Grand Gestures in SwiftUI

@jessielinden

Jessie Linden
she/her

jessielinden@
mastodon.online


Disney
Aladdin


BEETLEJUICE


COME FROM AWAY

Agenda

Agenda

Overview

Demo

Compose

protocol Gesture

`protocol Gesture`

- Each conforming type is a `struct`
- Has a `body`
- Has an `associatedType Value`

protocol Gesture

Grand Gestures

I Singles

II Doubles

III How To Use Them

protocol Gesture

Singles

TapGesture

LongPressGesture

DragGesture

RotateGesture

MagnifyGesture

TapGesture

Parameters

```
var count: Int
```

Example

```
TapGesture(count: 2)
    .onEnded {
        print("Tapped 2 times")
    }
```

TapGesture

Parameters

```
var count: Int
```

Example

```
TapGesture(count: 2)
    .onEnded {
        print("Tapped 2 times")
    }
```

```
// view
    .onTapGesture(count: 2) {
        print("Tapped 2 times")
    }
```

SpatialTapGesture

Parameters

```
var count: Int

var coordinateSpace: CoordinateSpace
```

Example

```
SpatialTapGesture()
    .onEnded { value in
        print("Tapped at (\(value.x), \(value.y))")
    }
```

SpatialTapGesture

Example

```
SpatialTapGesture()
    .onEnded { value in
        print("Tapped at (\(value.x), \(value.y))")
    }
```

```
// view
    .onTapGesture { value in
        print("Tapped at (\(value.x), \(value.y))")
    }
```

Parameters

```
var minimumDuration: Double
```

```
var maximumDistance: CGFloat
```

Example

```
LongPressGesture(minimumDuration: 1, maximumDistance: 50)
```

Example

```
LongPressGesture(minimumDuration: 1, maximumDistance: 50)
    .onChanged { _ in
        print("Pressing...")
    }
```

Fires immediately,
not per the `minimumDuration`

Example

```
LongPressGesture(minimumDuration: 1, maximumDistance: 50)
    .onChanged { _ in
        print("Pressing...")
    }
    .onEnded { ended in
        if ended {
            print("success")
        } else {
            print(“failure”)
        }
    }
```

Fires immediately,
not per the `minimumDuration`

Fires at `minimumDuration`,
even if still pressing

We don’t even get here

Example

```
// view
    .onLongPressGesture(minimumDuration: 1) {
       print("Long press successful")
    } onPressingChanged: { inProgress in
      if inProgress {
         print("pressing")
      } else {
         print(“complete")
      }
    }
```

Example

```
// view
    .onLongPressGesture(minimumDuration: 1) {
       print("Long press successful")
    } onPressingChanged: { inProgress in
       if inProgress {
          print("pressing")
       } else {
          print(“complete")
       }
    }
```

Fires at `minimumDuration`, *even if still pressing*

Fires immediately

Fires at `minimumDuration` or release, whichever comes *first*

👀 onRelease 👀

Parameters

```
var minimumDistance: CGFloat

var coordinateSpace: CoordinateSpace
```

Example

```
DragGesture()
    .onChanged { value in
        print(“Dragging”, value.location)
    }
    .onEnded { value in
        print(“Drag size”, value.translation)
    }
```

DragGesture

Parameters

```
var minimumDistance: CGFloat

var coordinateSpace: CoordinateSpace
```

Value

```
var location: CGPoint
var translation: CGSize
var startLocation: CGPoint
// more
```

Example

```
DragGesture()
    .onChanged { value in
        print(“Dragging”, value.location)
    }
    .onEnded { value in
        print(“Drag size”, value.translation)
    }
```

RotateGesture

Parameters

```
var minimumAngleDelta: Angle
```

Example

```
RotateGesture()
    .onChanged { value in
        print(“Rotating…”, value.rotation)
    }
    .onEnded { _ in
        print(“Rotation complete”)
    }
```

RotateGesture

Parameters

```
var minimumAngleDelta: Angle
```

Value

```
var rotation: Angle
var velocity: Angle
var startAnchor: UnitPoint
var startLocation: CGPoint
var time: Date
```

Example

```
RotateGesture()
    .onChanged { value in
        print(“Rotating…”, value.rotation)
    }
    .onEnded { _ in
        print(“Rotation complete”)
    }
```

MagnifyGesture

Parameters

```
var minumumScaleDelta: CGFloat
```

Value

```
var magnification: CGFloat
var velocity: CGFloat
var startAnchor: UnitPoint
var startLocation: CGPoint
var time: Date
```

Example

```
MagnifyGesture()
    .onChanged { value in
        print(“Magnifying…”, value.magnification)
    }
    .onEnded { _ in
        print(“Magnification complete”)
    }
```

protocol Gesture

Grand Gestures

I Singles

II Doubles

III How To Use Them

protocol Gesture

Grand Gestures

I Singles — `TapGesture` `LongPressGesture` `DragGesture` `RotateGesture` `MagnifyGesture`

II Doubles

III How To Use Them — `.onChanged` `.onEnded`

protocol Gesture

Aa Name	Parameters	Value	Convenience API	# Finger
TapGesture	`var count: Int`		`.onTapGesture(count:perform:)`	1
SpatialTapGesture	`var count: Int` `var coordinateSpace: CGFloat`	`var location: CGPoint` `var location3D: Point3D`	`.onTapGesture(count:perform:)`	1
LongPressGesture	`var minimumDuration: Double` `var maximumDistance: CGFloat`		`.onLongPressGesture(minimumDuration: maximumDistance:perform:onPressingChanged:)`	1

DragGesture	`var minimumDistance: CGFloat`
	`var coordinateSpace: CGFloat`
		`var location: CGPoint`
		`var translation: CGSize`
		`var startLocation: CGPoint`
		`// more`
				1

