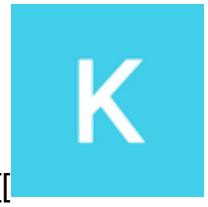
BACKDROP BLUR



[[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

- Installation
- Tutorials
- Liquid Glass
- Blur
- API
- FAQ

Ask

Get started

The **Backdrop** library (GitHub) can draw a copy of the **background (backdrop)** to a **foreground** with various effects.

You can achieve amazing **liquid glass** effect with the library.

Installation

maven-central v2.0.0-alpha03

build.gradle.kts

Copy

```
dependencies {
    implementation("io.github.kyant0:backdrop:<version>")
}
```

Tutorials

Liquid Glass

🦕 **You must read and practise these tutorials before using the library.**

- Glass Bottom Bar
- Interactive Glass Bottom Bar
- Glass Bottom Sheet
- Glass Slider
- Smoother rounded corners

Blur

- Progressive blur

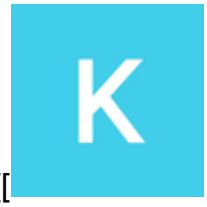
API

- Backdrops
- Backdrop effects

FAQ

NextGlass Bottom Bar

Last updated 3 months ago



Get started | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

- Goals
- What you will learn
- Steps
- Draw backdrop and add lens effect
- Draw the background to the backdrop (optional)
- Add blur effect
- Add surface for readability
- Final code

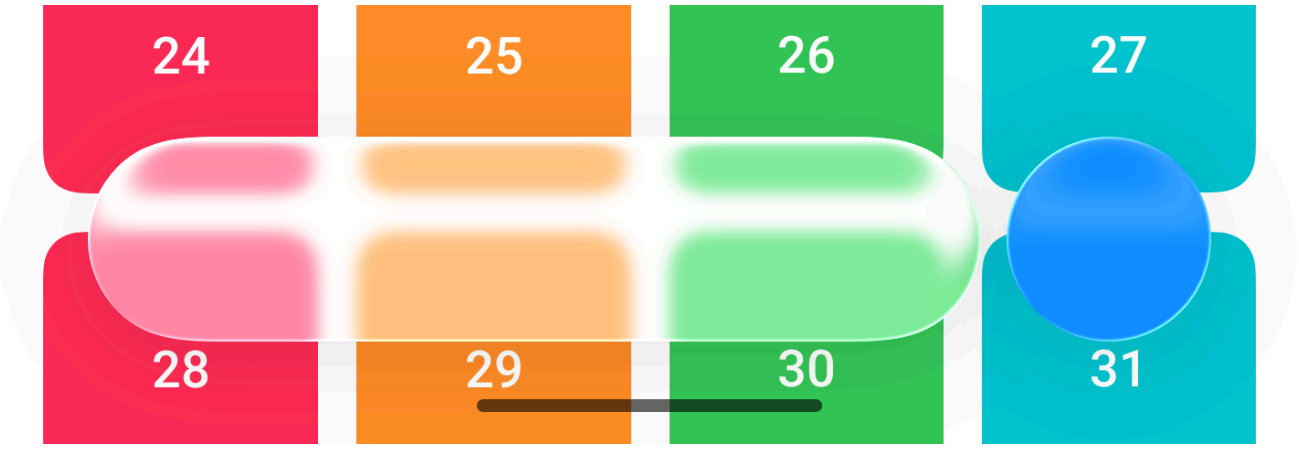
- Exercise: Add a tinted glass icon button

Ask

1. Tutorials

Glass Bottom Bar

Create a glass bottom bar



Goals

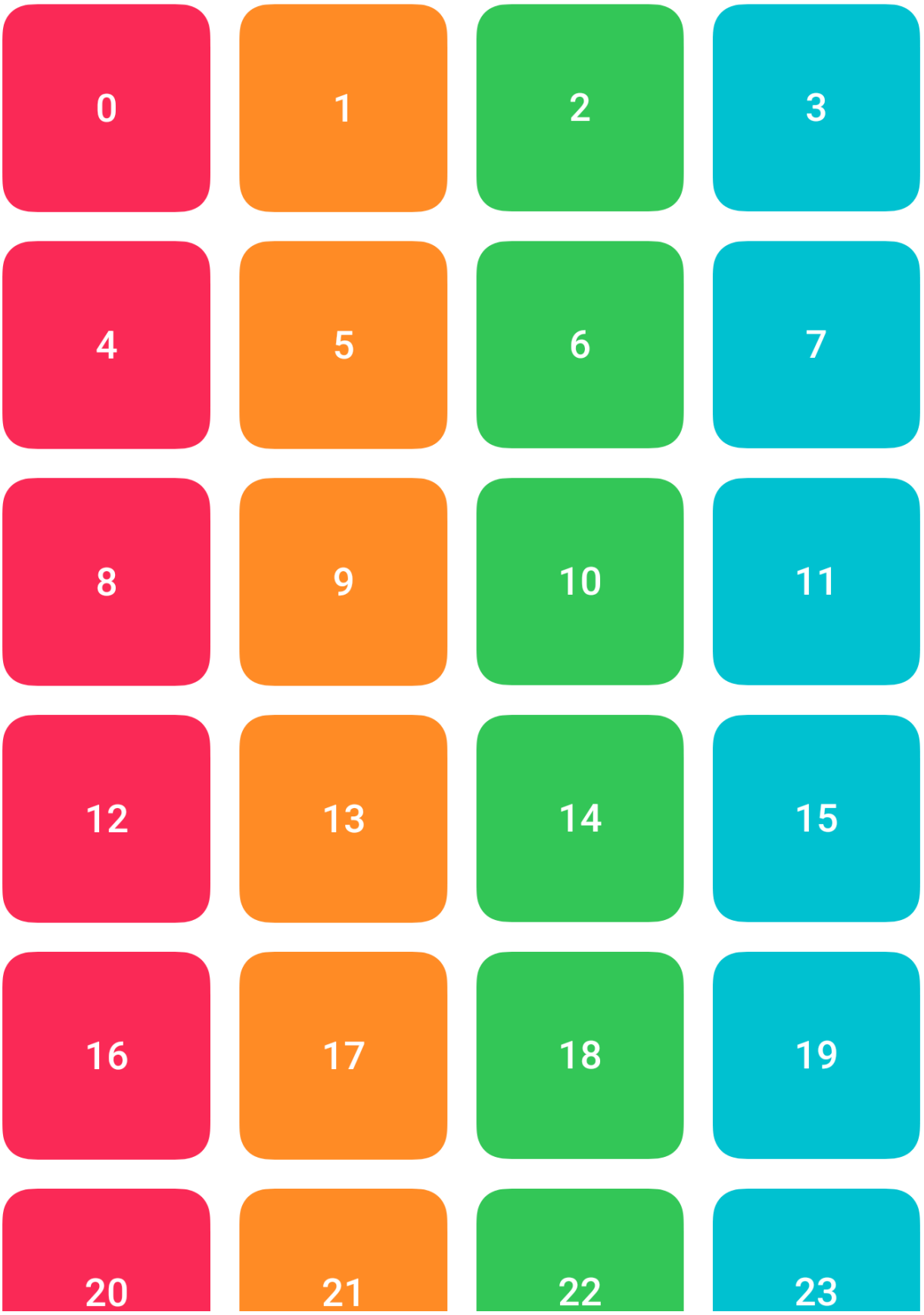
- Create a glass bottom bar over the `MainNavHost`.

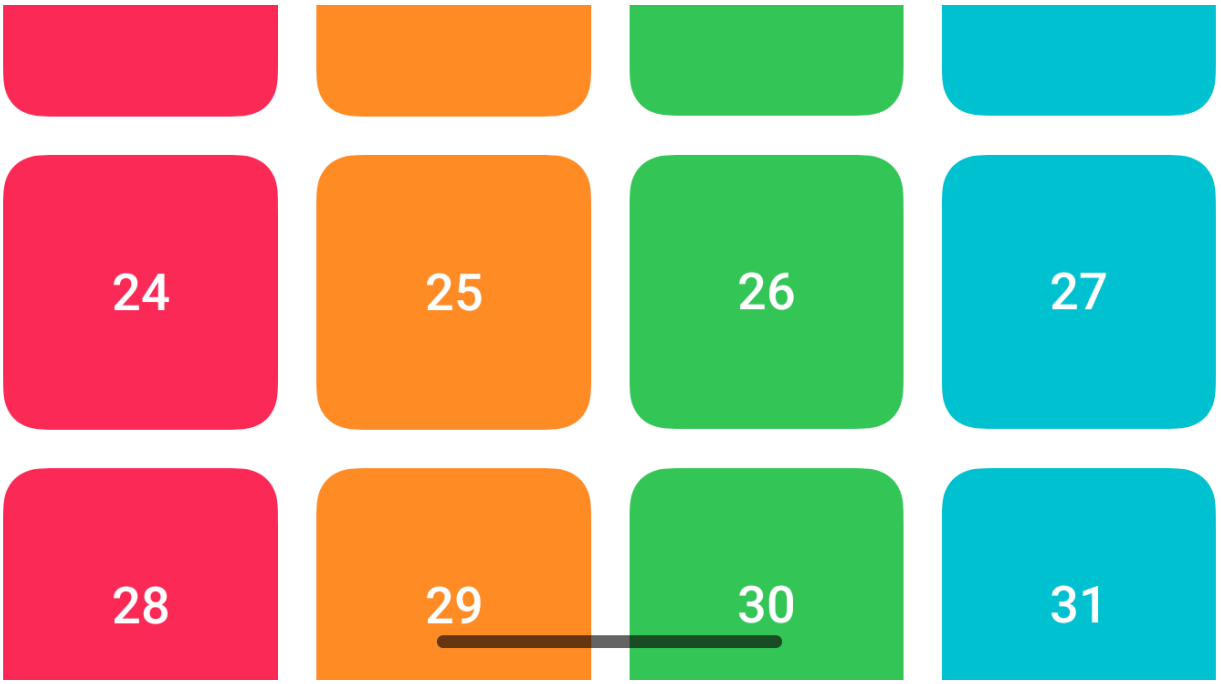
Copy

```
Box(Modifier.fillMaxSize()) {
    MainNavHost()
}
```

Here is what `MainNavHost` looks like:

