AGSLで画像にエフェクトを加える

米山 俊平（@Komeyama）

今回の内容

- AGSLを使ったエフェクトの例をいくつか紹介
- Jetpack Composeで実装する場合の要点を紹介

AGSL（Android Graphics Shading Language）

- 高度なグラフィックが要求されるAndroidアプリでシェーダーを使いたいときに使用できる。
- GLSL（OpenGL Shading Language）の構文と似ている。
- Android13以降で使用可能である。

など

エフェクトの例を紹介

エフェクトを加える画像

雨粒のようなエフェクト

Shaderは以下のGLSLのコードを参考にした。

- 『Heartfelt』Created by BigWIngs
  - https://www.shadertoy.com/view/ltffzl
- 『Sitting By The Window』Created by Kris_Katur
  - https://www.shadertoy.com/view/slfSzS

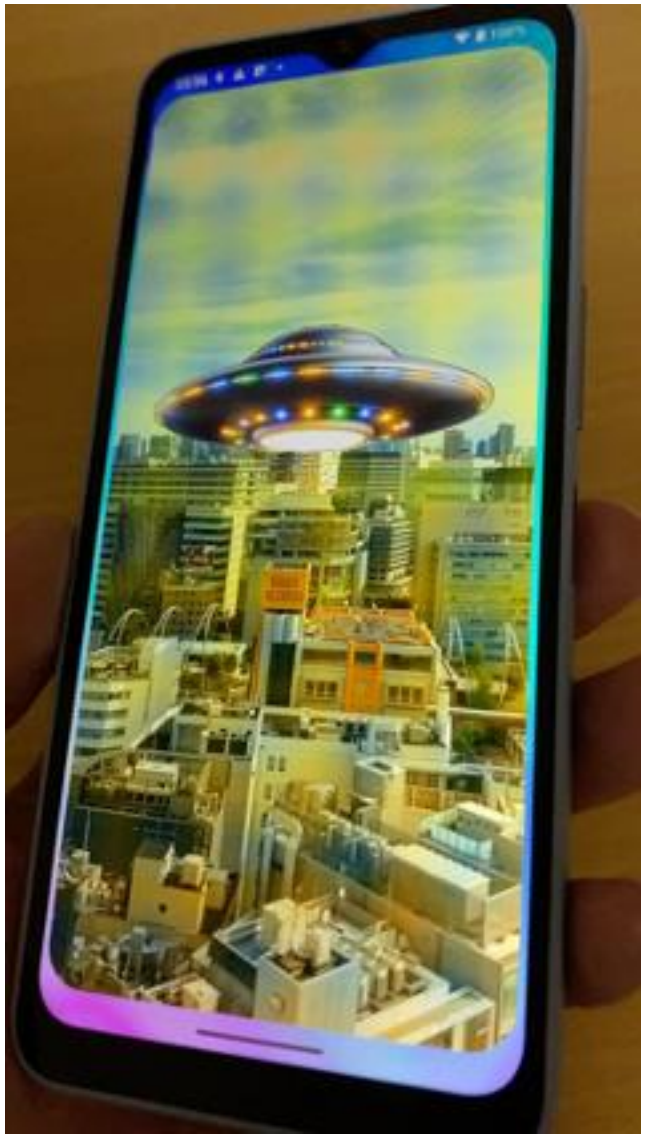
タップ操作によって変化する波紋風エフェクト

Shaderは以下のGLSLのコードを参考にした。

『RippleEffectFragmentShader』Created by abhi_bansal

- https://www.shadertoy.com/view/ldBXDD

デバイスの傾きによって変化するレアカード風エフェクト



Shaderは以下のGLSLのコードを参考にした。

『Voronoi Boronoi』Created by jcraw

- https://www.shadertoy.com/view/Mc2BD1

実装方法の要点を紹介

GLSLとの主な違いを簡単なコードで紹介

AGSL

```
uniform vec2 u_resolution;
uniform shader u_contents;

vec4 main(in vec2 fragCoord) {
   vec2 fragCoord.xy / u_resolution.xy;
   vec3 col = vec3(0,0);
   vec3 imgCol = u_contents.evel(uv * u_resolution.xy).rgb;
   col.r = imgCol.b;
   col.g = imgCol.r;
   col.b = imgCol.g;
   return vec4(col, 1.0);
}
```

GLSL

```
uniform vec2 u_resolution;
uniform sampler2D u_text;

void main() {
   vec2 uv = fragCoord.xy / u_resolution.xy
   vec3 col = vec3(0,0);
   vec4 texColor = texture(u_tex, uv);
   col.r = texColor.b;
   col.g = texColor.r;
   col.b = texColor.g;
   gl_FragColor = vec4(col, 1.0);
}
```

