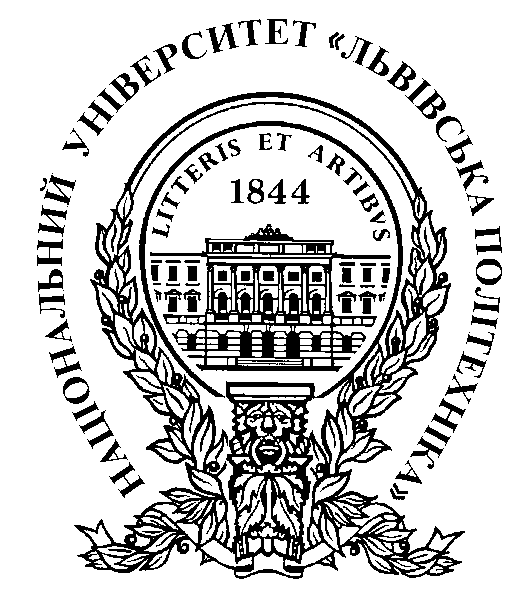
Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



Лабораторна робота №6

з дисципліни

«Програмування мобільних пристроїв»

Виконала:

ст. групи КН-311

Стефанишин Віта

Прийняв:

Григорович В. Г.

Львів 2019

**Лабораторна робота №6**

**Тема:** Створення крос-платформного мобільного додатку.

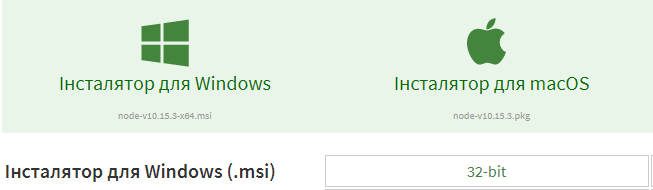
**Мета:** створити власний крос-платформний мобільний додаток використовуючи на вибір технології Xamarin, Unity3D, React Native.

**Завдання:**

1. Встановити на вибір середовище Xamarin, Unity3D, React Native.
2. Написати власний додаток або скомпілювати один з додатків-прикладів.
3. Виконати на мобільному телефоні.
4. Проаналізувати, з яких компонентів складається додаток, як він виконується.
5. Написати звіт.

**Хід роботи**

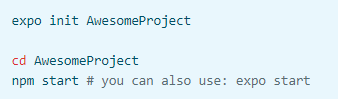
* 1. Інсталюю Node.JS

****

Встановивши Node 10+ , я можу використовувати npm для інсталяції утиліти командного рядка CLI Expo:

**

Потім виконую команди, щоб створити новий проект React Native під назвою "AwesomeProject":



Сервер розробки розпочато.

* 1. Встановлюю клієнтську програму Expo на телефон iOS і підключаюся до тієї ж бездротової мережі, що й комп'ютер. На iOS використовую програму Expo для сканування QR-коду з мого терміналу, щоб відкрити проект.



Expo CLI дозволяє легко запускати додаток React Native на фізичному пристрої без налаштування середовища розробки.

Після встановлення цих програм можна запустити програму на віртуальному пристрої iOS, запустивши її:



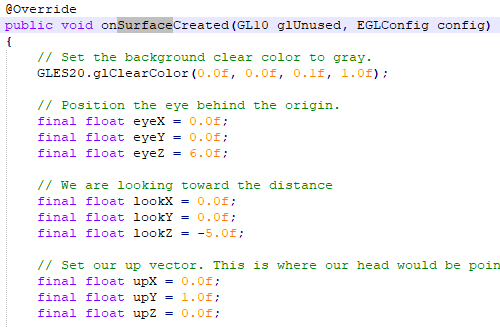
**Класи для відображення графіки у моєму проекті**

**Створення об'єкта GLSurfaceView**

Реалізація GLSurfaceView.Renderer класу або засобу візуалізації в програмі/ Цей клас контролює те, що витягується з [GLSurfaceView](https://developer.android.com/reference/android/opengl/GLSurfaceView.html)і з чим він пов'язаний. Існує три методи в засобі візуалізації, які викликаються системою Android для того, щоб з'ясувати, як малювати на [GLSurfaceView](https://developer.android.com/reference/android/opengl/GLSurfaceView.html):

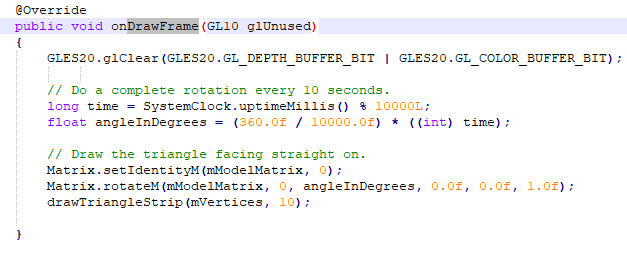
* [onSurfaceCreated()](https://developer.android.com/reference/android/opengl/GLSurfaceView.Renderer.html#onSurfaceCreated(javax.microedition.khronos.opengles.GL10,%20javax.microedition.khronos.egl.EGLConfig)) - Викликається один раз для налаштування середовища OpenGL ES перегляду.

