SwiftUI meets METAL

elevate your UI game

VERONIKA ZELINKOVÁ

hahaha

METAL

VERONIKA ZELINKOVÁ

@veronikacodes

5 / 40 colleagues

ever touched Metal

Unique
Fast

VERONIKA ZELINKOVÁ

@veronikacodes


IOS DEVELOPER
WHAT HAPPENED TO HIM?
HE TRIED TO LEARN METAL...
imgflip.com

1 / 5

What is this METAL?

VERONIKA ZELINKOVÁ

@veronikacodes

Metal (”Apple's OpenGL”)

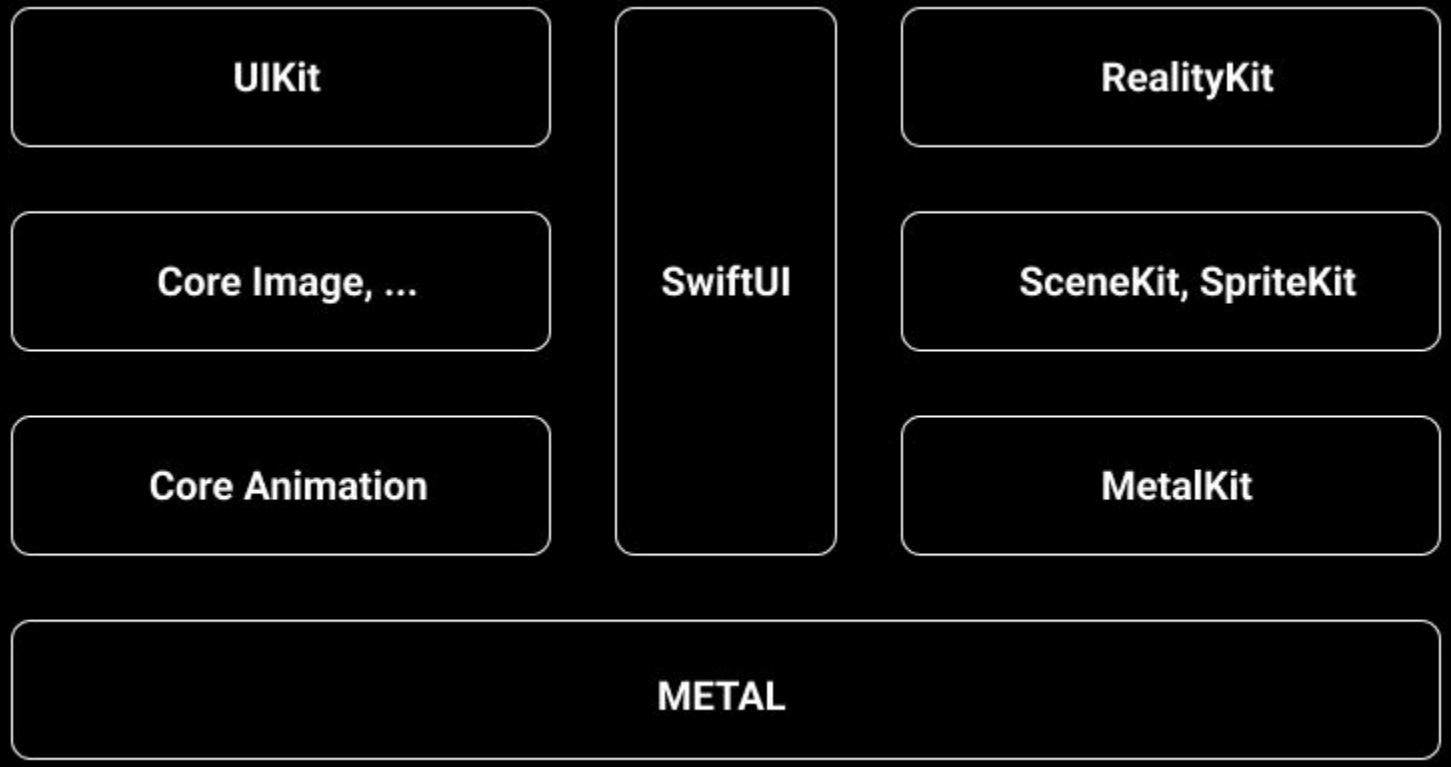
- Low level GPU programming framework for iOS
- Introduced on WWDC 2014
- Used in gaming, video processing, scientific computing, ...

VERONIKA ZELINKOVÁ
@veronikacodes

Metal vs. OpenGL

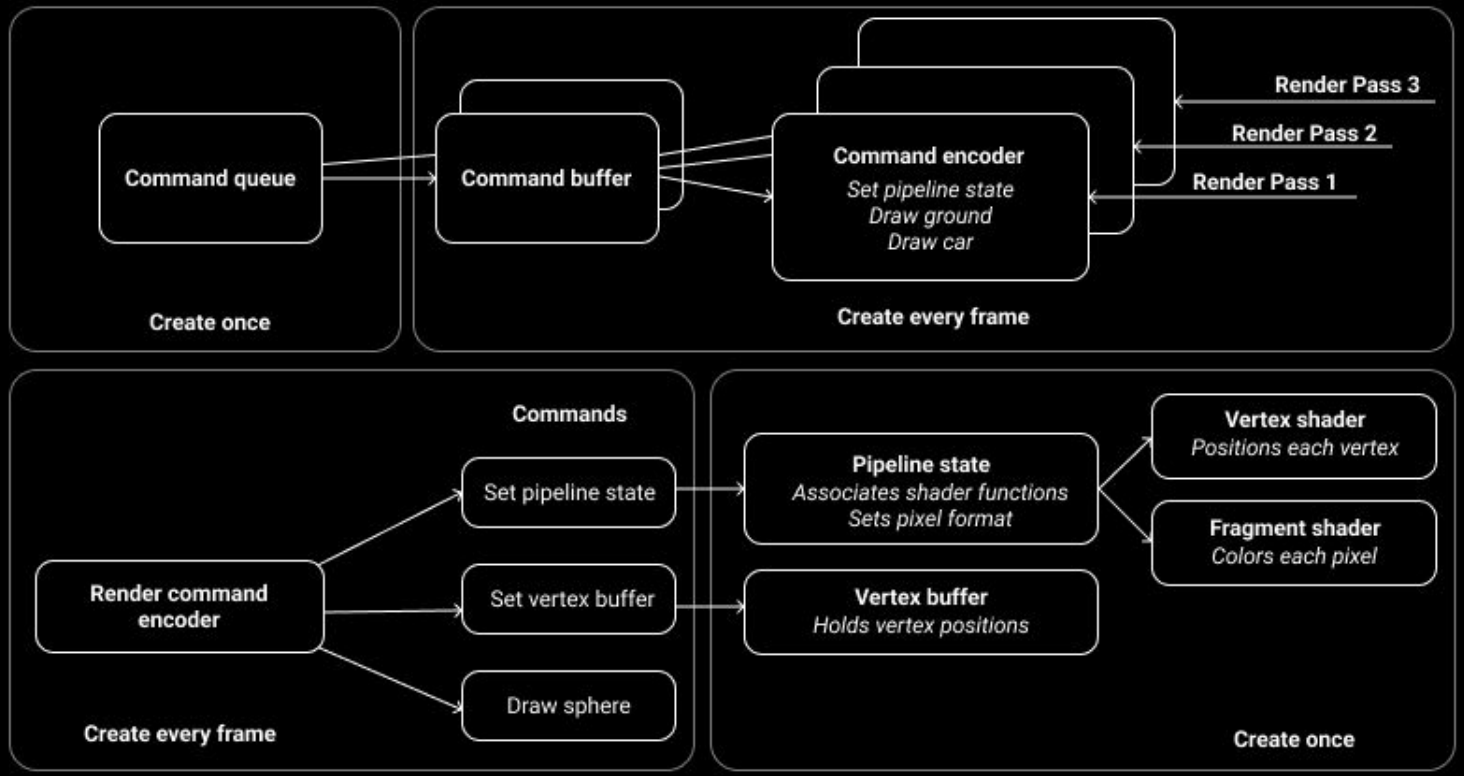
VERONIKA ZELINKOVÁ

@veronikacodes


UIKit
Core Image, ...
Core Animation
SwiftUI
RealityKit
SceneKit, SpriteKit
MetalKit
METAL