Liquid Glass on Android





What you will learn

- Create and draw backdrops
- Apply effects to the backdrops
- Handle the background drawing correctly
- Ensure the readability

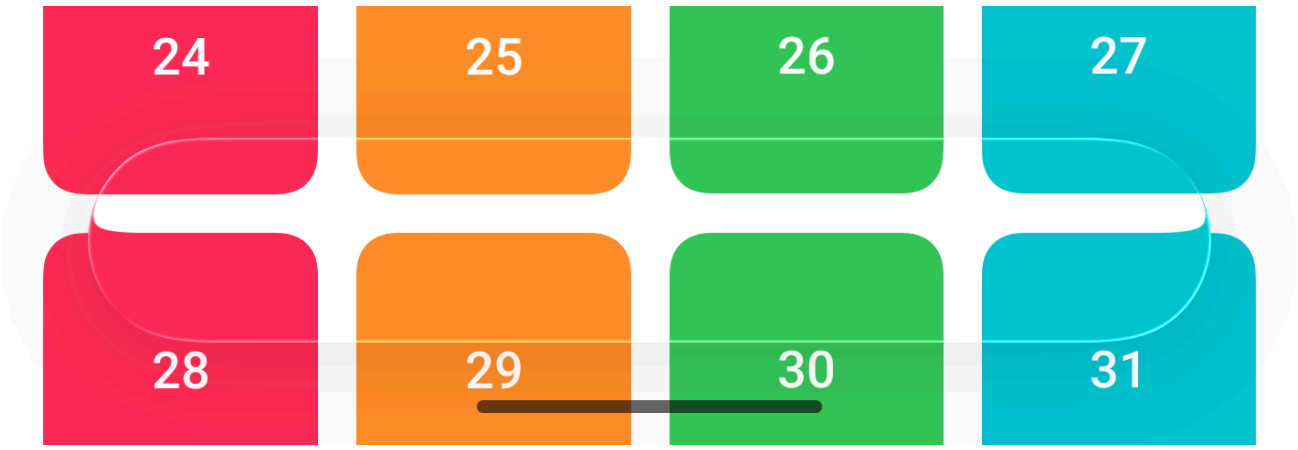
Steps

Draw backdrop and add lens effect

Copy

```
Box(Modifier.fillMaxSize()) {
    val backdrop = rememberLayerBackdrop()

    MainNavHost(
        modifier = Modifier.layerBackdrop(backdrop)
    )

    Box(
        Modifier
            .safeContentPadding()
            .drawBackdrop(
                backdrop = backdrop,
                shape = { CircleShape },
                effects = {
                    lens(16f.dp.toPx(), 32f.dp.toPx())
                }
            )
            .height(64f.dp)
            .fillMaxWidth()
            .align(Alignment.BottomCenter)
```

```
    )
}
```

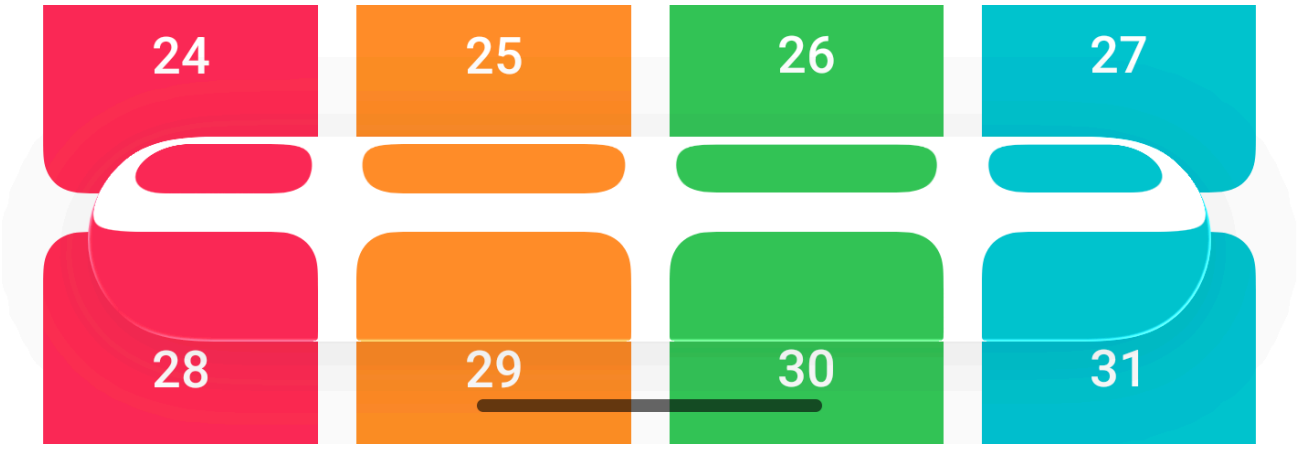


Noooo! The effect is wrong! There are transparent pixels in the bottom bar. Because we only draw the `MainNavHost` , **the background outside of** `MainNavHost` **should be drawn too**.

Draw the background to the backdrop (optional)

Copy

```
val backgroundColor = Color.White
val backdrop = rememberLayerBackdrop {
    drawRect(backgroundColor)
    drawContent()
}
```



Nice work!

Try to adjust the lens effect and observe what will happen.

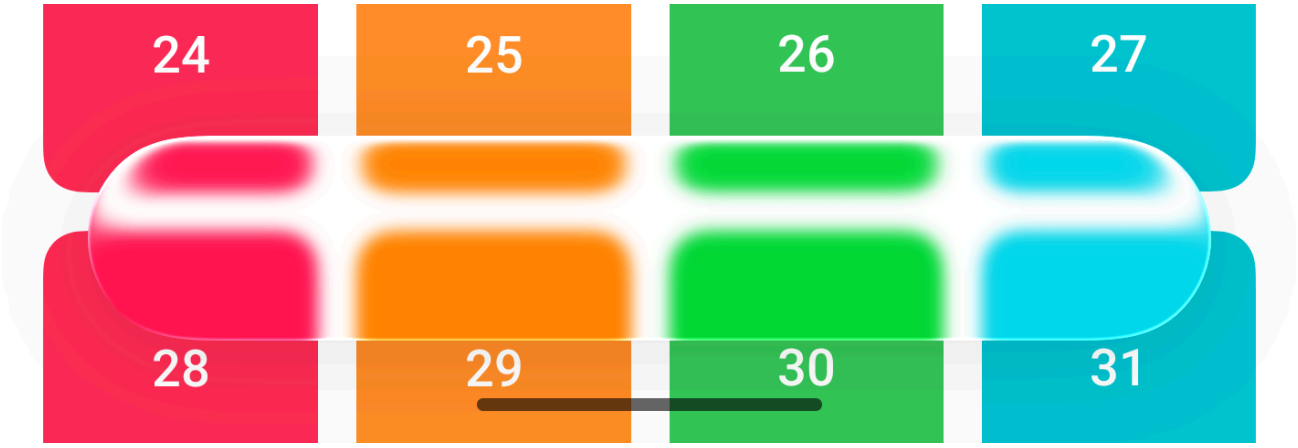
Add blur effect

Copy

```
Modifier.drawBackdrop(
    backdrop = backdrop,
```

```
    shape = { CircleShape },
    effects = {
        vibrancy()
        blur(4f.dp.toPx())
        lens(16f.dp.toPx(), 32f.dp.toPx())
    }
)
```

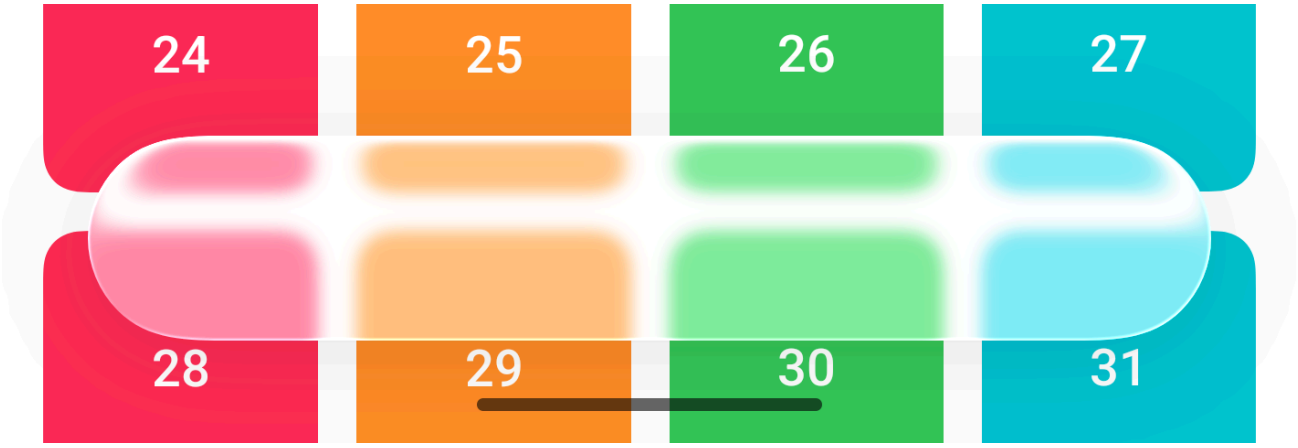
The use of `vibrancy()` enhances the saturation, giving us more visual impact.