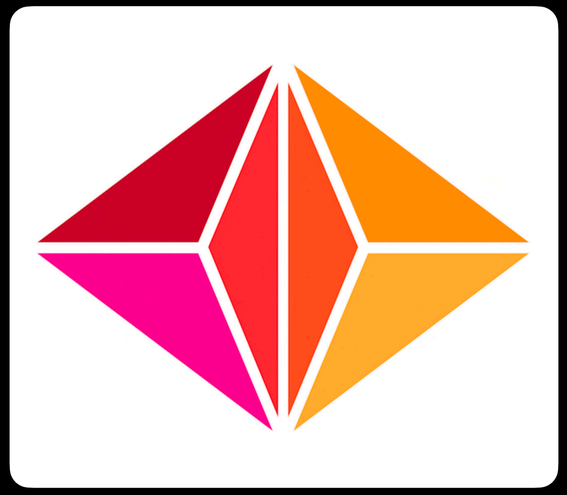
RotateGesture	`var minimumAngleDelta: Angle`	`var rotation: Angle`
		`var velocity: Angle`
		`var startAnchor: UnitPoint`
		`var startLocation: CGPoint`
		`var time: Date`
				2

MagnifyGesture	`var minimumScaleDelta: Angle`	`var magnification: CGFloat`
		`var velocity: CGFloat`
		`var startAnchor: UnitPoint`
		`var startLocation: CGPoint`
		`var time: Date`
				2

