

Очень странные шейдеры

Выворачиваем приложение наизнанку

Mobius · Максим Качинкин · Роман Могутнов



Разрешите представиться



Максим Качинкин

Android Tech Lead · Dodo Engineering

Больше 13 лет в мобильной разработке

Веду канал «Мобильное Чтиво»
t.me/mobilefiction

Программный директор Podlodka Android Crew

Пишу статьи на Хабре и Medium



Роман Могутнов

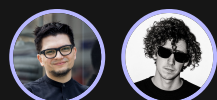
Principal Product Engineer · Dodo Engineering

Больше 10 лет создаю и развиваю цифровые продукты

ex-Yandex, Lensa (Prisma Labs), Mercuryo

Запускаю стартапы. Потому что обсуждать чужие идеи скучнее, чем проверять свои.

Могу поспорить с продактом, дизайнером и разработчиком. Даже когда все трое это я.



Dodo Engineering



IT-команда Dodo Brands



380+

человек работает в
Dodo Engineering

40+

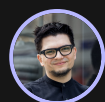
используемых систем
и сервисов

22+

млн клиентов наших
сервисов

847

заказов в минуту в
пике выдерживает
Dodo IS





Первомайская, 85 ▾
Открыты



 Хочу пиццу



485

ДОДОКОИНЫ



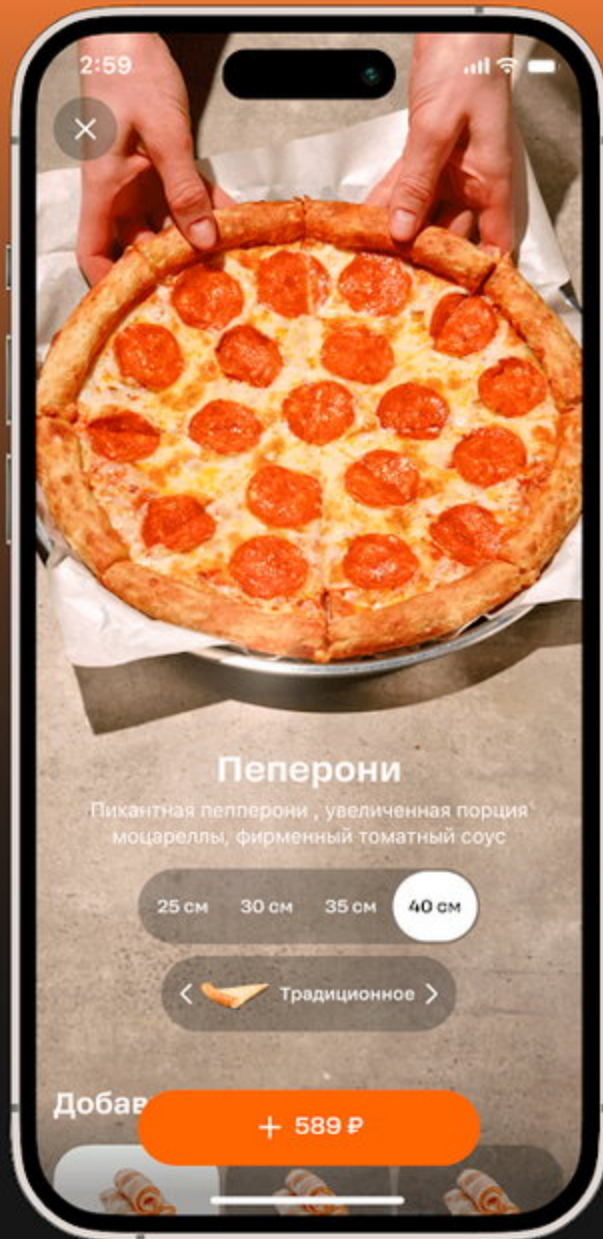
Мои заказы

12 ЗАКАЗОВ



Адреса доставки

2 АДРЕСА



10:00

улица Черняховского, 32 ▾
Доставка бесплатная около 35 минут

256

Всё Завтрак Промо Хиты Любят дети Новин

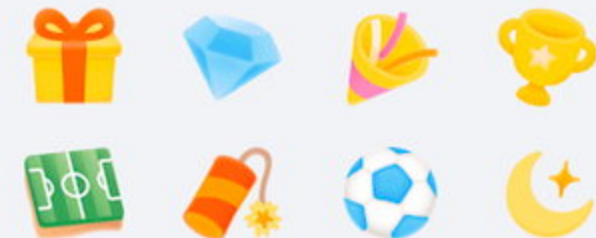
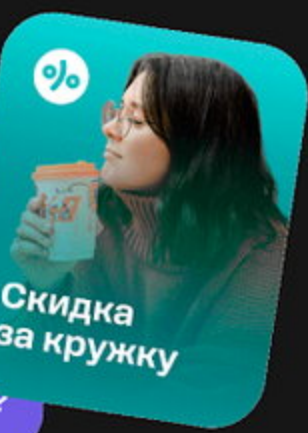
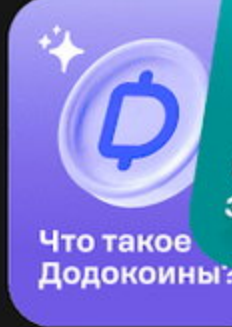
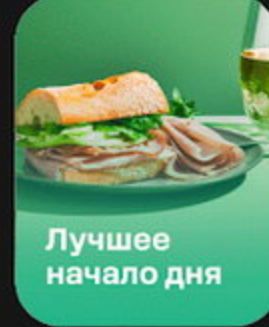
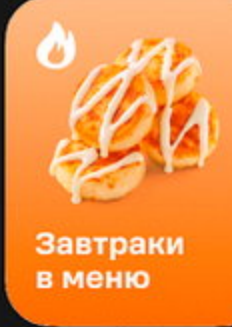


GENSHIN

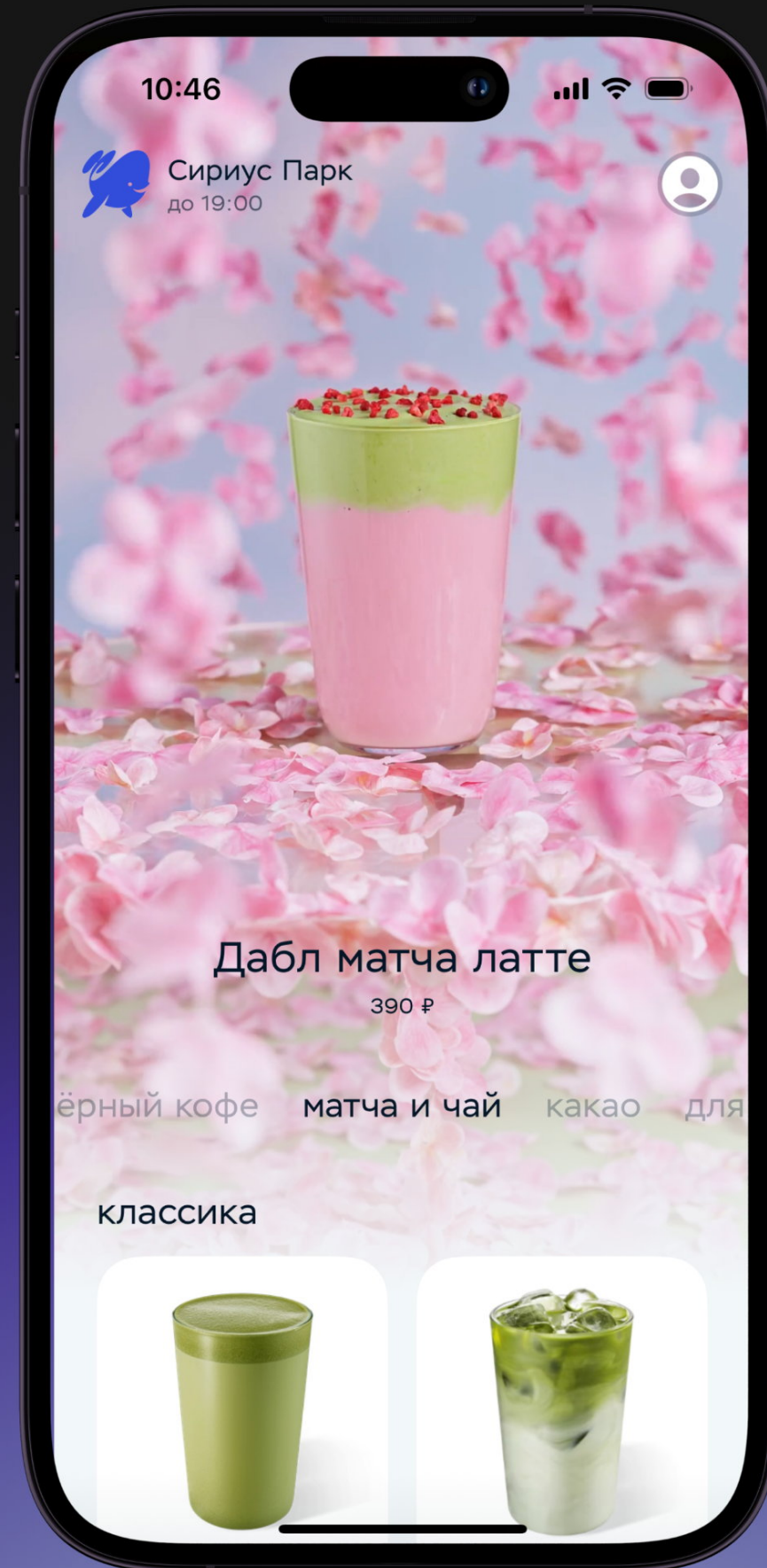
78

Последний шанс!

Успей забрать Баттше



05



Что забираем с собой



01

Написать эффект

Первый шейдер уже сегодня

02

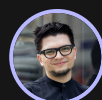
Понять рендеринг

UI без видео и с видео

03

Выбрать реализацию

Трейд-оффы по эффектам на видео





1. Простой шейдер

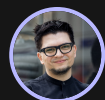
2. Шейдер Изнанки

3. Видео не работает

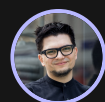
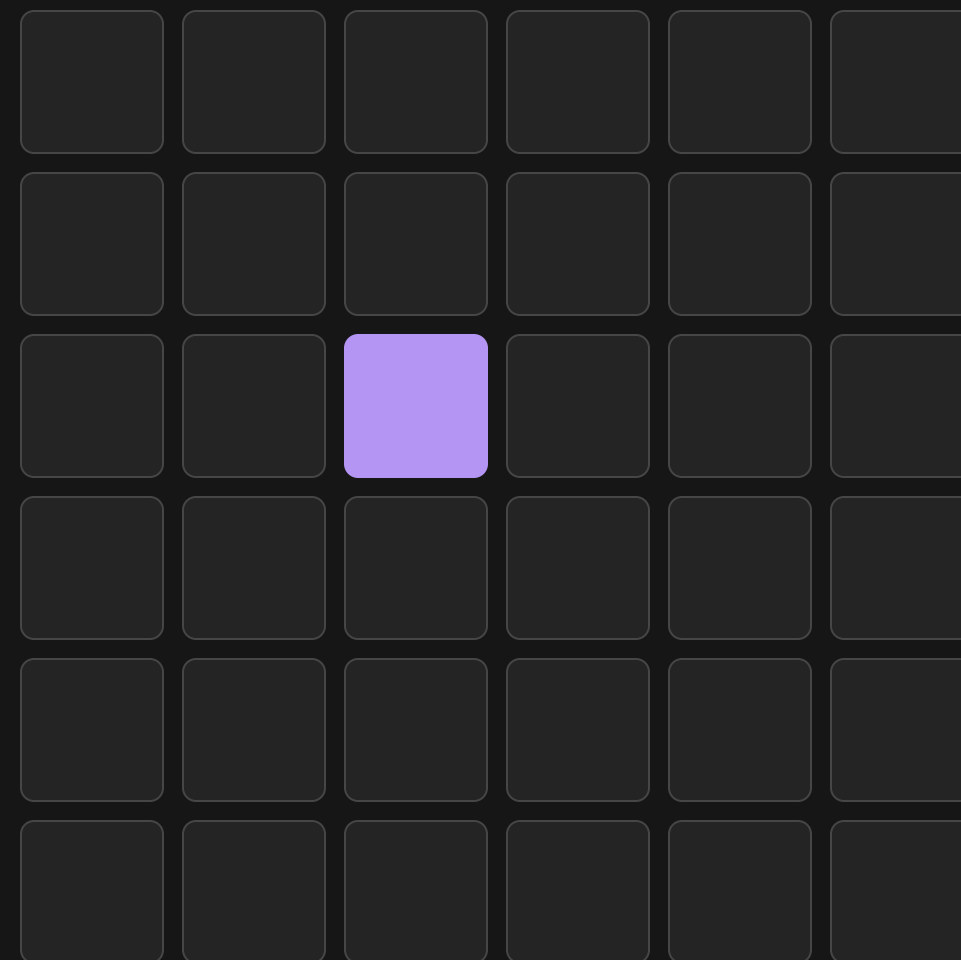
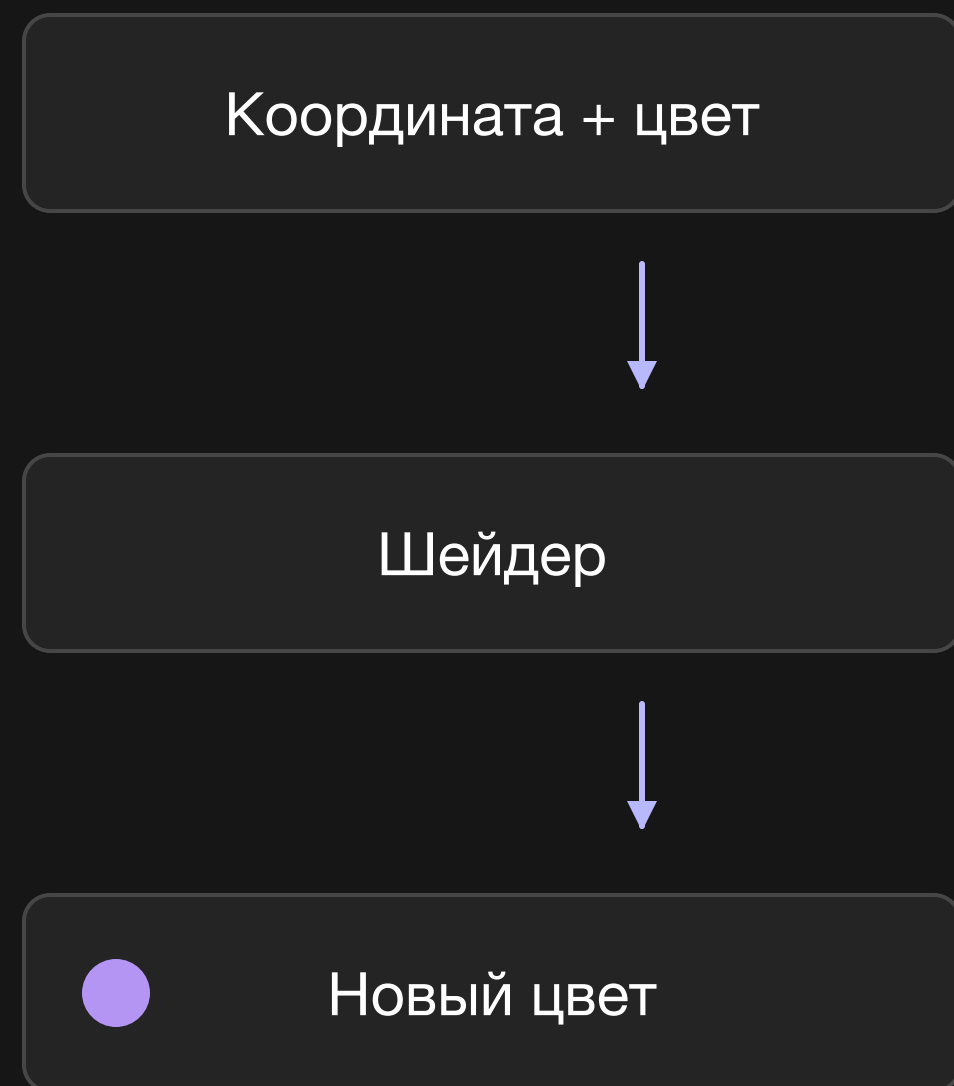
4. Шейдер для видео

5. Performance

6. Выводы



Что такое шейдер?



