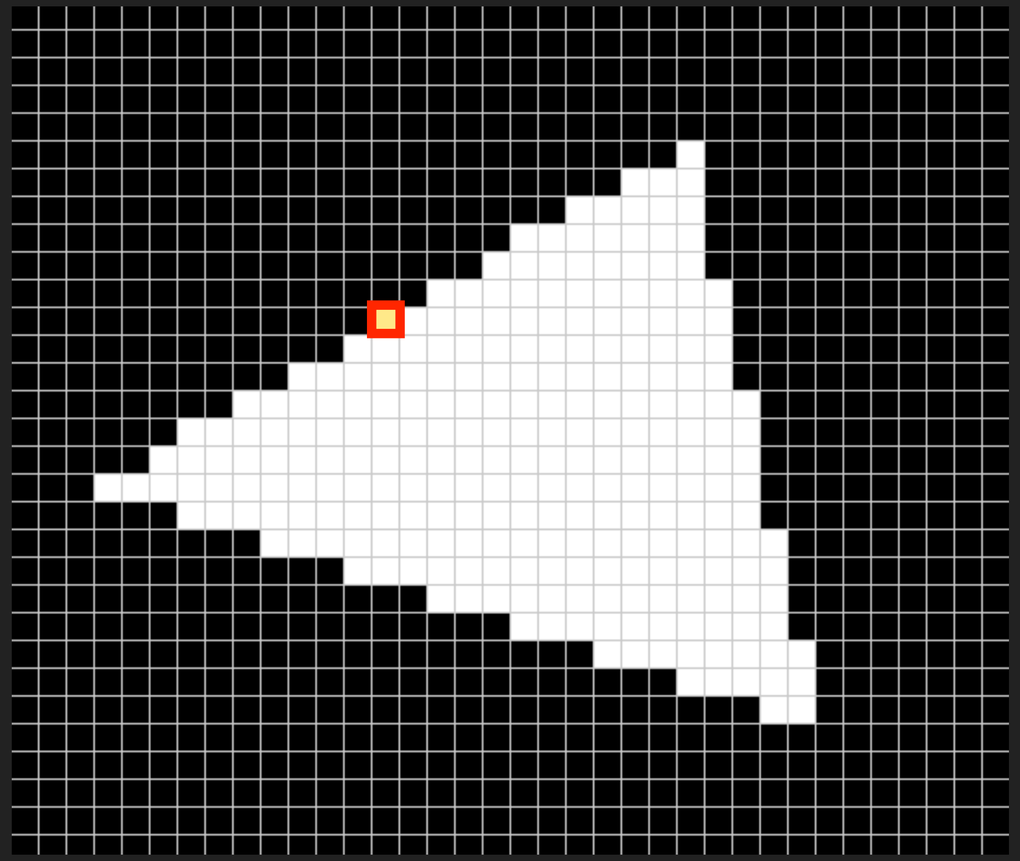
↓

フラグメントシェーダー

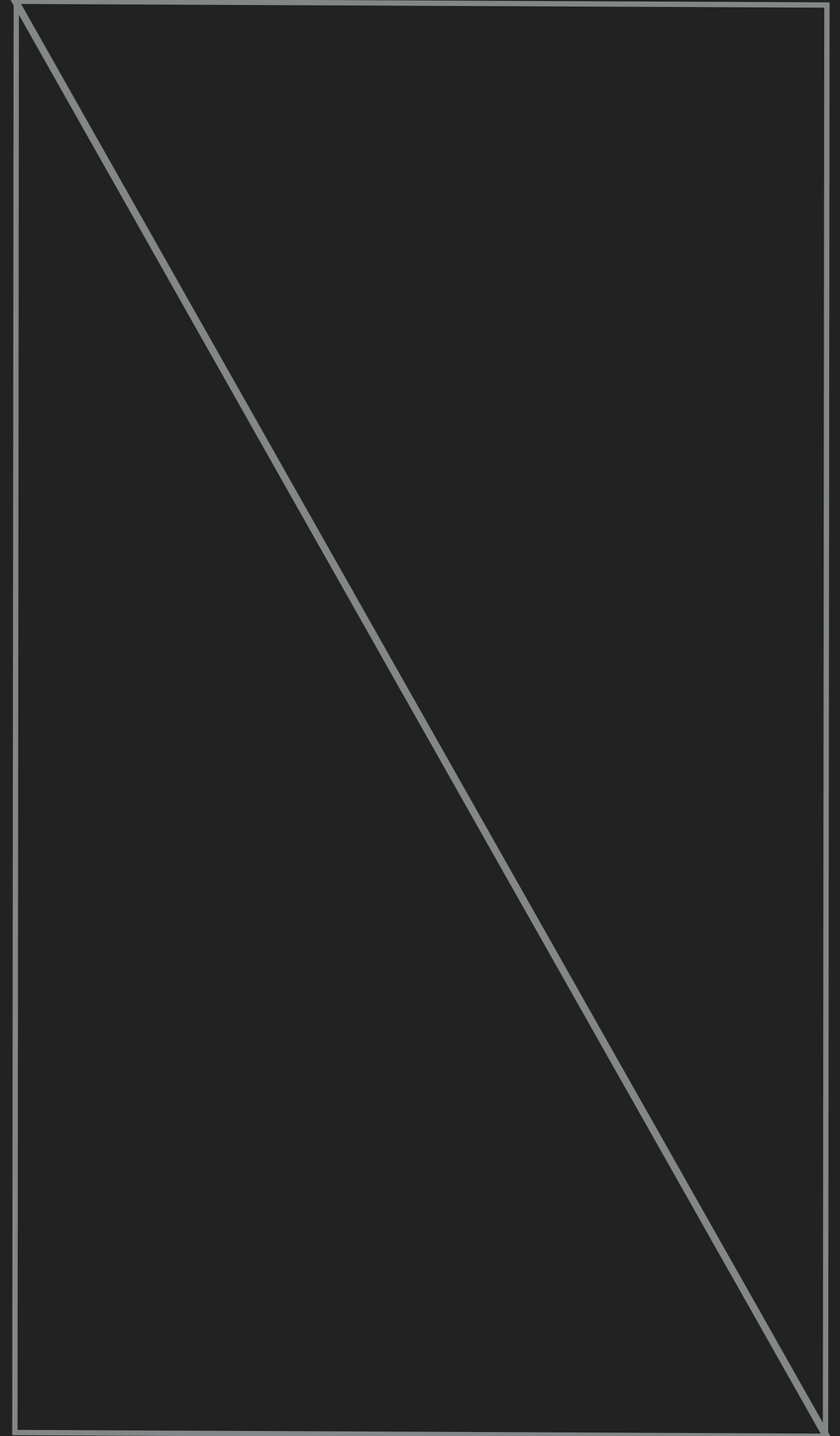
↓

色

フラグメントシェーダー：
フラグメントから色を決定する

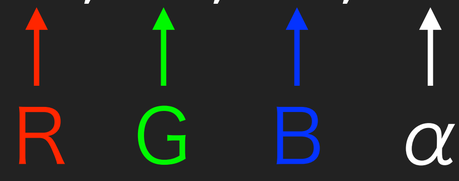


フラグメントシェーダーで絵を描くには？



画面全体を覆うように
三角形を2つ描画することで
全てのピクセルをフラグメントシェーダーで
制御する。

```
// 構造体
struct VertexOut {
    float4 pos [[position]];
};

// バーテックスシェーダー
vertex VertexOut vertexShader(
  const device packed_float2* vertex_array [[buffer(0)]],
  unsigned int vid [[vertex_id]])
{
    VertexOut v;
    v.pos = float4(vertex_array[vid], 1.0, 1.0);
    return v;
}

// フラグメントシェーダー
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]])