Agenda

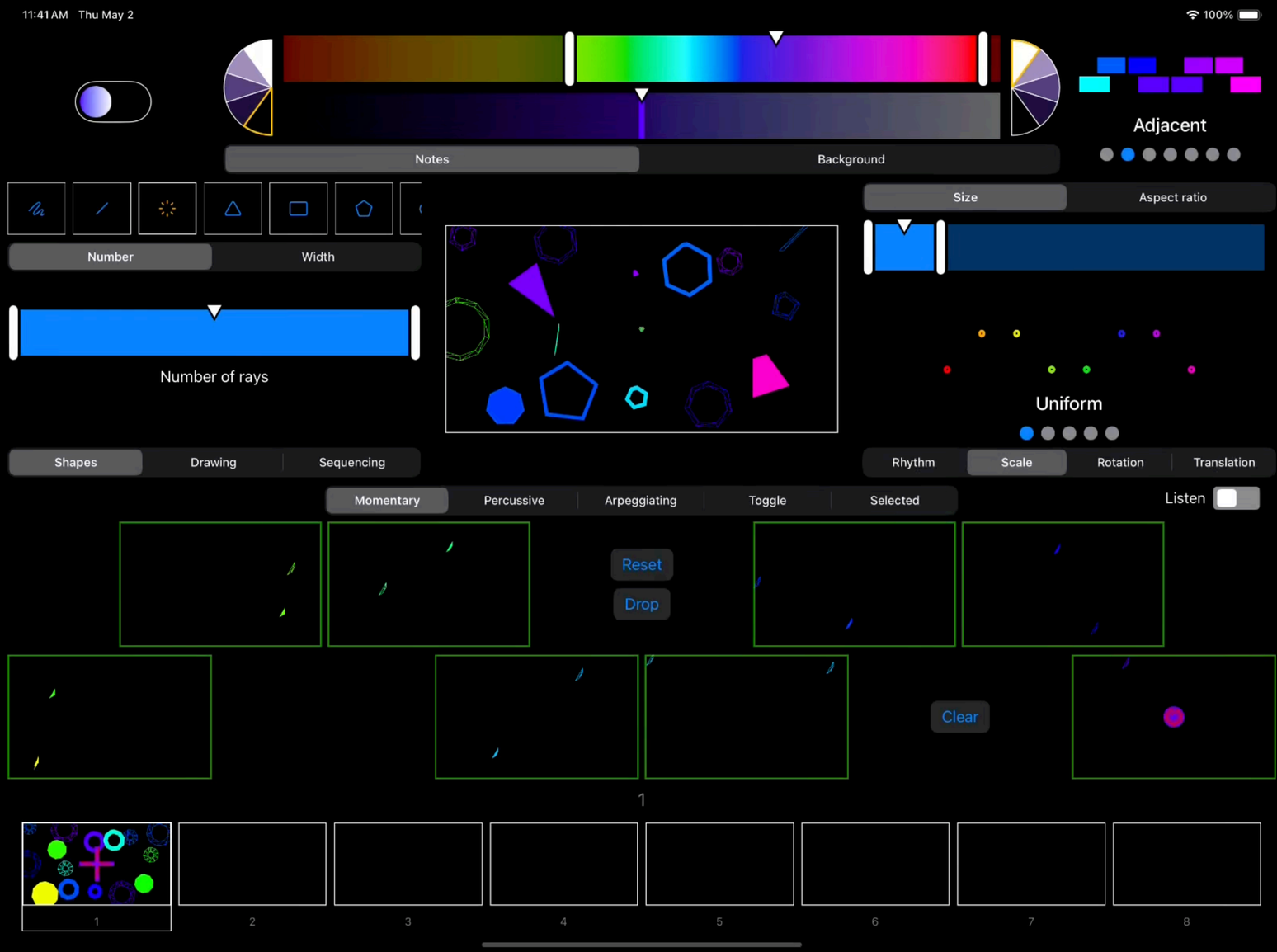
- ✓ Overview
- Demo
- Compose

Imager and the Ribbon Controller



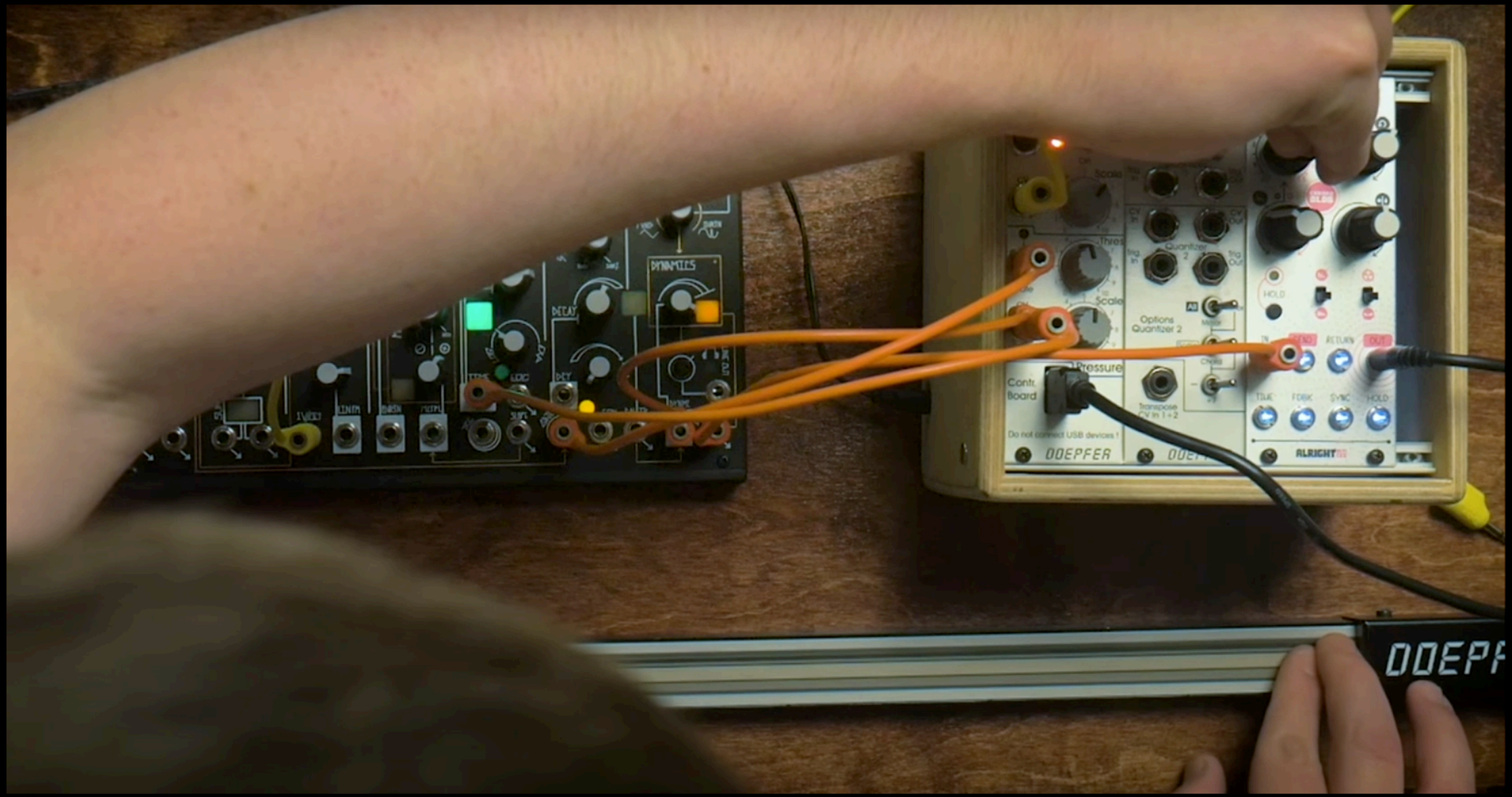
Imager and the Ribbon Controller

Fred



Donny