100+

lines of code


Command queue
Create once
Command buffer
Command encoder
Set pipeline state
Draw ground
Draw car
Render Pass 3
Render Pass 2
Render Pass 1
Create every frame
Commands
Set pipeline state
Render command encoder
Set vertex buffer
Draw sphere
Create every frame
Pipeline state
Associates shader functions
Sets pixel format
Vertex buffer
Holds vertex positions
Vertex shader
Positions each vertex
Fragment shader
Colors each pixel
Create once

iOS 17+

```
func colorEffect(
  _ shader: Shader,
  isEnabled: Bool = true
) -> some View
```

```
func distortionEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true
) -> some View
```

```
func layerEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true
) -> some View
```

iOS 17+

```
func colorEffect(
  _ shader: Shader,
  isEnabled: Bool = true
) -> some View
```

```
func distortionEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true
) -> some View
```

```
func layerEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true
) -> some View
```

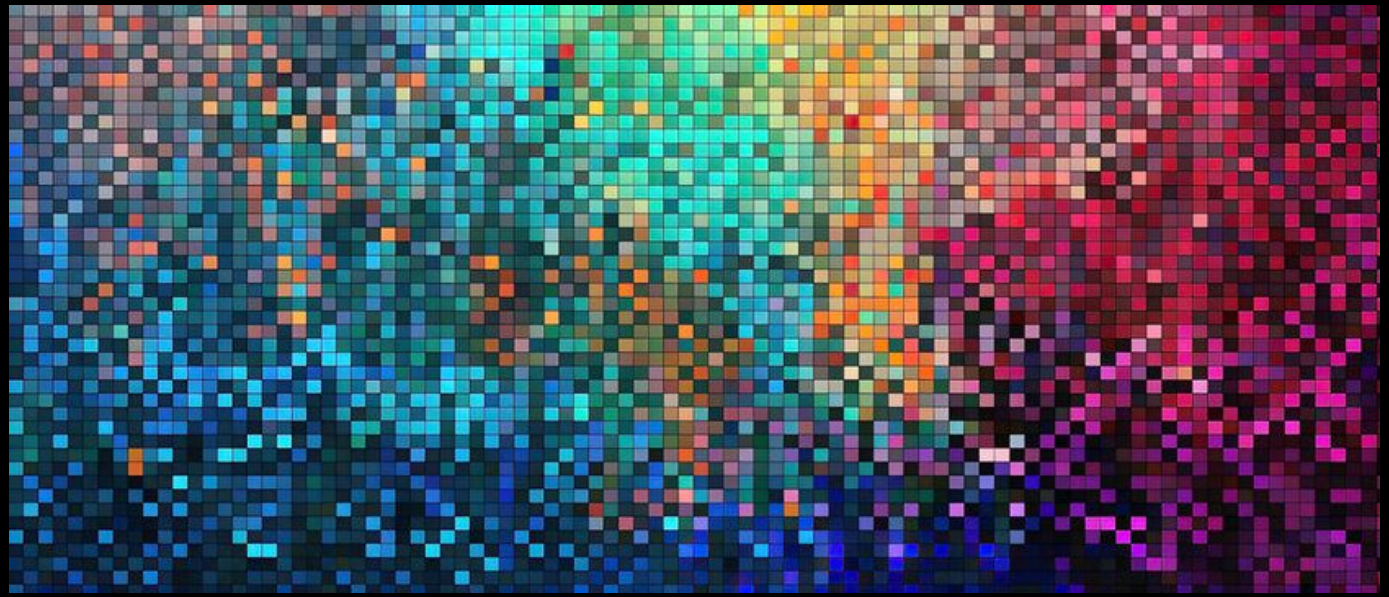
Shaders

= tiny functions that run on the GPU in parallel

VERONIKA ZELINKOVÁ

@veronikacodes

10 000 pixels



VERONIKA ZELINKOVÁ

@veronikacodes

GPU:

```
[[ stitchable ]] half4 name(float2 position, half4 color, args...)
```

- color
- image
- vector (position, frame, …)

CPU:

```
Image(name: .logo)
    .colorEffect(ShaderLibrary.name(
        .color(.purple),
        .image(image),
        .float2(touchPosition)
))
```


VERONIKA ZELINKOVÁ

@veronikacodes

Metal Shading Language 101

- float, float2, half4
- max(:), floor(:), length(:), sqrt(:), normalize(:)

```
- float2 vector1 = float2(2, 3);
- float2 vector2 = float2(3, 4);

- vector1 * 10 = float2(vector1.x * 10, vector1.y * 10) = (20, 30)
- vector1 + vector2 = float2(vector1.x + vector2.x, vector1.y + vector2.y) = (5, 7)
- vector1 * vector2 = float2(vector1.x * vector2.x, vector1.y * vector2.y) = (6, 12)

- length(vector2) = sqrt(3 * 3 + 4 * 4) = sqrt(25) = 5
- normalize(vector2) = vector2 / length(vector2) = float2(3, 4) / 5 = (0.6, 0.8)
```


