シリコンスタジオハッカソン
成果発表

電気通信大学大学院　情報学専攻　修士1年
高田 宗一郎

エモいポストエフェクト

- グリッチエフェクトが好き
- 自分の好きなもの詰め込んだ
  - 全部で6(7)エフェクト
- 全体的にこちらがとても参考になりました。
  - https://docs.google.com/presentation/d/1NMhx4HWuNZsjNRRlaFOu2ysjo04NgcpFlEhzodE8Rlg/edit#slide=id.g35d90bbe39_0_53

Glitch

- 横の線がランダムに入るグリッチ
- 少な目が好み
- 参考
  - https://qiita.com/amagitakayosi/items/65bdc3f3f7a446f9aa0a

```
float4 CustomPostProcess(Varyings input) : SV_Target
{
    UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);

    float2 uv = input.texcoord.xy;
    uint2 positionSS = uv * _ScreenSize.xy;
    float4 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS);

    // ノイズ(y座標ごと)
    float n = sin(uv.y * _Param1 + _Time.y * _Param2) * sin(uv.y * _Param3 + _Time.y * _Param4);


    uv.x = frac(uv.x + n * _Intensity);
    uv.y = frac(uv.y + n * _Intensity);
    uv.y *= frac(_Param * uv.y + n * _Param);
    float4 glitchColor = LOAD_TEXTURE2D_X(_InputTexture, uv * _ScreenSize.xy);

    outColor = lerp(outColor, glitchColor, step(frac(n * 170.), _Threshold));
    return outColor;
}
```

Glitch2

- 良い感じに横方向にずれるグリッチ
  - グリッチといっしょに色をずらすだけでめちゃくちゃ雰囲気出る
  - グレースケール化も良い感じ
- 参考
  - https://r-ngtm.hatenablog.com/entry/2019/01/23/081547
  - https://zenn.dev/kento_o/articles/08ec03e29ed636

```
float2 uv = input.texcoord;
uint2 positionSS = uv * _ScreenSize.xy;
float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;

// ポスタライズタイム
float posterize1 = floor(frac(perlinNoise(_SinTime.y) * 10.) / (1 / _FrameRate)) * (1 / _FrameRate);
float posterize2 = floor(frac(perlinNoise(_SinTime.y) * 5.) / (1 / _FrameRate)) * (1 / _FrameRate);
float posterize3 = floor(frac(perlinNoise(_SinTime.y) * 17.) / (1 / _FrameRate)) * (1 / _FrameRate);


// グリッチ発生フラグ
float flag = step(rand(posterize2), _GlitchProb);

// グリッチのずれ幅
float distort = ((2.0 * rand(posterize1) - 0.5) * 0.1) * flag;
// グリッチの高さ
float height = rand(2.0 * rand(posterize1) - 0.5);

// グリッチ生成
float glitchLine1 = step(uv.y, height);
float glitchLine2 = step(uv.y, height + _GlitchSize);
float glitch = saturate(glitchLine2 - glitchLine1);
uv.x = lerp(uv.x, uv.x + distort, glitch);

// RGBずらし
outColor.r = LOAD_TEXTURE2D_X(_InputTexture, (uv + float2(_ShiftX, _ShiftY) * flag) * _ScreenSize.xy).r;
outColor.b = LOAD_TEXTURE2D_X(_InputTexture, (uv - float2(_ShiftX, _ShiftY) * flag) * _ScreenSize.xy).b;

// グリッチ時確率でグレースケール化
float3 grayScale = Luminance(outColor).xxx;
outColor = lerp(outColor,grayScale, step(rand(posterize3),flag * _GrayProb));

return float4(outColor,1);
```

Sepia

- Sepia(大嘘)
- 色調を好きにできる
  - セピア調の雰囲気が合いそう

```
float4 CustomPostProcess(Varyings input) : SV_Target
{
    UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);

    uint2 positionSS = input.texcoord * _ScreenSize.xy;
    float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;

    outColor.r = outColor.r * _Rratio;
    outColor.g = outColor.g * _Gratio;
    outColor.b = outColor.b * _Bratio;

    return float4(outColor, 1);
}
```

