Flutter Shader

Scriviamone uno!

Chi sono

Simone Bonfrate

XR Engineer @ Wideverse

AI & Data Science Student @ Politecnico di Bari

Community Lead @ GDG Bari

Runner, D&D Player, Manga Reader...

Shader

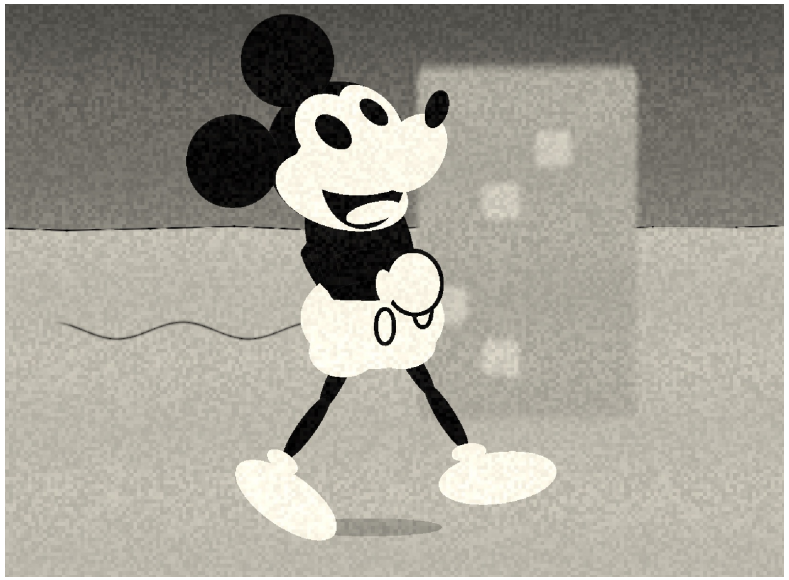
Shader: Cosa sono?

Sono dei programmi pensati per personalizzare l’aspetto grafico di un programma.

Vengono eseguiti dalla GPU per sfruttare l’elaborazione parallela ed ottimizzare il processo di rendering.

Shader: Perché imparare a scrivere gli shader?

Scrivere shader personalizzati consente di implementare effetti personalizzati senza intaccare le performance.