VERONIKA ZELINKOVÁ
@veronikacodes

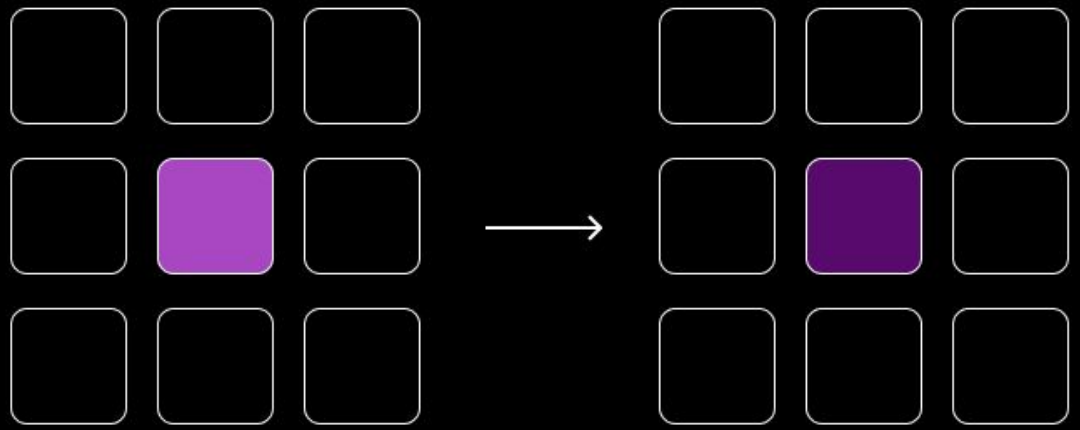
2 / 5

Color effect

VERONIKA ZELINKOVÁ

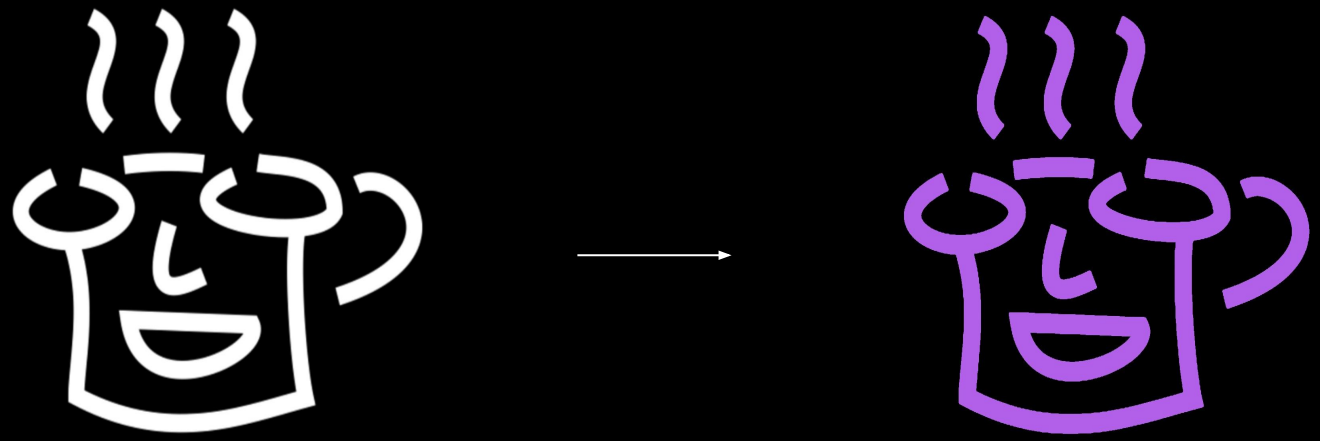
@veronikacodes

```
[[ stitchable ]] half4 name(float2 position, half4 color, args...)
```



```swift
func colorEffect(
  _ shader: Shader,
  isEnabled: Bool = true
) -> some View
```

Recolor shader



VERONIKA ZELINKOVÁ

@veronikacodes

Recolor shader

half4(1, 0, 0, 1)

half4(0, 0, 0, 1)

half4(1, 1, 1, 1)

half4(0.5, 0, 1, 1)

Recolor shader

```
// recolors non-transparent pixels
[[ stitchable ]] half4 recolor(float2 position,
                               half4 currentColor,
                               half4 newColor) {
    // TODO: Implementation



}
```

Recolor shader

```
// recolors non-transparent pixels
[[ stitchable ]] half4 recolor(float2 position,
                               half4 currentColor,
                               half4 newColor) {
    if(currentColor.a < 0.1) {
        return currentColor;
    }

}
```

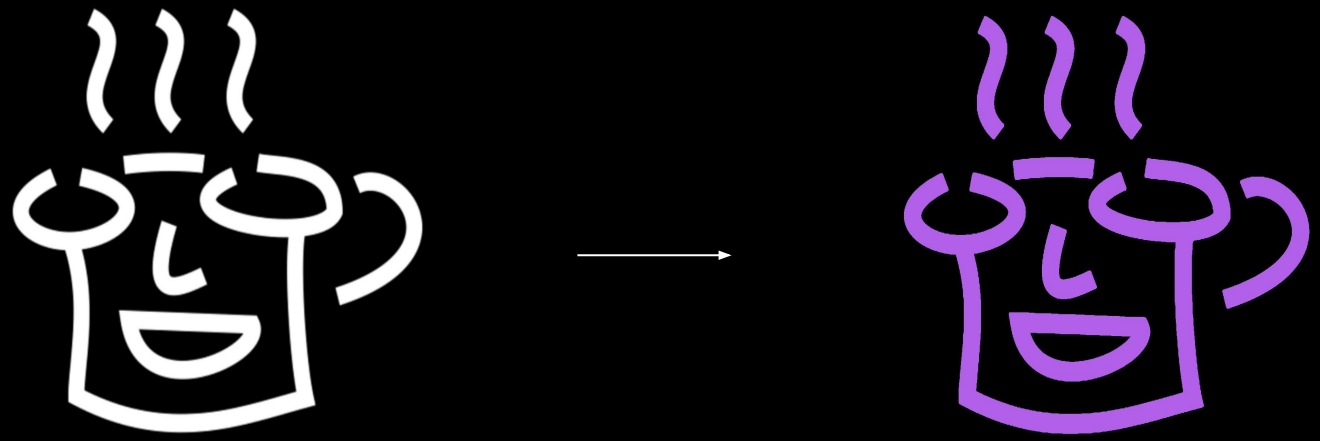
Recolor shader

```
// recolors non-transparent pixels
[[ stitchable ]] half4 recolor(float2 position,
                               half4 currentColor,
                               half4 newColor) {
    if(currentColor.a < 0.1) {
        return currentColor;
    }
    return newColor;
}
```


VERONIKA ZELINKOVÁ

@veronikacodes

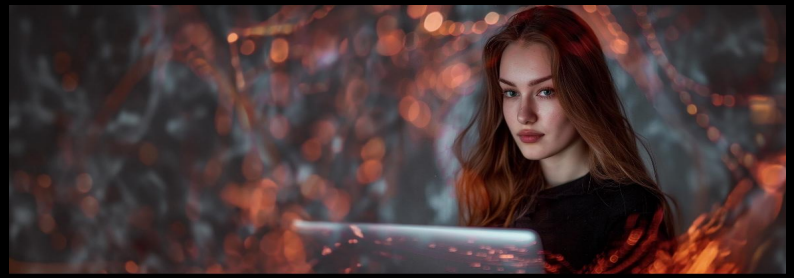
Recolor shader



VERONIKA ZELINKOVÁ

@veronikacodes

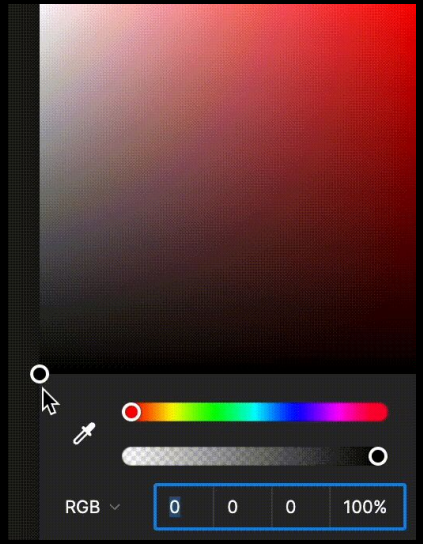
Black and White shader



VERONIKA ZELINKOVÁ

@veronikacodes

Black & White shader



VERONIKA ZELINKOVÁ

@veronikacodes

Black and White shader

```
[[ stitchable ]] half4 blackAndWhite(float2 position,
                                     half4 color) {
    // TODO: Implementation

}
```

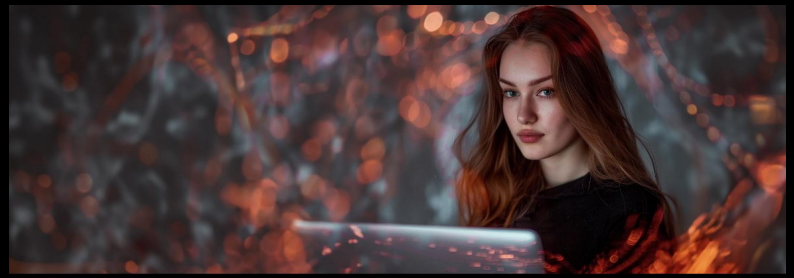
Black and White shader

```
[[ stitchable ]] half4 blackAndWhite(float2 position,
                                     half4 color) {
    float luminance = (color.r + color.g + color.b) / 3;

}
```

Black and White shader

```
[[ stitchable ]] half4 blackAndWhite(float2 position,
                                     half4 color) {
    float luminance = (color.r + color.g + color.b) / 3;
    return half4(luminance, luminance, luminance, 1);
}
```

