**App.js**

import \* as React from 'react';

import { Text, View } from 'react-native';

import { NavigationContainer } from '@react-navigation/native';

import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import Cau1 from '../Lab\_04\_05\_06\_Animation/Exercise/Cau1';

const Tab = createBottomTabNavigator();

function MyTabs() {

  return (

    <Tab.Navigator>

      <Tab.Screen name="Cau1" component={Cau1}></Tab.Screen>

    </Tab.Navigator>

  );

}

export default function App() {

  return (

    <NavigationContainer>   <MyTabs/>  </NavigationContainer>

  );

}

**C1**

const App = () => {

  const fadeAnim = useRef(new Animated.Value(0)).current;

  const fadeIn = () => {

    Animated.timing(fadeAnim, {

      toValue: 1,

      duration: 3000,

    }).start();

  };

  const fadeOut = () => {

    Animated.timing(fadeAnim, {

      toValue: 0,

      duration: 3000,

    }).start();

  };

  return (

    <SafeAreaView style={styles.container}>

      <Animated.View

        style={[ styles.fadingContainer,{opacity: fadeAnim,}, ]}>

        <View style={styles.innerContainer}> Hello 1 </View>

      </Animated.View>

      <Animated.View

        style={[ styles.fadingContainer, { opacity: fadeAnim, },]}>

        <View style={styles.innerContainer}> Hello 2 </View>

      </Animated.View>

      <View style={styles.buttonRow}>

        <Button title="Show" onPress={fadeIn}/>

        <Button title="Hide" onPress={fadeOut}/>

      </View>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: 'center',

    justifyContent: 'center',

  },

  fadingContainer: {

    padding: 20,

    backgroundColor: 'powderblue',

  },

  innerContainer: {

    width: 100,

    height: 100,

    textAlign: 'center',

    justifyContent: 'center',

    fontSize: 30,

  },

  buttonRow: {

    flexBasis: 100,

    justifyContent: 'space-evenly',

    marginVertical: 16,

  },

});

export default App;

**C2**

import React, { useRef } from 'react';

import { Animated, Image, View, StyleSheet, Button, SafeAreaView } from 'react-native';

const App = () => {

  const translateYAnim = useRef(new Animated.Value(0)).current;

  const moveUp = () => {

    Animated.timing(translateYAnim, {

      toValue: -200,

      duration: 2000,

    }).start();

  };

  const stop = () => {

    translateYAnim.stopAnimation(value => {translateYAnim.setValue(value);});};

  const moveDown = () => {

    Animated.timing(translateYAnim,{toValue: 0,duration: 2000,}).start();};

  return (

    <SafeAreaView style={styles.container}>

      <Animated.View

        style={[ styles.movingContainer, { transform: [{ translateY: translateYAnim }],},]}>

        <Image

          style={{ width: 100, height: 100 }}

          source={require("../Pic/ball.png")}

        />

      </Animated.View>

      <View style={styles.buttonRow}>

        <Button title="Start" onPress={moveUp}/>

        <Button title="Stop" onPress={stop}/>

        <Button title="Restart" onPress={moveDown}/>

      </View>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: 'center',

    justifyContent: 'center',

  },

  fadingContainer: {

    padding: 20,

    backgroundColor: 'powderblue',

  },

  innerContainer: {

    width: 100,

    height: 100,

    textAlign: 'center',

    justifyContent: 'center',

    fontSize: 30,

  },

  buttonRow: {

    flexBasis: 100,

    justifyContent: 'space-evenly',

    marginVertical: 16,

  },

});

export default App;

**C3**

const App = () => {

  const wiggleAnim = useRef(new Animated.Value(0)).current;

  const spinAnim = useRef(new Animated.Value(0)).current;

  const fadeAnim = useRef(new Animated.Value(0)).current;

  const colorAnim = useRef(new Animated.Value(0)).current;

  const colorAnimation = () => {Animated.loop(

      Animated.sequence([Animated.timing(colorAnim, { toValue: 1, duration: 500 }),

        Animated.timing(colorAnim, { toValue: 0, duration: 500 }),]),).start();};

  const fadeAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(fadeAnim, { toValue: 1, duration: 3000 }),