Шейдер, который ничего не делает



Android

```
uniform shader image;

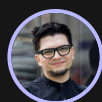
half4 main(float2 coord) {

    return image.eval(coord);
}
```

iOS

```
[[ stitchable ]]
half4 identity(float2 position, half4 color) {

    return color;
}
```



Красим в красный цвет



Android

```
uniform shader image;

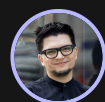
half4 main(float2 coord) {
    half4 c = image.eval(coord);

    return half4(
        c.r,
        c.g * 0.15,
        c.b * 0.15,
        c.a
    );
}
```

iOS

```
[[ stitchable ]] half4 redTint(
    float2 position,
    half4 color
) {
    half4 c = color;

    return half4(
        c.r,
        c.g * 0.15,
        c.b * 0.15,
        c.a
    );
}
```



Подключить к UI

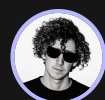


Android

```
val shader = remember(source) {  
    RuntimeShader(source)  
}  
  
val effect = remember(shader) {  
    RenderEffect.createRuntimeShaderEffect(  
        shader, "image"  
    ).asComposeRenderEffect()  
}  
  
Box(Modifier.graphicsLayer {  
    renderEffect = effect  
}) {  
  
    ProductDetails(  
        background = { Image(/* cover */) }  
    )  
}
```

iOS

```
let shader = ShaderLibrary.redTint()  
  
// Отдельного RenderEffect нет  
  
ProductDetails {  
    Image("cappuccino-cover")  
        .resizable()  
        .scaledToFill()  
}  
  
.colorEffect(shader)
```





1. Простой шейдер

2. Шейдер Изнанки

3. Видео не работает

4. Шейдер для видео

5. Performance

6. Выводы





Изнанка: четыре составляющие



Glitch

+



Цветокоррекция

+



Частицы

+



Лианы

→



Изнанка



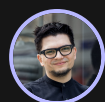
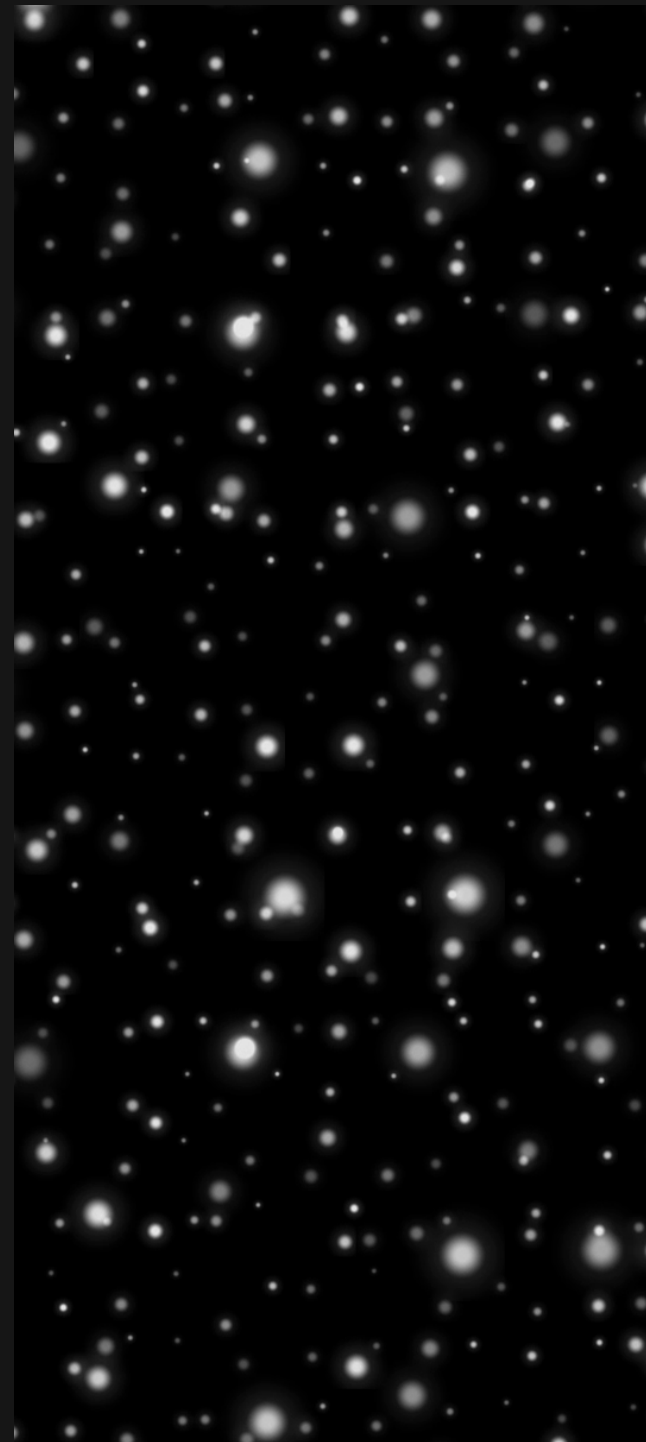
Сделаем цветокоррекцию



```
half4 c = image.eval(coord);  
float3 hsv = rgb2hsv(c.rgb / max(c.a, 0.001));  
hsv.x = fract(hsv.x + 0.55);  
hsv.y *= 0.2;  
hsv.z *= 0.65;  
return half4(hsv2rgb(hsv) * c.a, c.a);
```



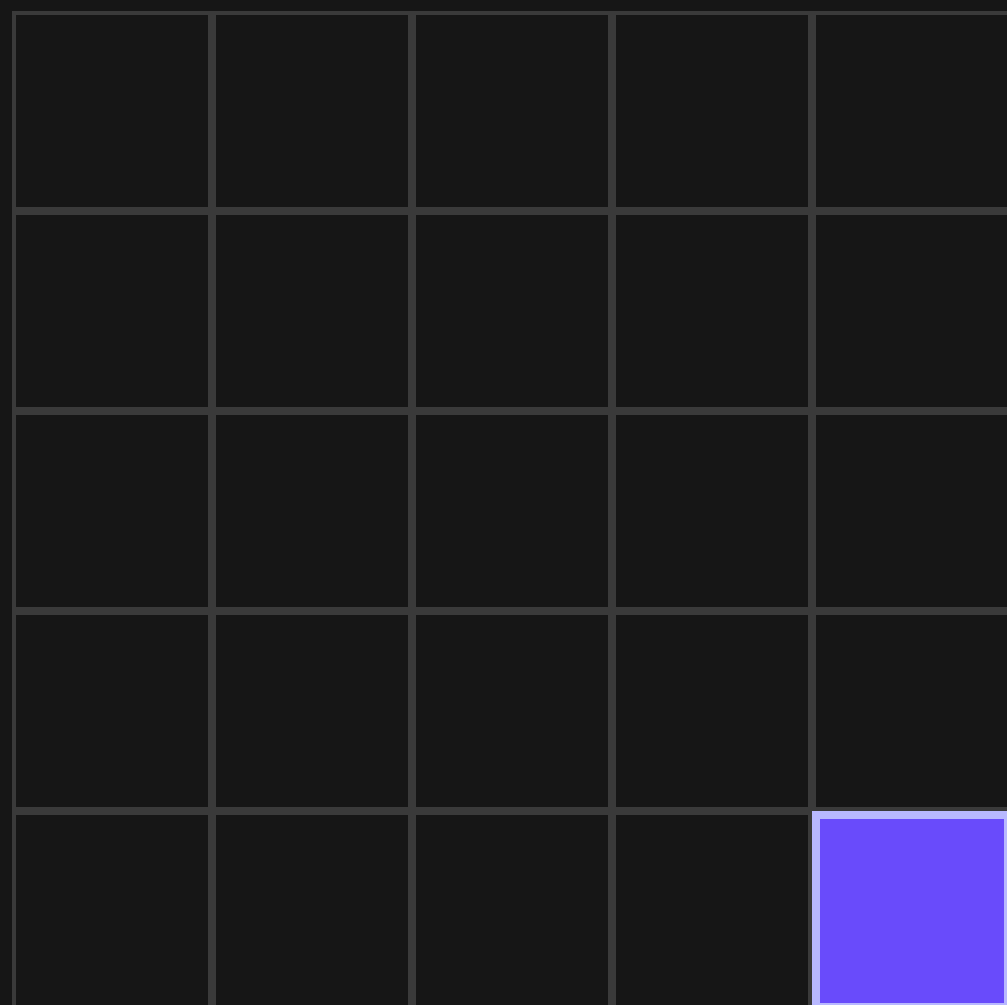
Частицы



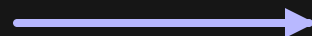
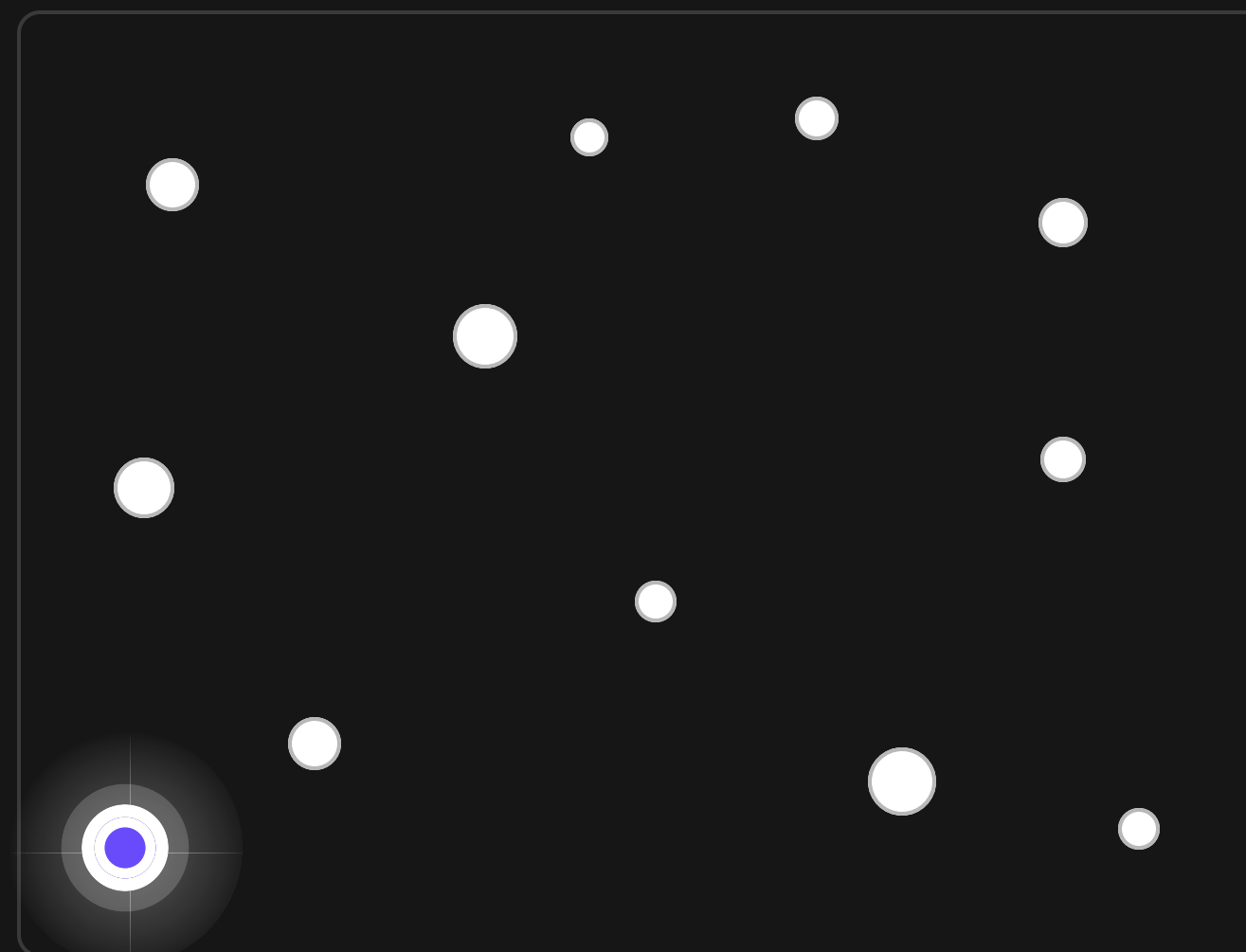
Частица