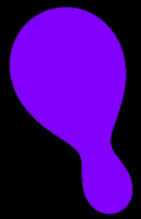
Black and White shader



VERONIKA ZELINKOVÁ

@veronikacodes

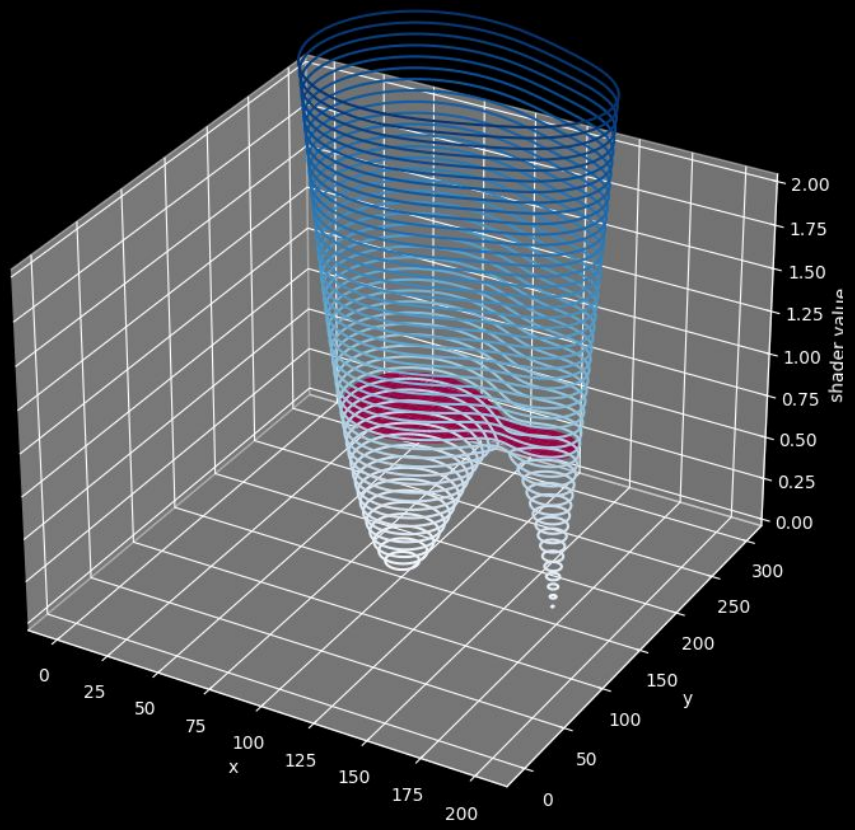
Dots shader



Dots shader

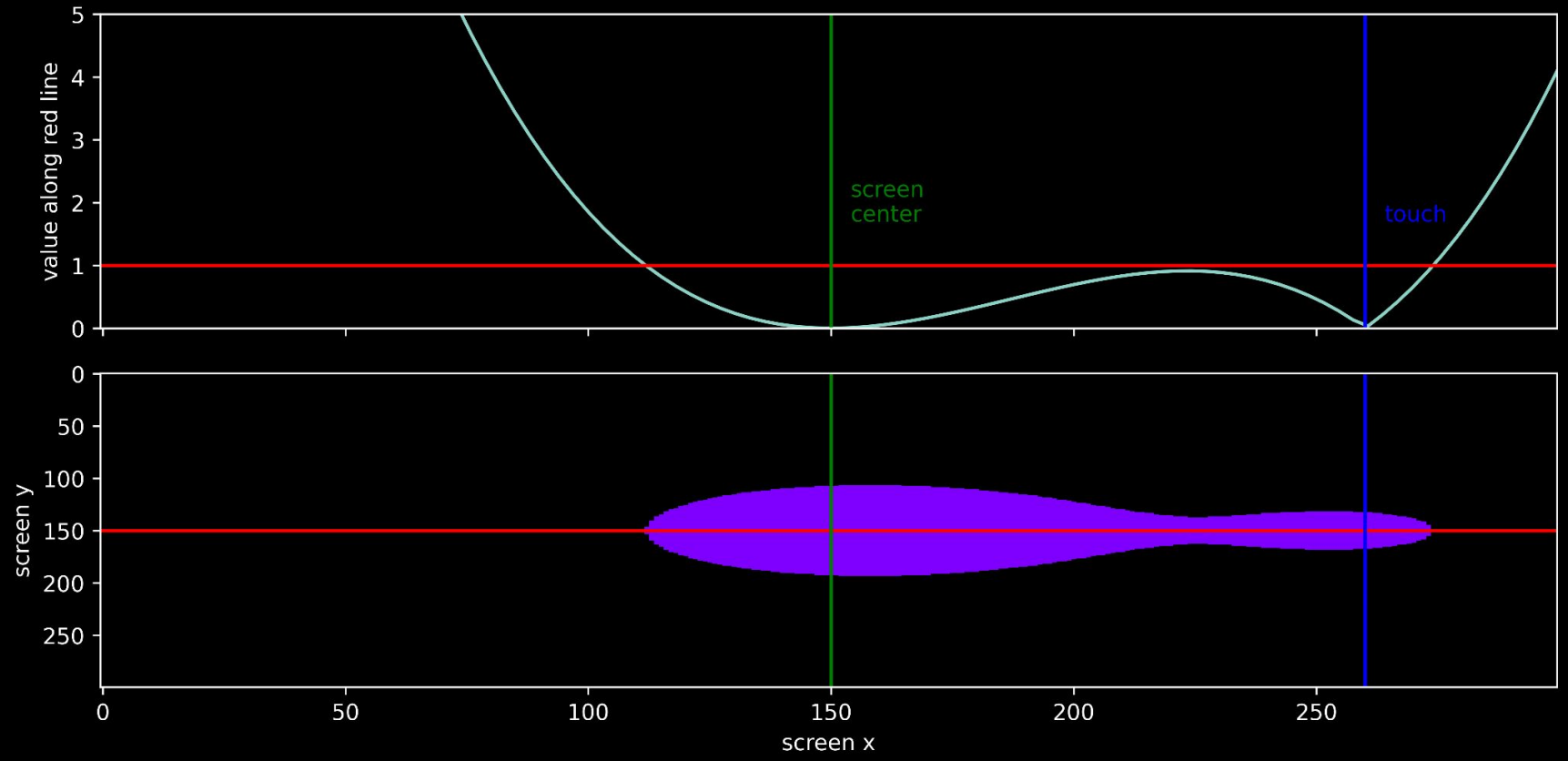
```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    // TODO: Implementation









}
```

Dots shader



VERONIKA ZELINKOVÁ

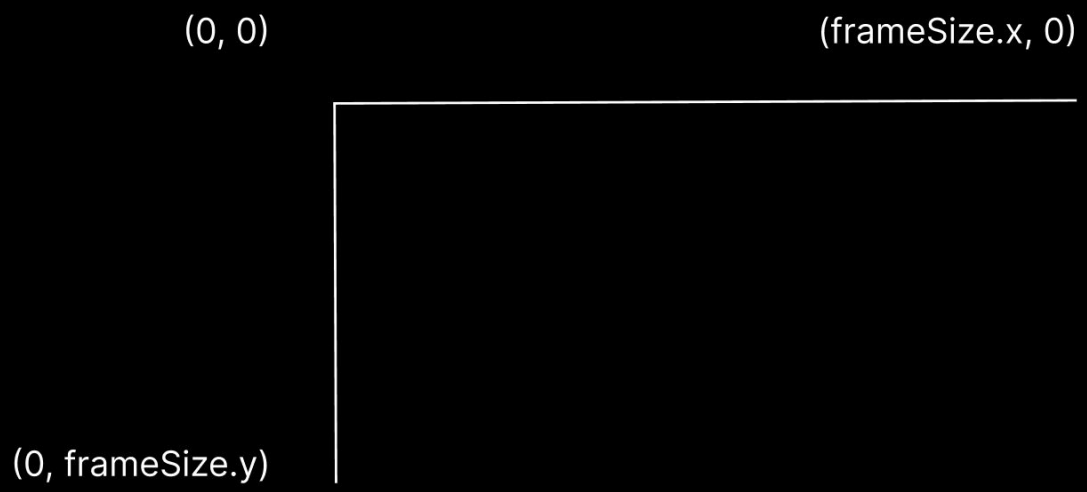
@veronikacodes



VERONIKA ZELINKOVÁ

@veronikacodes

Dots shader



Dots shader

```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 screenCenter = frameSize / 2;









}
```