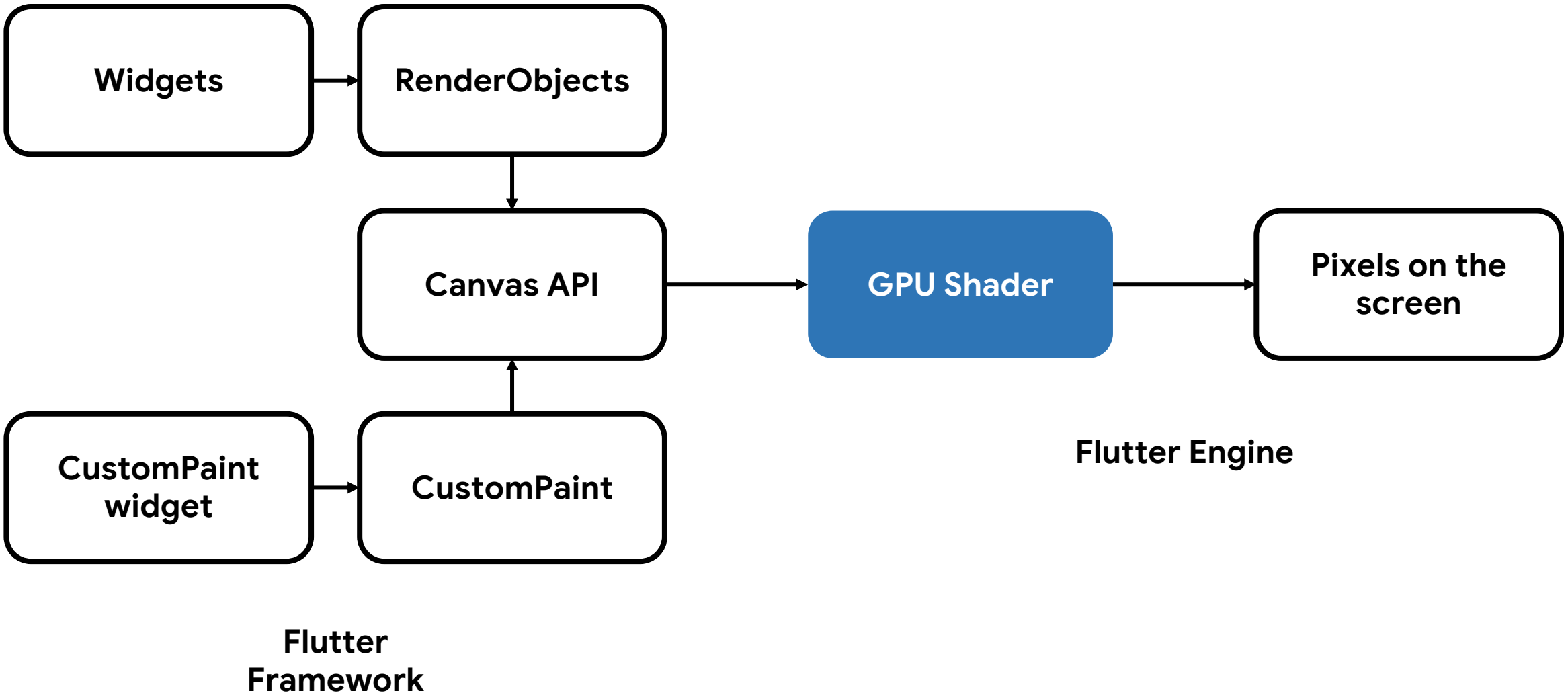
Shader: Tipologie

- Vertex Shader
- Tesselation Shader
- Geometry Shader
- FragmentShader

Shader: Tipologie

- Vertex Shader
- Tesselation Shader
- Geometry Shader
- FragmentShader

Shader: Flutter Architecture



Shader: Panoramica su Flutter Engine

Skia

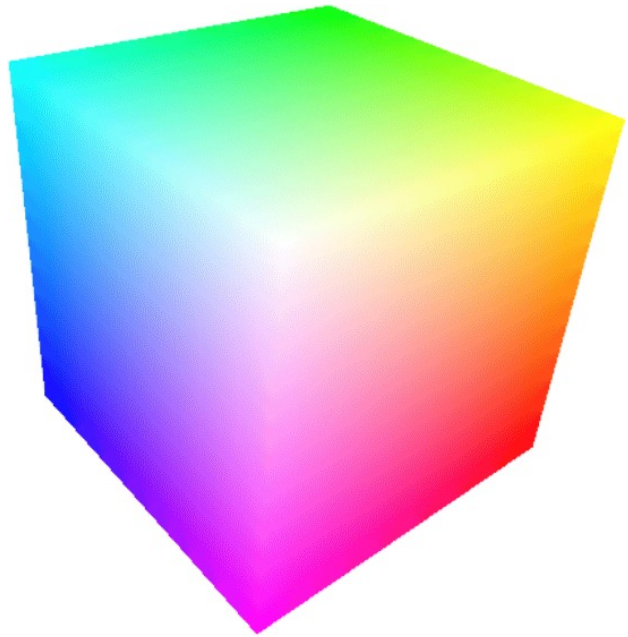
Compila gli shader a runtime, creando del jank e di conseguenza un drop del framerate

Impeller

Compila gli shader durante la build dell’applicazione, riducendo le risorse necessarie per il rendiering degli shader

GLSL: OpenGL Shading Language

E’ il linguaggio di programmazione, basato su C, per la realizzazione degli shader senza dover conoscere linguaggi specifici per ciascuna scheda video.



Scriviamo il nostro primo shader

```
//simple_shader.frag
```

Scriviamo il nostro primo shader

```
//simple_shader.frag

#version 460 core //Versione Shader

#include <flutter/runtime_effect.glsl> //Import dipendenze shader Flutter

out vec4 FragColor; //Variabile output

void main(){
}
```

Scriviamo il nostro primo shader

```
//simple_shader.frag

#version 460 core //Versione Shader

#include <flutter/runtime_effect.glsl> //Import dipendenze shader Flutter

out vec4 FragColor; //Variabile output

void main(){
    FragColor = vec4(0.4, 0.5, 0.3, 1.0);
}
```

Importiamolo su Flutter

```yaml
#pubspec.yaml

name: flutter_shaders_examples

#...

flutter:
  uses-material-design: true
```

Importiamolo su Flutter

```yaml
#pubspec.yaml

name: flutter_shaders_examples

#...

flutter:
  uses-material-design: true
  shaders:
    - assets/shaders/simple_shader.frag
```

Visualizziamo il nostro shader

```
FutureBuilder(
  future: FragmentProgram.fromAsset('assets/shaders/simple_shader.frag'),
  builder: (ctx, snap) {
    if (!snap.hasData) {
      return const Center(child: CircularProgressIndicator());
    }

    return CustomPaint(
      painter: ShaderPainter(
        shader: snap.data!.fragmentShader(),
      ),
    );
  },
);
```

Visualizziamo il nostro shader

```
class ShaderPainter extends CustomPainter {
  final FragmentShader shader;

  ShaderPainter({ super.repaint, required this.shader });

  @override
  void paint(Canvas canvas, Size size) {
    canvas.drawRect(
      Rect.fromLTWH(0, 0, size.width, size.height),
      Paint()..shader = shader,
    );
  }

  //...
}
```

TADAA!