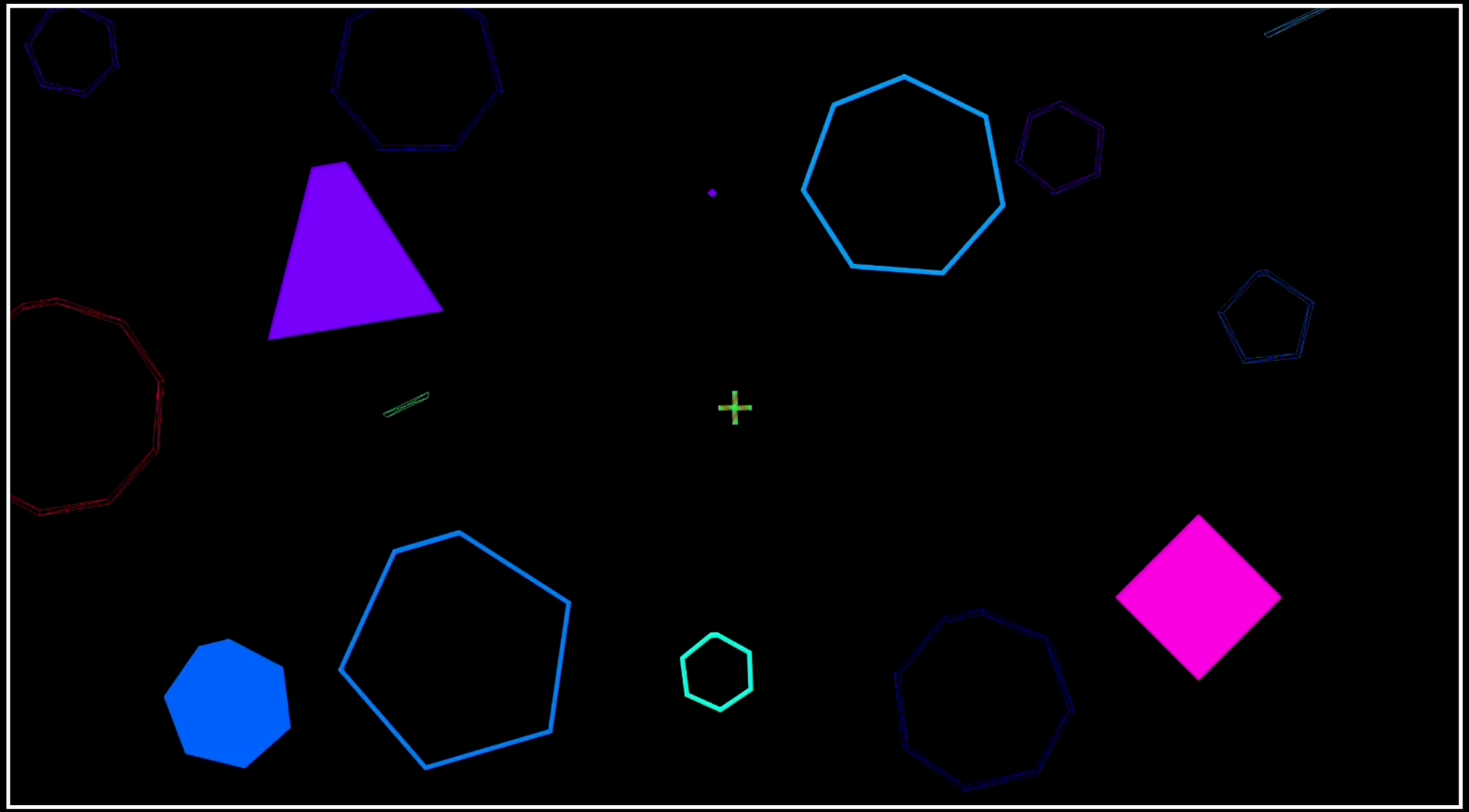

KURZWEIL
KURZWEIL


DYNAMICS
DECAY
Scale
Quantizer 2
Options
Quantizer 2
Contr.
Board
Pressure
Do not connect USB devices !
HOLD
RETURN
OUT
TRUE
FDBK
SYNC
HOLD
ALRIGHT!!
DOEPFER