VERONIKA ZELINKOVÁ

@veronikacodes

Dots shader

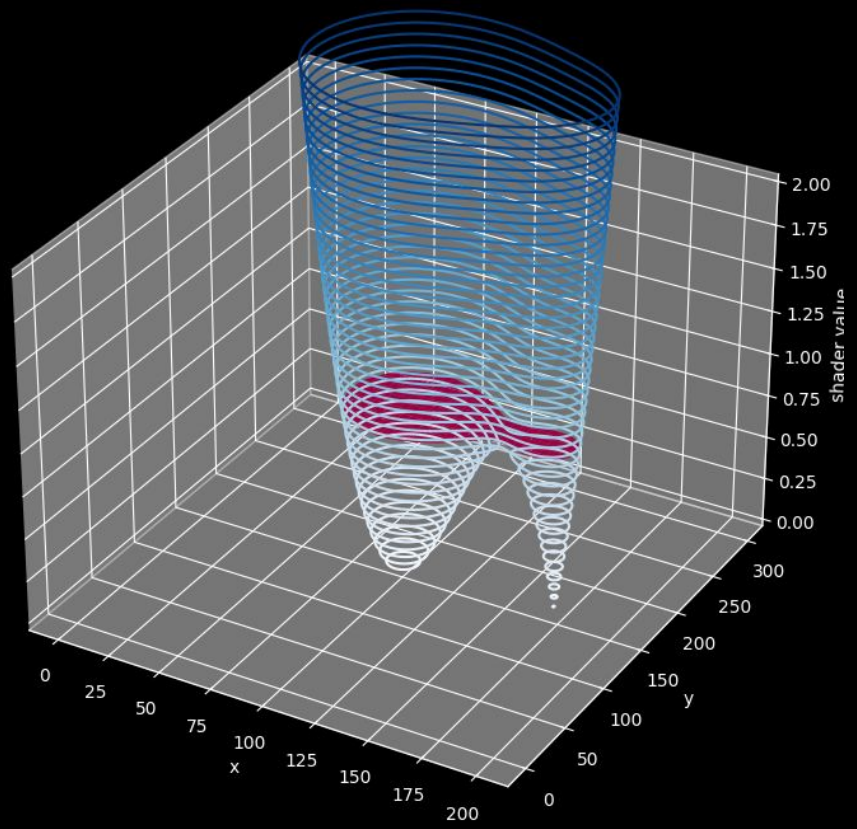
```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 screenCenter = frameSize / 2;

    float2 nPositionToCenter = (position - screenCenter) / maxSize;
    float cDistance = length(nPositionToCenter) * 10.0;









}
```

Dots shader

```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 screenCenter = frameSize / 2;

    float2 nPositionToCenter = (position - screenCenter) / maxSize;
    float cDistance = length(nPositionToCenter) * 10.0;

    float2 nTouchToPosition = nPositionToCenter - ((touch - screenCenter) / maxSize);
    float tDistance = length(nTouchToPosition) * 20.0;





}
```

Dots shader

```
cDistance * cDistance * tDistance
```



VERONIKA ZELINKOVÁ

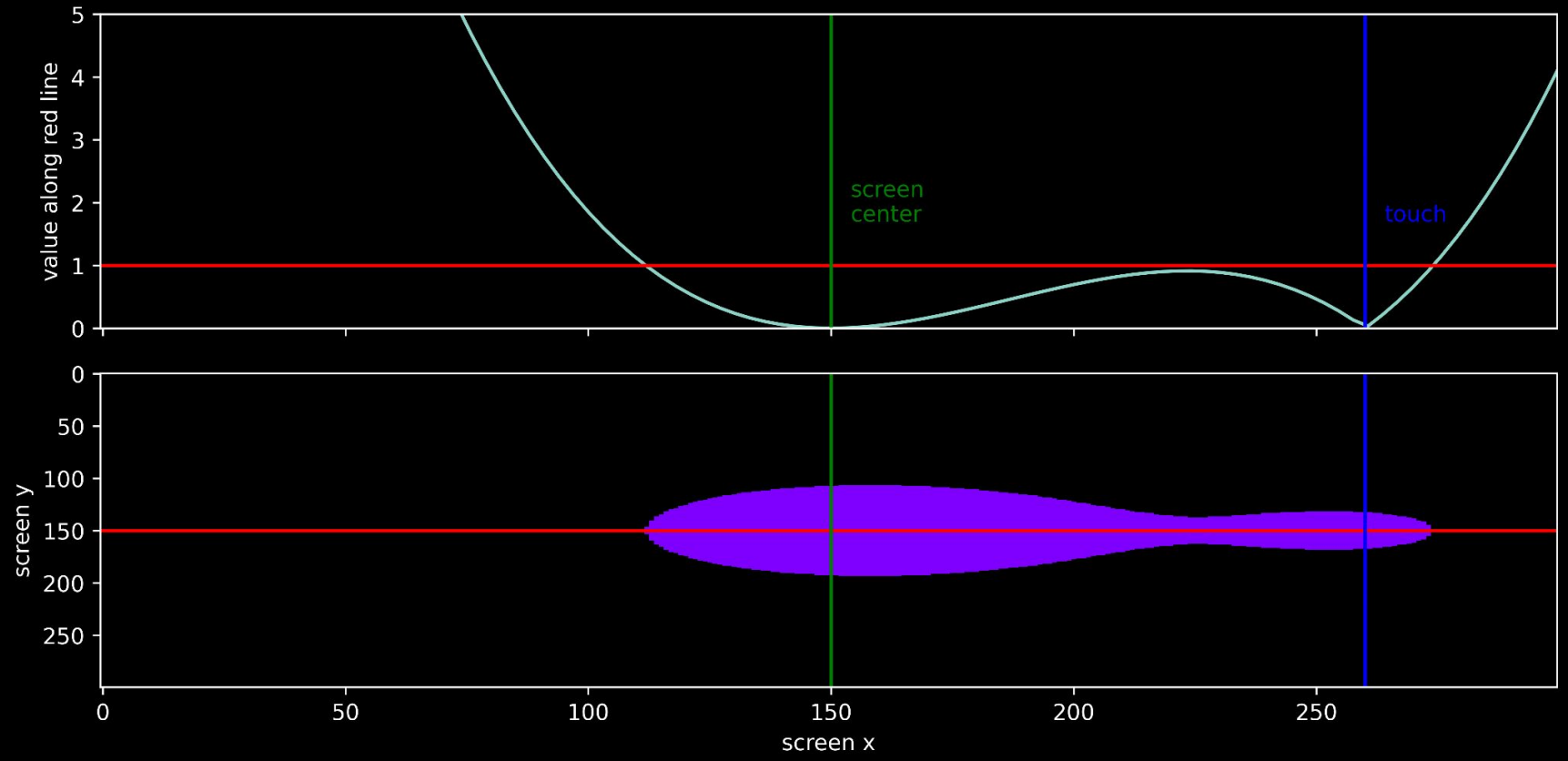
@veronikacodes

Dots shader

```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 screenCenter = frameSize / 2;

    float2 nPositionToCenter = (position - screenCenter) / maxSize;
    float cDistance = length(nPositionToCenter) * 10.0;

    float2 nTouchToPosition = nPositionToCenter - ((touch - screenCenter) / maxSize);
    float tDistance = length(nTouchToPosition) * 20.0;

    float distance = cDistance * cDistance * tDistance;





}
```