Add surface for readability

Copy

```
Modifier.drawBackdrop(
    backdrop = backdrop,
    shape = { CircleShape },
    effects = {
        vibrancy()
        blur(4f.dp.toPx())
        lens(16f.dp.toPx(), 32f.dp.toPx())
    },
    onDrawSurface = { drawRect(Color.White.copy(alpha = 0.5f)) }
)
```



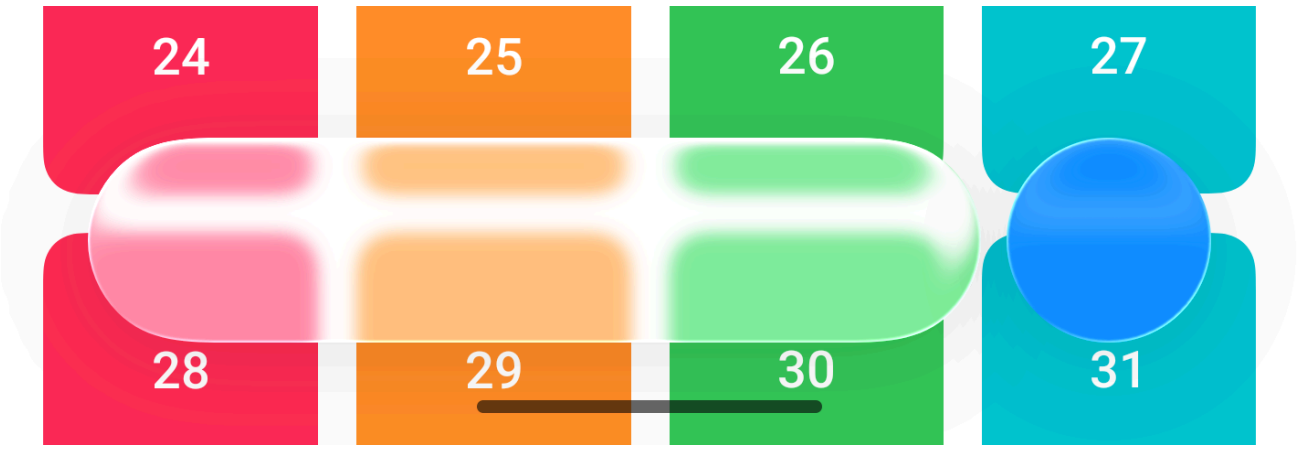
The readability has increased. **You must balance between beauty and readability.**

Final code

Copy

```
Box(Modifier.fillMaxSize()) {
    val backgroundColor = Color.White
    val backdrop = rememberLayerBackdrop {
        drawRect(backgroundColor)
        drawContent()
    }

    MainNavHost(
        modifier = Modifier.layerBackdrop(backdrop)
    )

    Box(
        Modifier
            .safeContentPadding()
            .drawBackdrop(
                backdrop = backdrop,
                shape = { CircleShape },
                effects = {
                    vibrancy()
                    blur(4f.dp.toPx())
                    lens(16f.dp.toPx(), 32f.dp.toPx())
                },
                onDrawSurface = { drawRect(Color.White.copy(alpha = 0.5f)) }
            )
            .height(64f.dp)
            .fillMaxWidth()
            .align(Alignment.BottomCenter)
    )
}
```

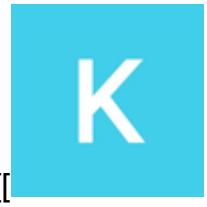
Exercise: Add a tinted glass icon button



It is recommended to use `BlendMode.Hue` , so that the hue of backdrop will adapt to the tint color.

PreviousGet startedNextInteractive Glass Bottom Bar

Last updated 4 months ago



Glass Bottom Bar | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

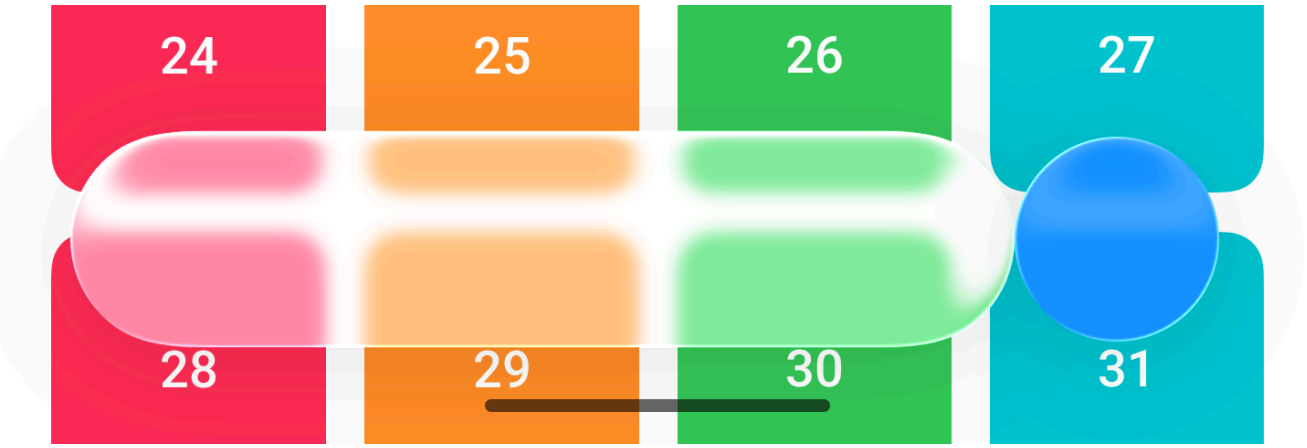
- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

- Goals
- What you will learn
- Steps
- Press to scale
- Prevent backdrop from scaling
- Final code

Ask

Interactive Glass Bottom Bar

Add interactive feedbacks for the glass bottom bar



Goals

- Add "press to scale" animation to the glass bottom bar

What you will learn

- Handle the transformations (scale, rotation) correctly
- Make use of `layerBlock` parameter of the `drawBackdrop` modifier

Steps

Press to scale

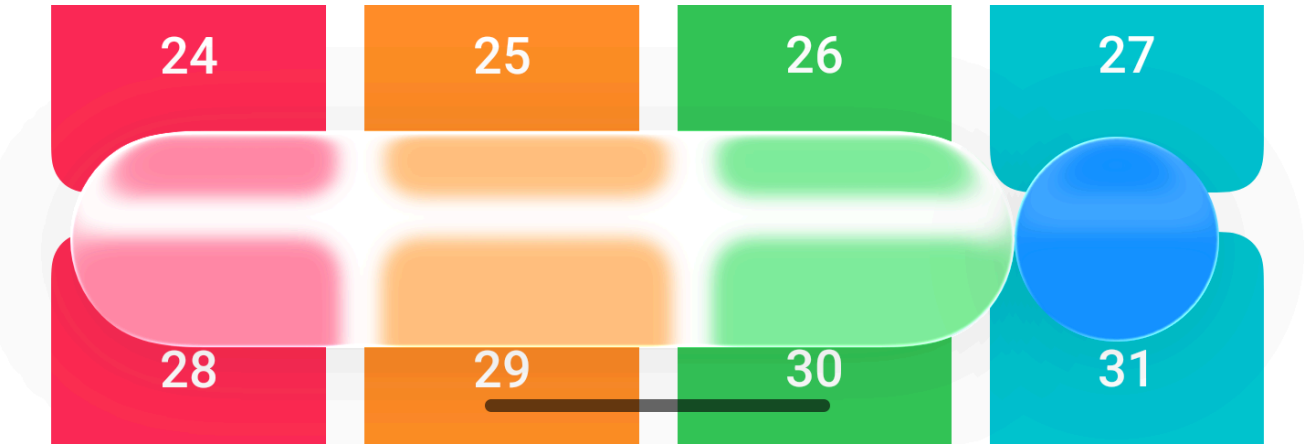
Update the code for the glass bottom bar.

Copy

```
val animationScope = rememberCoroutineScope()
val progressAnimation = remember { Animatable(0f) }

Box(
    Modifier
        .graphicsLayer {
            val progress = progressAnimation.value
            val maxScale = (size.width + 16f.dp.toPx()) / size.width
            val scale = lerp(1f, maxScale, progress)
            scaleX = scale
            scaleY = scale
        }
        .drawBackdrop(
            backdrop = backdrop,
            shape = { CircleShape },
            effects = {
                vibrancy()
                blur(4f.dp.toPx())
                lens(16f.dp.toPx(), 32f.dp.toPx())
            },
            onDrawSurface = { drawRect(Color.White.copy(alpha = 0.5f)) }
```

```
        )
        .clickable {}
        .pointerInput(animationScope) {
            val animationSpec = spring(0.5f, 300f, 0.001f)
            awaitEachGesture {
                // press
                awaitFirstDown()
                animationScope.launch {
                    progressAnimation.animateTo(1f, animationSpec)
                }

                // release
                waitForUpOrCancellation()
                animationScope.launch {
                    progressAnimation.animateTo(0f, animationSpec)
                }
            }
        }
        .fillMaxHeight()
        .weight(1f)
)
```

Press the bottom bar.



Oops! The backdrop is misplaced, it will also scale with the bottom bar. **But the backdrop shouldn't scale**.

Prevent backdrop from scaling

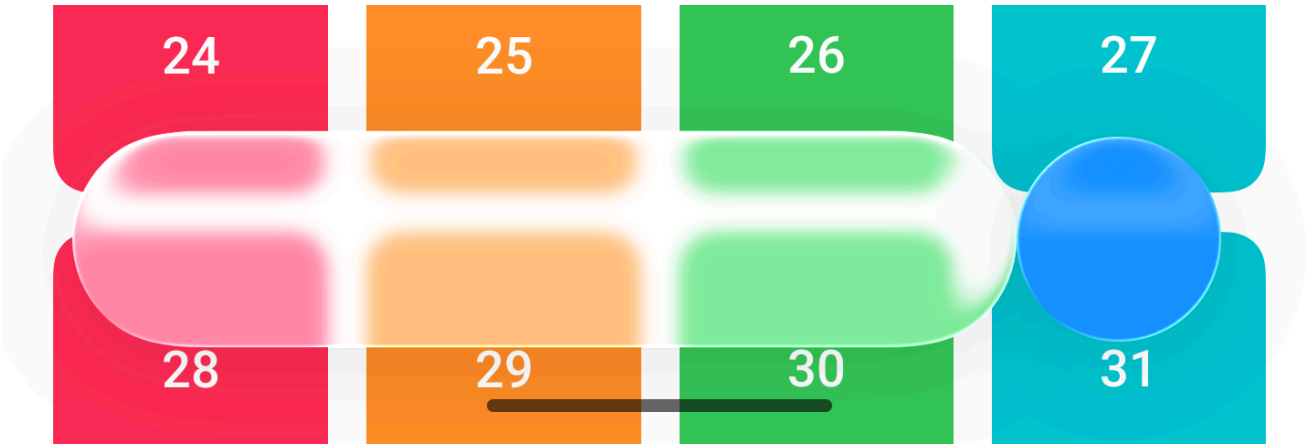
Move the code in `graphicsLayer` to `layerBlock` in the `drawBackdrop` modifier.