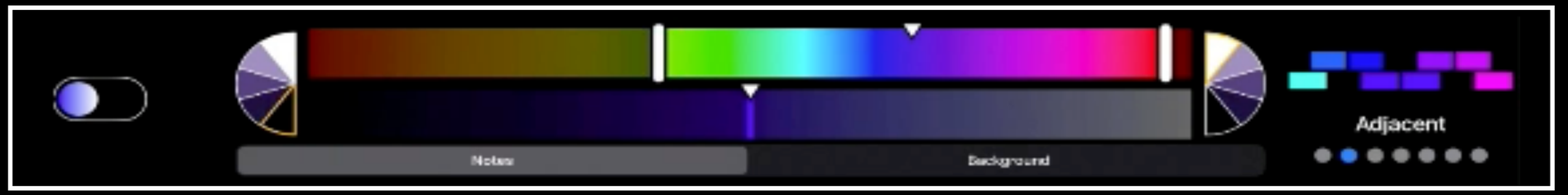
Arturia

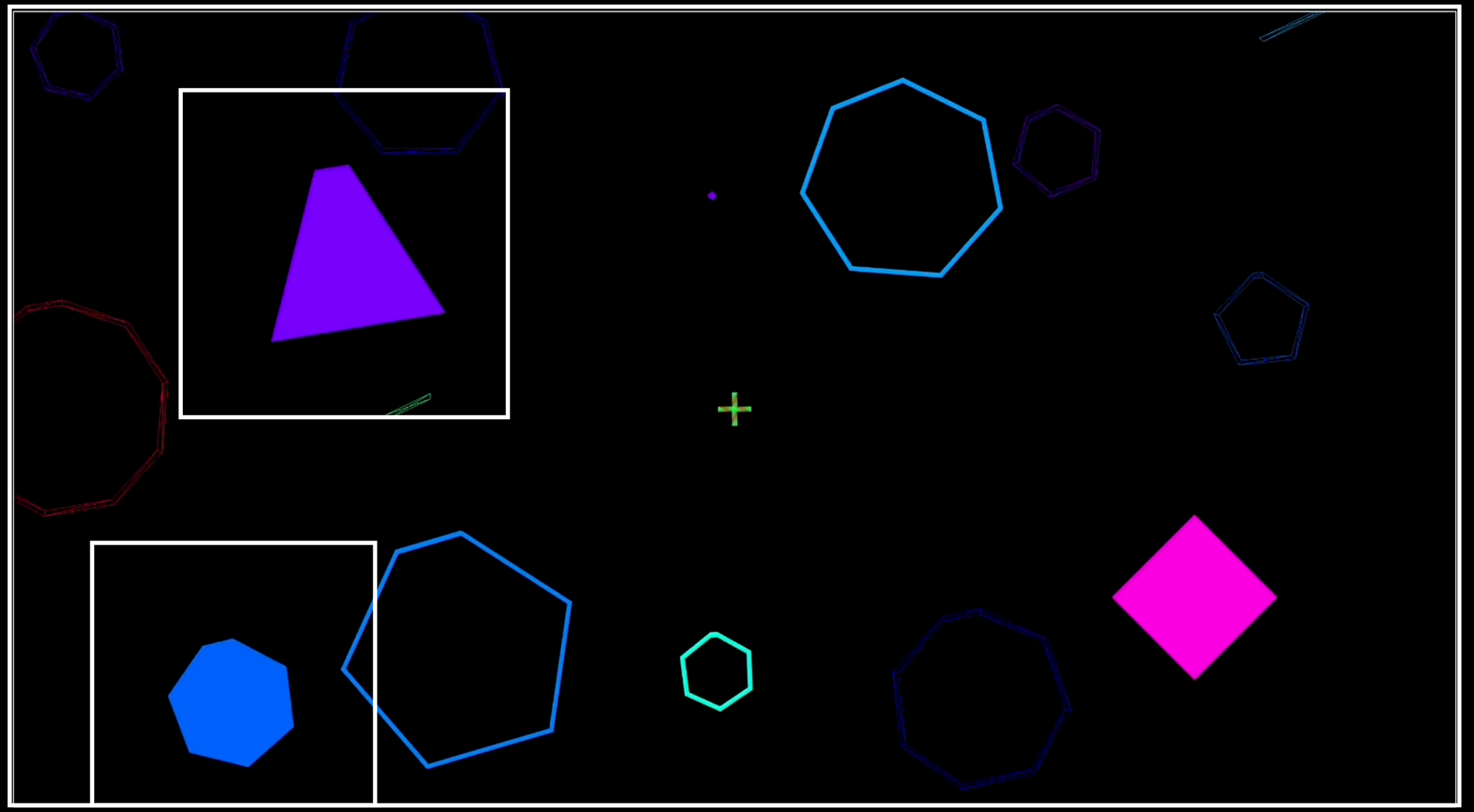
Origin Keyboard

Tutorials Series




Notes
Background
Adjacent
Number
Width
Number of rays
Shapes
Drawing
Sequencing
Size
Aspect ratio
Uniform
Rhythm
Scale
Rotation
Translation
Momentary
Percussive
Arpeggiating
Toggle
Selected
Listen
Reset
Drop
Clear
1
1
2
3
4
5
6
7
8


Notes
Background
Adjacent



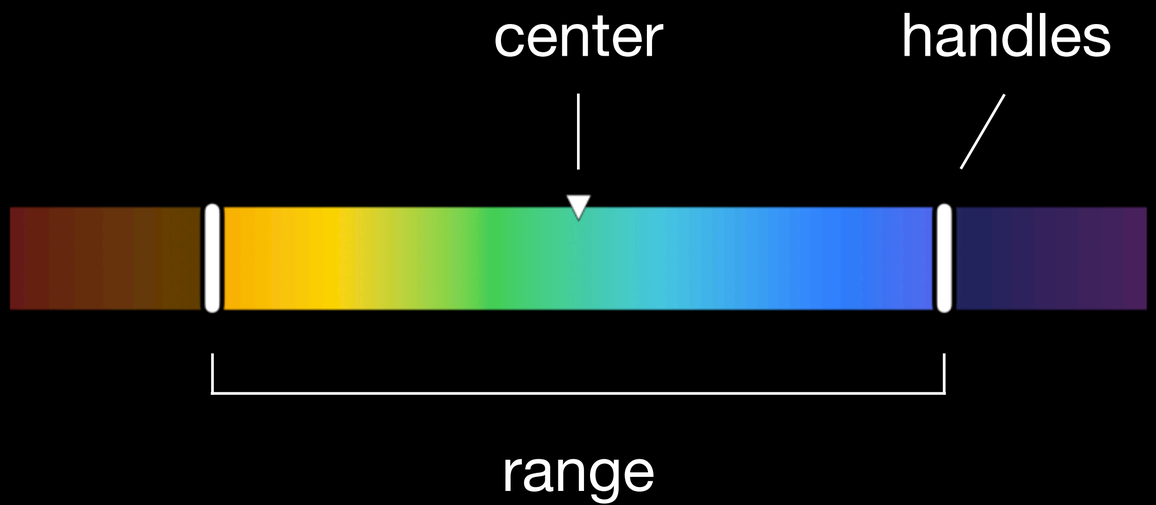
Building the ribbon

Join the beta!

Github

code deck

Terminology



Functions

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

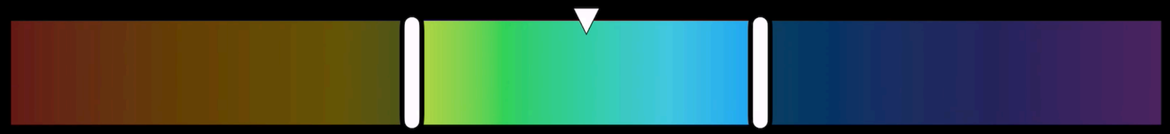
Functions

1. Move the center
2. Adjust the range

```
LongPressGesture
DragGesture
TapGesture(count: 2)
```

Functions

1. Move the center
2. Adjust the range
3. Open the door



`LongPressGesture`

Functions

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

3. Open the door

`LongPressGesture`

How To Use Them

Part II

`Gesture` Recognizers & Priorities

- Each type of gesture is a type of **recognizer**
- The last gesture added takes **priority***

*unless there's intervention

`Gesture` Recognizers & Priorities

Taps

```
SpatialTapGesture
```

```
TapGesture(count: 2)
```

Long Presses

```
LongPressGesture
```

```
LongPressGesture
```

Drags

```
DragGesture
```

```
DragGesture
```

```
DragGesture
```

How To Use Them
Part II

```
SpatialTapGesture                    var tap: some Gesture
TapGesture(count: 2)              var doubleTap: some Gesture

DragGesture                       var simpleDrag: some Gesture
DragGesture                       var anchorDrag: some Gesture
DragGesture                       var handleDrag: some Gesture

LongPressGesture                  var handlePress: some Gesture
LongPressGesture                  var openDoorPress: some Gesture
```