        Animated.timing(fadeAnim, { toValue: 0, duration: 3000 }),

      ]),

    ).start();

  };

  const fadeIn = () => {

    Animated.timing(fadeAnim, {

      toValue: 1,

      duration: 3000,

    }).start();

  };

  const fadeOut = () => {

    Animated.timing(fadeAnim, {

      toValue: 0,

      duration: 3000,

    }).start();

  };

  const wiggleAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(wiggleAnim, { toValue: 10, duration: 1000 }),

        Animated.timing(wiggleAnim, { toValue: -10, duration: 1000 }),

        Animated.timing(wiggleAnim, { toValue: 0, duration: 1000 }),

      ]),

    ).start();

  };

  const spinAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(spinAnim, {

          toValue: 1,

          duration: 3000,

          useNativeDriver: true,

        })

      ]),

    ).start();

  };

  const spin = spinAnim.interpolate({ inputRange: [0, 1], outputRange: ['0deg', '360deg'],});

  const textColor = colorAnim.interpolate({inputRange: [0, 1],outputRange: ['blue', 'green'],});

  useEffect(() => {

    wiggleAnimation();

    spinAnimation();

    // fadeAnimation();

    colorAnimation();

  }, []);

  return (

    <SafeAreaView style={styles.container}>

      <Animated.View

        style={[

          styles.absoluteImage,

          {

            transform: [{ translateX: wiggleAnim }],

          },

        ]}>

        <Image

          style={{ width: 50, height: 50, position: 'absolute', top: -50, left: 80, }}

          source={require("../Pic/Bell.png")}

        />

      </Animated.View>

      <Animated.View

        style={[

          styles.absoluteImage,

          {

            transform: [{ translateX: spinAnim }, { rotate: spin }],

          },

        ]}>

        <Image

          style={{ width: 150, height: 150 }}

          source={require("../Pic/Sodobatquai.jpg")}

        />

      </Animated.View>

      <Animated.View

        style={[

          styles.fadingContainer,

          {

            opacity: fadeAnim,

            alignItems: 'center',

          },

        ]}>

        <Animated.View style={[{ color: textColor, fontSize: 30 }]}>

          CallMeMaybe

        </Animated.View>

        <Animated.Text style={[{ color: textColor, fontSize: 30 }]}>

          Lalala

        </Animated.Text>

      </Animated.View>

      <Animated.View>

        <Button title="Show" onPress={fadeIn} />

        <Button title="Login" onPress={fadeOut} />

        <Button title="Cancel" onPress={fadeOut} />

      </Animated.View>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {flex: 1,alignItems: 'center',justifyContent: 'center',},

  innerContainer: {width: 100,height: 100,textAlign: 'center',justifyContent: 'center',

    fontSize: 30,},

  sol:{position: 'absolute',top: 20,right: 20,},

});

export default App;

**C4**

const App = () => {

  const wiggleAnim = useRef(new Animated.Value(0)).current;

  const fadeAnim = useRef(new Animated.Value(0)).current;

  const fadeAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(fadeAnim, { toValue: 1, duration: 3000 }),

        Animated.timing(fadeAnim, { toValue: 0, duration: 3000 }),

      ]),

    ).start();

  };

  const wiggleAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(wiggleAnim, { toValue: 100, duration: 1000, useNativeDriver: false }),

        Animated.timing(wiggleAnim, { toValue: -100, duration: 1000, useNativeDriver: false }),

        Animated.timing(wiggleAnim, { toValue: 0, duration: 1000, useNativeDriver: false }),

      ]),

    ).start();

  };

  useEffect(() => {

    wiggleAnimation();

    fadeAnimation();

  }, []);

  return (

    <SafeAreaView style={styles.container}>

      <Animated.View

        style={[

          styles.absoluteImage,

          {

            transform: [{ translateY: wiggleAnim }],

            opacity: fadeAnim,

          },

        ]}>

        <Image

          style={{ width: 100, height: 100}}

          source={require("../Pic/bongbong.jpg")}

        />

      </Animated.View>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {flex: 1,alignItems: 'center',justifyContent: 'center',},

  innerContainer: {

    width: 100,

    height: 100,

    textAlign: 'center',

    justifyContent: 'center',

    fontSize: 30,

  },

  sol: {position: 'absolute',top: 20,right: 20,},});

export default App;