Aggiungiuamo un parametro in input

```
#version 460 core //Versione Shader

#include <flutter/runtime_effect.glsl> //Import dipendenze shader Flutter

out vec4 FragColor; //Variabile output

void main(){
    FragColor = vec4(0.4, 0.5, 0.3, 1.0);
}
```

Aggiungiuamo un parametro in input

```
#version 460 core //Versione Shader

#include <flutter/runtime_effect.glsl> //Import dipendenze shader Flutter

uniform vec4 uColor; //Variabile input

out vec4 FragColor; //Variabile output

void main(){
    FragColor = vec4(uColor.r, uColor.g, uColor.b, uColor.a);
}
```

Aggiungiuamo un parametro in input

```
#version 460 core //Versione Shader

#include <flutter/runtime_effect.glsl> //Import dipendenze shader Flutter

uniform vec4 uColor; //Variabile input

out vec4 FragColor; //Variabile output

void main(){
    FragColor = uColor;
}
```

Aggiungiamo un parametro in input

```
class ShaderPainter extends CustomPainter {
  final FragmentShader shader;

  ShaderPainter({ super.repaint, required this.shader });

  @override
  void paint(Canvas canvas, Size size) {
    canvas.drawRect(
      Rect.fromLTWH(0, 0, size.width, size.height),
      Paint()..shader = shader,
    );
  }

  //...
}
```

Aggiungiamo un parametro in input

```
class ShaderPainter extends CustomPainter {
  final FragmentShader shader;

  ShaderPainter({ super.repaint, required this.shader });

  @override
  void paint(Canvas canvas, Size size) {
    shader.setFloat(0, color.red.toDouble() / 255);
    shader.setFloat(1, color.green.toDouble() / 255);
    shader.setFloat(2, color.blue.toDouble() / 255);
    shader.setFloat(3, color.alpha.toDouble() / 255);

    canvas.drawRect(
      Rect.fromLTWH(0, 0, size.width, size.height),
      Paint()..shader = shader,
    );
  }

  //...
}
```

Disegniamo forme geometriche

```
#version 460 core

#include <flutter/runtime_effect.glsl>

uniform vec4 uColor;

out vec4 FragColor;

void main(){
    FragColor = uColor;
}
```

Disegniamo forme geometriche

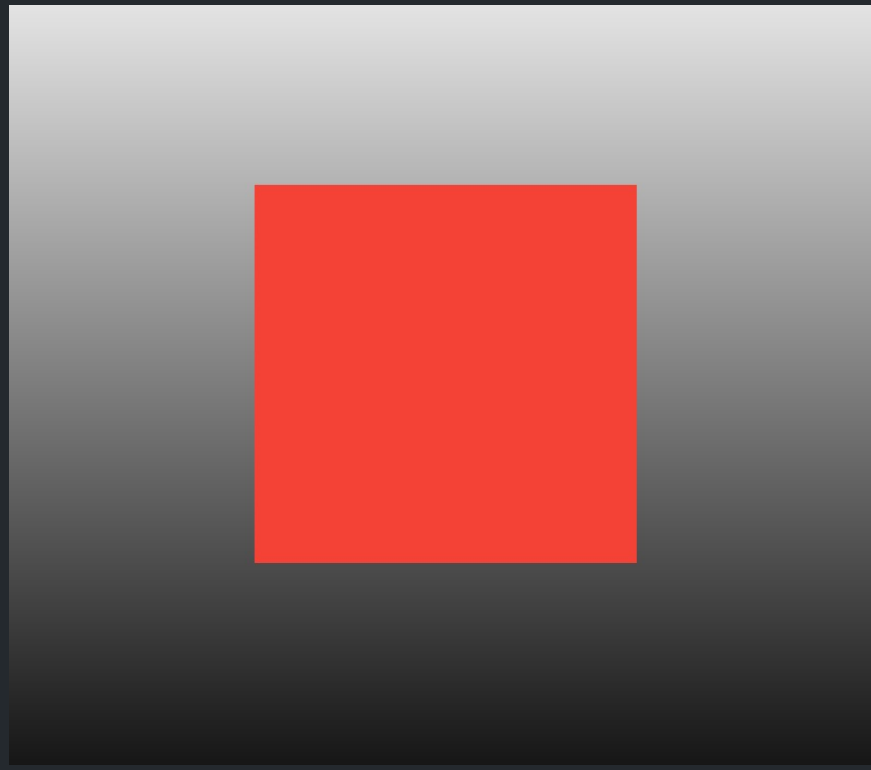
```
#version 460 core

#include <flutter/runtime_effect.glsl>

uniform vec4 uColor;

out vec4 FragColor;

void main(){
    // Otteniamo le coordinate del pixel
    vec2 position = FlutterFragCoord();


    FragColor = uColor;
}
```