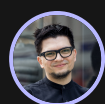
Экран: m пикселей



Список: n частиц



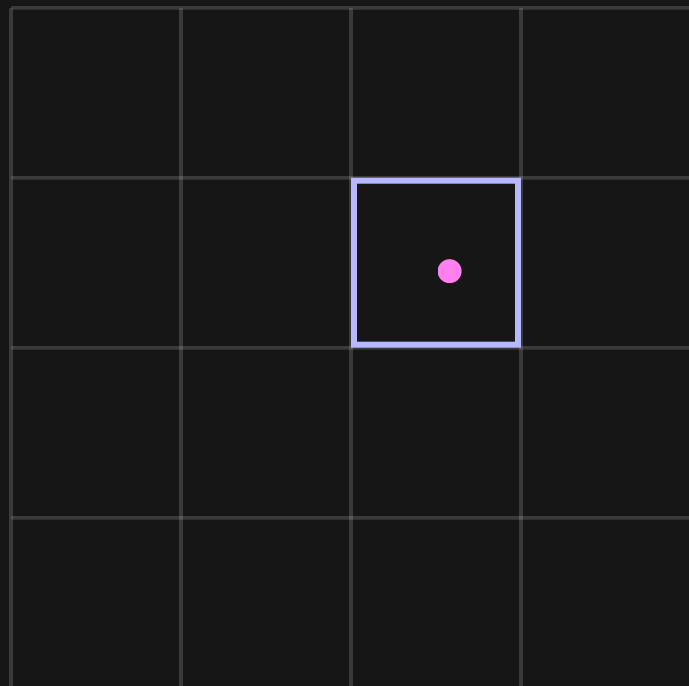
$O(m \times n)$



Частицы

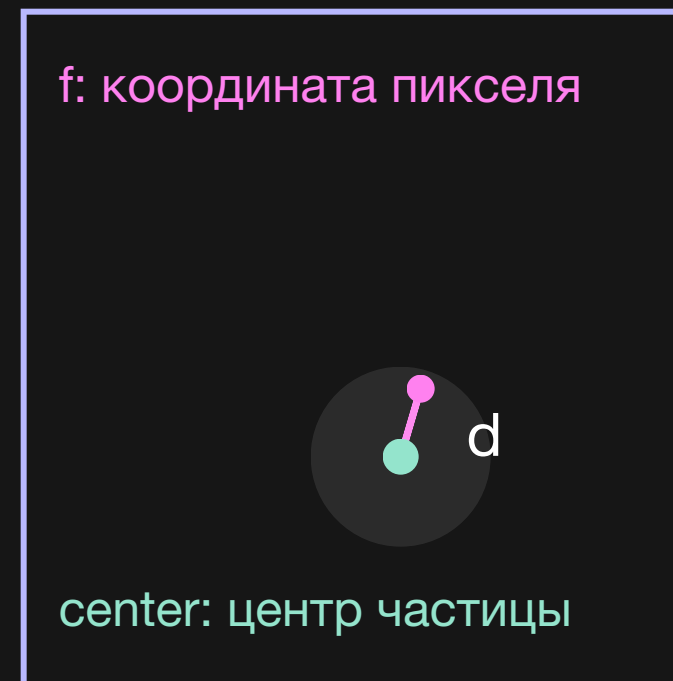


1. Найти ячейку



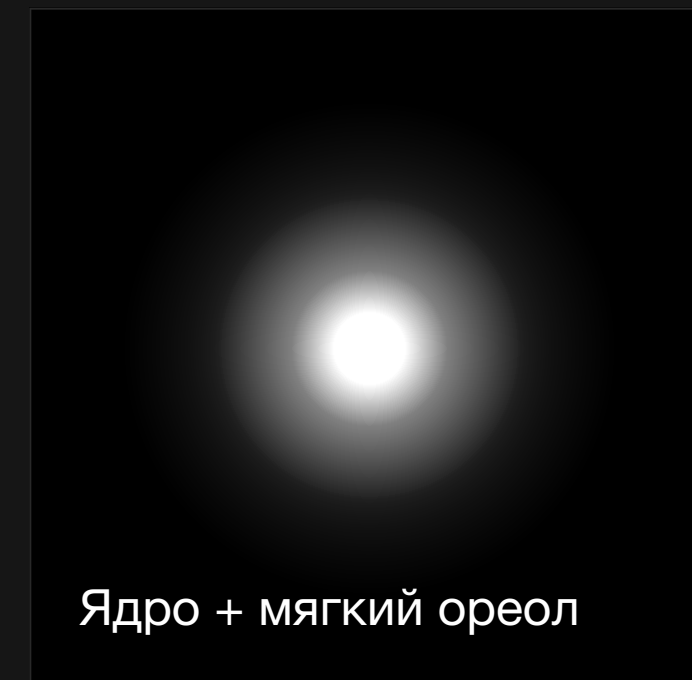
$q = (2.58, 1.55)$
cell = floor(q) = (2, 1)

2. Найти расстояние

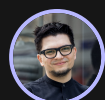


$f = q - \text{cell} = (0.58, 0.55)$
 $\text{center} = (0.55, 0.65)$
 $d = \text{length}(f - \text{center}) \approx 0.10$

3. Получить яркость

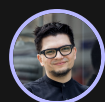
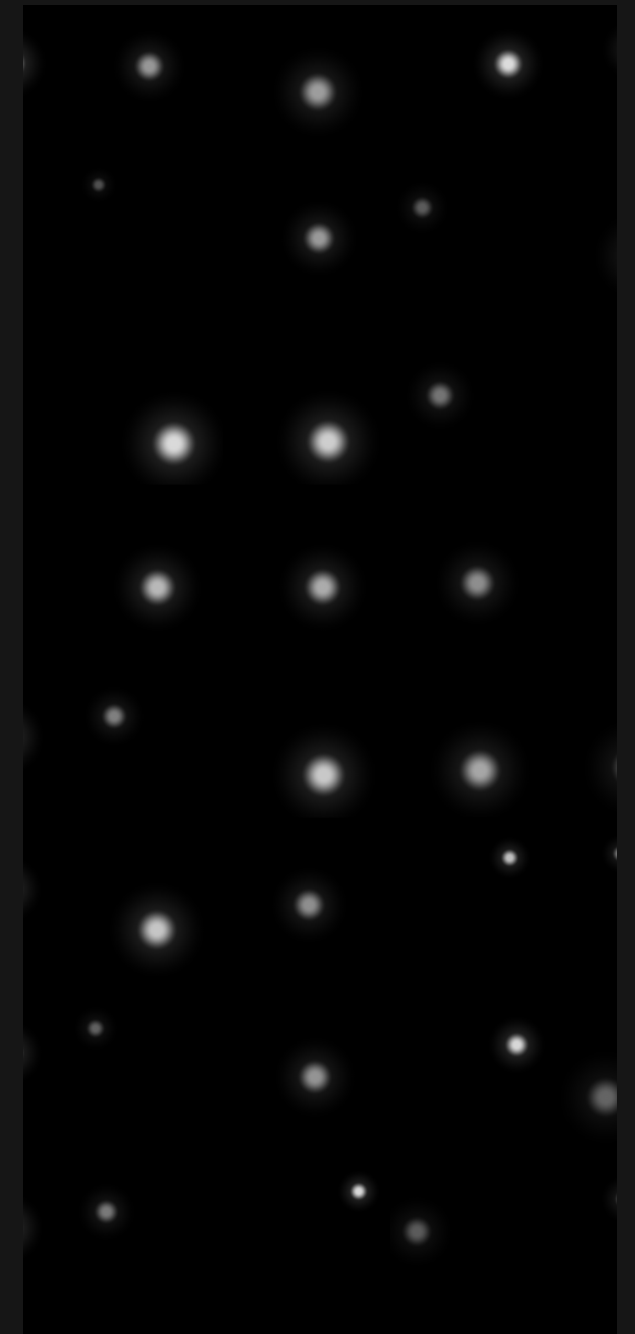


Маленькое d : светлый пиксель.
Далеко от центра: прозрачный.





```
float2 q = p * scale - velocity * t;  
float2 cell = floor(q);  
float2 f = q - cell;  
  
float2 rnd = hash2(cell);  
float2 center = float2(0.2) + 0.6 * rnd;  
  
float radius = 0.05 + 0.1 * rnd.y;  
  
float d = length(f - center);  
  
float core = smoothstep(radius, radius * 0.2, d);  
  
float halo = smoothstep(radius * 2.4, 0.0, d) * 0.25;  
return (core + halo) * (0.35 + 0.65 * rnd.x);
```



Частицы: оживляем



iOS

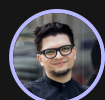
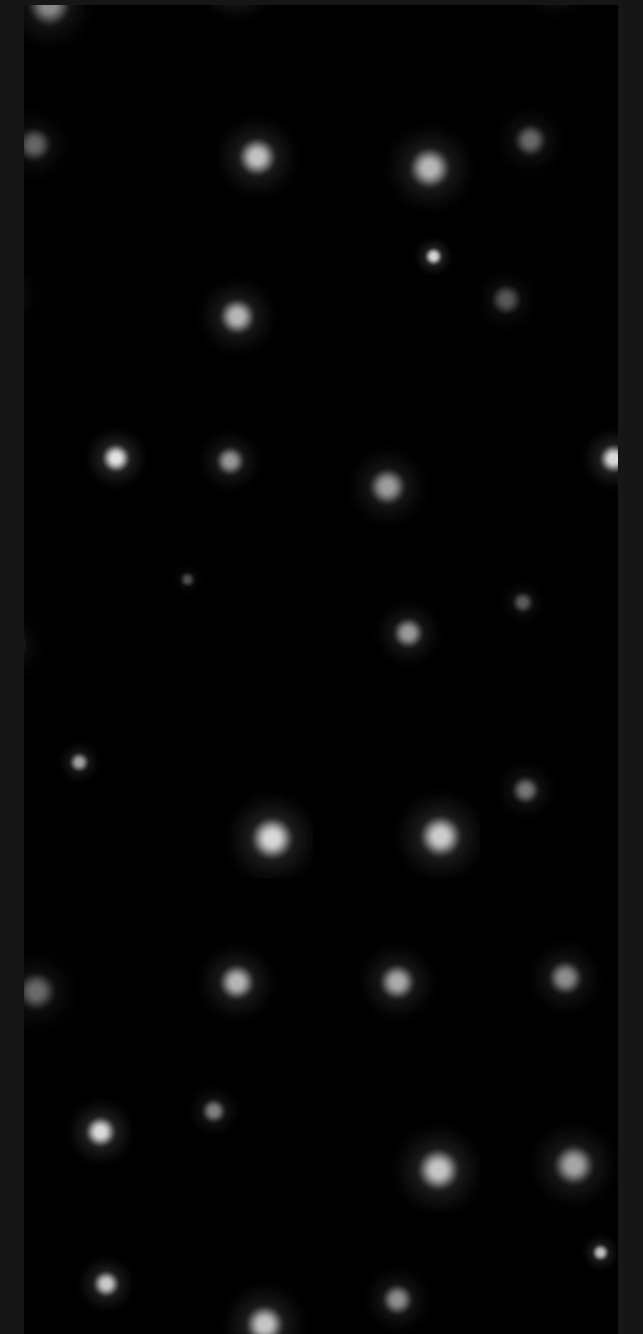
```
// draw(in:) вызывается для нового кадра
let now = CACurrentMediaTime()
let seconds = Float(
    now.truncatingRemainder(
        dividingBy: 3600
    )
)

var snow = SnowUniforms(
    size: vines.screenSize,
    time: seconds
)
encoder.setFragmentBytes(
    &snow,
    length: MemoryLayout<SnowUniforms>.stride,
    index: 0
)
```

Android

```
// Цикл кадров обновляет Compose state
val start = withFrameNanos { it }
while (true) {
    val now = withFrameNanos { it }
    time = 4f + (now - start) / 1e9f
}

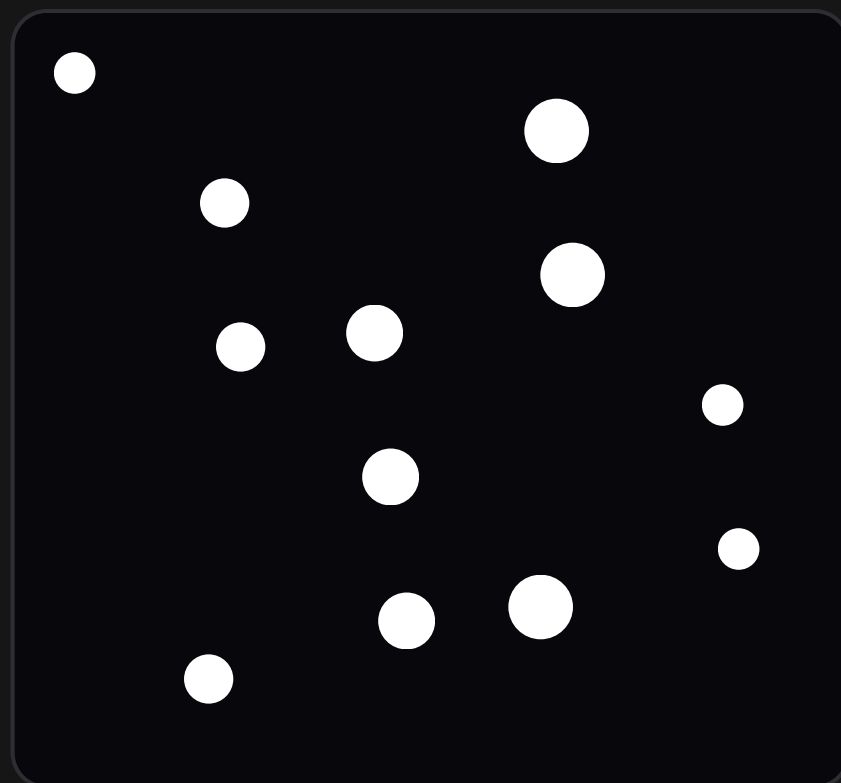
// Canvas читает time при отрисовке
shader.setFloatUniform("time", time)
drawRect(brush)
```



Три слоя создают глубину

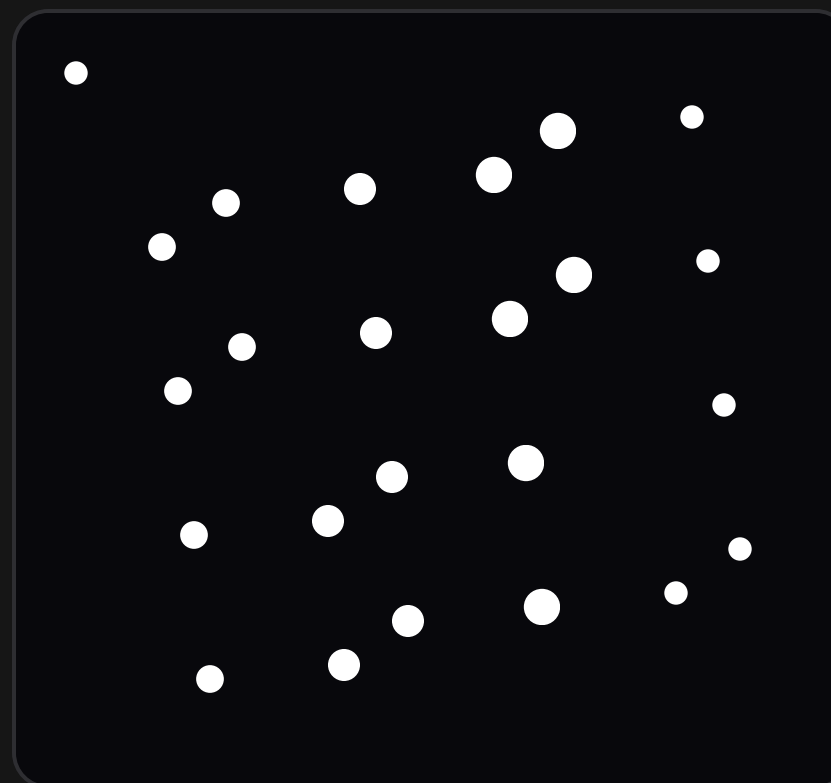


Ближний



Крупнее · быстро падают
8 ячеек по высоте

Средний



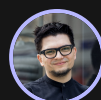
Мельче · медленно падают
14 ячеек по высоте

Дальний



Самые мелкие · сносит влево
22 ячейки по высоте

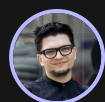
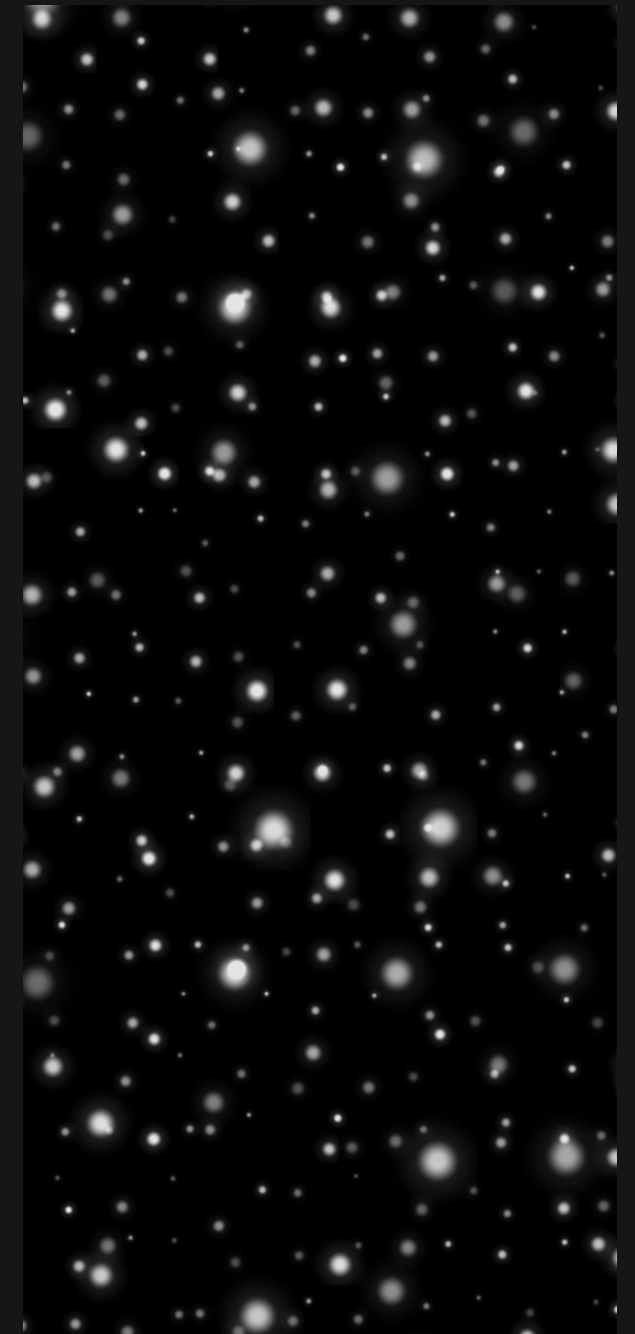
Глубину задаёт связь размера и движения, а не случайный размер.