**C5**

const App = () => {

  const colorAnim = useRef(new Animated.Value(0)).current;

  const wiggleAnim = useRef(new Animated.Value(0)).current;

  const scaleAnim = useRef(new Animated.Value(0)).current;

  const scalePictureAnim = useRef(new Animated.Value(0)).current;

  const spinAnim = useRef(new Animated.Value(0)).current;

  const colorAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(colorAnim, { toValue: 1, duration: 1000, useNativeDriver: true }),

        Animated.timing(colorAnim, { toValue: 0, duration: 1000, useNativeDriver: true }),

      ]),

    ).start();

  };

  const wiggleAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(wiggleAnim, { toValue: -150, duration: 1000, useNativeDriver: true }),

        Animated.timing(wiggleAnim, { toValue: 150, duration: 1000, useNativeDriver: true }),

        Animated.timing(wiggleAnim, { toValue: 0, duration: 1000, useNativeDriver: true }),

      ]),

    ).start();

  };

  const scaleAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(scaleAnim, { toValue: 1, duration: 3000, useNativeDriver: true }),

        Animated.timing(scaleAnim, { toValue: -1, duration: 3000, useNativeDriver: true }),

        Animated.timing(scaleAnim, { toValue: 0, duration: 3000, useNativeDriver: true })

      ]),

    ).start();

  };

  const scalePictureAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(scalePictureAnim, { toValue: 1, duration: 3000, useNativeDriver: true}),

        Animated.timing(scalePictureAnim, { toValue: -1, duration: 3000, useNativeDriver:true}),

        Animated.timing(scalePictureAnim,{toValue: 0, duration: 3000, useNativeDriver: true})]),

    ).start();

  };

  const spinAnimation = () => {

    Animated.loop(

      Animated.sequence([

        Animated.timing(spinAnim, { toValue: 1, duration: 3000, useNativeDriver: true }),

        Animated.timing(spinAnim, { toValue: -1, duration: 3000, useNativeDriver: true }),

        Animated.timing(spinAnim, { toValue: 1, duration: 3000, useNativeDriver: true })]),

    ).start();

  };

  const scale = scaleAnim.interpolate({

    inputRange: [0, 1],

    outputRange: [30, 40],

  });

  const scalePicture = scalePictureAnim.interpolate({inputRange: [0, 1],outputRange: [1, 2],});

  const textColor = colorAnim.interpolate({inputRange: [0, 1],outputRange: ['blue', 'green'],});

  const spin = spinAnim.interpolate({

    inputRange: [0, 1],

    outputRange: ['0deg', '45deg'],

  });

  useEffect(() => {

    wiggleAnimation();

    scaleAnimation();

    scalePictureAnimation();

    colorAnimation();

    spinAnimation();

  }, []);

  return (

    <SafeAreaView style={styles.container}>

      <Animated.View

        style={[

          styles.absoluteImage,

          {

            transform: [{

              translateX: wiggleAnim,

            },

            {rotate: spin}],

          },

        ]}>

        <Image

          style={{ width: 100, height: 100 }}

          source={require("../Pic/Shipper.png")}

        />

      </Animated.View>

      <Animated.View

        style={[

          styles.fadingContainer,

          {

            alignItems: 'center',

          },

        ]}>

        <Animated.Text style={[{ color: textColor, fontSize: scale }]}>

          Shopee cái gì cũng có ...