GLSLとの主な違いを簡単なコードで紹介

AGSL

```
uniform vec2 u_resolution;
uniform shader u_contents;

vec4 main(in vec2 fragCoord) {
    vec2 fragCoord.xy / u_resolution.xy;
    vec3 col = vec3(0,0);
    vec3 imgCol = u_contents.eval(
    col.r = imgCol.b;
    col.g = imgCol.r;
    col.b = imgCol.g;
    return vec4(col, 1.0);
}
```

GLSL

```
uniform vec2 u_resolution;
uniform sampler2D u_text;

void main() {
    vec2 uv = fragCoord.xy / u_resolution.xy
```

その他

- 座標について
  - AGSLは左上を基準、GLSLは左下が基準となる。
  - Developersに変換方法の参考例が記載されている。
    - https://developer.android.com/develop/ui/views/graphics/agsl/agsl-vs-glsl#coordinate_space
- AGSLには存在しないGLSLの関数はある。

など

Jetpack ComposeでAGSLを使う

```
val shader = RuntimeShader(AGSL_CODE)
Image(
  painter = painterResource(id = R.drawable.landscape),
  modifier = Modifier.
        onSizeChanged { size ->
           shader.setFloatUniform(
              shader.setFloatUniform("resolution", size.width.toFloat(),size.height.toFloat())
           )
        }.graphicsLayer {
           renderEffect = RenderEffect
              .createRuntimeShaderEffect(shader, "contents")
              .asComposeRenderEffect()
       }
)
```

Jetpack ComposeでAGSLを使う

```
val shader = RuntimeShader(AGSL_CODE)
Image(
  painter = painterRe
  modifier = Modifier
        onSizeChanged
           shader.set
              shader.                                    size.height.toFloat())
           )
        }.graphicsLay
           renderEffe
              .createRuntimeShaderEffect(shader,  contents )
              .asComposeRenderEffect()
      }
)
```

```
@Language("AGSL")
const val AGSL_CODE = """
  uniform float 2 resolution;
  uniform shader contens;

  vec4 main(in vec2 fragCoord) {

    . . .

""".trimIndent()
```

Jetpack ComposeでAGSLを使う

```
val shader = RuntimeShader(AGSL_CODE)
Image(
  painter = painterResource(id = R.drawable.landscape),
  modifier = Modifier.
        onSizeChanged { size ->
            shader.setFloatUniform(
                shader.setFloatUniform("resolution", size.width.toFloat(),size.height.toFloat())
            )
        }.graphicsLayer {
            renderEffect = RenderEffect
                .createRuntimeShaderEffect
                .asComposeRenderEffect()
        }
)
```

```
@Language("AGSL")
const val AGSL_CODE = """
  uniform float 2 resolution;
  uniform shader contens;

    . . .
```

Jetpack ComposeでAGSLを使う

```
val shader = RuntimeShader(AGSL_CODE)
Image(
  painter = painterResource(id = R.drawable.landscape),
  modifier = Modifier.
        onSizeChanged { size ->
           shader.setFloatUniform(
              shader.setFloatUniform("reso                    ))
           )
        }.graphicsLayer {
           renderEffect = RenderEffect
              .createRuntimeShaderEffect(shader, "contents")
              .asComposeRenderEffect()
        }
)
```

```
@Language("AGSL")
const val AGSL_CODE = """
  uniform float 2 resolution;
  uniform shader contens;

   . . .
```

Jetpack ComposeでAGSLを使う

```
val shader = RuntimeShader(AGSL_CODE)
Image(
  painter = painterResource(id = R.drawable.landscape),
  modifier = Modifier.
        onSizeChanged { size ->
            shader.setFloatUniform(
                shader.setFloatUniform("reso                    ))
            )
        }.graphicsLayer {
            shader.setFloatUniform("time", time)
            renderEffect = RenderEffect
                .createRuntimeShaderEffect(shader, "contents")
                .asComposeRenderEffect()
        }
)
```

時間やセンサー情報などのパラメーターをAGSLで使いたい場合、setFloatUnifom等をgraphicLayer内に記述する。

まとめ

- AGSLを使うことで面白いエフェクトをアプリに加えることができる。
- センサー情報等を使うことでモバイルデバイスならではの表現をアプリに加えられる。
- シェーダー自体の作成は難しいが、Shadertoyなどにある先人達の作ったGLSLのコードを参考にすれば、比較的楽にAndroidアプリに組み込むことができる。

ご清聴ありがとうございました