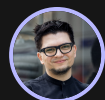
Три слоя создают глубину



```
float2 nearSpeed = float2(0.05, 0.22);  
float near = snowLayer(p, time, 8.0, nearSpeed);  
  
float2 midSpeed = float2(0.03, 0.10);  
float mid = snowLayer(p, time + 40.0, 14.0, midSpeed);  
  
float2 farSpeed = float2(-0.04, 0.16);  
float far = snowLayer(p, time + 80.0, 22.0, farSpeed);  
  
float flakes = 0.8 * near + 0.9 * mid + 0.7 * far;  
half a = half(clamp(flakes, 0.0, 1.0) * 0.9);  
return half4(a, a, a, a);
```



Glitch-эффeкт

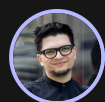


Glitch: сдвиг полос

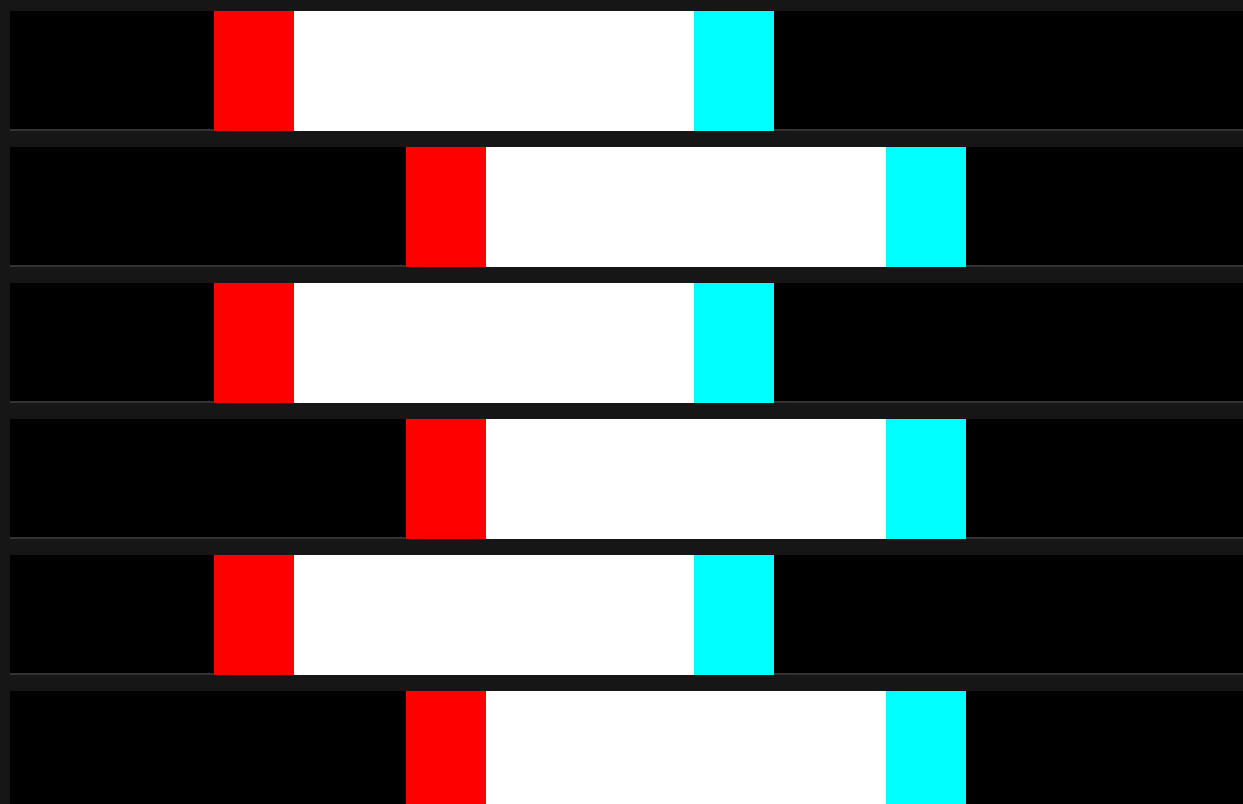


Сдвиг полос

```
float stripeH = imageSize.y / 12.0;  
float stripe = floor(coord.y / stripeH);  
  
float direction = mod(stripe, 2.0) * 2.0 - 1.0;  
float shift = direction * imageSize.x * 0.07;  
float2 displaced = coord + float2(shift, 0.0);  
  
// . . .  
return image.eval(displaced);
```

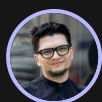


Glitch: откуда цветные края



■ R: красный

■ G+B: бирюзовый

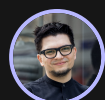
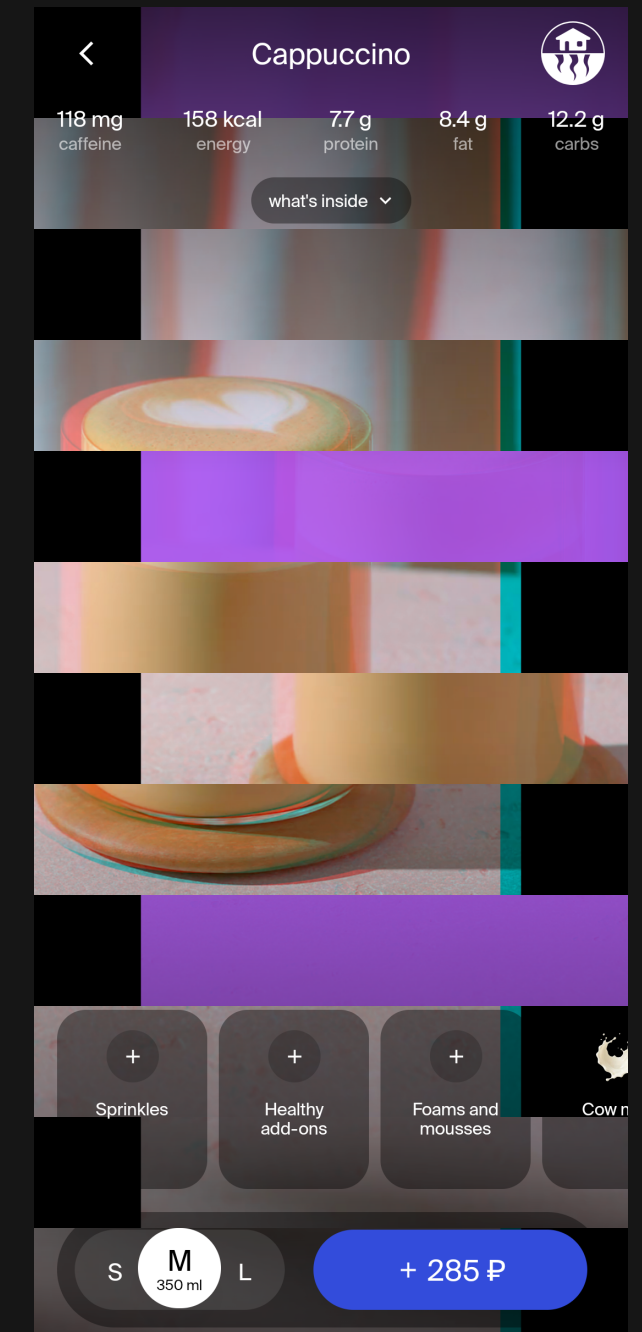


Glitch: каналы и цветные полосы

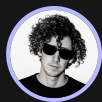
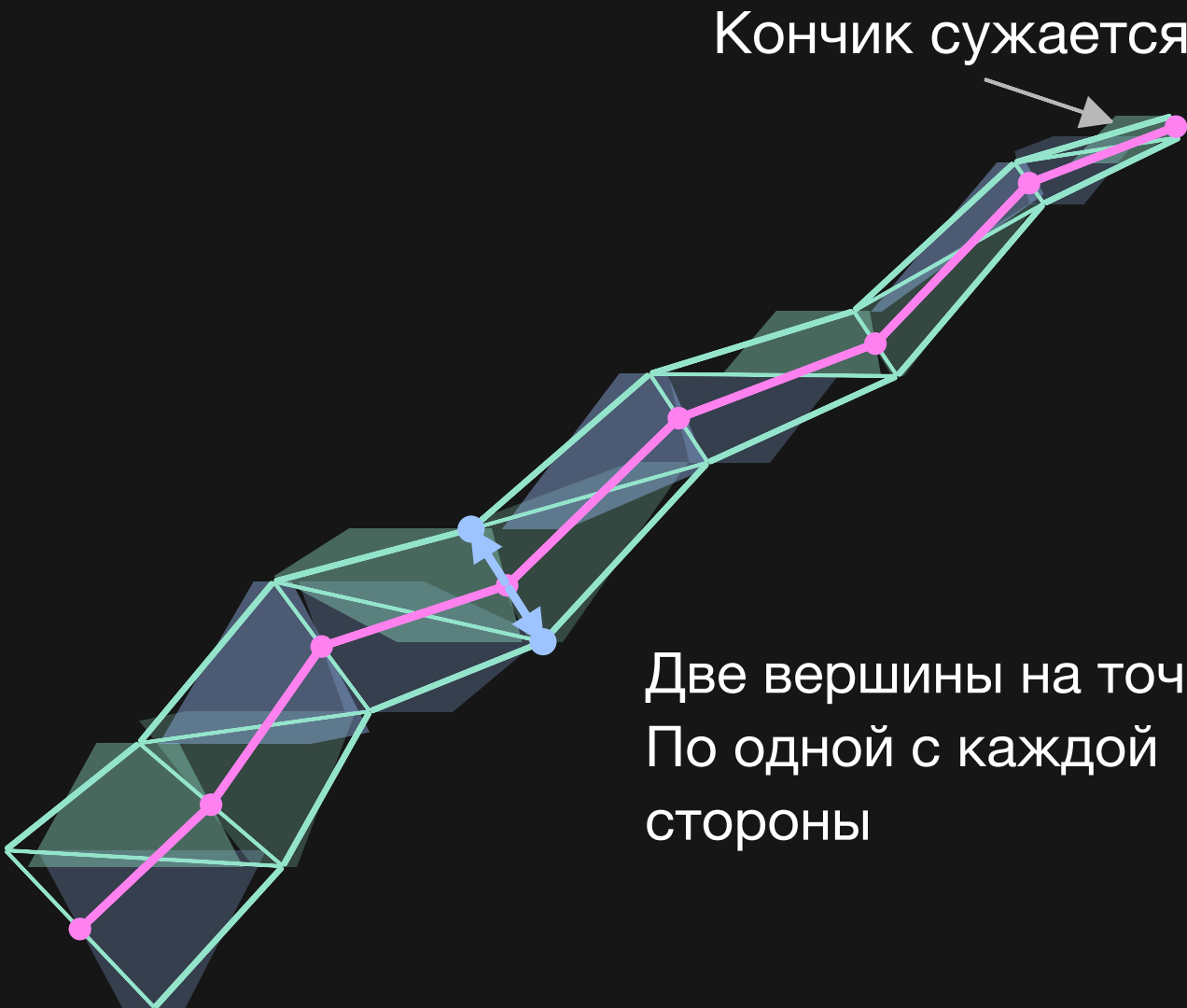


Разделение каналов Цветные полосы

```
float2 displaced = coord + float2(shift, 0.0);
half4 c = image.eval(displaced);
float2 delta = float2(imageSize.x * 0.035, 0.0);
half4 offset = image.eval(displaced + delta);
float red = offset.r / max(offset.a, 0.001) * c.a;
half4 split = half4(red, c.g, c.b, c.a);
float band = 1.0 - step(1.0, mod(stripe, 4.0));
half3 solid = half3(0.65, 0.30, 0.95) * c.a;
return half4(mix(split.rgb, solid, band * 0.85), c.a);
```