How To Use Them
Part II

View Modifiers

`.onTapGesture`

How To Use Them
Part II

View Modifiers

`.onTapGesture` `.onLongPressGesture`

View Modifiers

`.onTapGesture` `.onLongPressGesture`

`.gesture`

View Modifiers

`.onTapGesture` `.onLongPressGesture`

`.gesture`

`.highPriorityGesture` `.simultaneousGesture`

View Modifiers

`.gesture`

`.highPriorityGesture` `.simultaneousGesture`

```
func gesture<T>(
    _ gesture: T,
    including mask: GestureMask = .all
) -> some View where T : Gesture
```

View Modifiers

`.gesture`

`.highPriorityGesture` `.simultaneousGesture`

```
func gesture<T>(
    _ gesture: T,
    including mask: GestureMask = .all
) -> some View where T : Gesture
```

`.all`	all gestures in the hierarchy
`.gesture`	inline (disables subviews)
`.subviews`	subviews (disables inline)
`.none`	none

protocol Gesture

Grand Gestures

I Singles — `TapGesture` `LongPressGesture` `DragGesture` `RotateGesture` `MagnifyGesture`

II Doubles

III How To Use Them — `.onChanged` `.onEnded`

`.onTapGesture` `.onLongPressGesture`

`.gesture` `GestureMask` `.highPriorityGesture` `.simultaneousGesture`

How To Use Them
Part II

```
// ribbon
    .gesture(tap)
    .gesture(doubleTap)
    .gesture(simpleDrag)
    .gesture(anchorDrag)
    .gesture(handleDrag)
    .gesture(handlePress)
    .gesture(openDoorPress)
```

How To Use Them
Part II

```
// ribbon
    .gesture(tap)
    .gesture(doubleTap)
    .gesture(simpleDrag)
    .gesture(anchorDrag)
    .gesture(handleDrag)
    .gesture(handlePress)
    .gesture(openDoorPress)
```

Composition

Composition

Doubles

Composition

Doubles

Composition

1 & 2

1 → 2

1 || 2

Composition

1 & 2 `SimultaneousGesture`

1 → 2 `SequenceGesture`

1 || 2 `ExclusiveGesture`

protocol Gesture

SimultaneousGesture SequenceGesture

ExclusiveGesture

Value

first second

SimultaneousGesture SequenceGesture

ExclusiveGesture

```
struct Value {
    var first: First.Value
    var second: Second.Value
}
```

SimultaneousGesture SequenceGesture

ExclusiveGesture

```
SimultaneousGesture(LongPressGesture(), DragGesture())
            .onChanged { value in
                if let longPress = value.first {
                    print("Long press recognized?", longPress)
                } else if let drag = value.second {
                    print("Drag location", drag.location.x)
                }
            }
```

```
Long press recognized? true
Drag location 119.0
Drag location 120.0
Drag location 120.66665649414062
Drag location 121.33332824707031
```

```
SimultaneousGesture(LongPressGesture(),
                    SimultaneousGesture(MagnifyGesture(), RotateGesture()))
                    .onEnded { value in
                        if let longPress = value.first {
                            print("Pressed?", longPress)
                        }

                        if let simultaneous = value.second {
                            if let magnify = simultaneous.first {
                                print("Magnified by", magnify.magnification)
                            }
                            if let rotate = simultaneous.second {
                                print("Rotated", rotate.rotation.degrees)
                            }
                        }
                    }
```

```
Pressed? true
Magnified by 0.8814545615264194
Rotated -43.035197220611856
```

protocol Gesture

SimultaneousGesture SequenceGesture

ExclusiveGesture

```
enum Value {
   case first
   case second
}
```

SimultaneousGesture SequenceGesture

ExclusiveGesture

```
enum Value {
   case first   // the first hasn't ended
   case second  // the second has ended
}
```

SimultaneousGesture SequenceGesture

ExclusiveGesture

```
enum Value {
   case first  // the first has succeeded
   case second // the second has succeeded
}
```

protocol Gesture

Grand Gestures

I Singles

`TapGesture` `LongPressGesture` `DragGesture`
`RotateGesture` `MagnifyGesture`

II Doubles

`SimultaneousGesture` `SequenceGesture`
`ExclusiveGesture`

III How To Use Them

`.onChanged` `.onEnded`

`.onTapGesture` `.onLongPressGesture`

`.gesture` `GestureMask` `.highPriorityGesture` `.simultaneousGesture`

How To Use Them

Part III

Functions

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

3. Open the door

`LongPressGesture`

1. Move the center

SpatialTapGesture

DragGesture DragGesture

```
// ribbon
    .gesture(tap)
    .gesture(doubleTap)
    .gesture(simpleDrag)
    .gesture(anchorDrag)
    .gesture(handleDrag)
    .gesture(handlePress)
    .gesture(openDoorPress)
```

How To Use Them
Part III

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

```
// ribbon
    .gesture(tap)
    .gesture(simpleDrag)
    .gesture(anchorDrag)
```

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

```
var simpleDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            // apply the drag progress to the center
        }
}
```

1. Move the center

SpatialTapGesture

DragGesture DragGesture

```
var anchorDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            anchorPoints.forEach { anchor in
                // if the center is close enough,
                    stick it to the anchor point
            }
        }
}
```

1. Move the center

SpatialTapGesture

DragGesture DragGesture

```
var anchorDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            anchorPoints.forEach { anchor in
                // if the center is close enough,
                    stick it to the anchor point
            }
        }
        .simultaneously(with: simpleDrag)
}
```

1. Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

```
// ribbon
    .gesture(tap)
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
```

Functions

✔ Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

3. Open the door

`LongPressGesture`

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap)
    .gesture(doubleTap)
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress)
    .gesture(handleDrag)
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
var tap: some Gesture {
    SpatialTapGesture()
        .onEnded { tap in
            // move the center
        }
}
```

