Version: 2.6.0 – 2.12.0

PinchGestureHandler

> DANGER
>
> Consider using the new gestures API instead. The old API is not actively supported and is not receiving the new features. Check out RNGH 2.0 section in Introduction for more information.

A continuous gesture handler that recognizes pinch gesture. It allows for tracking the distance between two fingers and use that information to scale or zoom your content. The handler activates when fingers are placed on the screen and change their position. Gesture callback can be used for continuous tracking of the pinch gesture. It provides information about velocity, anchor (focal) point of gesture and scale.

The distance between the fingers is reported as a scale factor. At the beginning of the gesture, the scale factor is 1.0. As the distance between the two fingers increases, the scale factor increases proportionally. Similarly, the scale factor decreases as the distance between the fingers decreases. Pinch gestures are used most commonly to change the size of objects or content onscreen. For example, map views use pinch gestures to change the zoom level of the map.

The handler is implemented using UIPinchGestureRecognizer on iOS and from scratch on Android.

Properties

Properties provided to `PinchGestureHandler` do not extend common set of properties from base handler class.

Event data

See set of event attributes from base handler class. Below is a list of gesture event attributes specific to `PinchGestureHandler`:

`scale`

The scale factor relative to the points of the two touches in screen coordinates.

`velocity`

Velocity of the pan gesture the current moment. The value is expressed in point units per second.

`focalX`

Position expressed in points along X axis of center anchor point of gesture

`focalY`

Position expressed in points along Y axis of center anchor point of gesture

Example

See the scale and rotation example from Gesture Handler Example App.

```
export class PinchableBox extends React.Component {
  _baseScale = new Animated.Value(1);
  _pinchScale = new Animated.Value(1);
  _scale = Animated.multiply(this._baseScale, this._pinchScale);
  _lastScale = 1;
  _onPinchGestureEvent = Animated.event(
    [{ nativeEvent: { scale: this._pinchScale } }],
    { useNativeDriver: USE_NATIVE_DRIVER }
  );

  _onPinchHandlerStateChange = (event) => {
    if (event.nativeEvent.oldState === State.ACTIVE) {
      this._lastScale *= event.nativeEvent.scale;
      this._baseScale.setValue(this._lastScale);
      this._pinchScale.setValue(1);
    }
  };

  render() {
    return (
      <PinchGestureHandler
        onGestureEvent={this._onPinchGestureEvent}
        onHandlerStateChange={this._onPinchHandlerStateChange}>
        <View style={styles.container} collapsable={false}>
          <Animated.Image
            style={[
```

```
                styles.pinchableImage,
                {
                  transform: [{ perspective: 200 }, { scale: this._scale }],
                },
              ]}
            />
          </View>
        </PinchGestureHandler>
      );
    }
  }
```