{
  return half4(1.0, 0.0, 0.0, 1.0);
}
```



※以後構造体とバーテックスシェーダーは省略

円を描く

```
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]])
{
  return step(300.0, length(fragmentIn.pos.xy));
}
```

※step関数：
第一引数より第二引数が小さい場合0.0を返す
それ以外は1.0を返す

※スカラー値はベクトル値に暗黙的にキャストされ
その成分は全て元のスカラー値となる。
ex.) half4 color = 1.0; // (1.0, 1.0, 1.0, 1.0)

円をぼかして描く

```
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]])
{
  return smoothstep(300.0, 600.0,
    length(fragmentIn.pos.xy));
}
```

※smoothstep関数：

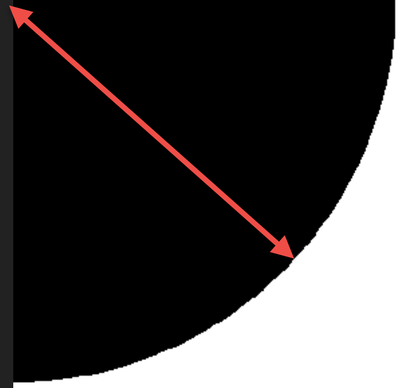
第一引数より第三引数が小さい場合0.0を返す

第二引数より大きい場合1.0を返す

第一引数と第二引数の間は滑らかに補間

(エルミート補間)