```
var doubleTap: some Gesture {
    TapGesture(count: 2)
        .onEnded { tap in
            // open and close the range
        }
}
```

`SimultaneousGesture`

1 & 2

`SequenceGesture`

1 → 2

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap)
    .gesture(doubleTap)
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress)
    .gesture(handleDrag)
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap.sequenced(before: doubleTap))
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress)
    .gesture(handleDrag)
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap.sequenced(before: doubleTap))
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress)
    .gesture(handleDrag)
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap.sequenced(before: doubleTap))
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress.sequenced(before: handleDrag))
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
var handlePress: some Gesture {
    LongPressGesture(minimumDuration: 0)
}

var handleDrag: some Gesture {
     DragGesture()
          .onChanged { value in
              adjustTheRange(fingerLocation: value.location.x)
          }
}
```

How To Use Them
Part III

```
@State private var selectedHandle: HandleType?

func handlePress(_ handle: HandleType) -> some Gesture {
     LongPressGesture(minimumDuration: 0)
         .onChanged { _ in
             selectedHandle = handle
         }
}

var handleDrag: some Gesture {
     DragGesture()
          .onChanged { value in
              adjustTheRange(fingerLocation: value.location.x)
          }
          .onEnded { _ in
              selectedHandle = nil
          }
 }
```

How To Use Them
Part III

```
@State private var selectedHandle: HandleType?

func handlePress(_ handle: HandleType) -> some Gesture {
    LongPressGesture(minimumDuration: 0)
        .onChanged { _ in
            selectedHandle = handle
        }
}

var handleDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            adjustTheRange(fingerLocation: value.location.x)
        }
        .onEnded { _ in
            selectedHandle = nil
        }
}
```

How To Use Them
Part III

```
@State private var selectedHandle: HandleType?

func handlePress(_ handle: HandleType) -> some Gesture {
    LongPressGesture(minimumDuration: 0)
        .onChanged { _ in
            selectedHandle = handle
        }
}

var handleDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            adjustTheRange(fingerLocation: value.location.x)
        }
        .onEnded { _ in
            selectedHandle = nil
        }
}
```

How To Use Them
Part III

```
@State private var selectedHandle: HandleType?

func handlePress(_ handle: HandleType) -> some Gesture {
    LongPressGesture(minimumDuration: 0)
        .onChanged { _ in
            selectedHandle = handle
        }
}

var handleDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            adjustTheRange(fingerLocation: value.location.x)
        }
        .onEnded { _ in
            selectedHandle = nil
        }
}
```

How To Use Them
Part III

```
@GestureState
```

How To Use Them
Part III

```
func updating<State>(
    _ state: GestureState<State>,
    body: @escaping (Self.Value, inout State, inout Transaction) -> Void
) -> GestureStateGesture<Self, State>
```

How To Use Them
Part III

```
func updating<State>(
    _ state: GestureState<State>,
    body: @escaping (Self.Value, inout State, inout Transaction) -> Void
) -> GestureStateGesture<Self, State>
```

```
@GestureState private var startLocation: CGPoint?

...

DragGesture()
    .onChanged { value in
        // utilize the startLocation
    }
    .updating($startLocation) { value, startLocation, _ in
        if startLocation == nil {
            startLocation = value.startLocation
        }
    }
```

Callbacks

`.updating`

updates the provided `GestureState` property as the gesture's value changes

should reset the state once gesture ends

↓

`@GestureState`

`.onChanged`

performs an action when the gesture's value changes

should not reset the state once gesture ends

↓

`@State`

`.onEnded`

performs an action when the gesture ends

Invoked to retrieve its final value and only when a gesture succeeds

How To Use Them
Part III

```
@State private var selectedHandle: HandleType?

func handlePress(_ handle: HandleType) -> some Gesture {
    LongPressGesture(minimumDuration: 0)
        .onChanged { _ in
            selectedHandle = handle
        }
}

var handleDrag: some Gesture {
    DragGesture()
        .onChanged { value in
            adjustTheRange(fingerLocation: value.location.x)
        }
        .onEnded { _ in
            selectedHandle = nil
        }
}
```

How To Use Them
Part III

```
@GestureState private var selectedHandle: HandleType?

var handlePress: some Gesture {
    LongPressGesture(minimumDuration: 0)
}

var handleDrag: some Gesture {
     DragGesture()
          .onChanged { value in
              adjustTheRange(fingerLocation: value.location.x)
          }
}
```

How To Use Them
Part III

```
@GestureState private var selectedHandle: HandleType?

var handlePress : some Gesture { ••• }
var handleDrag: some Gesture { ••• }


func rangeAdjustment(handle: HandleType) -> some Gesture {
    SequenceGesture(handlePress, handleDrag)
        .updating($selectedHandle) { (_, selectedHandle, _) in
                    selectedHandle = handle
        }
}
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
    .gesture(tap.sequenced(before: doubleTap))
    .gesture(anchorDrag.simultaneously(with: simpleDrag)
    .gesture(handlePress.sequenced(before: handleDrag))
```

2. Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

```
// ribbon
   .gesture(tap.sequenced(before: doubleTap))
   .gesture(anchorDrag.simultaneously(with: simpleDrag)

// leadingHandle
   .gesture(rangeAdjustment(handle: .leading))

// trailingHandle
   .gesture(rangeAdjustment(handle: .trailing))
```

Functions

✓ Move the center

`SpatialTapGesture`

`DragGesture` `DragGesture`

✓ Adjust the range

`TapGesture(count: 2)`

`LongPressGesture` `DragGesture`

3. Open the door

`LongPressGesture`

3. Open the door

`LongPressGesture`

```
// ribbon
   .gesture(anchorDrag.simultaneously(with: simpleDrag))
   .gesture(tap.sequenced(before: doubleTap))
   .gesture(openDoorPress)

// leadingHandle
   .gesture(rangeAdjustment(handle: .leading))

// trailingHandle
   .gesture(rangeAdjustment(handle: .trailing))
```