Copy

```
val animationScope = rememberCoroutineScope()
val progressAnimation = remember { Animatable(0f) }

Box(
    Modifier
```

```
            .drawBackdrop(
                backdrop = backdrop,
                shape = { CircleShape },
                effects = {
                    vibrancy()
                    blur(4f.dp.toPx())
                    lens(16f.dp.toPx(), 32f.dp.toPx())
                },
                layerBlock = {
                    val progress = progressAnimation.value
                    val maxScale = (size.width + 16f.dp.toPx()) / size.width
                    val scale = lerp(1f, maxScale, progress)
                    scaleX = scale
                    scaleY = scale
                },
                onDrawSurface = { drawRect(Color.White.copy(alpha = 0.5f)) }
            )
            .clickable(interactionSource = null, indication = null) {}
            .pointerInput(animationScope) {
                val animationSpec = spring(0.5f, 300f, 0.001f)
                awaitEachGesture {
                    // press
                    awaitFirstDown()
                    animationScope.launch {
                        progressAnimation.animateTo(1f, animationSpec)
                    }

                    // release
                    waitForUpOrCancellation()
                    animationScope.launch {
                        progressAnimation.animateTo(0f, animationSpec)
                    }
                }
            }
            .fillMaxHeight()
            .weight(1f)
    )
```

Press the bottom bar again.



It works correctly! The content will scale but the backdrop won't scale.

Final code

PreviousGlass Bottom BarNextGlass Bottom Sheet

Last updated 4 months ago

Interactive Glass Bottom Bar | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

- Goals
- What you will learn
- Steps
- Create a GlassBottomSheet
- Use the bottom sheet as a backdrop for the glass button (WRONG code)
- Use the bottom sheet as a backdrop for the glass button (CORRECT code)
- Final code

Ask

1. Tutorials

Glass Bottom Sheet

Create a glass bottom sheet

Goals

- Create a glass bottom sheet based on the code:

Copy

```
Box(Modifier.fillMaxSize()) {
    val backgroundColor = Color.White
    val backdrop = rememberLayerBackdrop {
        drawRect(backgroundColor)
        drawContent()
    }

    MainNavHost(
        modifier = Modifier.layerBackdrop(backdrop)
    )

    GlassBottomSheet(backdrop = backdrop)
}
```

What you will learn

- Handle the case of "glass on glass"
- Make use of `exportedBackdrop` parameter of the `drawBackdrop` modifier

Steps

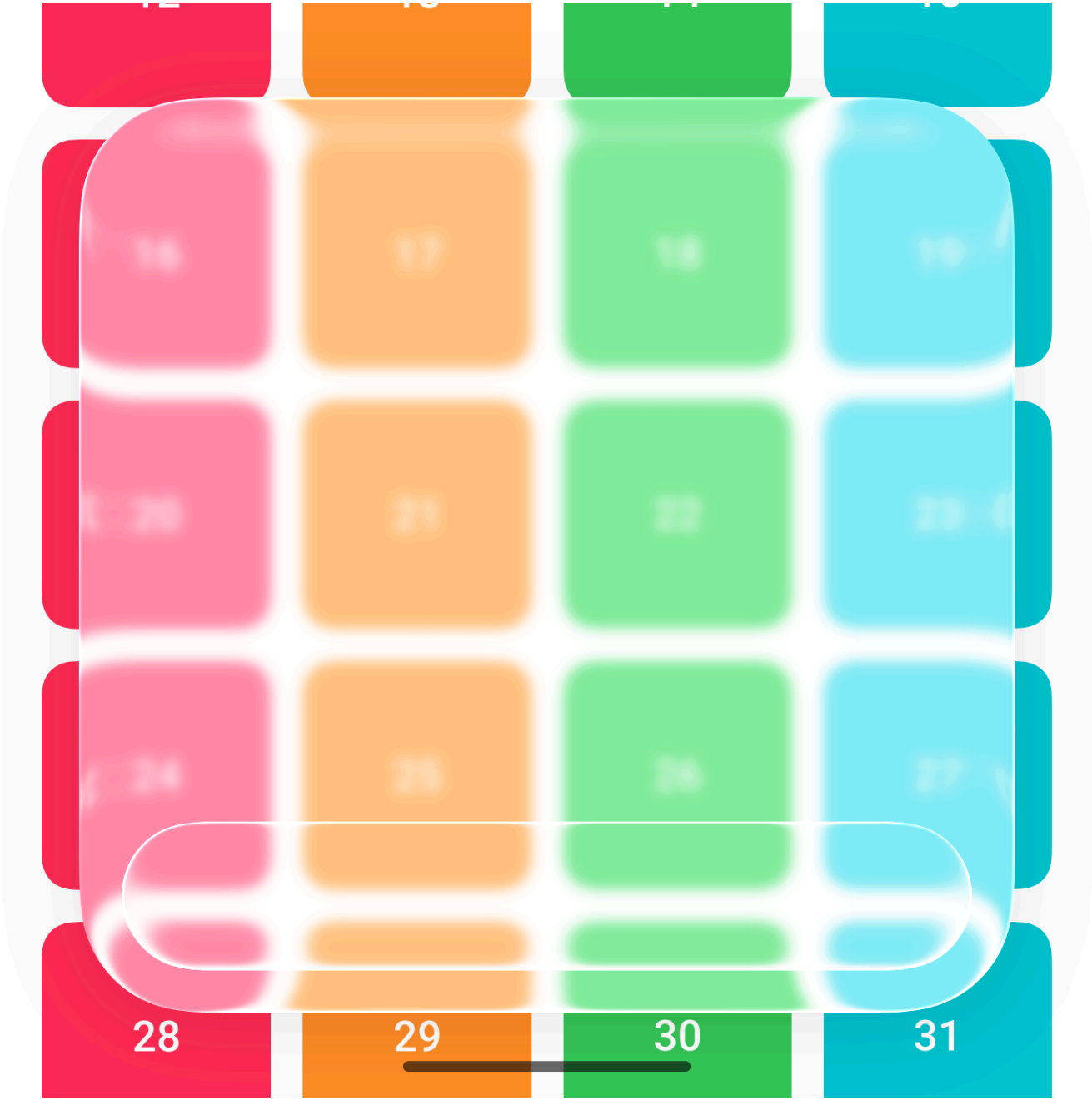
Create a GlassBottomSheet

GlassBottomSheet.kt

Copy

```
// your package

import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.BoxScope
import androidx.compose.foundation.layout.Column
import androidx.compose.foundation.layout.Spacer
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.safeContentPadding
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import com.kyant.backdrop.Backdrop
import com.kyant.backdrop.drawBackdrop
import com.kyant.backdrop.effects.blur
import com.kyant.backdrop.effects.lens
import com.kyant.backdrop.effects.vibrancy

@Composable
fun BoxScope.GlassBottomSheet(backdrop: Backdrop) {
    Column(
        Modifier
            .safeContentPadding()
            .drawBackdrop(
                backdrop = backdrop,
                shape = { RoundedCornerShape(44f.dp) },
                effects = {
                    vibrancy()
                    blur(4f.dp.toPx())
                    lens(24f.dp.toPx(), 48f.dp.toPx(), true)
                },
                onDrawSurface = { drawRect(Color.White.copy(alpha = 0.5f)) }
            )
            .fillMaxWidth()
            .align(Alignment.BottomCenter)
    ) {
        Spacer(Modifier.height(256f.dp))
        // glass button
        Box(
            Modifier
                .padding(16f.dp)
                .drawBackdrop(
                    backdrop = backdrop,
                    shape = { CircleShape },
```

```
                    shadow = null,
                    effects = {
                        vibrancy()
                        blur(4f.dp.toPx())
                        lens(16f.dp.toPx(), 32f.dp.toPx())
                    },
                    onDrawSurface = { drawRect(Color.White.copy(alpha =
0.5f)) }
                )
                .height(56f.dp)
                .fillMaxWidth()
        )
    }
}
```



The backdrop for the glass button is `backdrop`, but we want to include the bottom sheet.

Use the bottom sheet as a backdrop for the glass button (WRONG code)

The WRONG idea is to set a new LayerBackdrop after `drawBackdrop` .

Copy

```
val bottomSheetBackdrop = rememberLayerBackdrop()
Column(
    Modifier
        .layerBackdrop(bottomSheetBackdrop)
        .drawBackdrop()
) {
    // glass button
    Box(
        Modifier
            .drawBackdrop(
                backdrop = bottomSheetBackdrop,
            )
    )
}
```

You will get a crash:

Fatal signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0x **in tid** (RenderThread), pid __

Because the `layerBackdrop` modifier will draw the content to the `bottomSheetBackdrop` , and the content will draw the `bottomSheetBackdrop` , **it's a loop**!

