ชื่อ นายทธรรษ ธีรชูวิวัฒน์
รหัสนักศึกษา 63070065

Lab 11 : Gesture

จงพัฒนาแอปพลิเคชันที่แสดงการทำ Gesture และ Animation ซึ่งมีรายละเอียดดังนี้

- เมื่อเริ่มรันโปรแกรม โปรแกรมจะแสดงโลโก้ IT ที่บริเวณกลางหน้าจอ
- โปรแกรมสามารถจัดการการสัมผัสหน้าจอได้ดังนี้
  - เมื่อสัมผัสหน้าจอด้วยนิ้ว 1 นิ้ว (จุดสัมผัส 1 จุด) แล้วทำการลากนิ้ว (Drag)
    - โลโก้ IT จะเคลื่อนที่ตามนิ้วนั้น เมื่อปล่อยนิ้วจากหน้าจอ โลโก้ IT จะหยุดเคลื่อนที่ และแสดงที่ตำแหน่งล่าสุดที่ปล่อยนิ้ว
  - เมื่อสัมผัสหน้าจอด้วยนิ้ว 2 นิ้ว (จุดสัมผัส 2 จุด) แล้วทำการลากนิ้วทั้ง 2 นิ้วออกจากกัน ที่บริเวณมุมของรูปโลโก้ IT
    - โลโก้ IT จะขยายขนาดตามนิ้วทั้งสอง และเมื่อปล่อยนิ้วทั้งสองออกจากหน้าจอ โลโก้ IT จะย่อกลับมาที่ขนาดปกติ โดยมีลักษณะการย่อขนาดแบบ Spring
- รายละเอียดการทำงานของโปรแกรม แสดงใน onlearn.it.kmitl.ac.th

ข้อแนะนำในการปฏิบัติการ

1. นักศึกษาสามารถกำหนดสไตล์ชีทการแสดงส่วนประกอบต่างๆ ได้ (แต่ให้แสดงได้คล้ายกับหน้าจอตัวอย่าง)
2. นักศึกษาสามารถกำหนดโครงสร้างของโปรแกรมได้ตามที่ต้องการ
3. นักศึกษาสามารถเริ่มพัฒนาโปรแกรม โดยดูจากตัวอย่างโปรแกรมใน Slide การสอนได้ จะมีลักษณะใกล้เคียงกัน แต่ต้องปรับปรุงเล็กน้อย
4. นักศึกษาสามารถกำหนดค่าที่ใช้ในการทำ Animation (Animated.Value(), Animated.ValueXY()) ได้หลายค่า เพื่อใช้แสดงแอนิเมชันการย้ายโลโก้ และขยายขนาดของโลโก้ (Slide 21)
5. ข้อแนะนำในการตรวจสอบรูปแบบการสัมผัสหน้าจอ เช่น มีจุดสัมผัส 1 จุด หรือ 2 จุดเป็นต้น
   - ให้เขียนฟังก์ชัน onPanResponderMove ให้ทำงานตามต้องการ เมื่อมีการลากนิ้ว
   - รูปแบบ :

     onPanResponderMove: (evt, gestureState) => {

     ... //กำหนดการทำงานเมื่อมีการลากนิ้ว ... }
   - สามารถเรียกใช้ evt.nativeEvent.touches ซึ่งจะรีเทิร์นอะเรย์ของการสัมผัสหน้าจอทั้งหมด ณ ขณะนั้น (Slide 23)
   - สามารถหาขนาดของอะเรย์นั้น เพื่อหาจำนวนจุดสัมผัสหน้าจอ

- สามารถใช้จำนวนจุดสัมผัสบนหน้าจอ ในการกำหนดการทำงานของโปรแกรมในรูปแบบต่างๆ

6. ข้อแนะนำในการแสดงโลโก้ IT ตามการลากนิ้ว (จุดสัมผัส 1 จุด)
   - ดูตัวอย่างโปรแกรมใน Slide 19 ประกอบ
   - เมื่อคอมโพเนนต์ได้สิทธิ์เป็น Responder แล้ว (onPanResponderGrant ทำงาน)
     - ทำการกำหนด offset เริ่มต้นของ Animated.ValueXY() ให้เท่ากับตำแหน่งของคอมโพเนนต์ที่มีการปล่อยนิ้วครั้งล่าสุด โดยเรียกใช้ setOffset()
     - กำหนดตำแหน่ง x, y ของ Animated.ValueXY() ให้เป็น 0 โดยเรียกใช้ setValue()
   - เมื่อมีการลากนิ้วบนหน้าจอ ให้เขียนโปรแกรมใน onPanResponderMove เพิ่มเติม
     - จาก Slide 19 เราสามารถกำหนดค่า pan.x และ pan.y จากค่า dx และ dy ตามลำดับ ซึ่งถูกกำหนดจาก Animated.event()
     - แต่ก่อนหน้านี้ เรากำหนด onPanResponderMove ดังนี้

       onPanResponderMove: (evt, gestureState) => {
       ... //กำหนดการทำงานเมื่อมีการลากนิ้ว ... }
     - ในกรณีนี้ ค่า dx และ dy สามารถอ้างอิงได้จาก gestureState.dx และ gestureState.dy ตามลำดับ
     - เราสามารถกำหนดค่า pan ได้โดยทำการเรียกใช้ setValue()
   - เมื่อมีการปล่อยนิ้วจากหน้าจอ (onPanResponderRelease ทำงาน)
     - ทำการรวมค่า offset และ base value เพื่อนำไปใช้กำหนดค่า offset ของคอมโพเนนต์ในครั้งต่อไป (การทำงานในหัวข้อก่อนหน้า) โดยการเรียกใช้ flattenOffset()
   - ในส่วนของการแสดงรูปโลโก้
     - สามารถใช้ Animated.Image ได้
     - กำหนดตำแหน่งของการแสดงคอมโพเนนต์ (top, left) ด้วยการเรียก getLayout() ได้