3. Open the door

`LongPressGesture`

```
// ribbon
    .gesture(anchorDrag.simultaneously(with: simpleDrag))
    .gesture(tap.sequenced(before: doubleTap))
    .gesture(openDoorPress)
```

3. Open the door

`LongPressGesture`

```
// ribbon
    .gesture(anchorDrag.simultaneously(with: simpleDrag))
    .gesture(openDoorPress)
    .gesture(tap.sequenced(before: doubleTap))
```

`ExclusiveGesture`

1 || 2

3. Open the door

`LongPressGesture`

```
// ribbon
    .gesture(anchorDrag.simultaneously(with: simpleDrag))
    .gesture(openDoorPress.exclusively(before:
                tap.sequenced(before: doubleTap)))
```

Functions

- ✓ Move the center
  - `SpatialTapGesture`
  - `DragGesture` `DragGesture`
- ✓ Adjust the range
  - `TapGesture(count: 2)`
  - `LongPressGesture` `DragGesture`
- ✓ Open the door
  - `LongPressGesture`

How To Use Them
Part III

```
// ribbon
   .gesture(anchorDrag.simultaneously(with: simpleDrag))
   .gesture(openDoorPress.exclusively(before:
            tap.sequenced(before: doubleTap)))

// leadingHandle
   .gesture(rangeAdjustment(handle: .leading))

// trailingHandle
   .gesture(rangeAdjustment(handle: .leading))
```

protocol Gesture

Grand Gestures

I Singles

TapGesture LongPressGesture DragGesture
RotateGesture MagnifyGesture

II Doubles

SimultaneousGesture SequenceGesture
ExclusiveGesture

III How To Use Them

.onChanged .onEnded .updating
.onTapGesture .onLongPressGesture @GestureState
GestureMask .gesture .highPriorityGesture .simultaneousGesture
.simultaneously(with:) .sequenced(before:) .exclusively(before:)

Summary

- Gestures can **stand alone** or be **composed** into a new gesture
- The view hierarchy has inherent **priorities** it considers, that can be altered with modifiers
- With `@GestureState`, we can read and write state meant to be reset when the gesture ends

What didn't we cover?

`AnyGesture` `.map`

`defersSystemGestures(on:)`

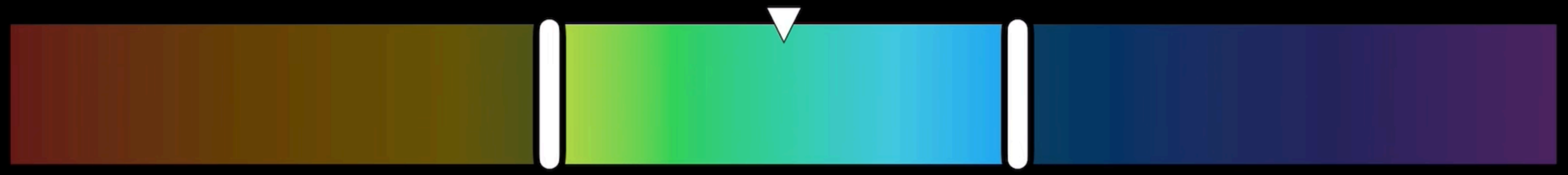
BONUS!

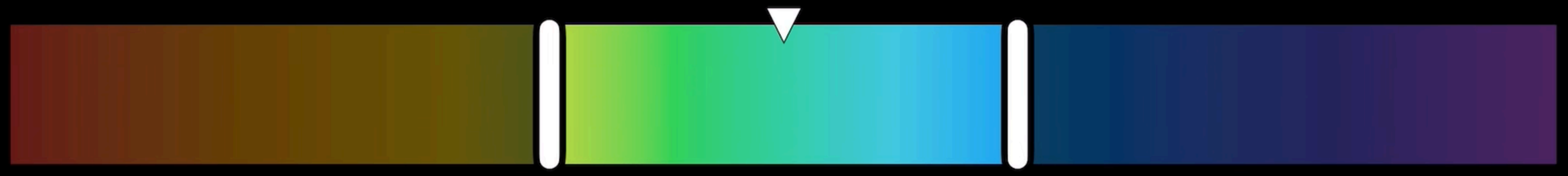
`LongPressReleaseDetectionGesture`

```
var openDoorPress: some Gesture {
    LongPressGesture(minimumDuration: 0.5)
        .onEnded { _ in
            // open the door
        }
}
```

```
struct LongPressReleaseDetectionGesture : Gesture {
    enum PressingState {
        case pressed, success, failure
    }

    let maxX: CGFloat
    let maxY: CGFloat
    let pressingState: (PressingState) -> Void
    let minimumDuration: CGFloat = 0.5

    public var body: some Gesture {
        SequenceGesture(
            LongPressGesture(minimumDuration: minimumDuration)
                .onEnded { _ in
                    pressingState(.pressed)
                },
            DragGesture(minimumDistance: 0)
                .onEnded { value in
                    pressingState((0...maxX).contains(value.location.x) &&
                                  (0...maxY).contains(value.location.y) ?
                            .success : .failure)
                }
        )
    }
}
```

```
var openDoorPress: some Gesture {
    LongPressReleaseDetectionGesture(
        maxX: width,
        maxY: height,
        pressingState: { pressingState in
            withAnimation {
                if pressingState == .pressed { // long press success
                    scaleEffect = 0.95
                } else { // release location determines success or failure
                    scaleEffect = 1

                    if pressingState == .success {
                        // open the door
                    }
                }
            }
        }
    )
}
```





Join the beta!

Github

code deck