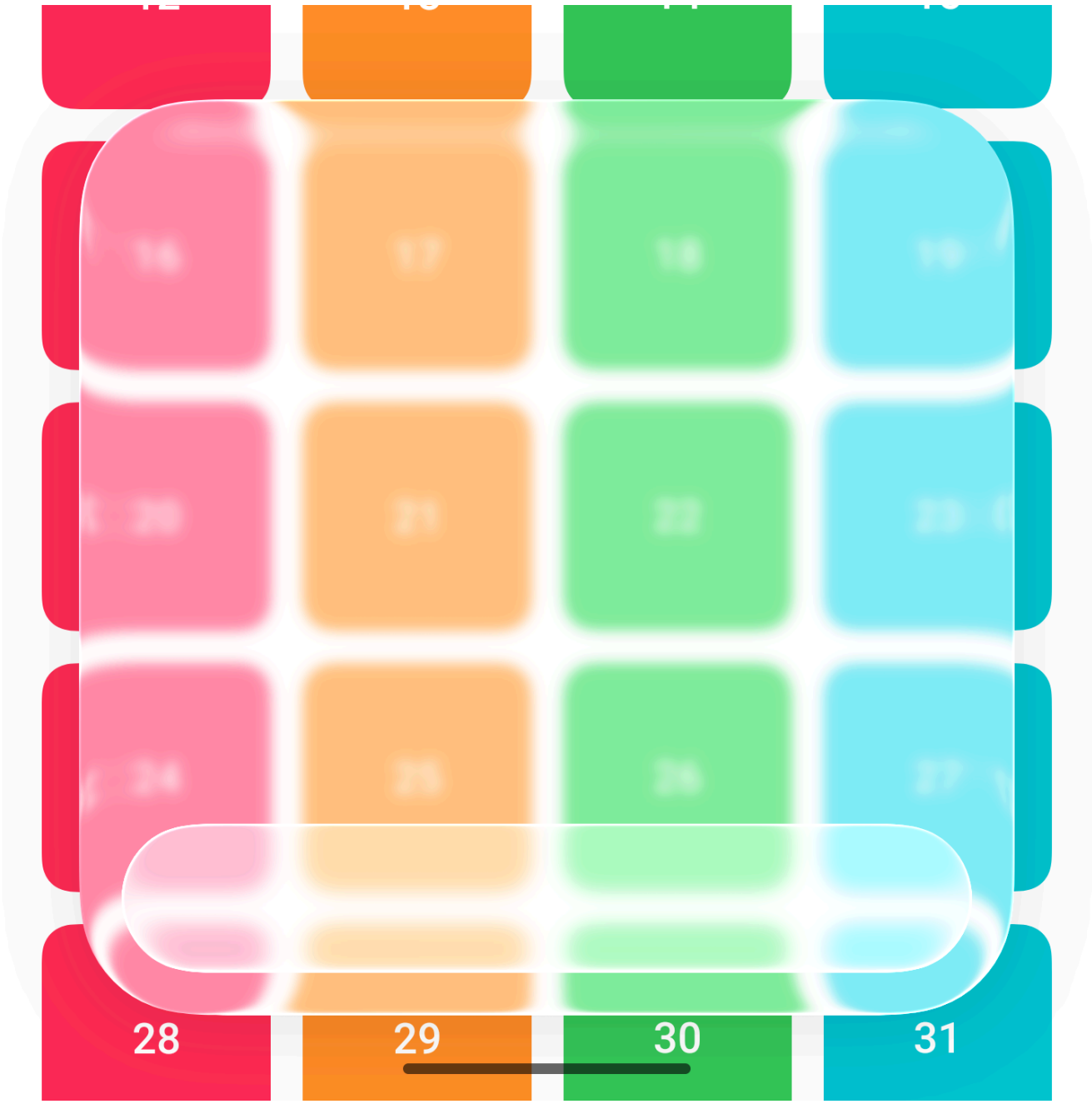
Use the bottom sheet as a backdrop for the glass button (CORRECT code)

Use `exportedBackdrop` in `drawBackdrop` modifier, **it will skip drawing the content**.

Copy

```
val bottomSheetBackdrop = rememberLayerBackdrop()
Column(
    Modifier
        .drawBackdrop(
            backdrop = backdrop,
            exportedBackdrop = bottomSheetBackdrop,
        )
) {
    // glass button
    Box(
        Modifier
            .drawBackdrop(
                backdrop = bottomSheetBackdrop,
            )
```

```
    )
}
```



Final code

PreviousInteractive Glass Bottom BarNextGlass Slider

Last updated 4 months ago

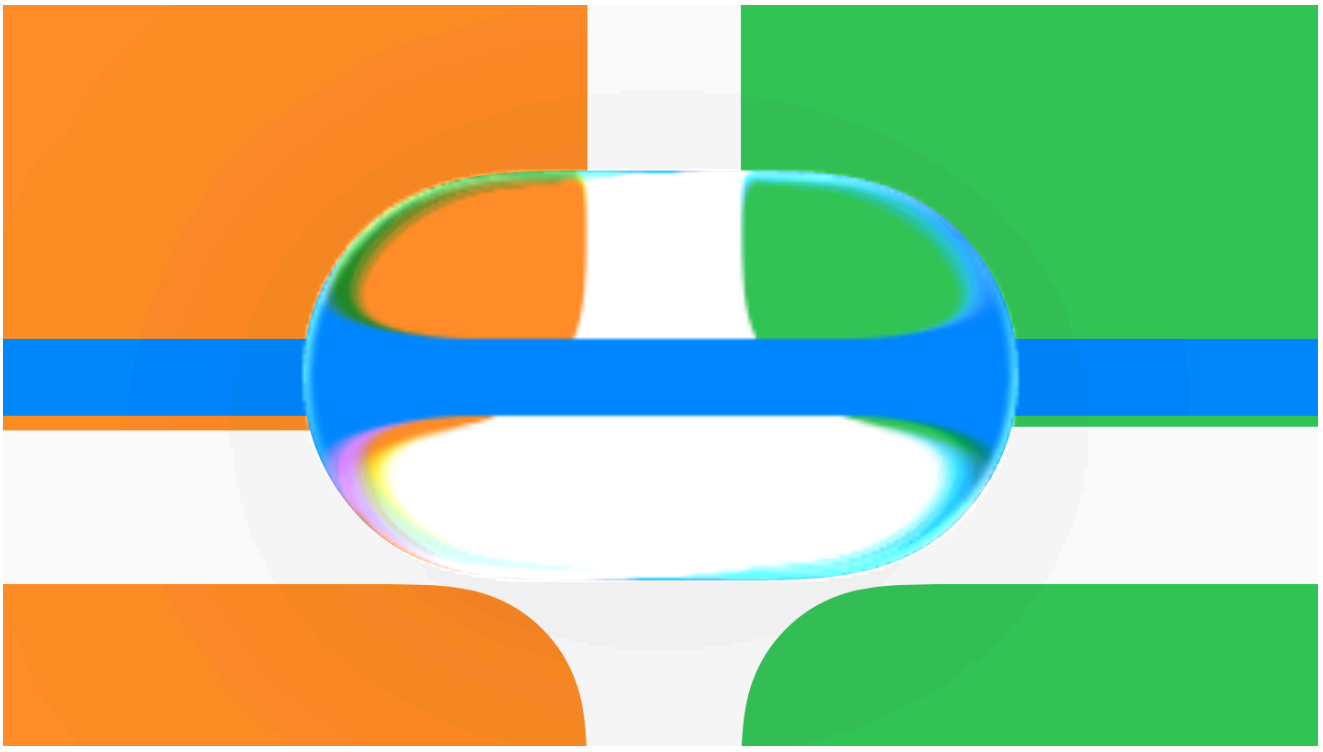
Glass Bottom Sheet | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Glass Slider

Create a glass slider



Goals

- Create a glass slider based on the code:

Copy

```
Box(Modifier.fillMaxSize()) {
    val backgroundColor = Color.White
    val backdrop = rememberLayerBackdrop {
        drawRect(backgroundColor)
        drawContent()
    }

    MainNavHost(
        modifier = Modifier.layerBackdrop(backdrop)
    )

    GlassSlider(backdrop = backdrop)
}
```

What you will learn

- Combine multiple backdrops by using the `rememberCombinedBackdrop` function

Steps

Create a GlassSlider

GlassSlider.kt

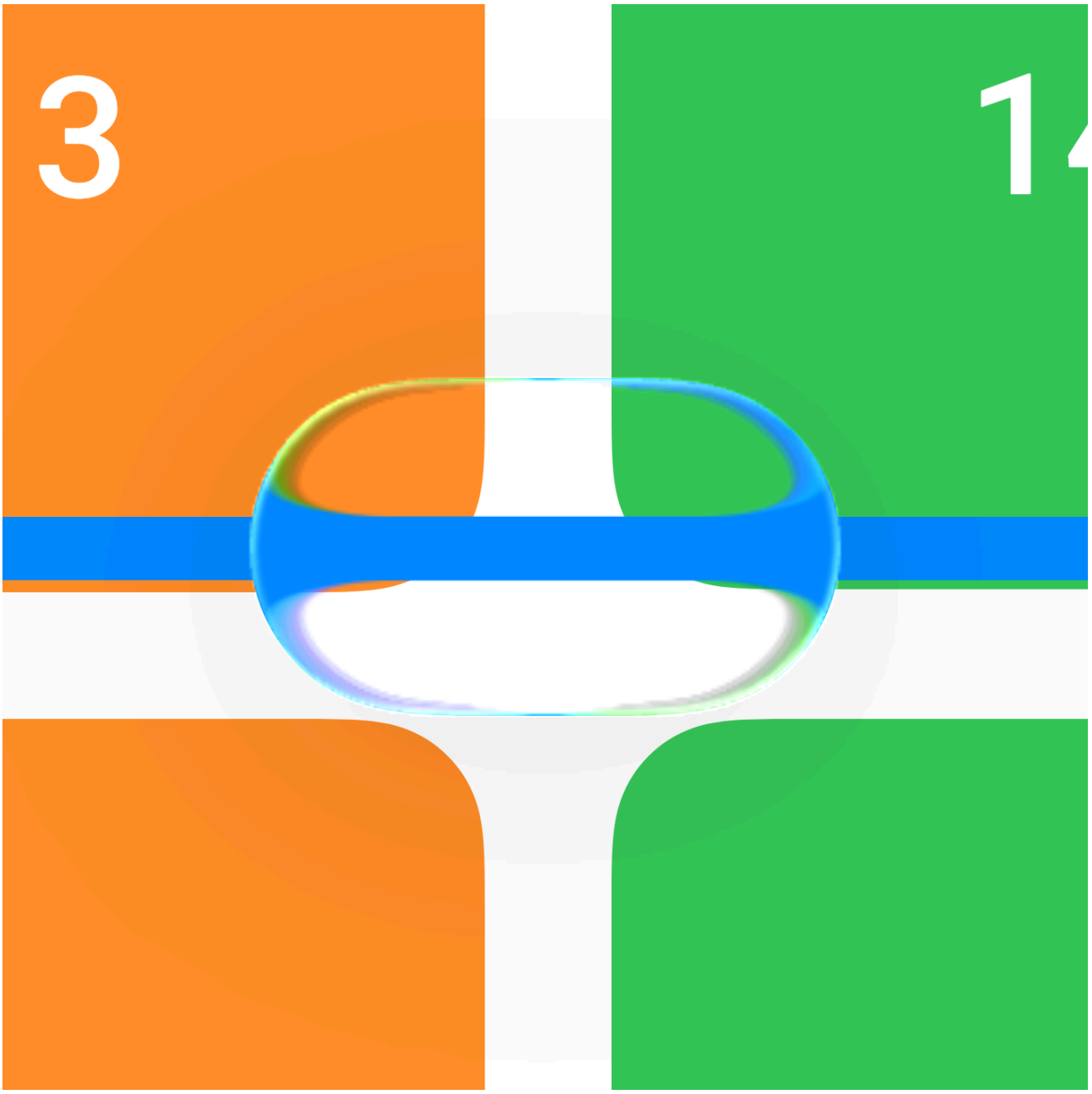
Copy

```
// your package

import androidx.compose.foundation.background
import androidx.compose.foundation.layout.Box
import androidx.compose.foundation.layout.BoxWithConstraints
import androidx.compose.foundation.layout.fillMaxWidth
import androidx.compose.foundation.layout.height
import androidx.compose.foundation.layout.offset
import androidx.compose.foundation.layout.padding
import androidx.compose.foundation.layout.size
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.runtime.Composable
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.unit.dp
import com.kyant.backdrop.Backdrop
import com.kyant.backdrop.backdrops.layerBackdrop
import com.kyant.backdrop.backdrops.rememberLayerBackdrop
import com.kyant.backdrop.drawBackdrop
import com.kyant.backdrop.effects.lens

@Composable
fun GlassSlider(
    backdrop: Backdrop,
    modifier: Modifier = Modifier
) {
    BoxWithConstraints(
        modifier
            .padding(horizontal = 24f.dp)
            .fillMaxWidth(),
        contentAlignment = Alignment.CenterStart
    ) {
        val trackBackdrop = rememberLayerBackdrop()

        // track
        Box(
            Modifier
                .layerBackdrop(trackBackdrop)
                .background(Color(0xFF0088FF), CircleShape)
                .height(6f.dp)
                .fillMaxWidth()
        )

        // thumb
        Box(
            Modifier
                .offset(x = maxWidth / 2f - 28f.dp)
                .drawBackdrop(
```