Disegniamo forme geometriche

```
#version 460 core

#include <flutter/runtime_effect.glsl>

uniform vec2 uSize;
uniform vec4 uColor;

out vec4 FragColor;

void main(){
    // Otteniamo le coordinate normalizzate del pixel
    vec2 position = FlutterFragCoord() / uSize;

    // Disegniamo un quadrato posizionato al centro
    vec2 absDistanceFromCenter = abs(position.xy - vec2(0.5));

    if(absDistanceFromCenter.x < 0.2 && absDistanceFromCenter.y < 0.2){
        FragColor = uColor;
    }else{
        FragColor = mix(vec4(1.0), vec4(0.0), position.y);
    }
}
```

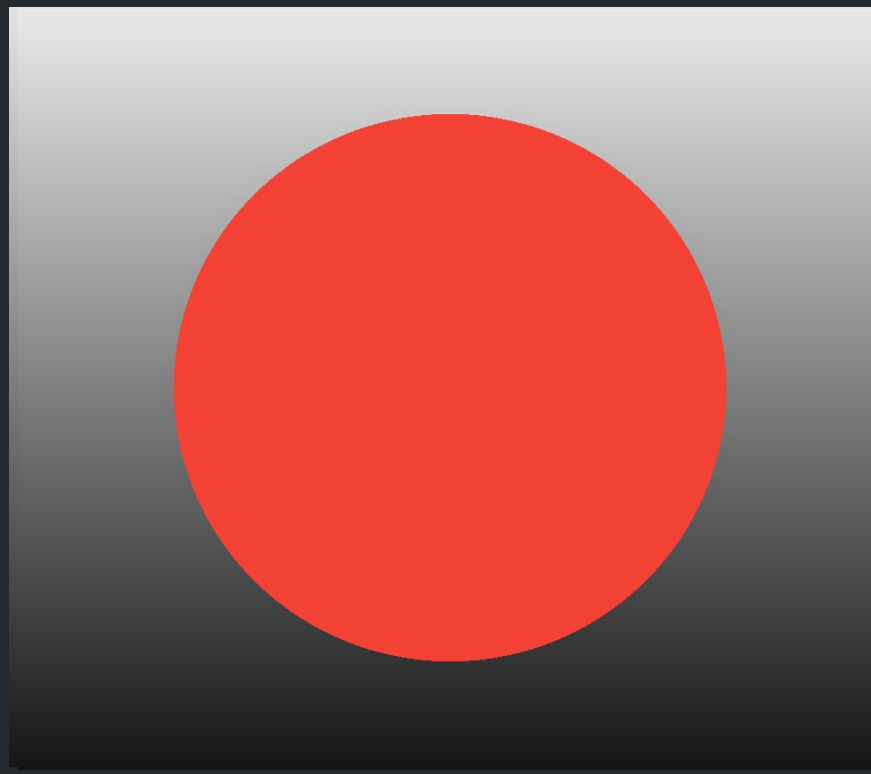
Disegniamo forme geometriche

```
#version 460 core

#include <flutter/runtime_effect.glsl>

uniform vec2 uSize;
uniform vec4 uColor;

out vec4 FragColor;

void main(){
    // Otteniamo le coordinate normalizzate del pixel
    vec2 position = FlutterFragCoord() / uSize;

    // Disegniamo un quadrato posizionato al centro
    vec2 absDistanceFromCenter = abs(position.xy - vec2(0.5));

    if(absDistanceFromCenter.x < 0.2 && absDistanceFromCenter.y < 0.2){
        FragColor = uColor;
    }else{
        FragColor = mix(vec4(1.0), vec4(0.0), position.y);
    }
}
```



Disegniamo delle forme geometriche

```
#version 460 core

#include <flutter/runtime_effect.glsl>

uniform vec2 uSize;
uniform vec4 uColor;

out vec4 FragColor;

void main(){
    float value = distance(position, vec2(0.5));
    float circle = step(0.3, value);
    vec3 circleColor = vec3(circle);
    circleColor= circleColor * uColor.rgb;

    if(circle ==0.){
         FragColor = uColor;
    }else{
        FragColor = mix(vec4(1.0), vec4(0.0), position.y);
    }
}
```



Domande?

Grazie per l’attenzione

bonfry

bonfry.com

Lascia un feedback