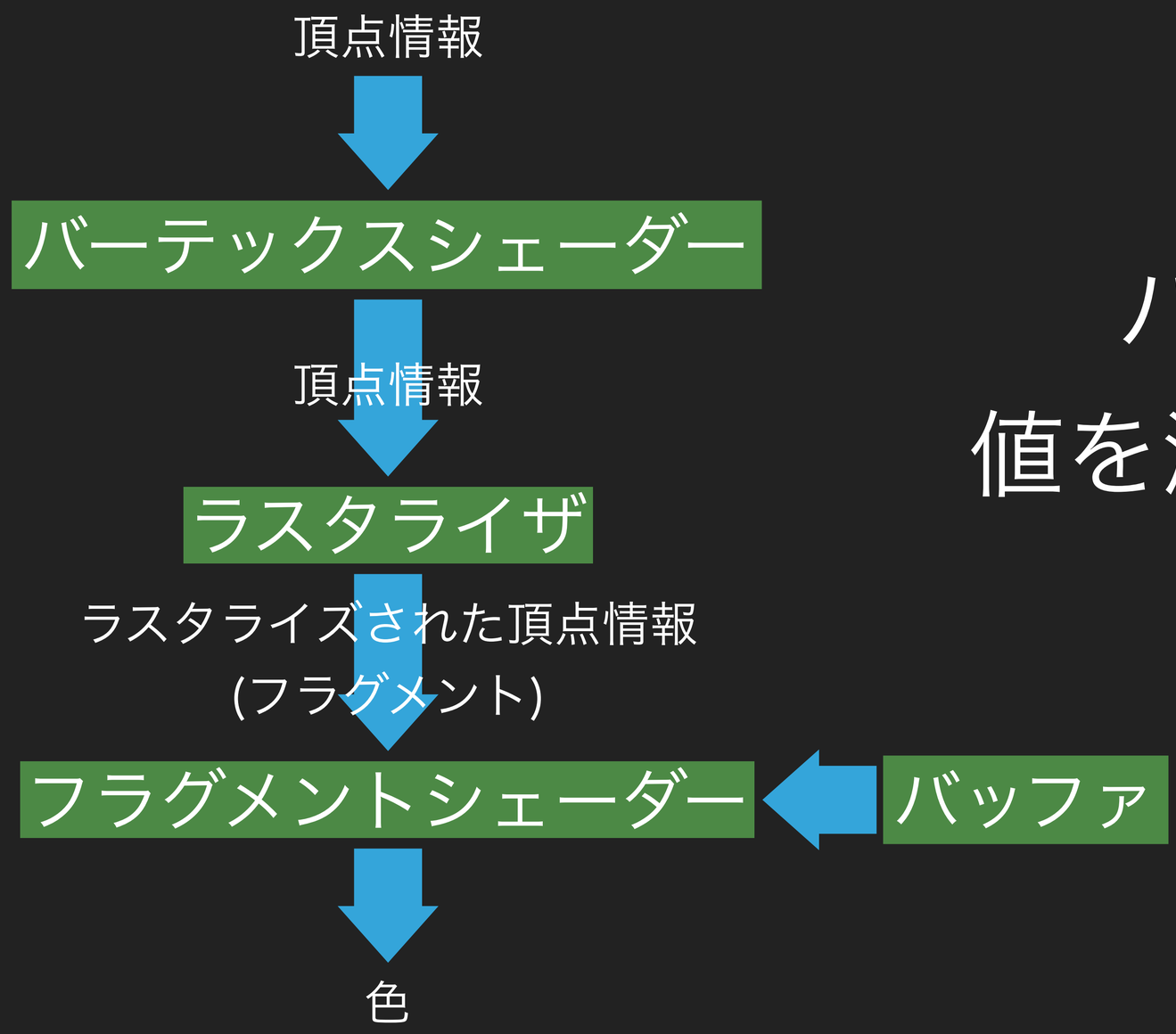
座標の正規化



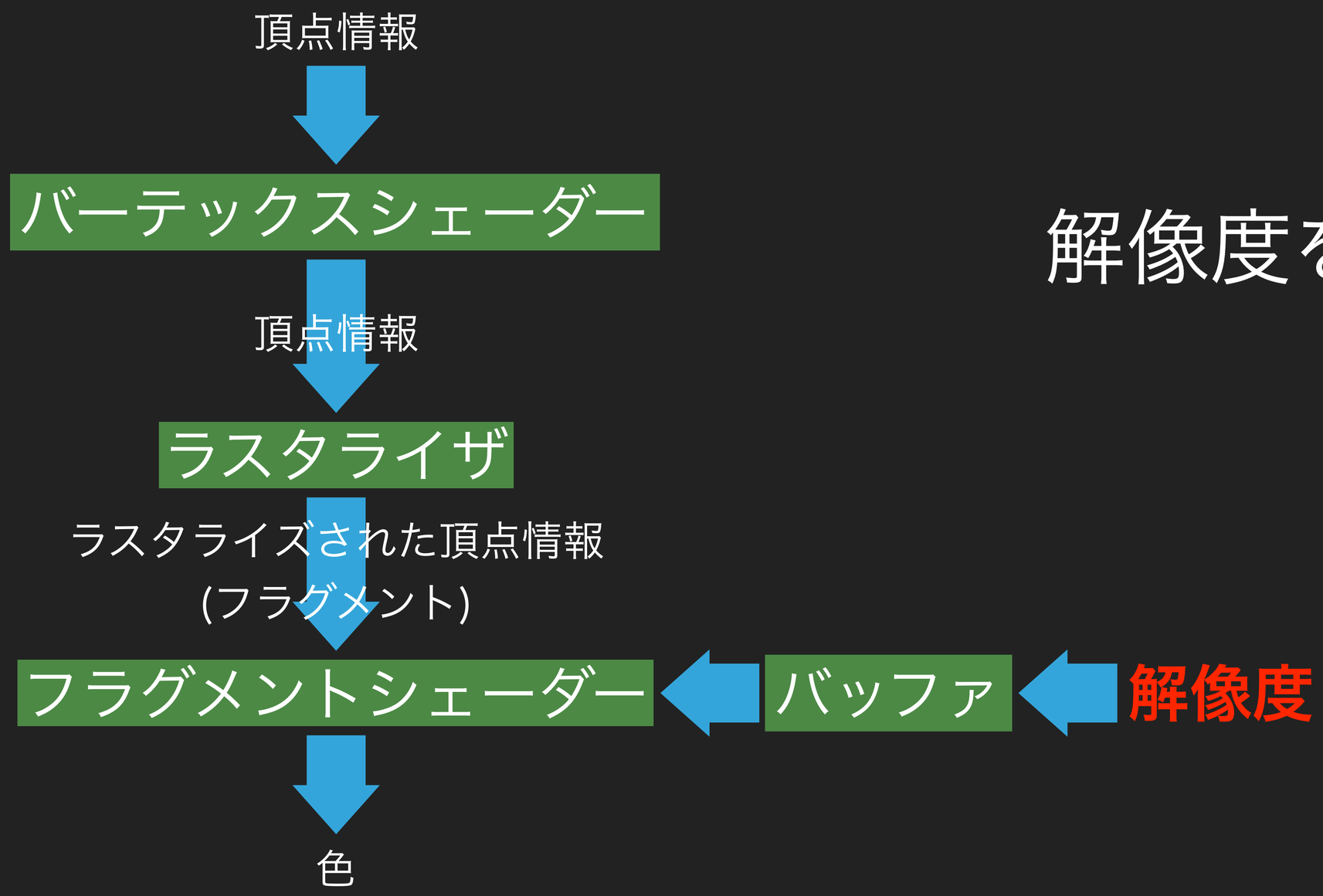
```
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]])
{
  return step(300.0, length(fragmentIn.pos.xy));
}
```