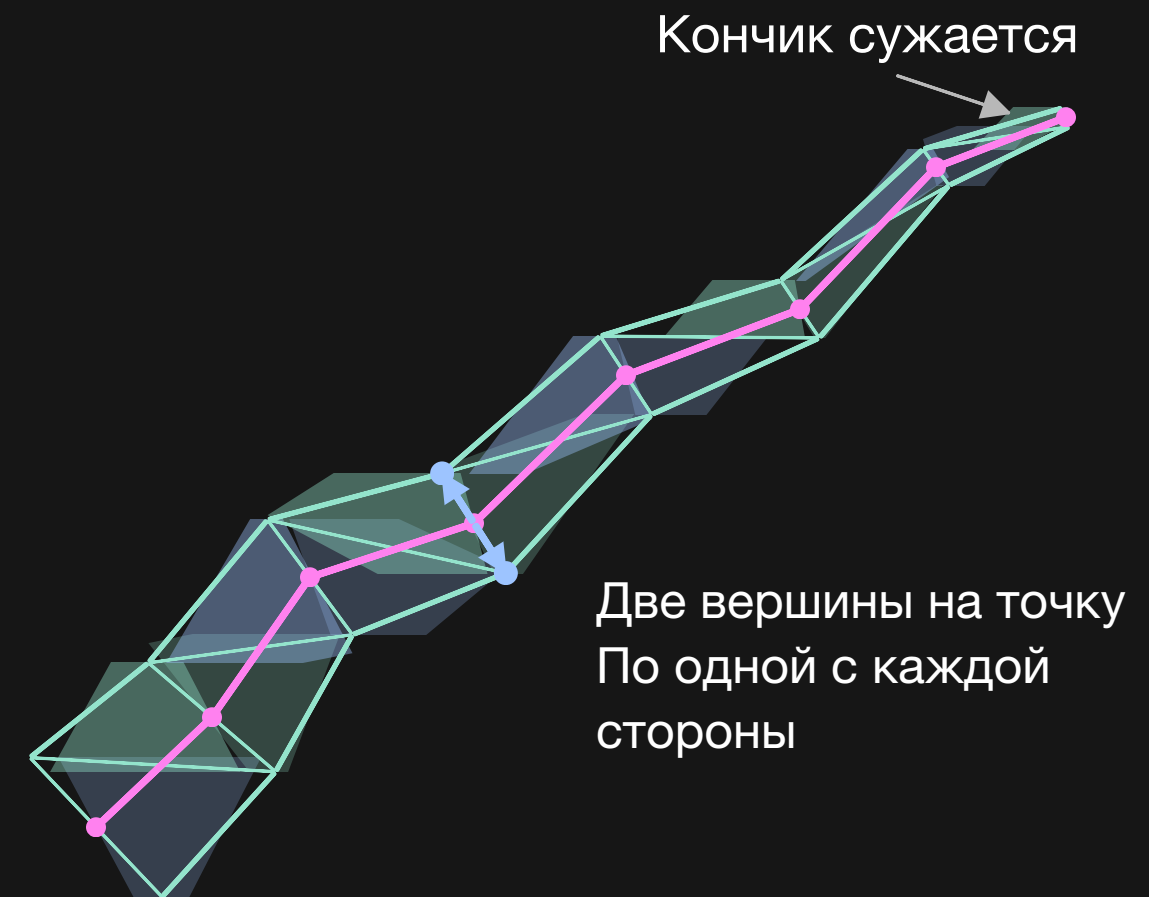
Рисуем лианы



Три шага построения лианы



```
float targetLen = maxLen * progress;  
  
float steps = targetLen / 4.0;  
float grown = steps > 0.0 ? ceil(steps) : 0.0;  
float validPoints = min(pointCount, 1.0 + grown);  
  
float t = i / max(1.0, validPoints - 1.0);  
  
halfWidth = mix(mainWidth, 6.0, t) * 0.5;  
  
float dLen = length(d);  
float2 normal = float2(0.0, 1.0);  
if (dLen > 0.001) normal = float2(-d.y, d.x) / dLen;  
  
varyings.position = p + normal * halfWidth * side;
```



1. Длина · 2. Ширина · 3. Вершины



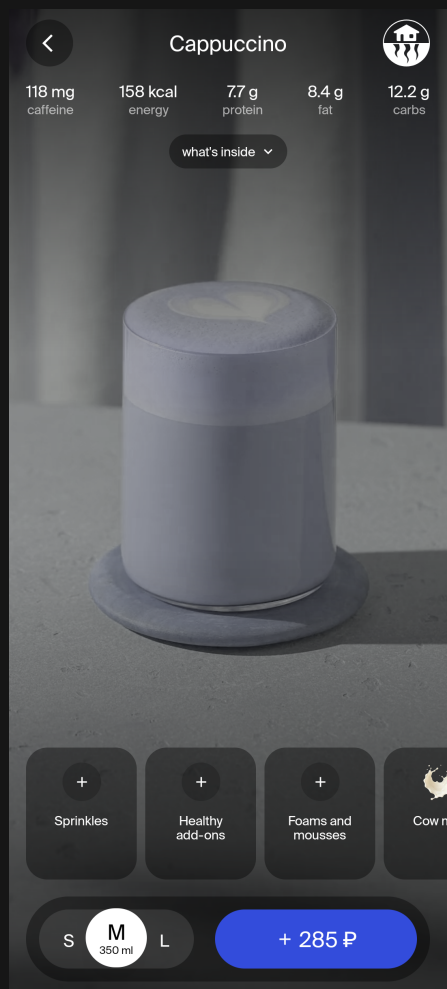
Четыре части. Один эффект.



Glitch

✓ Готово

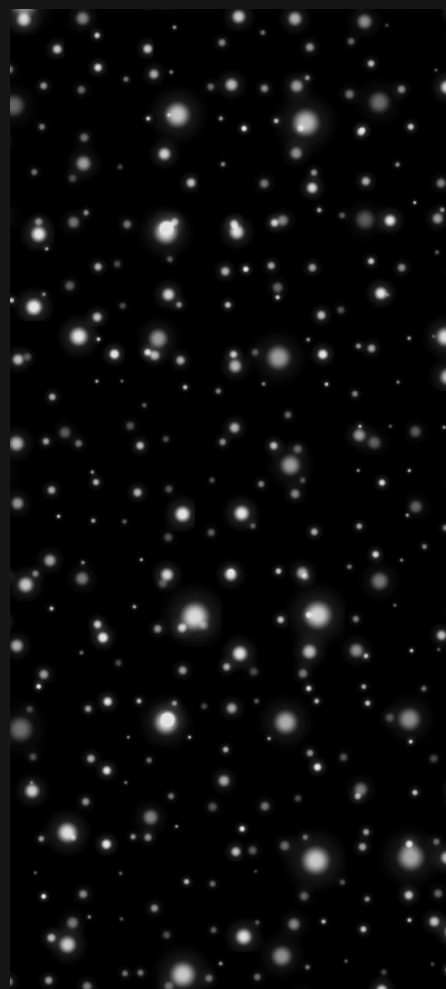
+



Цветокоррекция

✓ Готово

+



Частицы

✓ Готово

+



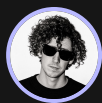
Лианы

✓ Готово

→

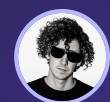


Изнанка





Спасибо!
До встречи.



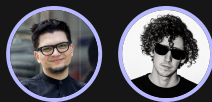
Постойте. Мы добавили видео.



IOS



ANDROID



План



1. Простой шейдер

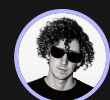
2. Шейдер Изнанки

3. Видео не работает

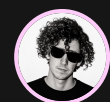
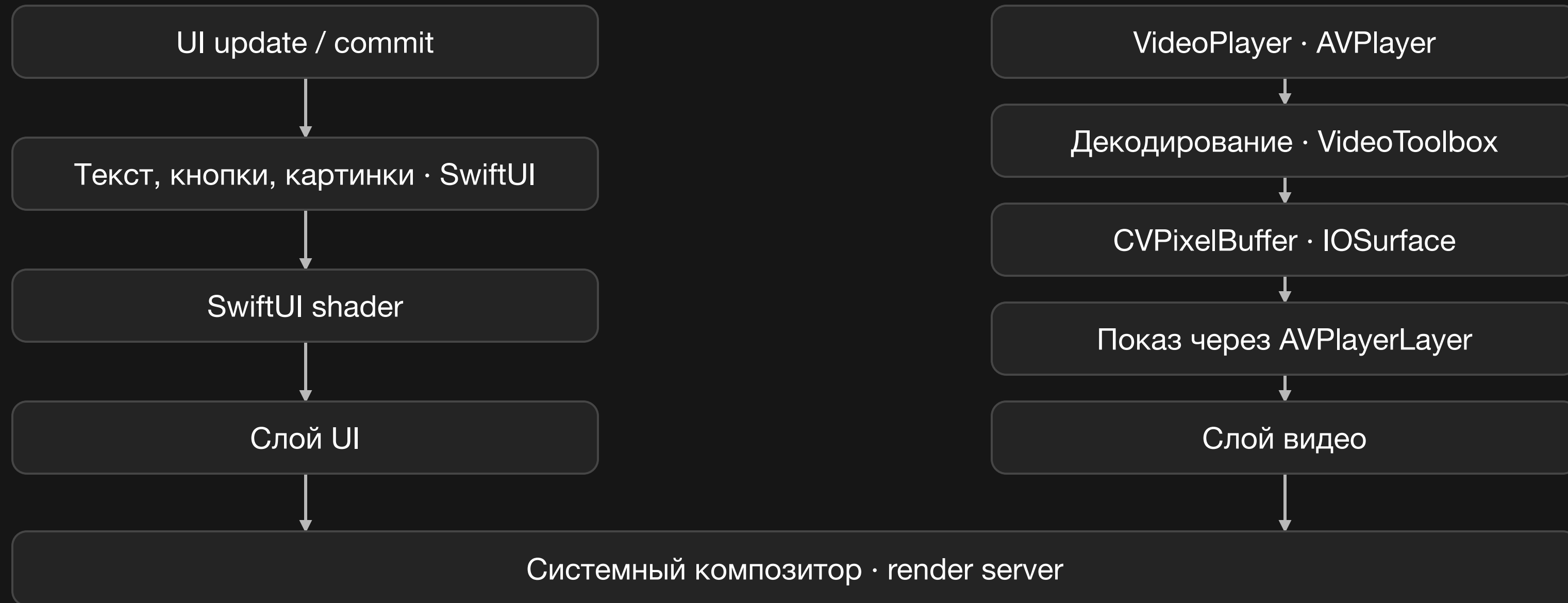
4. Шейдер для видео

5. Performance

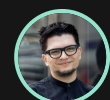
6. Выводы



iOS: видео проходит отдельно



Android: видео проходит отдельно





1. Простой шейдер

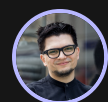
2. Шейдер Изнанки

3. Видео не работает

4. Шейдер для видео

5. Performance

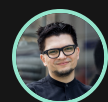
6. Выводы



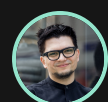
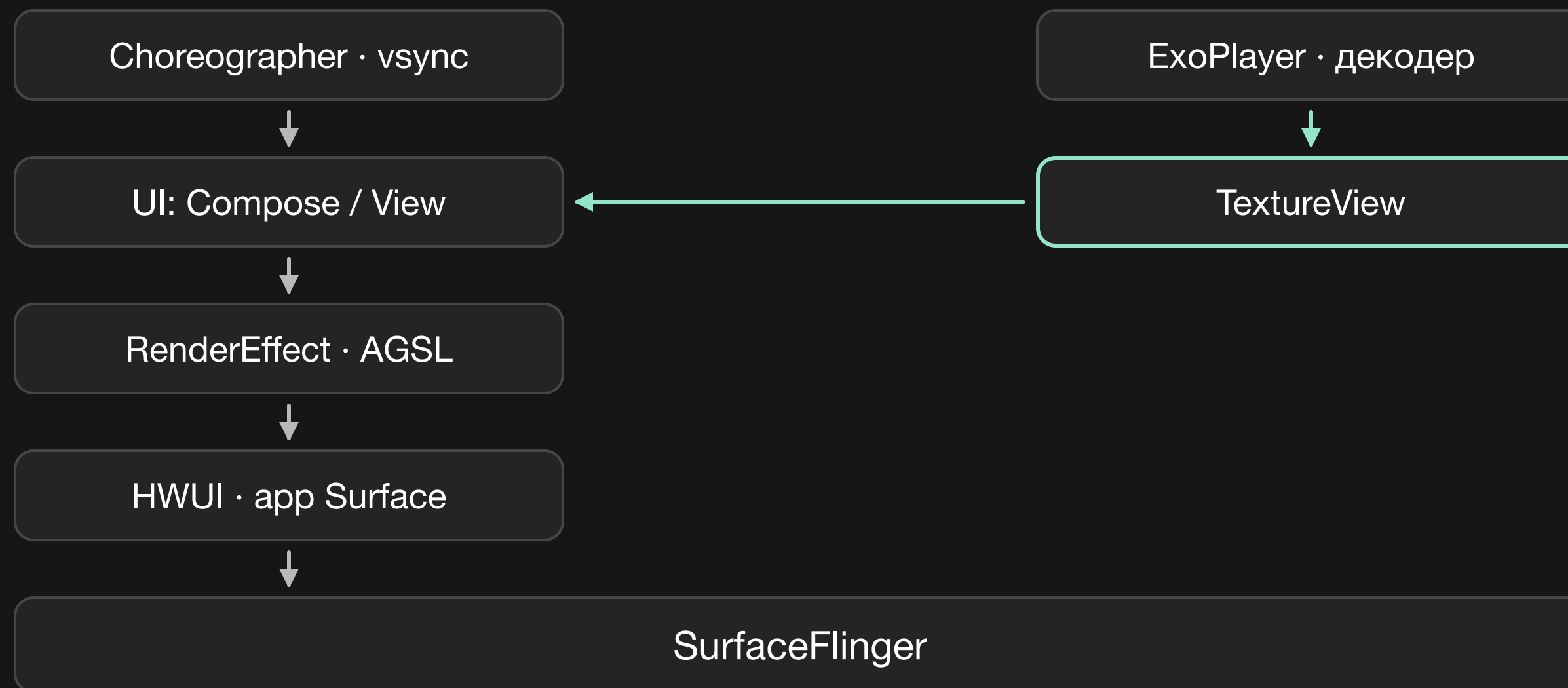
Android: три пути



- 01 TextureView + AGSL
- 02 Media3: GLError
- 03 Свой GL: UI + видео



TextureView + AGSL

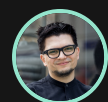


TextureView + AGSL

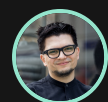
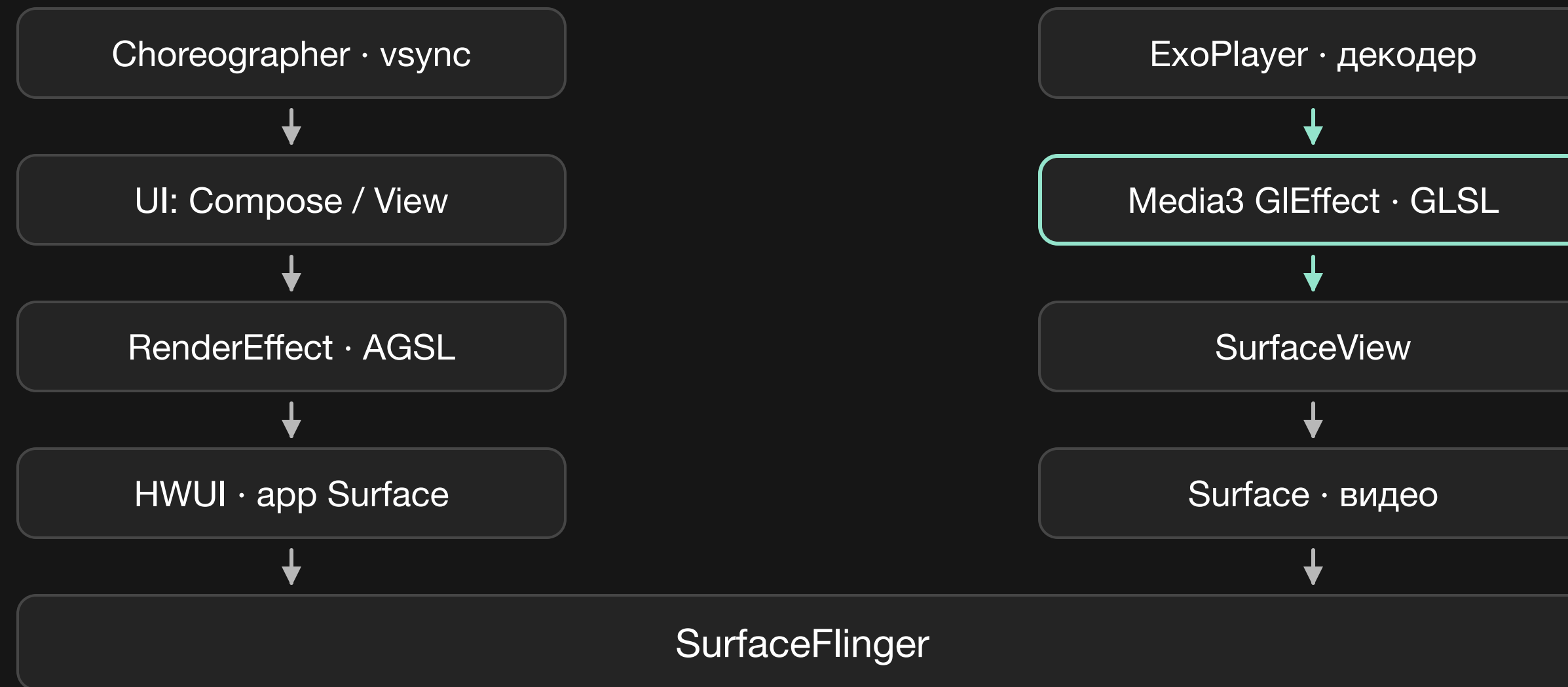


```
AndroidView(  
    factory = { TextureView(it) },  
    update = { player.setVideoTextureView(it) }  
)
```

```
// Родительский контейнер:  
Modifier.upsideDownAgs1(effect)
```



