Version: 2.6.0 – 2.12.0

Gesture states & events

Every gesture can be treated as "state machine". At any given time, each handler instance has an assigned state that can change when new touch events occur or can be forced to change by the touch system in certain circumstances.

A gesture can be in one of the six possible states:

- **UNDETERMINED**

  This is the initial state of each gesture recognizer and it goes into this state after it's done recognizing a gesture.

- **FAILED**

  A gesture recognizer received some touches but for some reason didn't recognize them. For example, if a finger travels more distance than a defined `maxDist` property allows, then the gesture won't become active but will fail instead. Afterwards, it's state will be reset to `UNDETERMINED`.

- **BEGAN**

  Gesture recognizer has started receiving touch stream but hasn't yet received enough data to either fail or activate.

- **CANCELLED**

  The gesture recognizer has received a signal (possibly new touches or a command from the touch system controller) resulting in the cancellation of a continuous gesture. The gesture's state will become `CANCELLED` until it is finally reset to the initial state, `UNDETERMINED`.

- **ACTIVE**

  Recognizer has recognized a gesture. It will become and stay in the `ACTIVE` state until the gesture finishes (e.g. when user lifts the finger) or gets cancelled by the touch system. Under normal circumstances the state will then turn into `END`. In the case that a gesture is cancelled by the touch system, its state would then become `CANCELLED`.

- **END**

  The gesture recognizer has received touches signalling the end of a gesture. Its state will become `END` until it is reset to `UNDETERMINED`.

State flows

The most typical flow of state is when a gesture picks up on an initial touch event, then recognizes it, then acknowledges its ending and resets itself back to the initial state.

The flow looks as follows (longer arrows represent that there are possibly more touch events received before the state changes):

`UNDETERMINED` -> `BEGAN` ------> `ACTIVE` ------> `END` -> `UNDETERMINED`

Another possible flow is when a handler receives touches that cause a recognition failure:

`UNDETERMINED` -> `BEGAN` ------> `FAILED` -> `UNDETERMINED`

At last, when a handler does properly recognize the gesture but then is interrupted by the touch system the gesture recognition is canceled and the flow looks as follows:

`UNDETERMINED` -> `BEGAN` ------> `ACTIVE` ------> `CANCELLED` -> `UNDETERMINED`

Events

There are three types of events in RNGH2: `StateChangeEvent`, `GestureEvent` and `PointerEvent`. The `StateChangeEvent` is send every time a gesture moves to a different state, while `GestureEvent` is send every time a gesture is updated. The first two carry a gesture-specific data and a `state` property, indicating the current state of the gesture. `StateChangeEvent` also carries a `oldState` property indicating the previous state of the gesture. `PointerEvent` carries information about raw touch events, like touching the screen or moving the finger. These events are handled internally before they are passed along to the correct callbacks:

`onBegin`

Is called when a gesture transitions to the `BEGAN` state.

`onStart`

Is called when a gesture transitions to the `ACTIVE` state.

`onEnd`

Is called when a gesture transitions from the `ACTIVE` state to the `END`, `FAILED`, or `CANCELLED` state. If the gesture transitions to the `END` state, the `success` argument is set to `true` otherwise it is set to `false`.

`onFinalize`

Is called when a gesture transitions to the `END`, `FAILED`, or `CANCELLED` state. If the gesture transitions to the `END` state, the `success` argument is set to `true` otherwise it is set to `false`. If the gesture transitions from the `ACTIVE` state, it will be called after `onEnd`.

`onUpdate`

Is called every time a gesture is updated while it is in the `ACTIVE` state.

`onPointerDown`

Is called when new pointers are placed on the screen. It may carry information about more than one pointer because the events are batched.

`onPointerMove`

Is called when pointers are moved on the screen. It may carry information about more than one pointer because the events are batched.

`onPointerUp`

Is called when pointers are lifted from the screen. It may carry information about more than one pointer because the events are batched.

`onPointerCancelled`

Is called when there will be no more information about this pointer. It may be called because the gesture has ended or was interrupted. It may carry information about more than one pointer because the events are batched.

`onPointerChange`

Is called before `onPointerDown`, `onPointerMove`, `onPointerUp` and `onPointerCancelled` with the same event, which may be useful in case you share logic between them. It may carry information about more than one pointer because the events are batched.