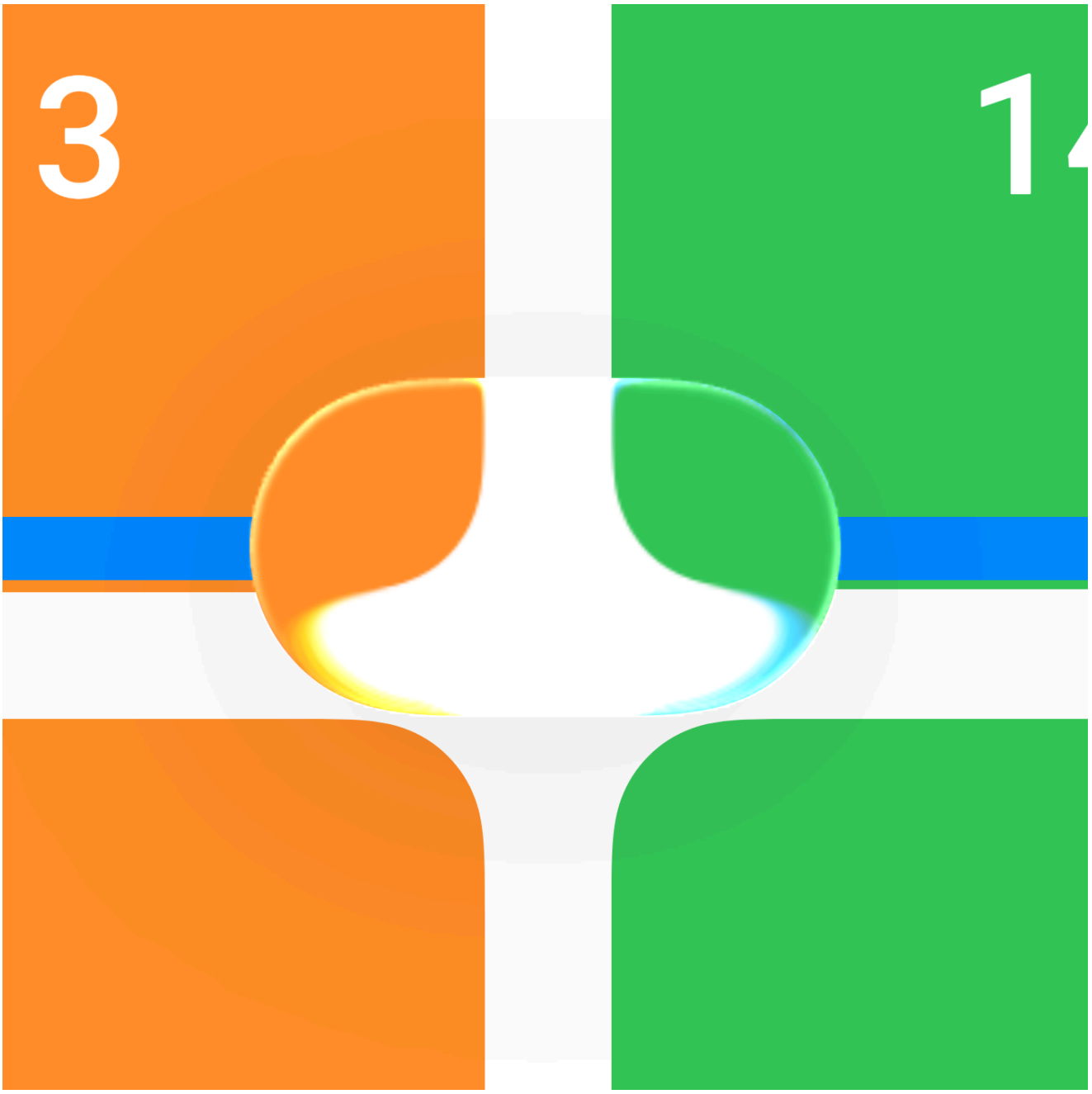
```
                    // We want to draw both of `backdrop` and
`trackBackdrop`
                    backdrop = trackBackdrop,
                    shape = { CircleShape },
                    effects = {
                        lens(
                            refractionHeight = 12f.dp.toPx(),
                            refractionAmount = 16f.dp.toPx(),
                            chromaticAberration = true
                        )
                    }
                )
                .size(56f.dp, 32f.dp)
        )
    }
}
```

Replace line 51 in GlassSlider.kt with

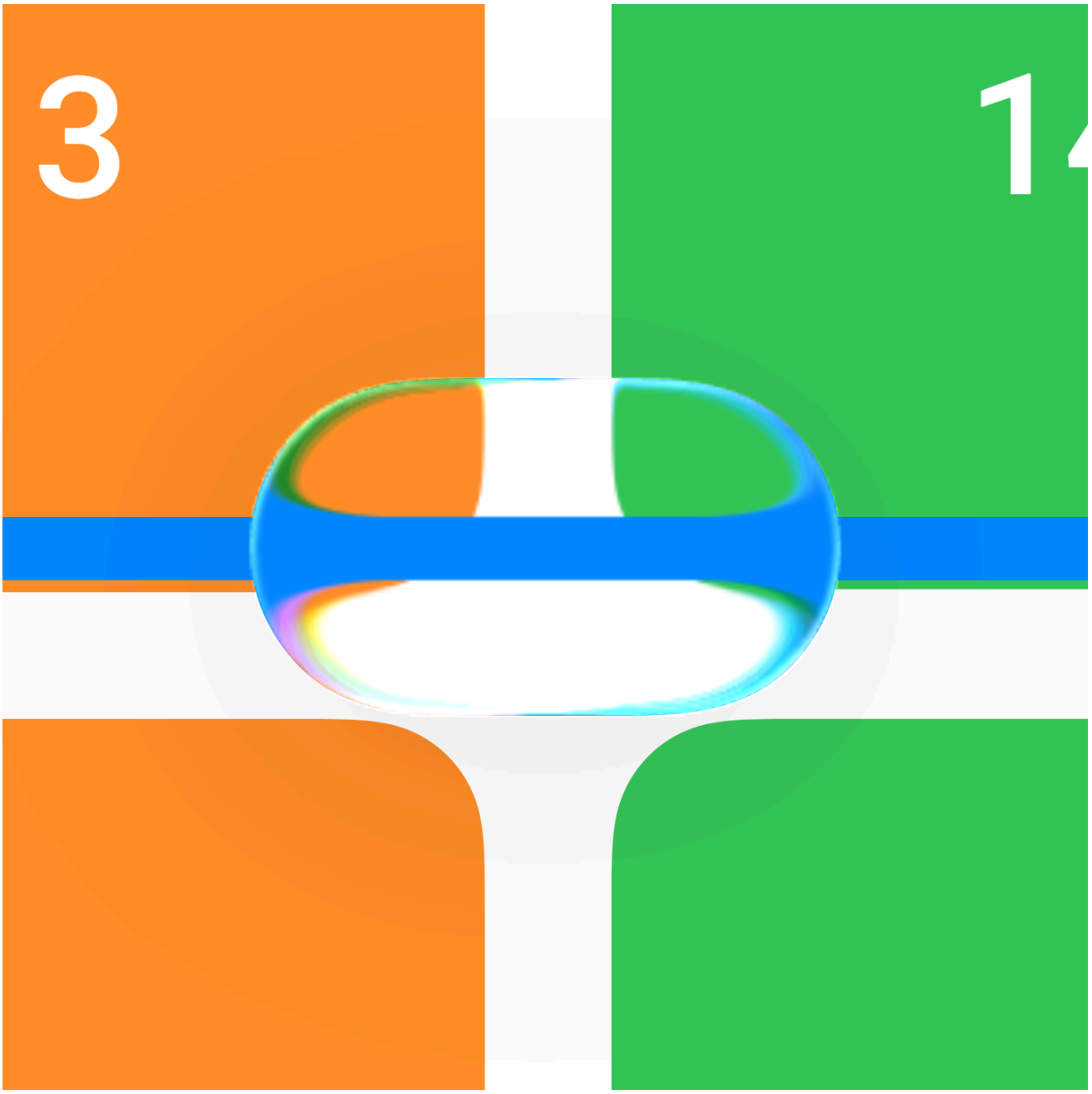
1. `trackBackdrop`



1. `backdrop`



1. `rememberCombinedBackdrop(backdrop, trackBackdrop)`



Background and track are refracted by thumb simultaneously.

Background and track are refracted by thumb simultaneously.