У **onSurfaceCreated** ставимо чорним дефолтний колір очищення. Потім, методами класу ShaderUtils створюємо шейдери, отримуємо їх id: vertexShaderId (верховий) і fragmentShaderId (Фрагментний), створюємо з них програму programId, і методом **glUseProgram** повідомляємо системі, що цю програму треба використовувати для побудови зображення. Далі, методом bindData передаємо дані в шейдери.

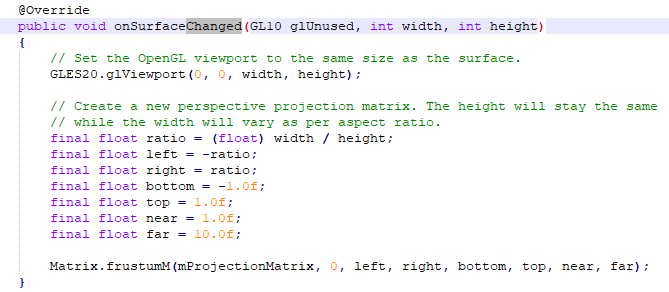


* [onDrawFrame()](https://developer.android.com/reference/android/opengl/GLSurfaceView.Renderer.html#onDrawFrame(javax.microedition.khronos.opengles.GL10)) - Викликається для кожного перемальовування зображення.

Ось основна реалізація засобу візуалізації, яка не робить нічого, крім того, щоб намалювати чорний фон у [GLSurfaceView](https://developer.android.com/reference/android/opengl/GLSurfaceView.html):



* Методом **onSurfaceChanged**  задаємо область малювання на всю поверхню surface-компонента.



* 1. Створюю *react-native init NewProject*

Мій *App.js* файл:

import React, {Component} from 'react';

import {Platform, StyleSheet, Text, View, Button, TouchableOpacity} from 'react-native';

type Props = {}; /// (1)

export default class App extends Component<Props> {

constructor(props)

{

super(props)

this.state = { count: 0 } /// (2)

}

onPress = () => { /// (3)

this.setState({ /// (4)

count: this.state.count+1

})

}

render() { /// (5)

return ( /// (6)

<View style={styles.container}>

<TouchableOpacity style={styles.button} onPress={this.onPress} >

<Text>Touch Here</Text>

</TouchableOpacity>

<Button style={styles.button} onPress={this.onPress} title="Button"/>

<Text style={styles.welcome}>Welcome to React Native!</Text>

<Text style={styles.text}>Count = {this.state.count}</Text>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1, /// (7)

justifyContent: 'center',

alignItems: 'center',

backgroundColor: '#F5FCFF',

},

welcome: {

fontSize: 20,

textAlign: 'center',

margin: 10,

},

text: {

textAlign: 'center',

color: '#333333',

marginBottom: 5,

},

button: {

alignItems: 'center',

backgroundColor: '#DDDDDD',

padding: 20,

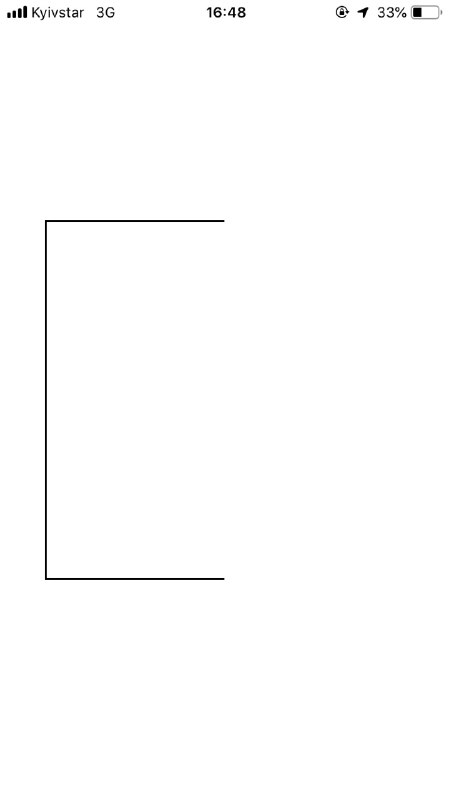
margin: 10,

},

});

Тестування додатку:

Запущений графічний додаток з першою С-буквою прізвища(Стефанишин) на iOS.



**Висновки:** в результаті виконання цієї лабораторної роботи було створено власний крос-платформний мобільний додаток використовуючи технологію **React Native**.