# Hoe maak je fruit ninja in React Native?

In deze instructie leer je hoe je Fruit Ninja maakt in React Native. Voor de instructie nemen we aan dat je al een Expo init gedaan hebt en dus de standaard bestanden al tot beschikking hebt.

Stap 1, Assets.

Maak een nieuwe map aan genaamd “Assets” hierin zetten we de afbeeldingen die we gaan gebruiken in de app. Zorg dat de afbeeldingen genummerd zijn.

Stap 2, Imports.

Om het programma werkend te krijgen hebben we een paar imports nodig vanuit React.   
Kopieer en plak de volgende code bovenaan in het bestand. Overschrijf dus de oude imports

import React, { useRef, useEffect } from 'react';

import { Text, SafeAreaView, ImageBackground, Animated, Dimensions, Easing, Image, TouchableOpacity } from 'react-native';

import tw from 'tailwind-react-native-classnames';

Stap 3, Nodige variabele.

Voor het programma hebben we een aantal variabele nodig.  
Kopieer en plak de volgende code in de App() function.

  // Use state to store some values

  const [points, setPoints] = React.useState(0);

  const [isPlaying, setIsPlaying] = React.useState(true);

  const [height, setHeight] = React.useState(Dimensions.get('window').height \* 0.25);

Stap 4, De array.

Kopieer en plak deze array onder de code die je zojuist gekopieerd hebt.

  // Images that are used in the game

  const [image, setImage] = React.useState(

    require('./assets/1.png'),

    require('./assets/2.png'),

    require('./assets/3.png'),

    require('./assets/4.png'),

    require('./assets/5.png'),

    require('./assets/6.png'),

  );

We maken hier een const die een array aan images heeft.

Stap 5, random afbeelding selectie.

Kopieer en plak de volgende code onder de array

  // Function that changes the image to a random image onclick

  generateImage = () => {

    const randomImage = Math.floor(Math.random() \* 6) + 1;

    setImage(randomImage);

  }

We maken hier een const die een random nummer genereert van 1 t/m 6. Daarna wordt dit nummer gebruikt om de juiste image uit de array te halen.

Stap 6, De animatie.

Voor fruit ninja moeten we wat animatie hebben dat zorgt dat het fruit naar boven ‘gegooid’ wordt. Het moet dus omhoog en weer naar beneden ‘vallen’.

Kopieer en plak deze code onder de image selectie.

  // The values of the animations

  const moveValue = useRef(new Animated.ValueXY({

    x: 0,

    y: Dimensions.get('window').height + 70

  })).current;

  const spinValue = new Animated.Value(0);

  // Config for the spin animation

  const spin = spinValue.interpolate({

    inputRange: [0, 1],

    outputRange: ['0deg', '360deg']

  })

We hebben hier een stuk code dat de hoogte van het scherm vind en vanuit dat punt de animatie begint. Voor een beter effect hebben we ook een draai animatie voor het fruit, dat is de spin const die we hier hebben.

Kopieer en plak de volgende code in het bestand.

// Function that run the animation when the game starts

  function cycleAnimation() {

    // Loop the animation

    Animated.sequence([

      Animated.timing(moveValue, {

        toValue: { x: getRandomWidth(), y: Dimensions.get('window').height + 70 },

        duration: 1000,

        ease: Easing.ease,

        delay: 0,

        useNativeDriver: false

      }),

      Animated.timing(moveValue, {

        toValue: { x: getRandomWidth(), y: height },

        duration: 1000,

        ease: Easing.ease,

        delay: 0,

        useNativeDriver: false

      })

    ]).start(({ finished }) => {

      // Check if the animation is finished and the game is still running

      if (finished && isPlaying) {

        // Restart the animation

        cycleAnimation();

        setTimeout(() => {

          // Generate a new image

          generateImage();

        }, 1000);

      }

      else {

        // Reset the values

        moveValue.setValue({ x: 0, y: Dimensions.get('window').height + 70 });

      }

    });

  }

  // Fruit rotate function

  function cycleRotation() {

    // Loop the rotation

    Animated.sequence([

      Animated.timing(

        spinValue,

        {

          toValue: 1,

          duration: 3000,

          easing: Easing.linear,

          useNativeDriver: false

        }

      ),

      Animated.timing(

        spinValue,

        {

          toValue: 0,

          duration: 1,

          easing: Easing.linear,

          useNativeDriver: false

        }

      )

    ]).start(({ finished }) => {

      if (finished) {

        // Restart the rotation

        cycleRotation();

      }

    });

  }

Deze code zorgt dat het fruit omhoog en omlaag gaat op een vrijwel random manier.

Stap 7, Spawn locatie.

We hebben al de hoogte van de start aangegeven in de animatie hierboven maar het fruit moet ook op een random breedte punt gezet worden zodat het niet altijd vanuit hetzelfde punt omhoog gaat.

Kopieer en plak de volgende code onder de animatie

  // Generate a random width

  function getRandomWidth() {

    return Math.floor(Math.random() \* (Dimensions.get('window').width - 70));

  }

De code is vrij simpel, het genereert een random nummer gebaseerd op de breedte van het scherm zonder dat het fruit in een rand gespawnt wordt.

Stap 8, Gameplay.

Voor de gameplay moeten we zorgen dat als je op een fruit klikt/tikt dat het verdwijnt en een nieuwe verschijnt onderaan het scherm.

ClickedObject = () => {

    // Stop the game

    setIsPlaying(false);

    // Wait 500ms to restart the game

    setTimeout(() => {

      // Generate a new image

      generateImage();

      // Start the game again

      setIsPlaying(true);

    }, 500);

  }

  useEffect(() => {

    if (isPlaying) {

      // Restart the animation

      cycleAnimation();

    }

    else {

      // Reset the values and stop the animation

      moveValue.stopAnimation();

      moveValue.setValue({ x: 0, y: Dimensions.get('window').height + 70 });

    }

    cycleRotation();

  }, [isPlaying, cycleRotation, cycleAnimation]);

Stap 9, De view.

Als laatste maken we de view zelf. Om te beginnen maken we de achtergrond en de regio waarin de animatie zich afspeelt.

Kopieer en plak de volgende code return.

    <SafeAreaView style={tw.style('flex-1 bg-black')}>

      <ImageBackground source={require('./assets/background.jpg')} resizeMode="cover" style={tw.style('opacity-90 w-full h-full')}>

        <Text style={tw.style('text-white text-xl font-bold mt-10 ml-5')}>Score: {points}</Text>

        <Animated.View

          style={[

            {

              transform: [{ rotate: spin }],

              marginLeft: 10,

              width: 100,

              height: 100,

            },

            moveValue.getLayout(),

          ]}

        >

Nu moeten we zorgen dat de fruiten aanraakbaar zijn en als ze aangeraakt worden de code die we nu al hebben activeren zodat het spel ook daadwerkelijk speelbaar is.

Kopieer en plak de volgende code onder de code hieronder, dus nog in de return.

          <TouchableOpacity onPress={() => {

            ClickedObject()

            // Adds 1 to the score

            setPoints(points + 1)

          }}>

Als laatste moeten we zorgen dat alles netjes afgesloten is.

Kopieer en plak de volgende code aan het eind van het bestand.

            <Image source={image} style={tw.style('')} />

          </TouchableOpacity>

        </Animated.View>

      </ImageBackground>

    </SafeAreaView>

  );

Gefeliciteerd, je hebt nu een werkende Fruit Ninja project gemaakt in React Native! Veel plezier met het spel