Media3: GLEffect



Media3: GLEffect

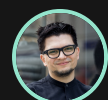


```
val videoEffect = remember(effect) {
    UpsideDownGLEffect(effect)
}

val player = rememberDemoPlayer(
    effects = persistentListOf(videoEffect)
)

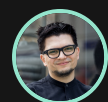
// Внутри rememberDemoPlayer:
setVideoEffects(effects)
prepare()

SurfaceViewVideo(
    player = player,
    modifier = Modifier.matchParentSize()
)
```

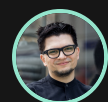
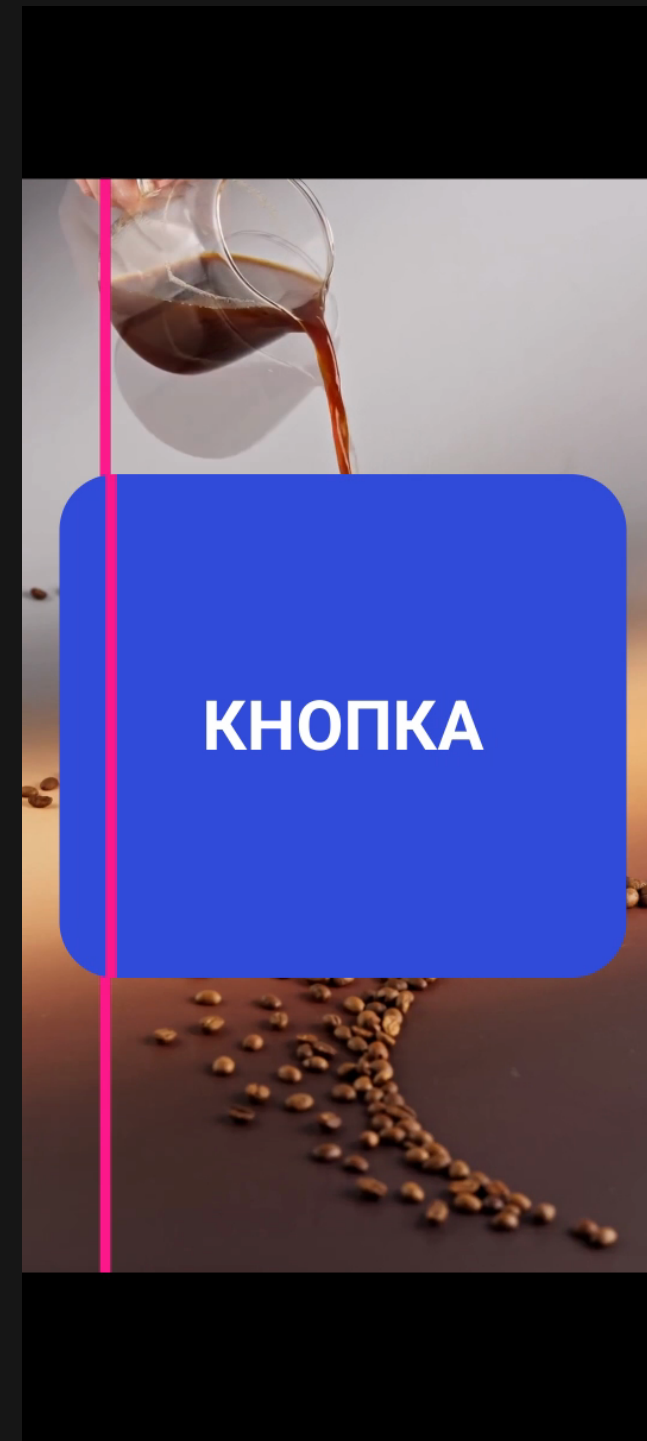




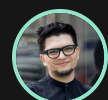
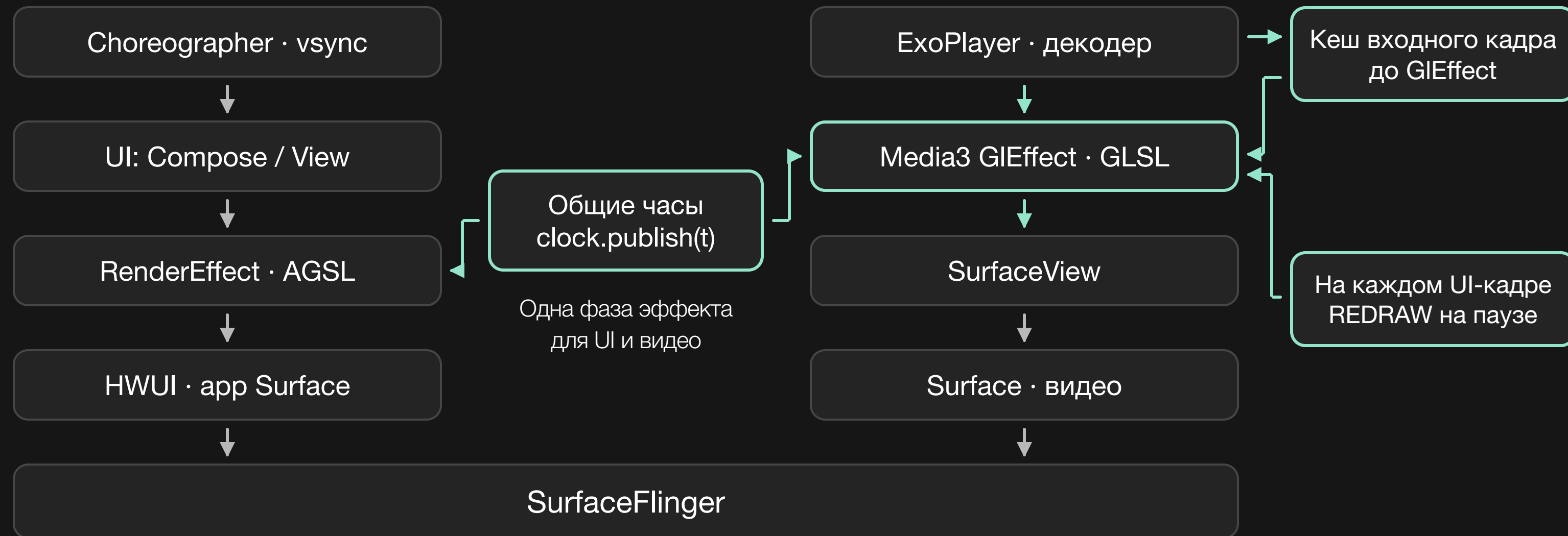
```
val elapsedMillis =  
    (frameNanos - startNanos) / NanosPerMilli  
  
state.publish(snapshotAt(elapsedMillis))  
  
// В publish(value): параметры для UI  
snapshot = value  
  
// В том же publish: параметры для GL  
glSnapshot = value
```



Пауза: это проблема



Синхронизация RenderEffect и Media3 GLEffect



Media3: кеш кадра и явный REDRAW



```
// При создании video graph: запомнить вход
.setEnableReplayableCache(true)

// Каждый UI-кадр, даже во время паузы
val now = withFrameNanos { it }
val elapsedMs = (now - start) / NanosPerMilli
clock.publish(elapsedMs)

// На паузе: обработать прежний кадр ещё раз
if (paused && demo.redrawOnPause) {
    player.setVideoEffects(
        VideoFrameProcessor.REDRAW
    )
}
```

Подготовка

Кеш сохраняет декодированный кадр до эффекта.

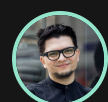
Общие часы

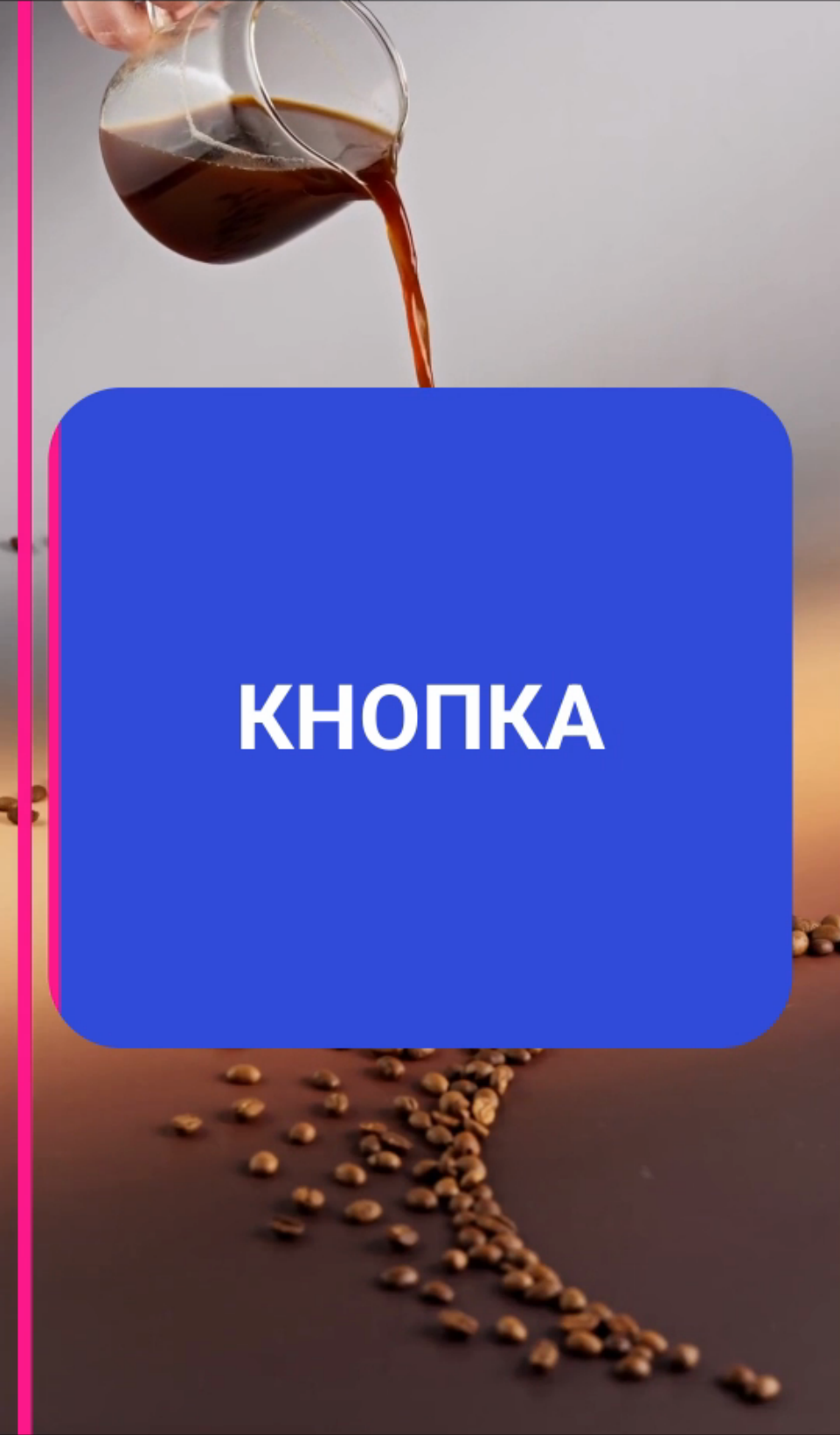
UI и GLEffect читают опубликованное время.

Повторное рисование

REDRAW берёт кадр из кеша. GLEffect рисует его с новым временем.

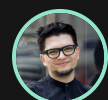
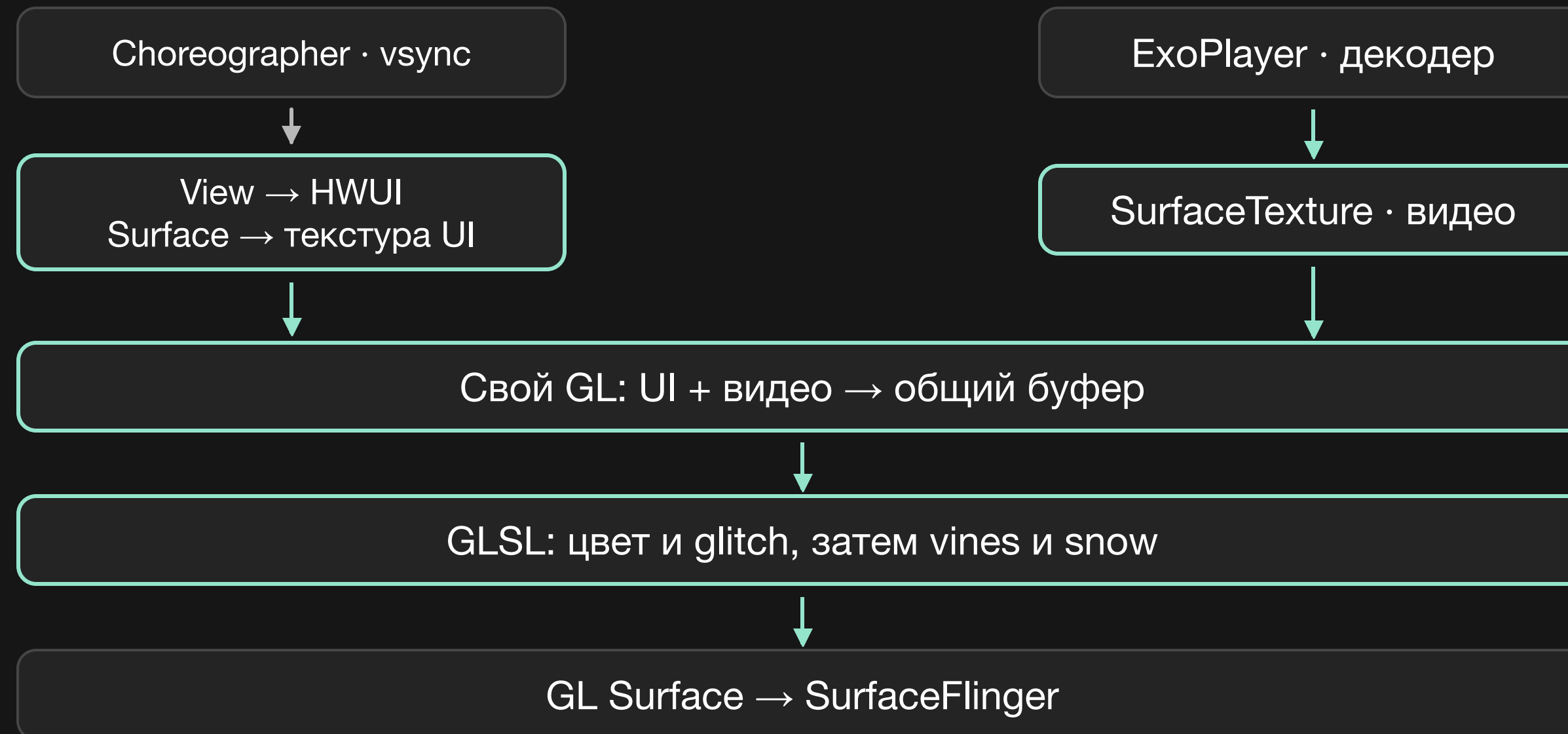
Media3 1.11.0 · RestrictedApi



A background image showing a hand pouring dark coffee from a glass carafe into a cup. Below the cup, a trail of coffee beans is scattered on a dark surface. A bright pink vertical line is on the left side of the image.