        </Animated.Text>

      </Animated.View>

      <Animated.View style={{ flexDirection: "row", transform: [{scale: scalePicture}], }}>

        <Animated.View

          style={[

            styles.absoluteImage,

            // {transform: [{scale: scalePicture}],}

          ]}>

          <Image

            style={[{

              width: 100,

              height: 100,

            }]}

            source={require("../Pic/MyTom.png")}

          />

        </Animated.View>

        <Animated.View

          style={[

            styles.absoluteImage,

            // {transform: [{scale: scalePicture}],}

          ]}>

          <Image

            style={{ width: 100, height: 100 }}

            source={require("../Pic/Coca.png")}

          />

        </Animated.View>

        <Animated.View

          style={[

            styles.absoluteImage,

            // {transform: [{scale: scalePicture}],}

          ]}>

          <Image

            style={{ width: 100, height: 100 }}

            source={require("../Pic/Snack.png")}

          />

        </Animated.View>

      </Animated.View>

    </SafeAreaView>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: 'center',

    justifyContent: 'center',

  },

  innerContainer: {

    width: 100,

    height: 100,

    textAlign: 'center',

    justifyContent: 'center',

    fontSize: 30,

  },

  sol: {

    position: 'absolute',

    top: 20,

    right: 20,

  },

});

export default App;

**App.js (PanResponder)**

import \* as React from 'react';

import { Text, View } from 'react-native';

import { NavigationContainer } from '@react-navigation/native';

import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

import DragpanResponderView from './View/DragpanResponderView';

import DragAndSpringpanResponderView from './View/DragAndSpringpanResponderView';

const Tab = createBottomTabNavigator();

function MyTabs() {

  return (

    <Tab.Navigator>

      <Tab.Screen name="DragpanResponderView" component={DragpanResponderView}></Tab.Screen>

      <Tab.Screen name="DragAndSpringpanResponderView" component={DragAndSpringpanResponderView}></Tab.Screen>

    </Tab.Navigator>

  );

}

export default function App() {

  return (

    <NavigationContainer>

      <MyTabs/>

    </NavigationContainer>

  );

}

**DragpanResponderView.js**

import React, {useRef} from 'react';

import {Animated, View, StyleSheet, PanResponder, Text} from 'react-native';

const App = () => {

  const pan = useRef(new Animated.ValueXY()).current;

  const panResponder = useRef(

    PanResponder.create({

      onMoveShouldSetPanResponder: () => true,

      onPanResponderMove: (event, gesture) => {

        Animated.event(

            [

                null,

                {   dx: pan.x,

                    dy: pan.y

                }

            ],{ useNativeDriver: false }

        )(event, gesture);

      },

      onPanResponderRelease: (event, gesture) => {

        pan.extractOffset();

        console.log('Dragged to:', { x: gesture.moveX, y: gesture.moveY });

      },

    }),

  ).current;

  console.log(pan);

  return (

    <View style={styles.container}>

      <Text style={styles.titleText}>Drag this box!</Text>

      <Animated.View

        style={{

          transform: [{translateX: pan.x}, {translateY: pan.y}],

        }}

        {...panResponder.panHandlers}>

        <View style={styles.box} />

      </Animated.View>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: 'center',

    justifyContent: 'center',

  },

  titleText: {

    fontSize: 14,

    lineHeight: 24,

    fontWeight: 'bold',

  },

  box: {

    height: 150,

    width: 150,

    backgroundColor: 'lightblue',

    borderRadius: 5,

  },

});

export default App;

**DragAndSpringpanResponderView.js**

import React, {useRef} from 'react';

import {Animated, View, StyleSheet, PanResponder, Text} from 'react-native';

const App = () => {

  const pan = useRef(new Animated.ValueXY()).current;

  const panResponder = useRef(

    PanResponder.create({

      onMoveShouldSetPanResponder: () => true,

      onPanResponderMove: (event, gesture) => {

        Animated.event(

            [

                null,

                {   dx: pan.x,

                    dy: pan.y

                }

            ],{ useNativeDriver: false }

        )(event, gesture);

      },

      onPanResponderRelease: (event, gesture) => {

        pan.extractOffset();

        console.log('Dragged to:', { x: gesture.moveX, y: gesture.moveY });

      },

    }),

  ).current;

  console.log(pan);

  return (

    <View style={styles.container}>

      <Text style={styles.titleText}>Drag this box!</Text>

      <Animated.View

        style={{

          transform: [{translateX: pan.x}, {translateY: pan.y}],

        }}

        {...panResponder.panHandlers}>

        <View style={styles.box} />

      </Animated.View>

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    alignItems: 'center',

    justifyContent: 'center',

  },

  titleText: {

    fontSize: 14,

    lineHeight: 24,

    fontWeight: 'bold',

  },

  box: {

    height: 150,

    width: 150,

    backgroundColor: 'lightblue',

    borderRadius: 5,

  },

});

export default App;