画面サイズに対してのバランスがわかりにくい

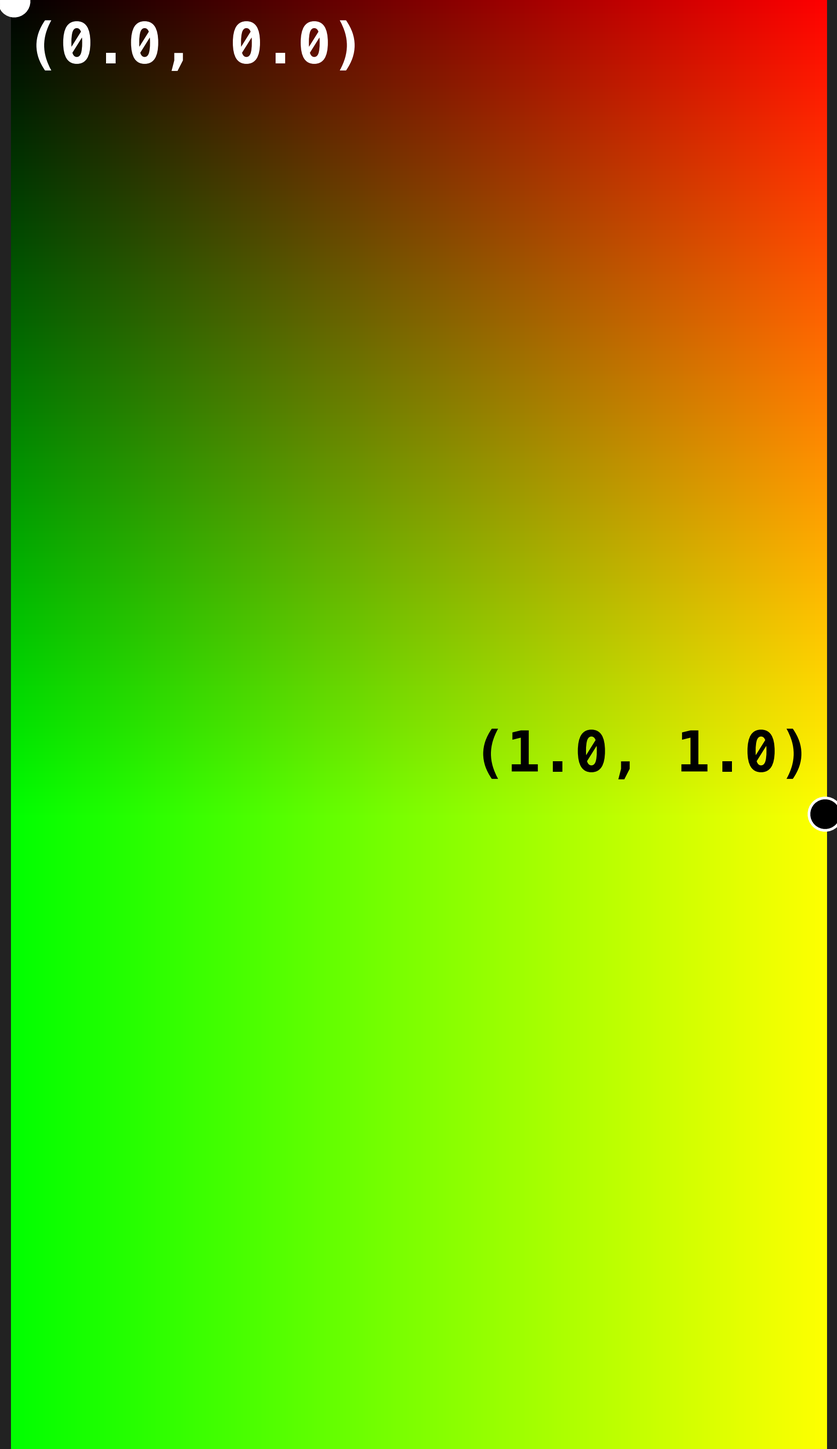
しかしピクセル座標しか入力情報がない



シェーダーには
バッファを通して
値を渡すことができる。

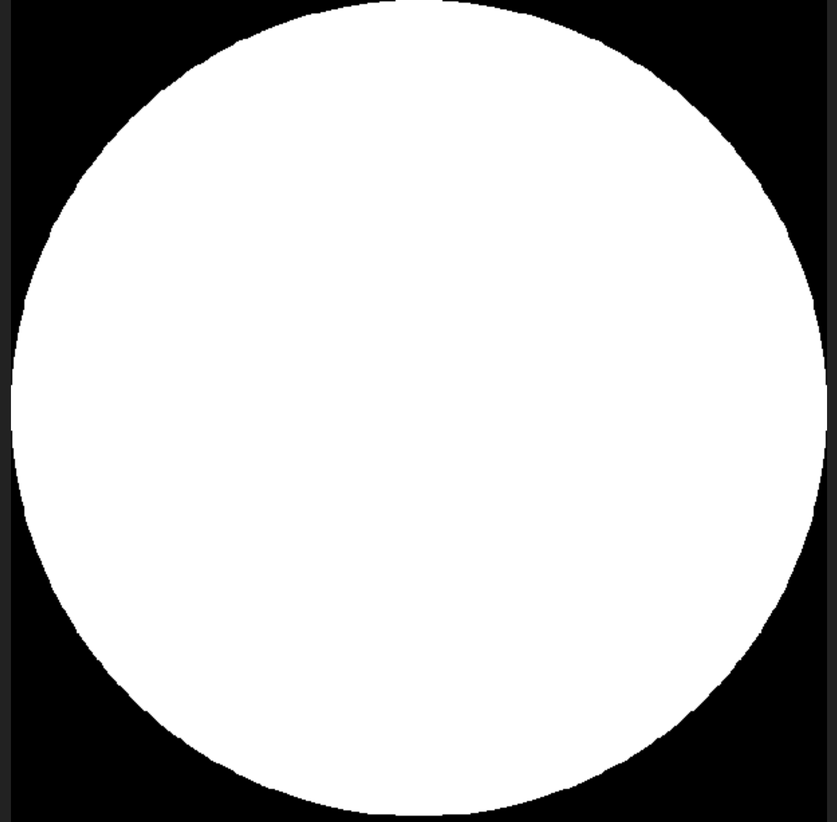

頂点情報
バーテックスシェーダー
頂点情報
ラスタライザ
ラスタライズされた頂点情報
(フラグメント)
フラグメントシェーダー
バッファ
解像度
色

解像度を渡す



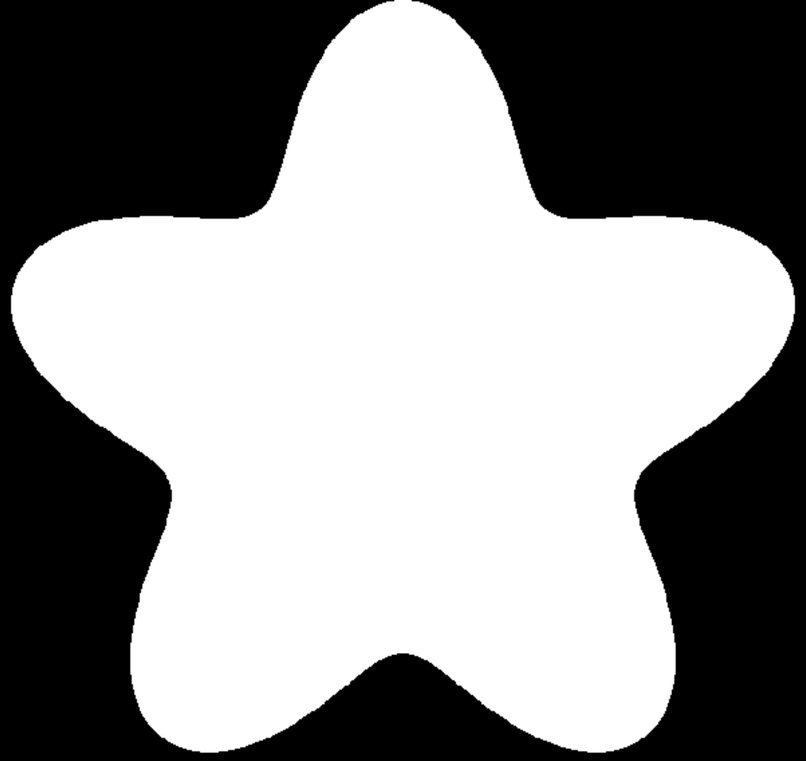
フラグメントシェーダー側では引数として受け取る

```
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]] )
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  return half4(p.x, p.y, 0.0, 1.0);
}
```