КНОПКА

Свой GL: собираем UI и видео



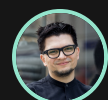
Свой GL: два входа в текстуры



```
// attachVideo: вход декодера
val texture = stream.surfaceTexture
val surface = Surface(texture)
if (videoSurfaces.replace(surface))
    player.setVideoSurface(surface)
```

```
// attachUi: отдельный вход UI
val texture = stream.surfaceTexture
val surface = Surface(texture)
uiHost.setTarget(surface, stream)
```

```
// UiTextureHost: HardwareRenderer
renderer.setSurface(target)
```



Свой GL: эффект после композиции

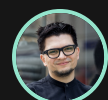


```
val scene = snapshots()
val effect = scene.effect

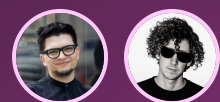
GLS20.glBindFramebuffer(
    GLS20.GL_FRAMEBUFFER, sceneFramebuffer)
drawComposite()

GLS20.glBindFramebuffer(GLS20.GL_FRAMEBUFFER, 0)
drawGrade(effect)

// Наложения: blending включён в полном коде
if (scene.drawsVines) drawVines(scene.vines)
drawSnow(effect.timeSeconds)
```



А что в iOS?

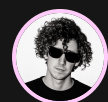


iOS: два пути

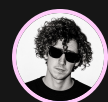
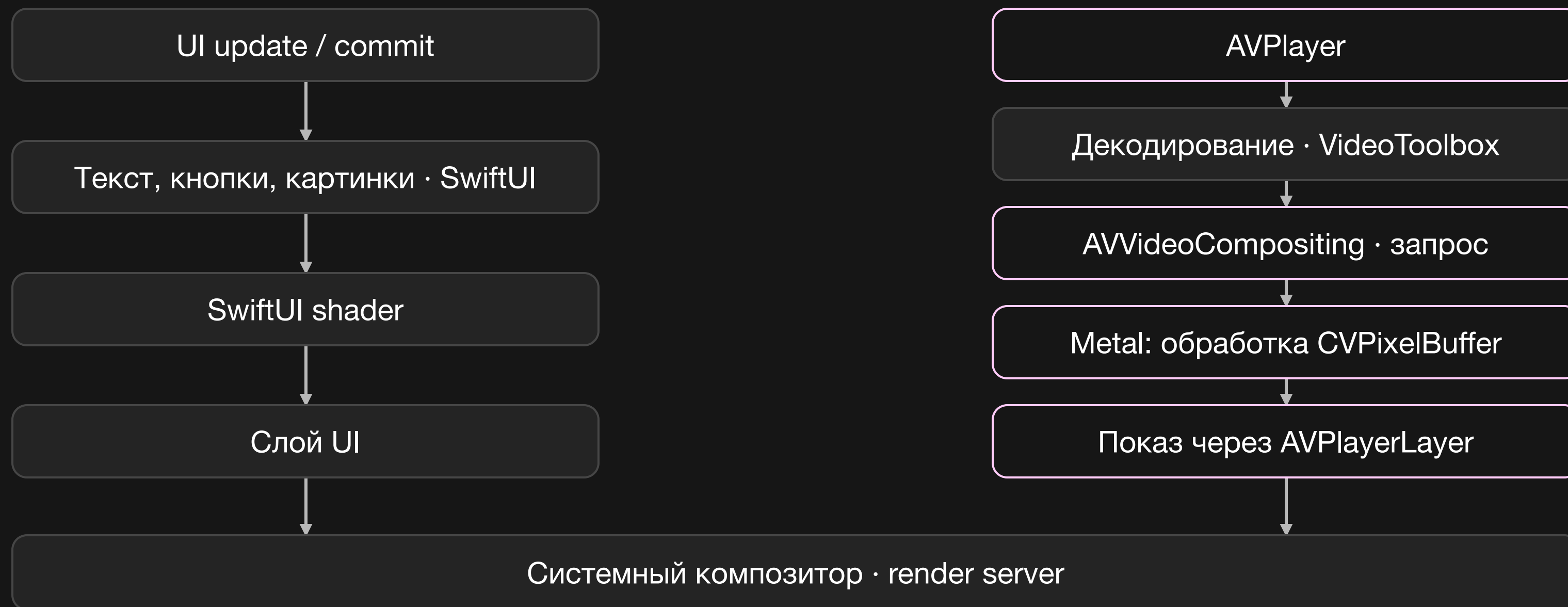


01 AVVideoComposition

02 Кастомный рендеринг



AVVideoComposition: эффект внутри видеоветки



AVVideoComposition: обработка кадра

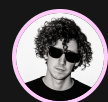


```
guard let trackID = request.sourceTrackIDs.first,
      let source = request.sourceFrame(
        byTrackID: trackID.int32Value),

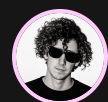
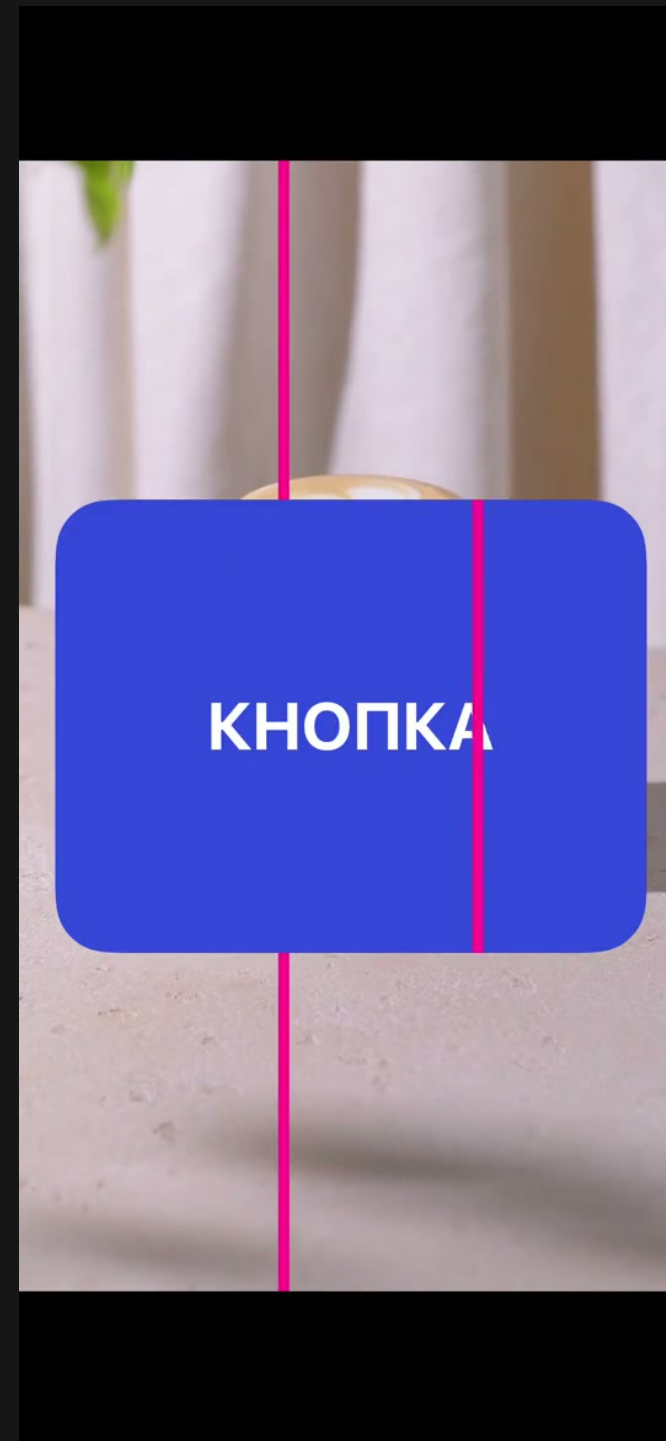
      let destination = request.renderContext.newPixelBuffer()
else {
  request.finish(with: CompositorError.missingFrame)
  return
}

let uniforms = Self.effect.current
  .uniforms(at: CACurrentMediaTime())

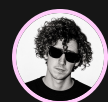
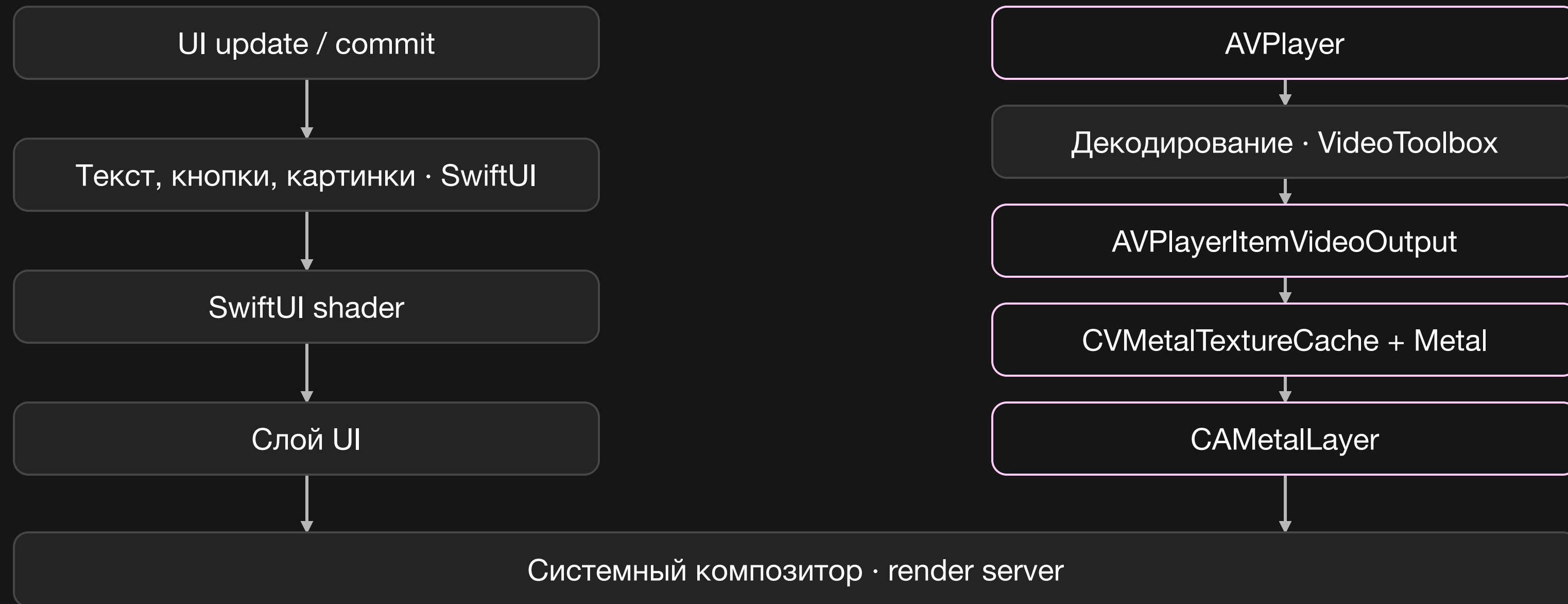
renderer.render(source, into: destination,
               uniforms: uniforms) {
  request.finish(withComposedVideoFrame: destination)
}
```