VERONIKA ZELINKOVÁ

@veronikacodes



VERONIKA ZELINKOVÁ

@veronikacodes

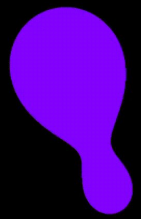
Dots shader

```
[[ stitchable ]] half4 dots(float2 position, half4 color
                            float2 touch, float2 frameSize) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 screenCenter = frameSize / 2;

    float2 nPositionToCenter = (position - screenCenter) / maxSize;
    float cDistance = length(nPositionToCenter) * 10.0;

    float2 nTouchToPosition = nPositionToCenter - ((touch - screenCenter) / maxSize);
    float tDistance = length(nTouchToPosition) * 20.0;

    float distance = cDistance * cDistance * tDistance;

    if (distance <= 1) {
        return half4(0.5, 0, 1, 1); // purple
    }
    return half4(0, 0, 0, 1); // black
}
```

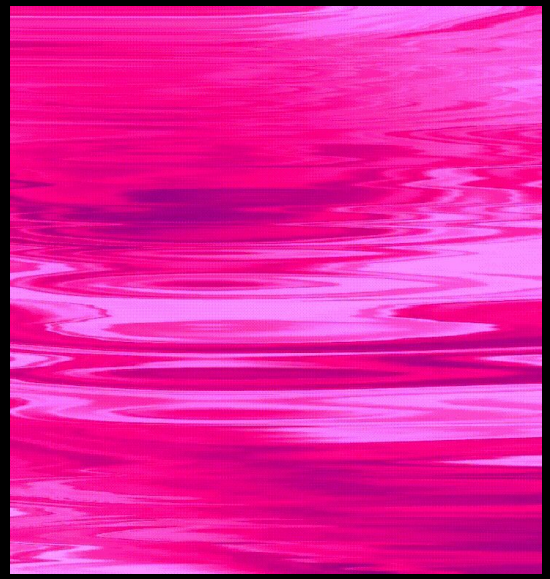

VERONIKA ZELINKOVÁ

@veronikacodes

Dots shader



Texture shader

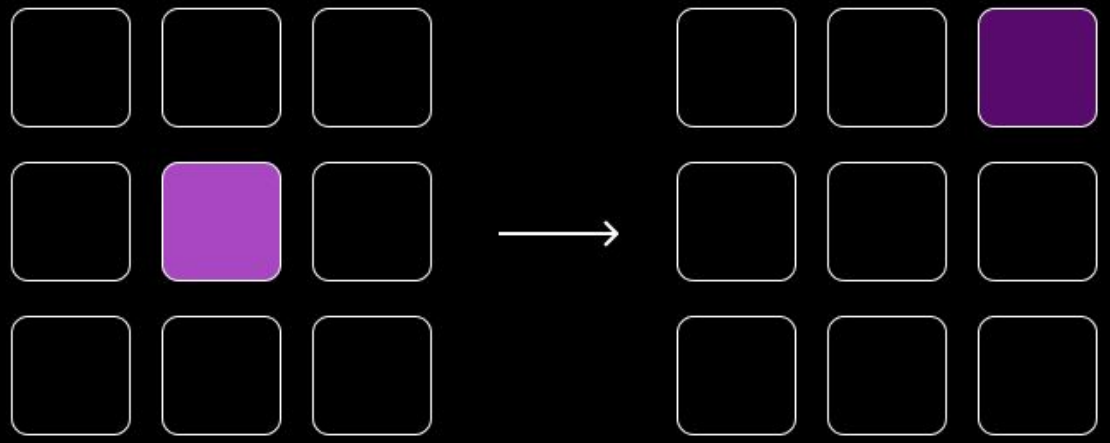


Loading shaders

3 / 5

Distortion effect

```
[[ stitchable ]] float2 name(float2 position, args...)
```



VERONIKA ZELINKOVÁ

@veronikacodes

```swift
func distortionEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true,
) -> some View
```

maxSampleOffset

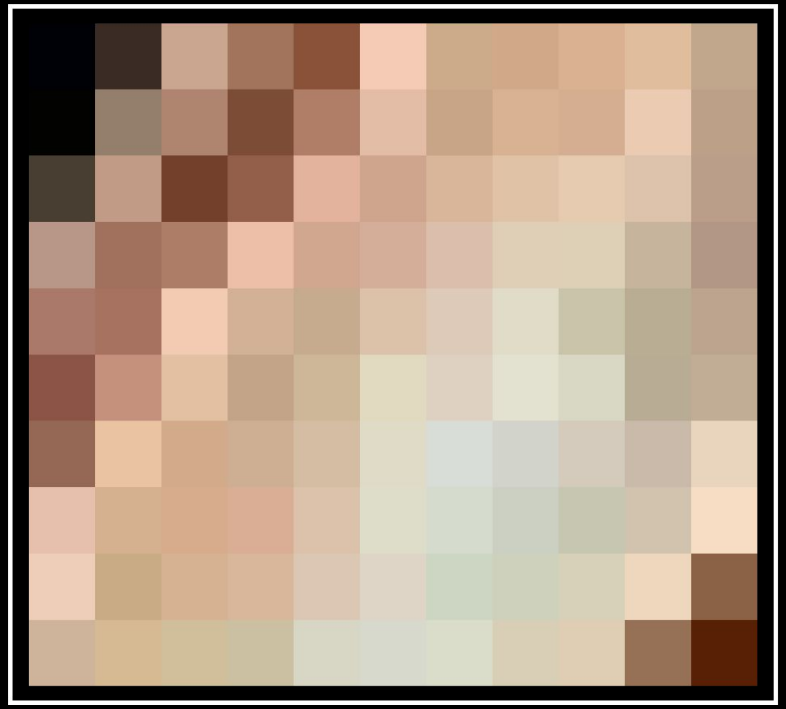
```
return float2(position.x + time, position.y + time);
maxSampleOffset: .zero
```

maxSampleOffset

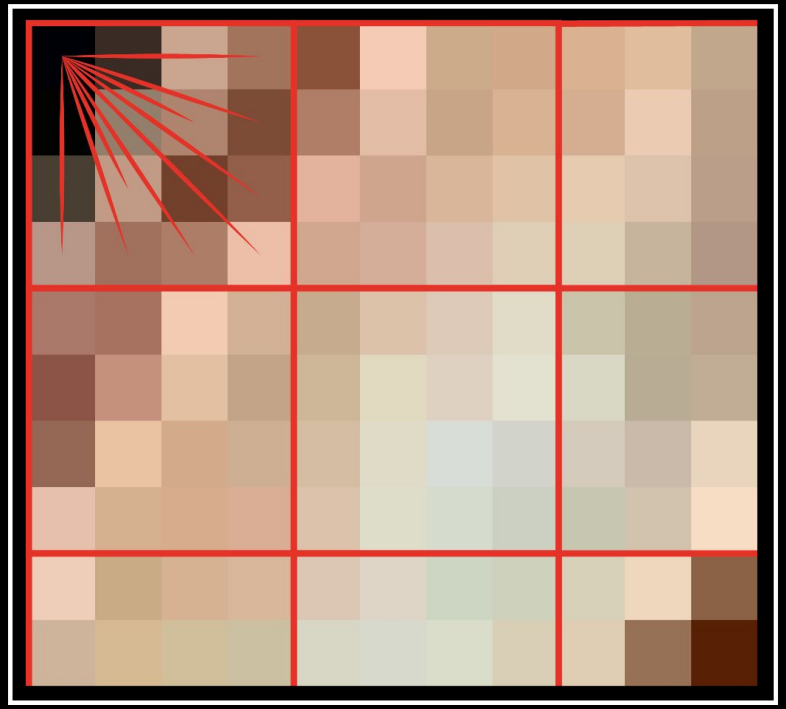
```
return float2(position.x + time, position.y + time);
maxSampleOffset: .init(width: time, height: time)
```