画面いっぱいに円を描く

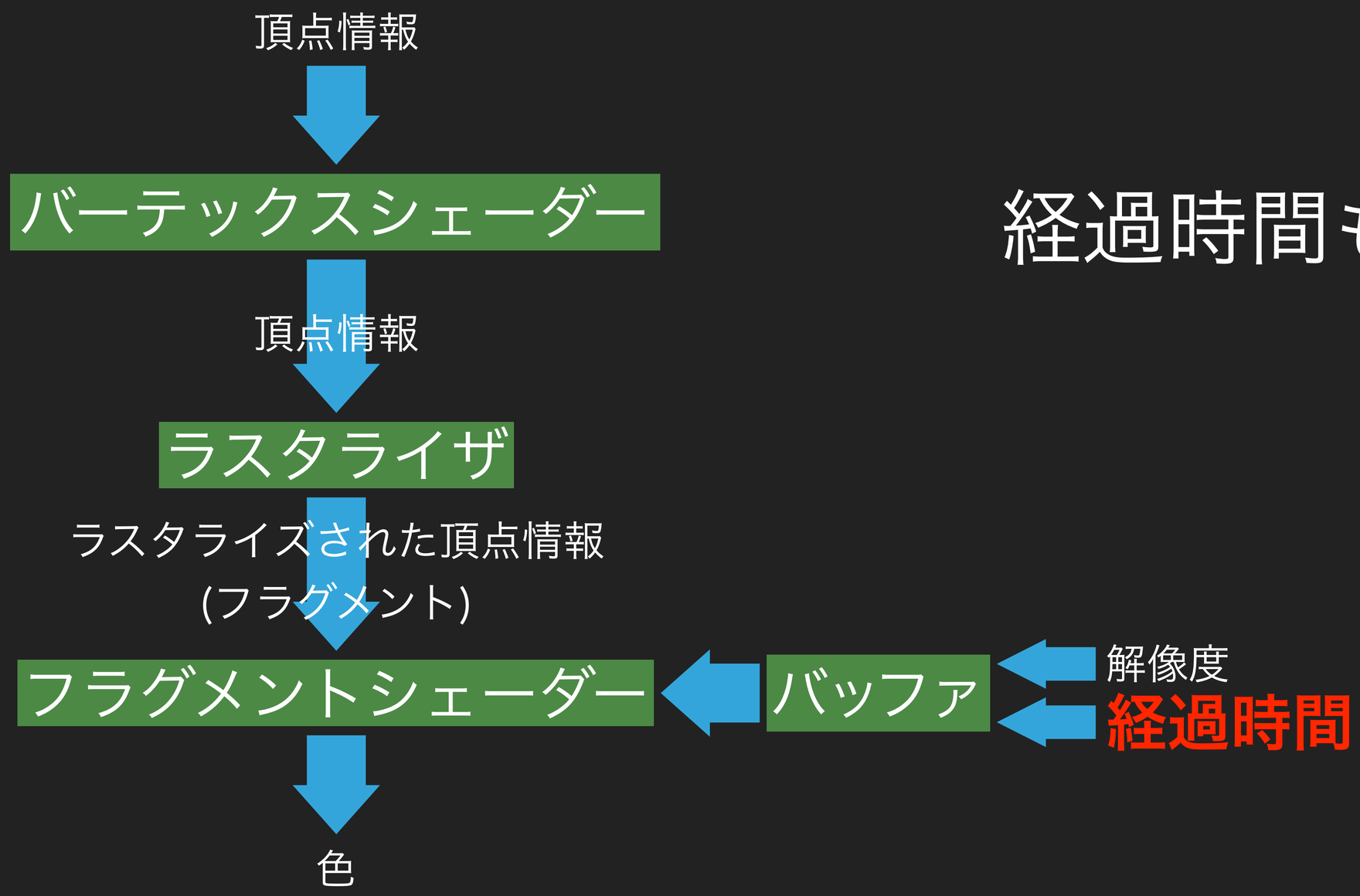
```
float circle(float2 p)
{
  // p - 0.5 は  p - float2(0.5, 0.5) と同義
  return step(length(p - 0.5), 0.5);
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]])
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  return circle(p);
}
```



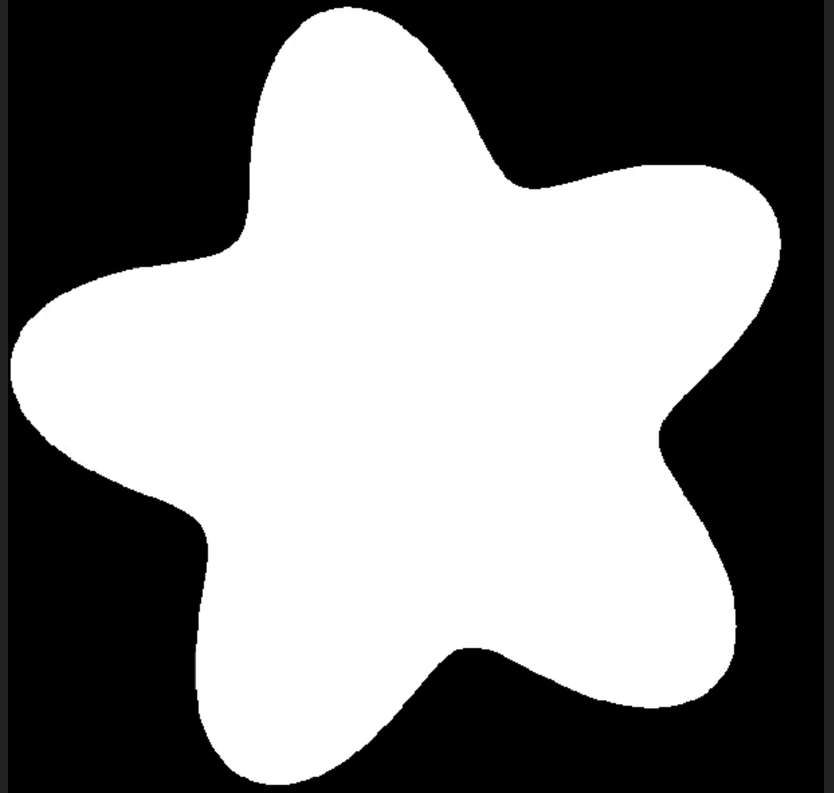
偏角に応じて閾値を変える

```
float star(float2 p)
{
  float2 c = p - 0.5;
  float a = atan2(c.y, c.x);
  return step(length(c) + 0.1*sin(5.0*a) + 0.1, 0.5);
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]])
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  return star(p);
}
```