PreviousGlass Bottom SheetNextSmoother rounded corners

Last updated 4 months ago

Glass Slider | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

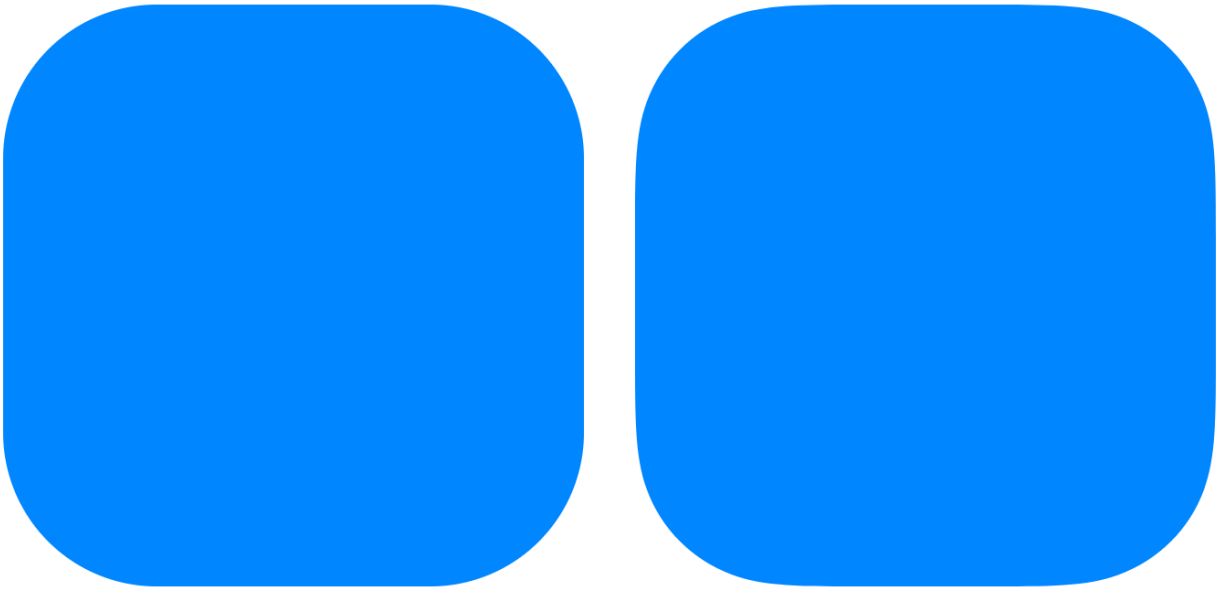
Ask

Smoother rounded corners

The Capsule library allows you to create beautiful and smooth G2 continuous rounded rectangles.

In the previous tutorials, all shapes are created by the library.

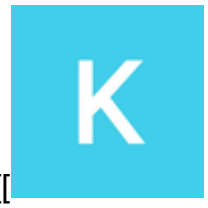
Can you tell the difference?



If you prefer the right one, the Capsule library is your right choice.

PreviousGlass SliderNextProgressive blur

Last updated 4 months ago

Smoother rounded corners | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

- Alpha-masked progressive blur
- Exact progressive blur

Ask

1. Tutorials

Progressive blur

Create progressive blur effect

Alpha-masked progressive blur

Copy

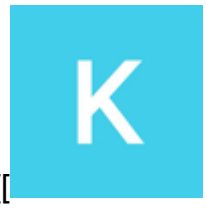
```
Modifier.drawPlainBackdrop(
    backdrop = backdrop,
    shape = { RectangleShape },
    effects = {
        blur(4f.dp.toPx())
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            effect(
                RenderEffect.createRuntimeShaderEffect(
                    obtainRuntimeShader(
                        "AlphaMask",
                        """
uniform shader content;

uniform float2 size;
layout(color) uniform half4 tint;
uniform float tintIntensity;

half4 main(float2 coord) {
    float blurAlpha = smoothstep(size.y, size.y * 0.5, coord.y);
    float tintAlpha = smoothstep(size.y, size.y * 0.5, coord.y);
    return mix(content.eval(coord) * blurAlpha, tint * tintAlpha,
tintIntensity);
}"""
                    ).apply {
                        setFloatUniform("size", size.width, size.height)
                        setColorUniform("tint", tintColor.toArgb())
                        setFloatUniform("tintIntensity", 0.8f)
                    },
                    "content"
                )
            )
        }
    }
)
```

Exact progressive blur

TBD.

PreviousSmoother rounded cornersNextBackdrops

Last updated 4 months ago



Progressive blur | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

- Backdrop
- Layer backdrop
- Combined backdrop
- Canvas backdrop
- Empty backdrop

Ask

1. API

Backdrops

Backdrop

`rememberBackdrop` can draw a backdrop with custom commands.

Layer backdrop

`rememberLayerBackdrop` must be used with `Modifier.layerBackdrop` to draw the Composable's content, or it's derived by using `exportedBackdrop` parameter of the `drawBackdrop` modifier. It is coordinates-dependent.

Combined backdrop

`rememberCombinedBackdrop` can merge multiple backdrops into one backdrop. It is useful to create components such as tabs and sliders.

Canvas backdrop

`rememberCanvasBackdrop` can draw custom content to a empty backdrop. It is coordinates-independent.

Empty backdrop

`emptyBackdrop` draws nothing.

PreviousProgressive blurNextBackdrop effects

Last updated 4 months ago

Backdrops | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

- Color filter
- Custom ColorFilter
- Opacity
- Color controls (brightness, contrast, saturation)
- Vibrancy
- Exposure adjustment
- Gamma adjustment (Android 13+)
- Blur
- Blur effect
- Lens (Android 13+)
- Lens effect
- RenderEffect
- Custom RenderRffect

Ask

1. API

Backdrop effects

Backdrop effects are `RenderEffect`s radically. They only take effect with Android 12 and above. Some effects involving with `RuntimeShader` need Android 13 and above.

The order of effects matters. To create the right visual effects, you must apply them with the following order:

color filter ⇒ blur ⇒ lens

Color filter

Custom ColorFilter

Copy

```
colorFilter(colorFilter: (Android)ColorFilter)
```

Copy

```
colorFilter(colorFilter: (Compose)ColorFilter)
```

Opacity

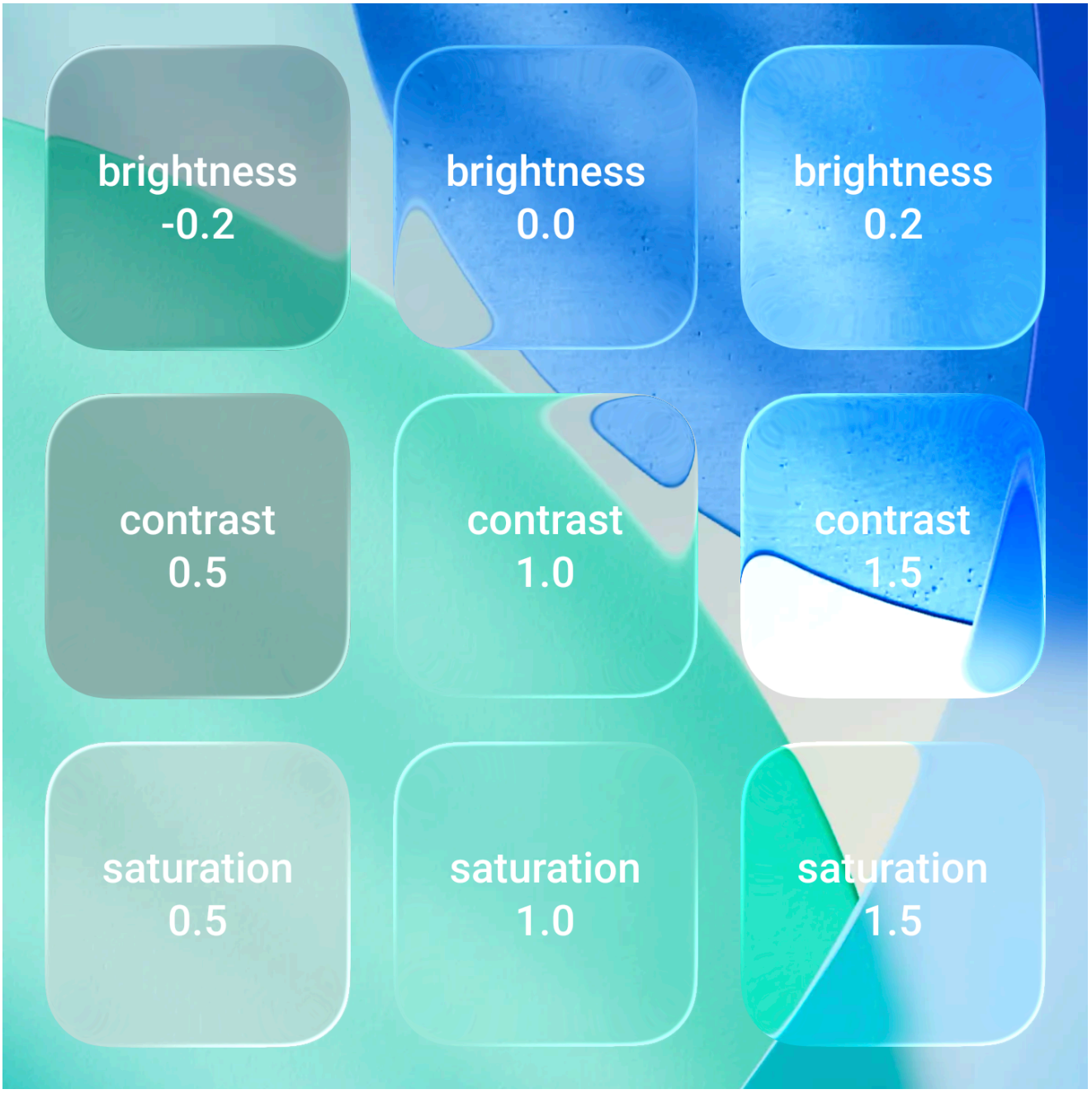
Copy

```
opacity(alpha: Float)
```

Color controls (brightness, contrast, saturation)

Copy

```
colorControls(
    brightness: Float = 0f,
    contrast: Float = 1f,
    saturation: Float = 1f
)
```



Vibrancy

Multiply saturation with 1.5. It is equivalent to `colorControls(saturation = 1.5f)` .