**ClonePanResponder**

import React, {useRef} from 'react';

import {Animated, View, StyleSheet, PanResponder, Text, Dimensions} from 'react-native';

const App = () => {

  const windowWidth = Dimensions.get('window').width;

  let images = [

    'https://images.unsplash.com/photo-1596854372745-0906a0593bca?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8Mnx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D',

    'https://images.unsplash.com/photo-1584197176155-304c9a3c03ce?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NHx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D',

    'https://images.unsplash.com/photo-1584197176155-304c9a3c03ce?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NHx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D'

  ];

  const pan = useRef(new Animated.ValueXY()).current;

  const pan\_width = useRef(new Animated.Value(300)).current;

  const panResponder = useRef(

    PanResponder.create({

      onStartShouldSetPanResponder: (evt, gestureState) => true,

      onStartShouldSetPanResponderCapture: (evt, gestureState) => true,

      onMoveShouldSetPanResponder: (evt, gestureState) => true,

      onMoveShouldSetPanResponderCapture: (evt, gestureState) => true,

      onPanResponderMove: Animated.event(

        [

          null,

          {dx: pan.x, dy: pan.y}

        ]

      ),

      onPanResponderRelease: (evt, gestureState) => {

        console.log(gestureState.moveX + " " + gestureState.moveY + gestureState.dx + " " + gestureState.dy);

        if (gestureState.moveX <= windowWidth \* 3 / 4) {

          Animated.spring(pan, {

            toValue: {x: 0, y: 0},

            friction: 1,

            tension: 100,

            useNativeDriver: true

          }).start();

        } else {

          // Animated.spring(pan, {

          //   toValue: {x: windowWidth, y: 0},

          //   friction: 2,

          //   tension: 140,

          //   useNativeDriver: true

          // }).start();

          Animated.spring(pan\_width, {

            toValue: 0,

            friction: 2,

            tension: 140,

            useNativeDriver: true

          }).start();

        }

      }

    }),

  ).current;

  return (

    <View style={styles.container}>

      <Animated.Image

        style={{

          width: pan\_width,

          aspectRatio: 3 / 4,

          transform: [

            {translateX: pan.x}, {translateY: pan.y}

          ],

        }}

        {...panResponder.panHandlers}

        source={{uri:'https://images.unsplash.com/photo-1584197176155-304c9a3c03ce?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NHx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D'}}

      >

      </Animated.Image>

      {/\* <Animated.Image

        style={{

          width: 200,

          aspectRatio: 3 / 4,

          transform: [

            {translateX: pan.x}, {translateY: pan.y}

          ],

        }}

        {...panResponder.panHandlers}

        source={{uri:'https://images.unsplash.com/photo-1584197176155-304c9a3c03ce?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NHx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D'}}

      >

      </Animated.Image>

      <Animated.Image

        style={{

          width: 200,

          aspectRatio: 3 / 4,

          transform: [

            {translateX: pan.x}, {translateY: pan.y}

          ],

        }}

        {...panResponder.panHandlers}

        source={{uri:'https://images.unsplash.com/photo-1584197176155-304c9a3c03ce?w=800&auto=format&fit=crop&q=60&ixlib=rb-4.0.3&ixid=M3wxMjA3fDB8MHxzZWFyY2h8NHx8Y2F0JTIwcG9ydHJhaXR8ZW58MHx8MHx8fDA%3D'}}

      >

      </Animated.Image> \*/}

    </View>

  );

};

const styles = StyleSheet.create({

  container: {

    flex: 1,

    flexDirection: 'row',

    alignItems: 'center',

    justifyContent: 'center',

    backgroundColor: '#121212'

  },

  titleText: {

    fontSize: 14,

    lineHeight: 24,

    fontWeight: 'bold',

    color: 'white',

    userSelect: false

  },

  box: {

    height: 150,

    width: 150,

    backgroundColor: 'blue',

    borderRadius: 5,

  },

});

export default App;