アニメーション

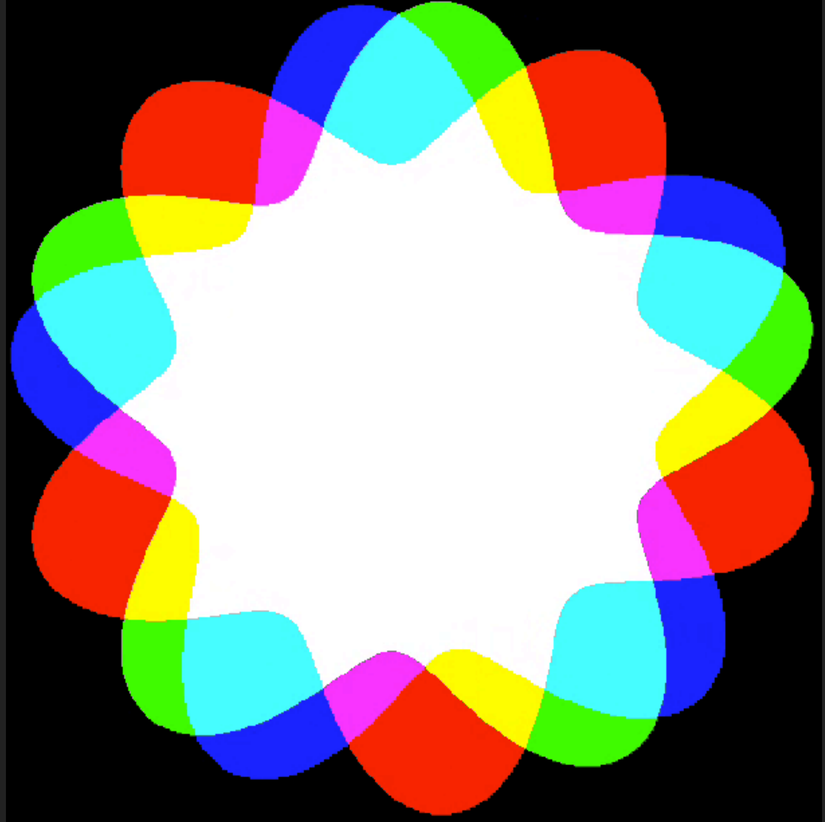

頂点情報
バーテックスシェーダー
頂点情報
ラスタライザ
ラスタライズされた頂点情報
(フラグメント)
フラグメントシェーダー
バッファ
解像度
経過時間
色

経過時間も渡す



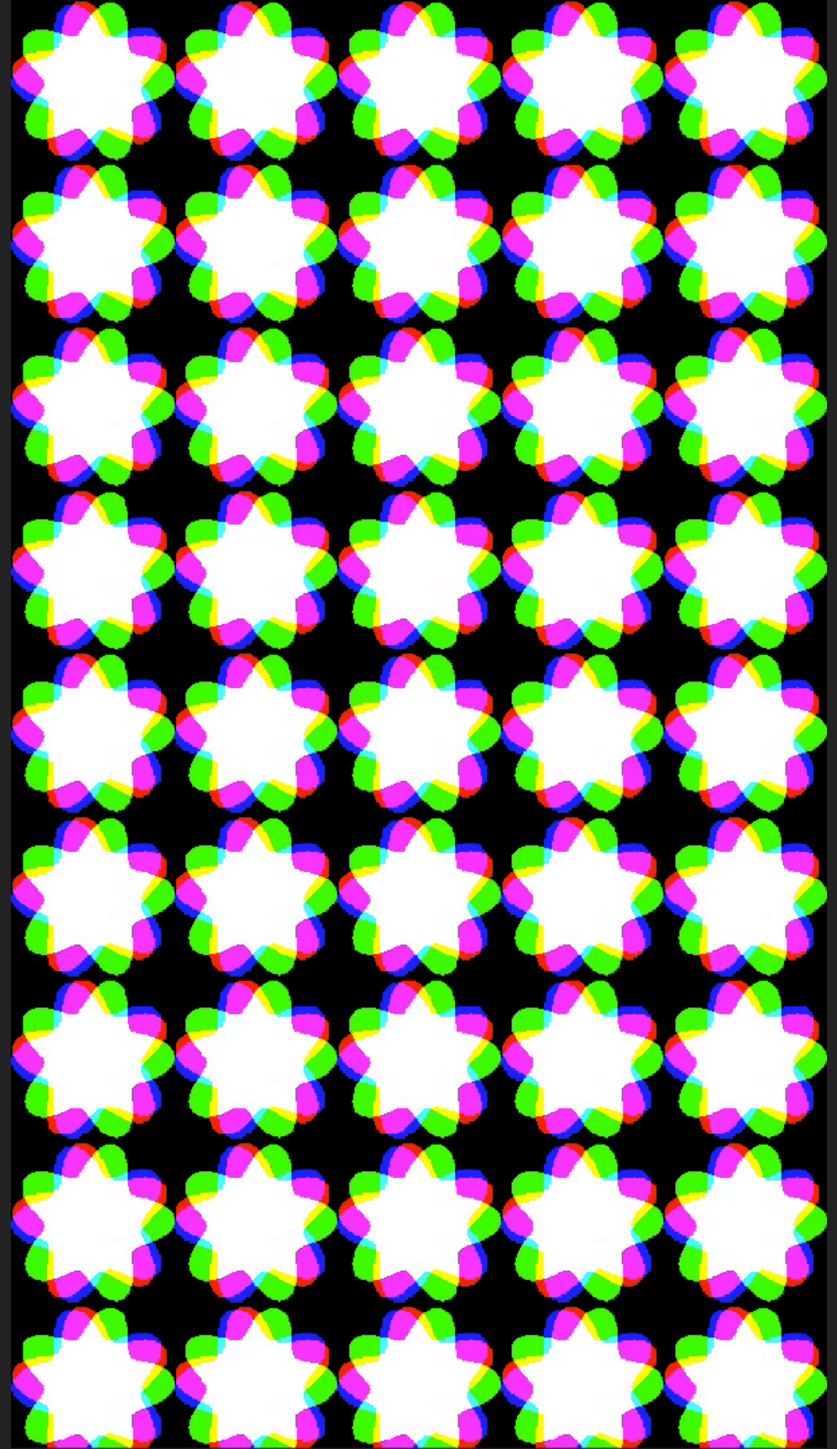
時間を位相として偏角に加える

```
float star(float2 p, float phase)
{
  float2 c = p - 0.5;
  float a = atan2(c.y, c.x) + phase;
  return step(length(c) + 0.1*sin(5.0*a) + 0.1, 0.5);
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]],
  constant float &time [[buffer(1)]])
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  return star(p, time);
}
```



RGB要素毎に位相を変える

```
float star(float2 p, float phase)
{
  float2 c = p - 0.5;
  float a = atan2(c.y, c.x) + phase;
  return step(length(c) + 0.1*sin(5.0*a) + 0.1, 0.5);
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]],
  constant float &time [[buffer(1)]])
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  return half4(star(p, time),
               star(p, sin(time)),
               star(p, 2.0*time),
               1.0);
}
```