Pixelate shader

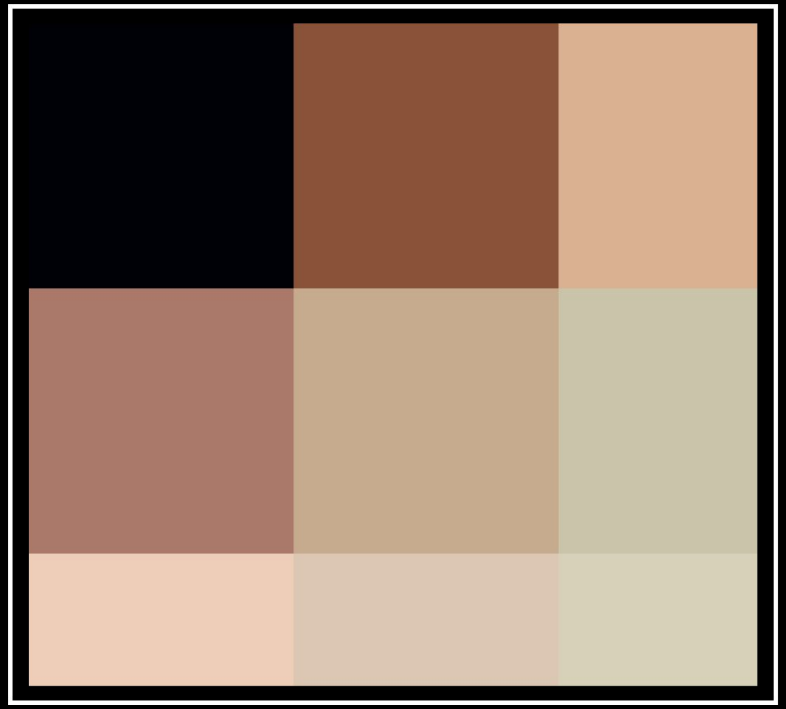
Pixelate shader



Pixelate shader



Pixelate shader



Pixelate shader

```
[[ stitchable ]] float2 pixelate(float2 position,
                                 float strength) {
    // TODO: Implementation



}
```

Pixelate shader

```
[[ stitchable ]] float2 pixelate(float2 position,
                                 float strength) {
    float x = strength * floor(position.x / strength);
    float y = strength * floor(position.y / strength);


}
```

Pixelate shader

```
[[ stitchable ]] float2 pixelate(float2 position,
                                 float strength) {
    float x = strength * floor(position.x / strength);
    float y = strength * floor(position.y / strength);

    return float2(x, y);
}
```

Pixelate shader

4 / 5

Layer effect

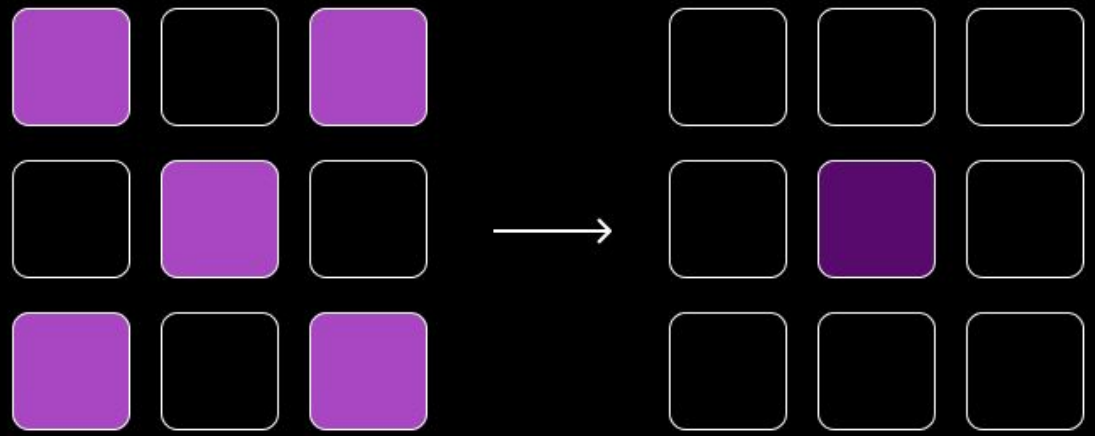
Layer effect

ISSUES:

- Color effect can't access its surroundings
- Distortion effect can't return a custom color

- Less efficient

```
#include <SwiftUI/SwiftUI.h>

[[ stitchable ]] half4 name(float2 position, SwiftUI::Layer layer, args...)
```



Layer effect

- `#include <SwiftUI/SwiftUI.h>`

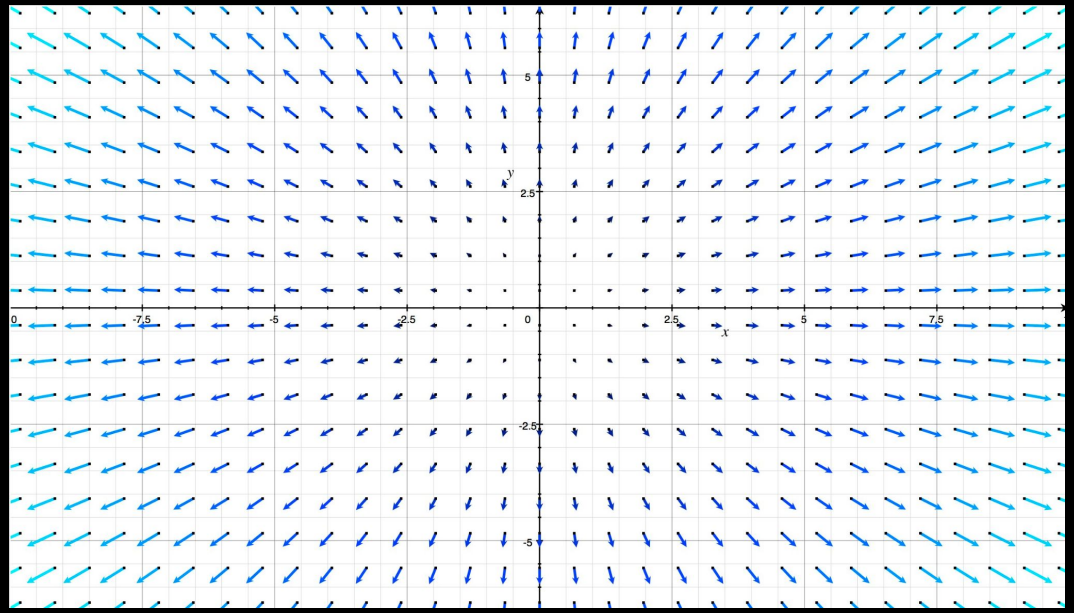
```
struct Layer {
  metal::texture2d<half> tex;
  float2 info[5];

  /// Samples the layer at `p`, in user-space coordinates,
  /// interpolating linearly between pixel values. Returns an RGBA
  /// pixel value, with color components premultipled by alpha (i.e.
  /// [R*A, G*A, B*A, A]), in the layer's working color space.
  half4 sample(float2 p) const {
    p = metal::fma(p.x, info[0], metal::fma(p.y, info[1], info[2]));
    p = metal::clamp(p, info[3], info[4]);
    return tex.sample(metal::sampler(metal::filter::linear), p);
  }
};
```

```swift
func layerEffect(
  _ shader: Shader,
  maxSampleOffset: CGSize,
  isEnabled: Bool = true
) -> some View
```