7. ข้อแนะนำในการขยายขนาดโลโก้ IT ตามการลากนิ้วแยกออกจากกัน (จุดสัมผัส 2 จุด)
   - เมื่อมีการลากนิ้วทั้งสองนิ้วออกจากกัน ให้เขียนโปรแกรมใน onPanResponderMove เพิ่มเติม
     - แนวคิดหลัก : อาจคิดจากตำแหน่ง x, y ของจุดที่นิ้วสัมผัส (มี 2 จุดสัมผัส) นักศึกษาอาจหาระยะห่างระหว่างจุด 2 จุด แล้วนำไปใช้กำหนดขนาดของโลโก้ หรือคำนวณ scale ของรูปที่เปลี่ยนไปได้ โดยอาจเก็บค่านั้นใน Animated.Value ซึ่งนำไปใช้กำหนด style ในการแสดงผล

- กรณีที่ต้องการหาตำแหน่งจุดกดของแต่ละนิ้ว สามารถทำได้โดยการอ้างอิง location ในแกน x และ y ของแต่ละจุดสัมผัส เช่น
  - evt.nativeEvent.touches[0].locationX คือตำแหน่งการสัมผัสตามแนวแกน x ของจุดสัมผัสที่ 0 (รายละเอียด Slide 10)
  - evt.nativeEvent.touches[1].locationX คือตำแหน่งการสัมผัสตามแนวแกน x ของจุดสัมผัสที่ 1
- นักศึกษาสามารถใช้ค่าดังกล่าวในการกำหนดขนาดของรูปโลโก้ได้ตามต้องการ

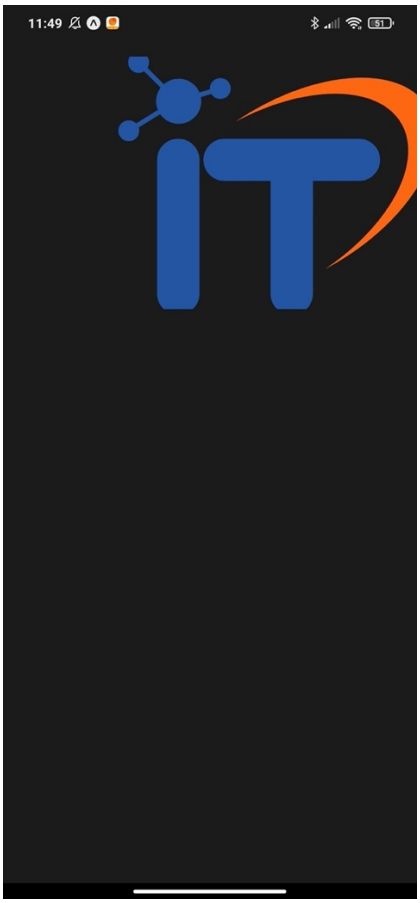
- เมื่อมีการปล่อยนิ้วทั้งสองจากหน้าจอ ให้เขียนโปรแกรมในฟังก์ชัน onPanResponderRelease เพิ่มเติม
  - ดูตัวอย่างโปรแกรมใน Slide ที่ 21 ประกอบ
  - ให้กำหนดรูปแบบการทำ Animation แบบ Spring เพิ่มเติม ให้ปรับขนาดของคอมโพเนนต์กลับมาเป็นปกติ
  - ในส่วนของการแสดงโลโก้ IT อาจต้องมีการกำหนดแอตทริบิวท์ transform ในการกำหนด style ของโลโก้ IT ให้ปรับขนาดตาม Animated.Value

```jsx
import { StatusBar } from "expo-status-bar";
import React, { useRef } from "react";
import { Animated, PanResponder, StyleSheet, View , LogBox} from "react-native";
LogBox.ignoreAllLogs();
export default function App() {
  const pan = useRef(new Animated.ValueXY()).current;
  const scale = useRef(new Animated.Value(1)).current;
  const panResponder = PanResponder.create({
    onStartShouldSetPanResponder: () => true,
    onPanResponderGrant: (evt, gestureState) => {
      pan.setOffset({ x: pan.x._value, y: pan.y._value });
      pan.setValue({ x: 0, y: 0 });
    },
    onPanResponderMove: (evt, gestureState) => {
      const touches = evt.nativeEvent.touches;
      if (touches.length == 1) {
        Animated.event([
          null,
          {
            dx: pan.x,
            dy: pan.y,
          },
        ])(evt, gestureState);
      } else if (touches.length >= 2) {
        Animated.spring(scale, {
          toValue: 3,
          friction: 3,
          useNativeDriver: false,
        }).start();
      }
    },
    onPanResponderRelease: () => {
      pan.flattenOffset();
      Animated.spring(scale, {
        toValue: 1,
        friction: 3,
        useNativeDriver: false,
      }).start();
    },
  });
  return (
    <View style={styles.container}>
      <Animated.Image
        {...panResponder.panHandlers}
        style={[pan.getLayout(), styles.box, { transform: [{ scale: scale }] }]}
        source={require("./assets/IT_Logo.png")}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: "#fff",
    alignItems: "center",
    justifyContent: "center",
  },
  box: {
    width: 100,
    height: 80,
    borderRadius: 4,
  },
});

```


11:48


11:49