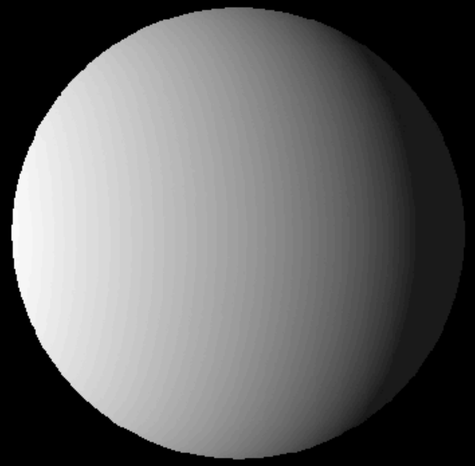
繰り返す

```
float star(float2 p, float time)
{
  float2 c = p - 0.5;
  float a = atan2(c.y, c.x) + time;
  return step(length(c) + 0.1*sin(5.0*a) + 0.1, 0.5);
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]],
  constant float &time [[buffer(1)]])
{
  float2 p = fragmentIn.pos.xy/min(res.x, res.y);

  float2 p1 = fract(p * 5.0);
  return half4(star(p1, time),
               star(p1, sin(time)),
               star(p1, 2.0*time),
               1.0);
}
```

※fract(x) = x - floor(x)

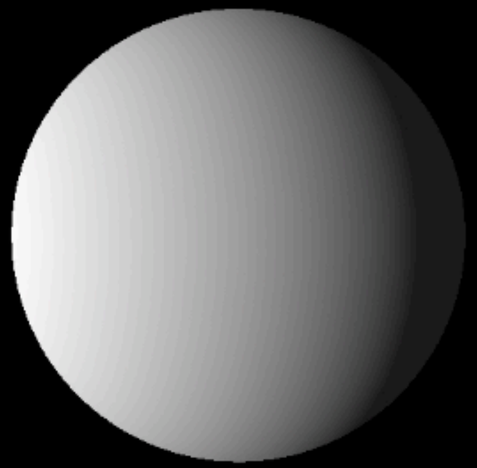
3次元的な表現

レイマーチング



レイトレーシングの一種
(説明は割愛)

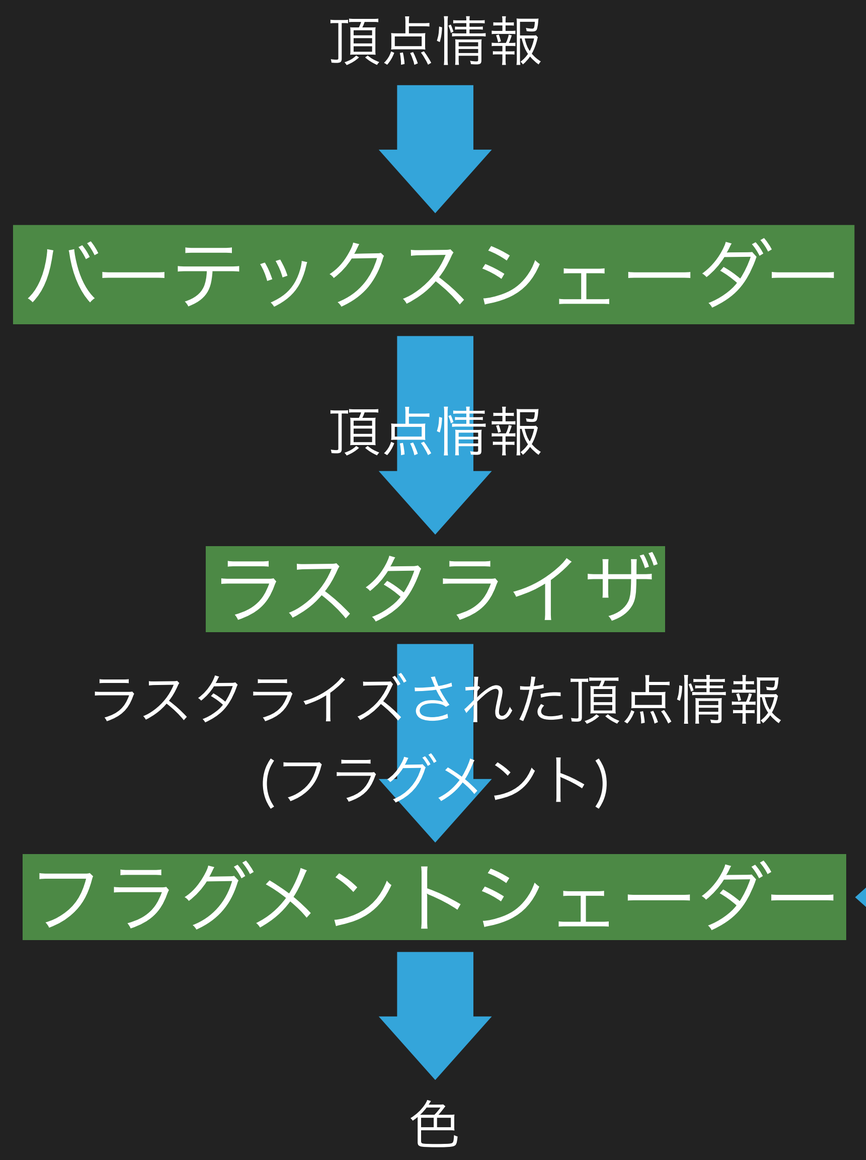
https://wgld.org/d/glsl/g011.html
をMSLに移植＆アニメーションを追加