Copy

```
vibrancy()
```

Exposure adjustment

Copy

```
exposureAdjustment(ev: Float)
```

Gamma adjustment (Android 13+)

Copy

```
gammaAdjustment(power: Float)
```

Blur

Blur effect

Copy

```
blur(
    radius: Float,
    edgeTreatment: TileMode = TileMode.Clamp
)
```

Lens (Android 13+)

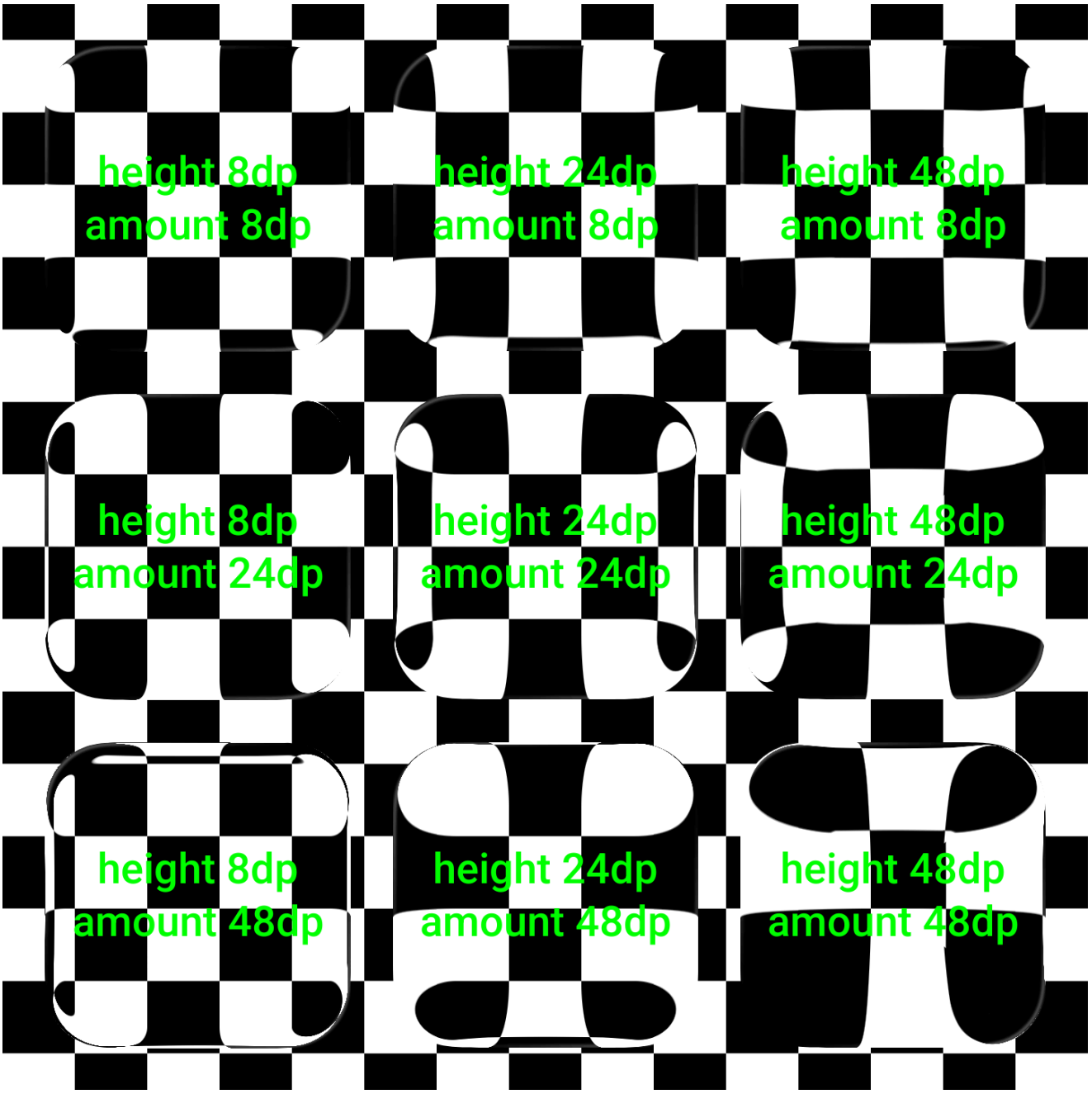
⚠️ To use the lens effect, your `shape` must be `CornerBasedShape` .

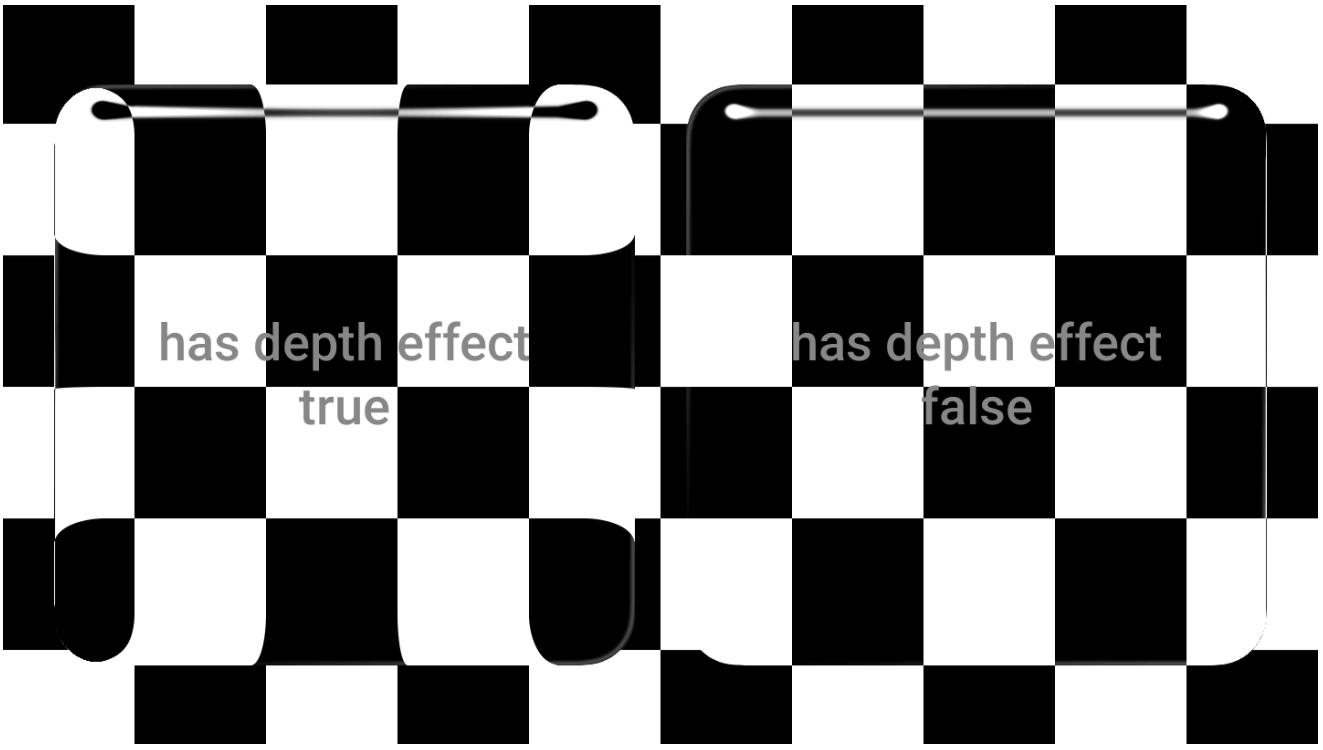
Lens effect

Copy

```
lens(
    refractionHeight: Float,
    refractionAmount: Float = height,
    depthEffect: Boolean = false,
    chromaticAberration: Boolean = false
)
```

- `height` must be in [0, `shape.minCornerRadius`]. If it exceeds, it will have discontinuities at some corners, but it's acceptable.
- `amount` must be in [0, `size.minDimension`].


height 8dp
amount 8dp
height 24dp
amount 8dp
height 48dp
amount 8dp
height 8dp
amount 24dp
height 24dp
amount 24dp
height 48dp
amount 24dp
height 8dp
amount 48dp
height 24dp
amount 48dp
height 48dp
amount 48dp


has depth effect
true
has depth effect
false

RenderEffect

Custom RenderRffect

Copy

```
effect(effect: (Android)RenderEffect)
```

Copy

```
effect(effect: (Compose)RenderEffect)
```

PreviousBackdropsNextFAQ

Last updated 4 months ago

Backdrop effects | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

Ask

FAQ

Why doesn't the effect work?

Because you may not use the `Modifier.layerBackdrop(backdrop: LayerBackdrop)` modifier, this article may help you.

Why did it crash?

If you get crash: Fatal signal 11 (SIGSEGV), code 2 (SEGV_ACCERR), fault addr 0x **in tid** (RenderThread), pid __ , this article may help you.

PreviousBackdrop effects

Last updated 4 months ago

FAQ | Backdrop] [[

Backdrop

](https://kyant.gitbook.io/backdrop)

Ctrlk

Ask

- Get started
- Tutorials
  - Glass Bottom Bar
  - Interactive Glass Bottom Bar
  - Glass Bottom Sheet
  - Glass Slider
  - Smoother rounded corners
  - Progressive blur
- API
  - Backdrops
  - Backdrop effects
- FAQ

Powered by GitBook

- Installation
- Tutorials

- Liquid Glass
- Blur
- API
- FAQ

Ask

Get started

The **Backdrop** library (GitHub) can draw a copy of the **background (backdrop)** to a **foreground** with various effects.

You can achieve amazing **liquid glass** effect with the library.

Installation

maven-central v2.0.0-alpha03

build.gradle.kts

Copy

```
dependencies {
    implementation("io.github.kyant0:backdrop:<version>")
}
```

Tutorials

Liquid Glass

🦕 **You must read and practise these tutorials before using the library.**

- Glass Bottom Bar
- Interactive Glass Bottom Bar
- Glass Bottom Sheet
- Glass Slider
- Smoother rounded corners

Blur

- Progressive blur

API

- Backdrops
- Backdrop effects

FAQ

NextGlass Bottom Bar

Last updated 3 months ago

Get started | Backdrop]