Wave shader

Wave shader

Wave shader step 1: create distortion



Wave shader step 1: create distortion

Wave shader step 1: create distortion

```
[[ stitchable ]] half4 waveStep1(float2 position, SwiftUI::Layer layer,
                                 float2 frameSize, float spread) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nScreenCenter = (frameSize / 2) / maxSize;
    float2 directionToCenter = nPosition - nScreenCenter;




}
```

Wave shader step 1: create distortion

```
[[ stitchable ]] half4 waveStep1(float2 position, SwiftUI::Layer layer,
                                 float2 frameSize, float spread) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nScreenCenter = (frameSize / 2) / maxSize;
    float2 directionToCenter = nPosition - nScreenCenter;

    float2 displacement = normalize(directionToCenter) * spread;
    float2 displacementPosition = (nPosition - displacement) * maxSize;
    return layer.sample(displacementPosition);
}
```

Wave shader step 2: add touch gesture

VERONIKA ZELINKOVÁ

@veronikacodes

Wave shader step 2: add touch gesture

```
[[ stitchable ]] half4 waveStep2(float2 position, SwiftUI::Layer layer, float2 frameSize,
                                 float2 touch, float spread) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nTouch = touch / maxSize;
    float2 directionToTouch = nPosition - nTouch;

    float2 displacement = normalize(directionToTouch) * spread;
    float2 displacementPosition = (nPosition - displacement) * maxSize;
    return layer.sample(displacementPosition);
}
```

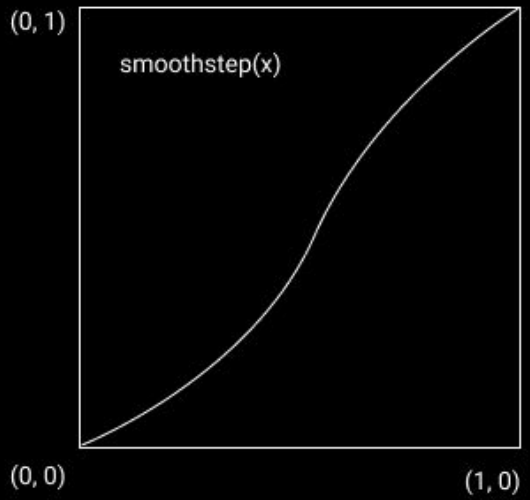
Wave shader step 3: limit outer distortion

Wave shader step 3: limit outer distortion

```
float smoothstep(float edge0, float edge1, float x)
```

- handy to create smooth transitions

```
0                          x <= edge0
1                          x >= edge1
Hermite interpolation     edge0 > x < edge1
```



Wave shader step 3: limit outer distortion

```
float outerMap = 1.0 - smoothstep(spread - width, spread, length(directionToTouch));
return layer.sample(float2(outerMap, outerMap));
```

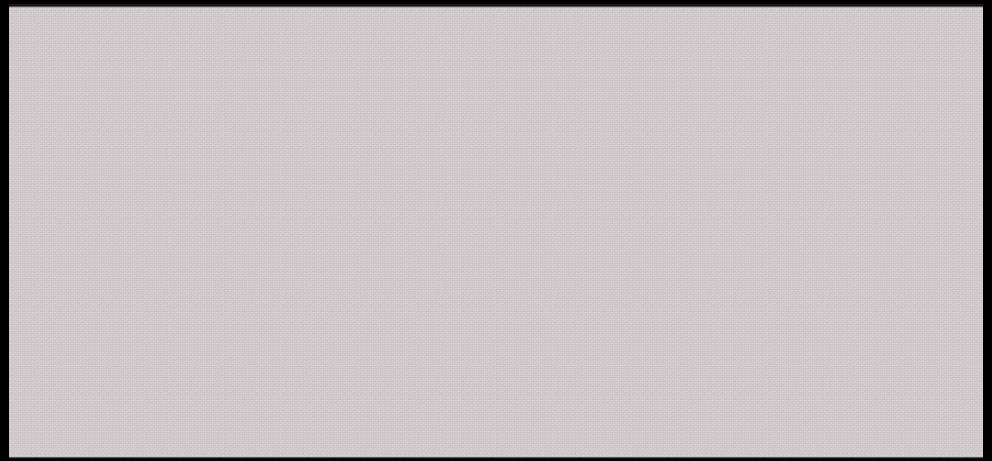
Wave shader step 3: limit outer distortion

```
[[ stitchable ]] half4 waveStep3(float2 position, float2 frameSize, SwiftUI::Layer layer,
                                 float2 touch, float spread, float width) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nTouch = touch / maxSize;
    float2 directionToTouch = nPosition - nTouch;

    float outerMap = 1.0 - smoothstep(spread - width, spread, length(directionToTouch));
    return layer.sample(float2(outerMap, outerMap));
}
```

Wave shader step 4: limit inner distortion

Wave shader step 4: limit inner distortion

```
float outerMap = 1.0 - smoothstep(spread - width, spread, length(directionToTouch));
float innerMap = smoothstep(spread - (width * 2.0), spread - width, length(directionToTouch));
float map = outerMap * innerMap;
return layer.sample(float2(map, map));
```



Wave shader

Wave shader

```
[[ stitchable ]] half4 wave(float2 position, SwiftUI::Layer layer, float2 frameSize, float2 touch,
                            float spread, float width, float amount) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nTouch = touch / maxSize;
    float2 directionToTouch = nPosition - nTouch;

    float outerMap = 1.0 - smoothstep(spread - width, spread, length(directionToTouch));
    float innerMap = smoothstep(spread - (width * 2.0), spread - width, length(directionToTouch));
    float map = outerMap * innerMap;

    float2 displacement = normalize(directionToTouch) * amount * map;
    float2 displacementPosition = (nPosition - displacement) * maxSize;
    return layer.sample(displacementPosition);
}
```

Wave shader with tint

Wave shader with tint

```
[[ stitchable ]] half4 wave(float2 position, SwiftUI::Layer layer, float2 frameSize, float2 touch,
                            half4 tint, float spread, float width, float amount) {
    float maxSize = max(frameSize.x, frameSize.y);
    float2 nPosition = position / maxSize;
    float2 nTouch = touch / maxSize;
    float2 directionToTouch = nPosition - nTouch;

    float outerMap = 1.0 - smoothstep(spread - width, spread, length(directionToTouch));
    float innerMap = smoothstep(spread - (width * 2.0), spread - width, length(directionToTouch));
    float map = outerMap * innerMap;

    float2 displacement = normalize(directionToTouch) * amount * map;
    float2 displacementPosition = (nPosition - displacement) * maxSize;
    return layer.sample(displacementPosition) + map * tint;
}
```

5 / 5

Final thoughts

Summarization

- 3 view modifiers available from iOS 17

- **Color effect** - filters, drawing
- **Distortion effect** - distortions, transitions
- **Layer effect**

Limitations

- SwiftUI views only (placeholder warning image)
- Only one image(_:) parameter can be passed to a shader
- Low performance, lagging with more complex shaders
- No control

SwiftUI modifiers
vs.
manual setup

Next steps

- https://www.shadertoy.com/
- https://github.com/twostraws/Inferno
- https://developer.apple.com/metal/
- Metal by Tutorials (book by Kodeco)
- Metal Programming Guide (book by Janie Clayton)

linkedin.com/in/**vzelinkova**

instagram: @veronikacodes