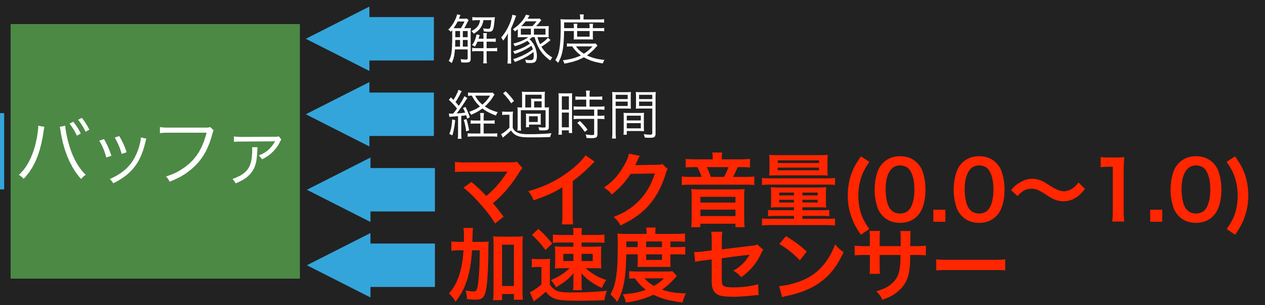
```
float sphere(float3 p){
 return length(p) - 1.0;
}

float3 getNormal(float3 p){
  float d = 0.0001;
  return normalize(
    float3(
      sphere(p + float3(  d, 0.0, 0.0)) - sphere(p + float3( -d, 0.0, 0.0)),
      sphere(p + float3(0.0,   d, 0.0)) - sphere(p + float3(0.0,  -d, 0.0)),
      sphere(p + float3(0.0, 0.0,   d)) - sphere(p + float3(0.0, 0.0,  -d))
    )
  );
}

fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res[[buffer(0)]],
  constant float &time[[buffer(1)]])
{
  float2 p1 = (fragmentIn.pos.xy * 2.0 - res) / min(res.x, res.y);
  float2 p = p1 + float2(0.3*sin(time), 0.3*cos(time));

  float angle = 90.0;
  float fov = angle * 0.5 * M_PI_F / 180.0;
  float3 cPos1 = float3(0.0, 0.0, 2.0);
  float3 lightDir = float3(-0.577, 0.577, 0.577);

  float phase1 = 0.7*sin(time);
  float phase2 = 0.5*sin(time*1.5);
  float3 ray1 = normalize(float3(sin(fov) * p.x, sin(fov) * p.y, -cos(fov)));
  float3x3 rotation = float3x3(cos(phase1), -sin(phase1), 0.0,
                               sin(phase1), cos(phase1), 0.0,
                               0.0, 0.0, 1.0) *
                      float3x3(1.0, 0.0, 0.0,
                               0.0, cos(phase2), -sin(phase2),
                               0.0, sin(phase2), cos(phase2));
  float3 ray = rotation * ray1;
  float3 cPos = rotation * cPos1;

  float distance = 0.0;
  float rLen = 0.0;
  float3  rPos = cPos;
  for(int i = 0; i < 16; i++){
    distance = sphere(rPos);
    rLen += distance;
    rPos = cPos + ray * rLen;
  }

  if(abs(distance) < 0.001){
    float3 normal = getNormal(rPos);
    float diff = clamp(dot(lightDir, normal), 0.1, 1.0);
      return half4(diff, diff, diff, 1.0);
  }
  else{
    return half4(0.0, 0.0, 0.0, 1.0);
  }
}
```

おまけ＆DEMO

もっと情報を渡す



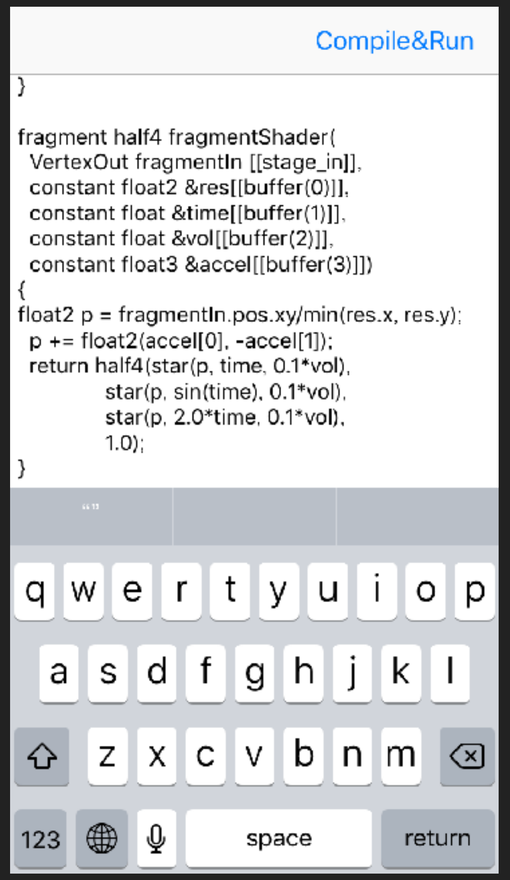

```
fragment half4 fragmentShader(
  VertexOut fragmentIn [[stage_in]],
  constant float2 &res [[buffer(0)]],
  constant float &time [[buffer(1)]],
  constant float &vol [[buffer(2)]],
  constant float3 &accel [[buffer(3)]])
{
  …
}
```

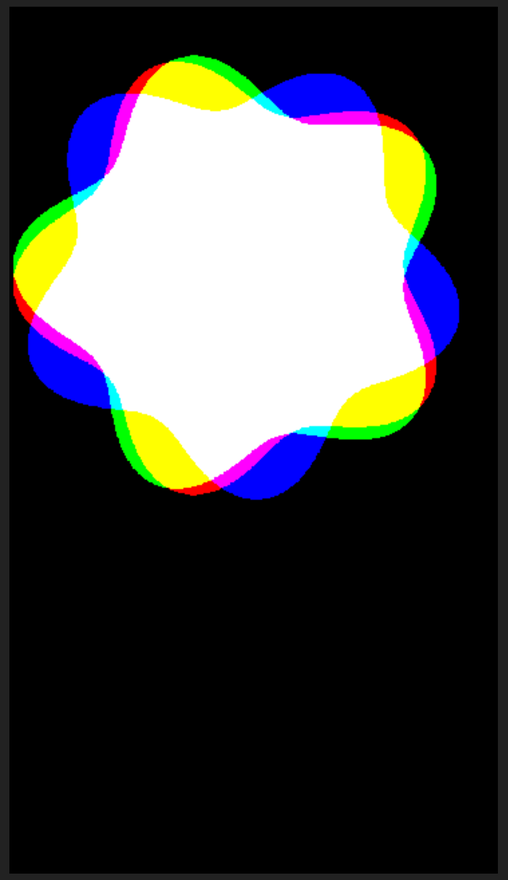


遊んでみよう！

ShaderPlayground

https://github.com/ta-ka-tsu/ShaderPlayground





参考

The Book of Shaders

https://thebookofshaders.com/

楽しい！Unityシェーダー お絵かき入門！

https://docs.google.com/presentation/d/1NMhx4HWuNZsjNRRIaFOu2ysjo04NgcpFlEhzodE8RIg/edit#slide=id.g368d0406a6_1_184

wgld.org | GLSL contents

https://wgld.org/d/glsl/