Пауза: это проблема



Metal: рисуем видеокадр сами



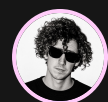
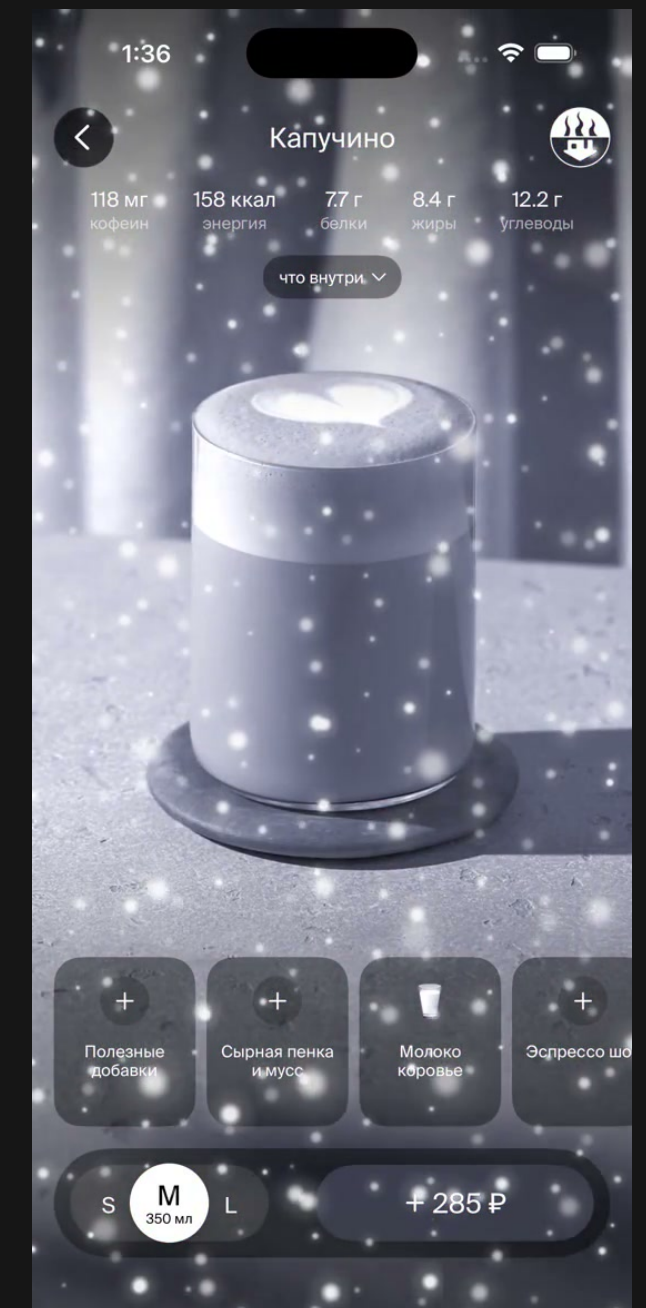
Metal: получаем видеокадр



```
item.add(videoOutput)

let itemTime =
    videoOutput.itemTime(forHostTime: hostTime)
guard videoOutput.hasNewPixelBuffer(
    forItemTime: itemTime) else { return nil }

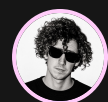
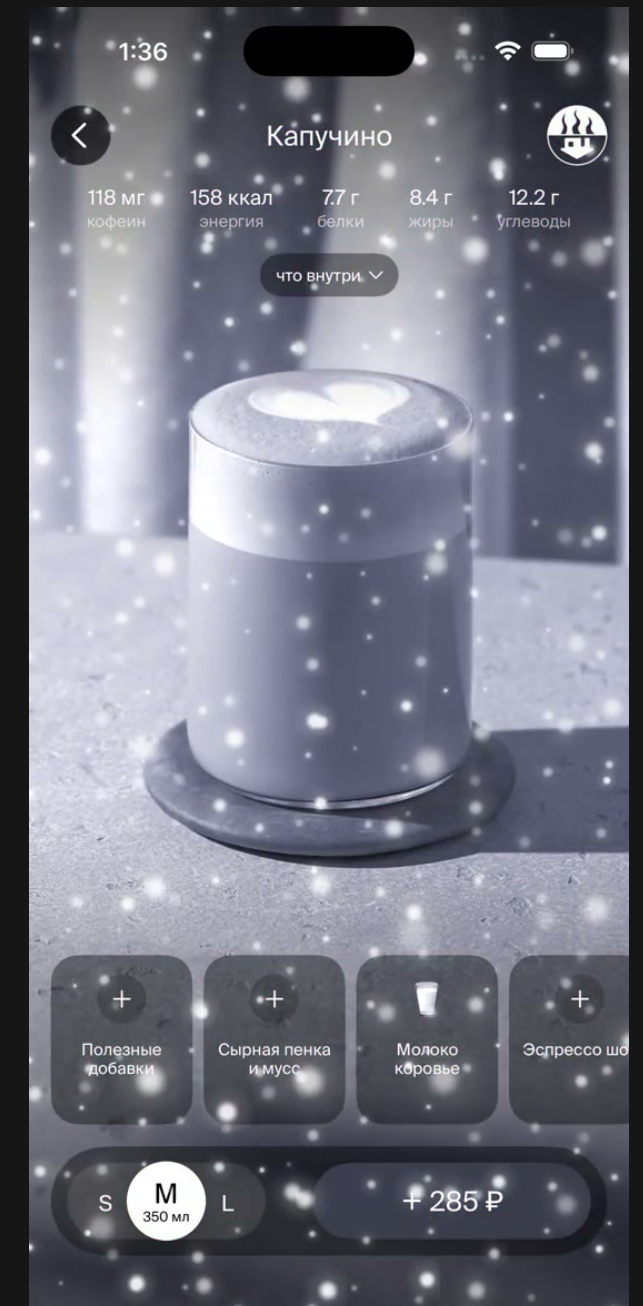
return videoOutput.copyPixelBuffer(
    forItemTime: itemTime,
    itemTimeForDisplay: nil)
```



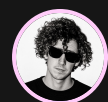
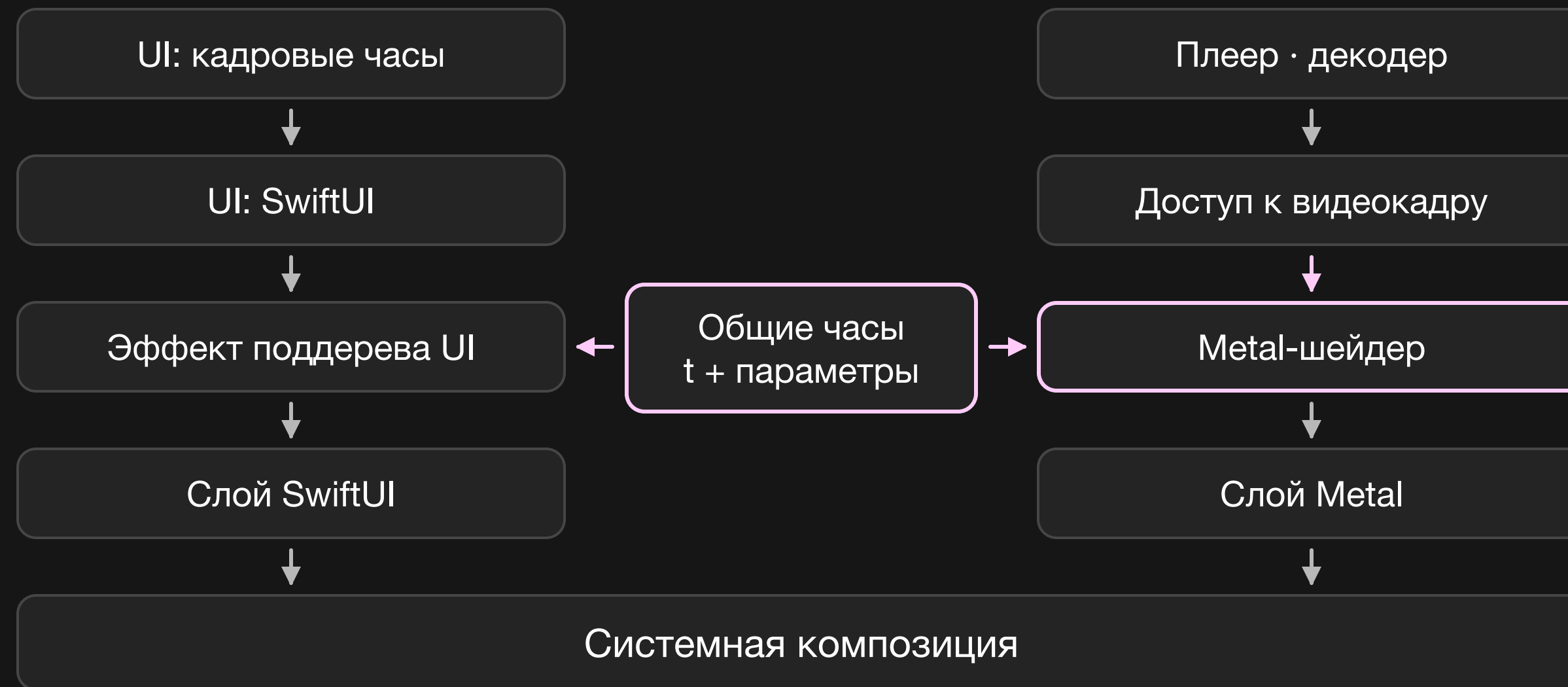
Metal: текстура и рисование



```
CVMetalTextureCacheCreateTextureFromImage(  
    kCFAllocatorDefault,  
    textureCache,  
    pixelBuffer,  
    nil,  
    .bgra8Unorm,  
    CVPixelBufferGetWidth(pixelBuffer),  
    CVPixelBufferGetHeight(pixelBuffer),  
    0,  
    &textureRef  
)  
  
let uniforms =  
    model.uniforms(at: CACurrentMediaTime())  
  
renderer.draw(  
    to: view.metalLayer,  
    uniforms: uniforms,  
    fillsBounds: fillsBounds  
)
```



iOS: один эффект на UI и видео



План



1. Простой шейдер

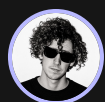
2. Шейдер Изнанки

3. Видео не работает

4. Шейдер для видео

5. Performance

6. Выводы



Плавность, время кадра и цена эффекта



iOS

Hitch time ratio

Время задержек на секунду анимации, мс/с.

UI commit time · p95

Instruments: подготовка и commit UI
внутри приложения, мс.

GPU power impact

Power Profiler: вклад процесса.
Относительная шкала, не мВт.

Android

Janky frames

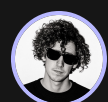
Доля UI-кадров, опоздавших к дедлайну, %.

UI frame time · p95

gfxinfo: длительность UI-кадра
до завершения рендеринга, мс.

GPU rail power

Счётчики питания GPU всего устройства.
Физическая мощность, мВт.

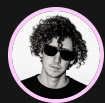


iOS: результаты измерений



iPhone Air · iOS 27.0.1 · Release · короткая серия

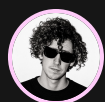
Путь	Hitch time ratio мс/с	UI commit time p95 · мс	GPU power impact
UI без эффектов запусков: 1	3,68	1,17	0,00
UI + видео без эффектов запусков: 2	0,00	1,61 1,46...1,75	0,00
AVVideoComposition запусков: 1	8,29	7,36	3,54
Кастомный рендеринг запусков: 1	4,61	8,11	3,57



iOS: что выбрать



Путь	Когда подходит	Что учитывать
AVVideoComposition	Фильтр следует за воспроизведением видео	Кадры запрашивает плеер. Обновление эффекта на паузе нужно инициировать отдельно.
Кастомный рендеринг	Эффект реагирует на жесты и продолжает двигаться на паузе	Сами храним кадр и запускаем отрисовку. Видео и звук остаются у AVPlayer.



Android: результаты измерений



Pixel 7 · Android 16

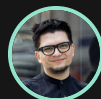
Путь	Janky frames, % иллюстративно	UI frame time p95 мс	GPU rail power мВт
Без видео, без эффекта	0,01	12	81
Без видео + AGSL	0,11	22	229
TextureView + AGSL	0,11	21	230
Media3 GLError	0,15	23	216
Свой GL*	0,03	15	232

*Для «Свой GL» первые два показателя относятся к вспомогательным HWUI-рендерерам, а не ко всему GL-конвейеру. Напрямую с остальными строками не сравниваются.

Android: что выбрать



Путь	Когда подходит	Что учитывать
TextureView + AGSL	Без видео, небольшое видео	Замерять производительность, если видео большое
Media3 + GLError	Эффект только на видео, без UI	Пауза, RestrictedApi
Свой GL	Нет возможности использовать AGSL: Android 13+, Mesh: Android 14+	Не обязательно быстрее. Обычные UI-замеры не показывают полную стоимость GL



План



1. Простой шейдер

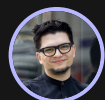
2. Шейдер Изнанки

3. Видео не работает

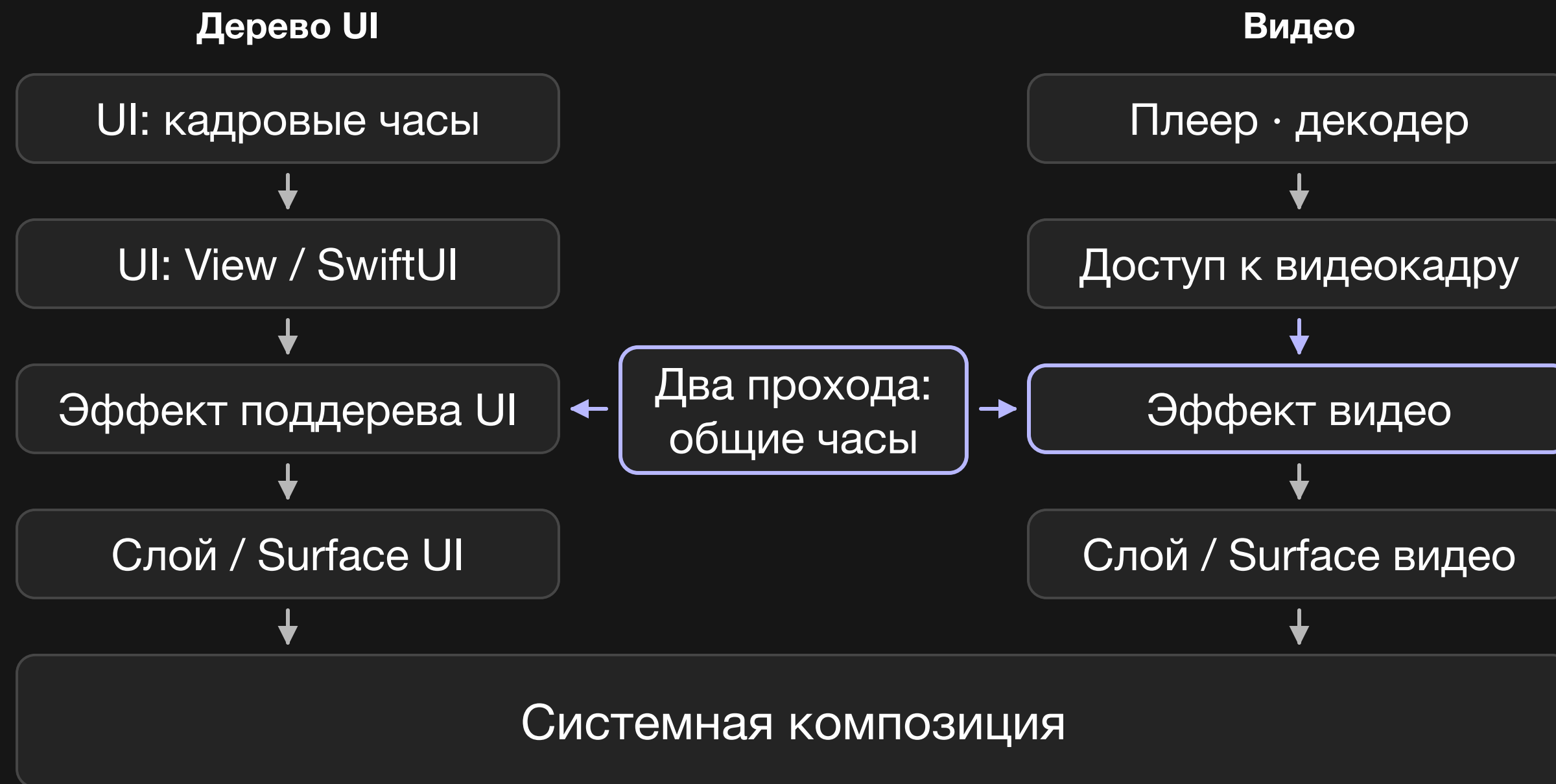
4. Шейдер для видео

5. Performance

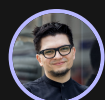
6. Выводы



Помните, как работает UI pipeline



Два пути: объединить входы до эффекта или согласовать отдельные проходы.



Спасибо за внимание!

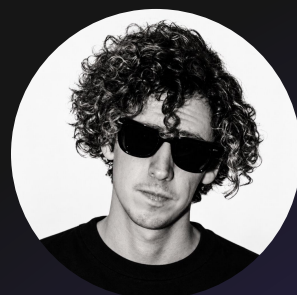
Давайте вопросы



Максим Качинкин

Android Tech Lead

t.me/mobilefiction · Мобильное Чтиво



Роман Могутнов

Principal Product Engineer

Исходный код
Android + iOS



github.com/iammxrn/stranger-shaders