ValueNoise

- ValueNoise（大嘘）
- 一部分だけ画面焼けみたいな効果
- 地味だけど割とこだわりポイント

```
float4 CustomPostProcess(Varyings input) : SV_Target
{
    UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);

    float2 uv = input.texcoord;

    uint2 positionSS = input.texcoord * _ScreenSize.xy;
    float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;

    float sizeX = (random(floor(_Time.x)) + 1.) * _ScaleX;
    float sizeY = (random(floor(_Time.y)) + 1.) * _ScaleY;

    float2 coord = uv * float2(sizeX, sizeY);

    coord.x += random(floor(_Time)) * sizeX;
    float n = random(random(floor(coord)) * floor(_Time));

    float3 pixColor = lerp(0.8,1.0,LOAD_TEXTURE2D_X(_InputTexture, floor(coord)).xyz);

    outColor = lerp(outColor * pixColor, outColor, step(n, _Threshold));
    return float4(outColor , 1);
}
```

Distance

- 良い感じに画面の端を暗くする
- 上下左右と画面の角のカーブ

```
float4 CustomPostProcess(Varyings input) : SV_Target
{
    UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);

    float2 uv = input.texcoord;
    uint2 positionSS = input.texcoord * _ScreenSize.xy;
    float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;

    // 上下左右
    float m = max(abs(uv.x - 0.5), abs(uv.y - 0.5)) * 2;
    float c1 = 1 - smoothstep(_Intensity, 1.0, m) * 0.9;

    // かど
    float d = distance(float2(0.5, 0.5), uv);
    float c2 = 1 - smoothstep(0.5, _Intensity2, d) * 0.9;

    outColor *= c1 * c2;

    return float4(outColor, 1);
}
```

DepthCollapse

- サンプルシーン、せっかく動き回れるのでそれを利用して何かやりたい

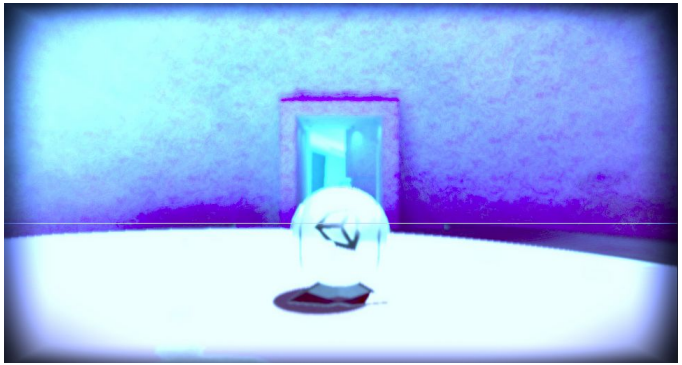
  →深度！

- ものに近づくと良い感じにぶれる
  - 色付きノイズとどっちにするか悩んだ
    - 色付きもおもしろい

```
float2 uv = input.texcoord;
uint2 positionSS = input.texcoord * _ScreenSize.xy;
float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;


// ピクセルノイズ
float ratioX = 1 / (float)_PixelX;
float ratioY = 1 / (float)_PixelY;
float2 coord = float2((int)(uv.x / ratioX) * ratioX, (int)(uv.y / ratioY) * ratioY);
float c = step(_NoiseRatio, rand(coord * floor(_Time.y * _NoiseSpeed)));

// 歪み
float dis1 = sin(_Time * 100.0) + uv.y;
float dis2 = sin(_Time * 50.0) + uv.y * 2.;
uv.x += (dis1 * dis1 + dis2 * dis2 - 1) * _DistortIntensity / 10.;

float3 collapsedColor = lerp(outColor,LOAD_TEXTURE2D_X(_InputTexture, uv * _ScreenSize.xy


// 深度で効果
float depth = Linear01Depth(LoadCameraDepth(positionSS),_ZBufferParams);
outColor = lerp(collapsedColor, outColor, smoothstep(_NoiseMin,_NoiseMax,depth));

return float4(outColor, 1);
```

（おまけ）Horror

- 深度いじってたらできた偶然の産物
- あまり意識してなかったけど今回のポストエフェクトがめちゃくちゃホラーな雰囲気に合う



```
float4 CustomPostProcess(Varyings input) : SV_Target
{
    UNITY_SETUP_STEREO_EYE_INDEX_POST_VERTEX(input);

    uint2 positionSS = input.texcoord * _ScreenSize.xy;
    float3 outColor = LOAD_TEXTURE2D_X(_InputTexture, positionSS).xyz;
    float depth = LinearEyeDepth(LoadCameraDepth(positionSS),_ZBufferParams);

    float flag = 1 - step(_Intensity, depth);

    float mosaic = LOAD_TEXTURE2D_X(_InputTexture, floor(positionSS)).xyz;
    outColor = lerp(outColor, mosaic, depth);
    return float4(outColor, 1);
}
```