КАФЕДРА

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

должность, уч. степень, звание	подпись, дата	инициалы, фамилия

Отчет о лабораторной работе №6

Организация доступа к аппаратным компонентам мобильного устройства

По дисциплине: Программирование мобильных устройств

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №			
		подпись, дата	инициалы, фамилия

Санкт-Петербург 2024

Задание:
Дополнить приложение, разработанное в рамках одной из предыдущих лабораторных работ, добавив в него работу с аппаратным обеспечение мобильного устройства (камера, светодиод подсветки, датчики и т.п.)

Цель работы:
Получение навыков использования в разрабатываемых мобильных приложения доступа к компонентам аппаратного обеспечения мобильных устройств.

Выполнение задания:

В рамках последних изменений в мобильном приложении был проведен ряд модификаций и добавлений новых функций. Эти обновления включают в себя добавление нового экрана настроек, интеграцию функции мигания фонарика при выигрыше, а также доработку интерфейса и логики приложения. Ниже представлен обзор ключевых модификаций.

1. Новый экран настроек

- **Описание**: В приложение был добавлен новый экран настроек, расширяющий функциональность и улучшающий пользовательский опыт. Этот экран предназначен для настройки персональных данных пользователя и управления различными параметрами приложения.
- **Функции экрана настроек**:
    - **Выбор и установка аватара**: Пользователи могут выбрать изображение из галереи устройства или сделать новое фото с камеры для установки в качестве аватара.
    - **Ввод и редактирование персональных данных**: Предусмотрена возможность ввода и изменения имени и фамилии пользователя.
- **Техническая реализация**: Для выбора изображения аватара использован **UIImagePickerController**, позволяющий доступ к камере и фотогалерее устройства.

2. Функция мигания фонарика при выигрыше

- **Описание**: Для повышения интерактивности приложения внедрена функция мигания фонарика.
- **Реализация**: Использован **AVCaptureDevice**, чтобы обратиться к фонарику устройства. В случае выигрыша в игре, фонарик мигает три раза, сигнализируя о победе.
- **Логирование**: Введено системное логирование состояний фонарика (**NSLog**) для отслеживания его активации и деактивации.

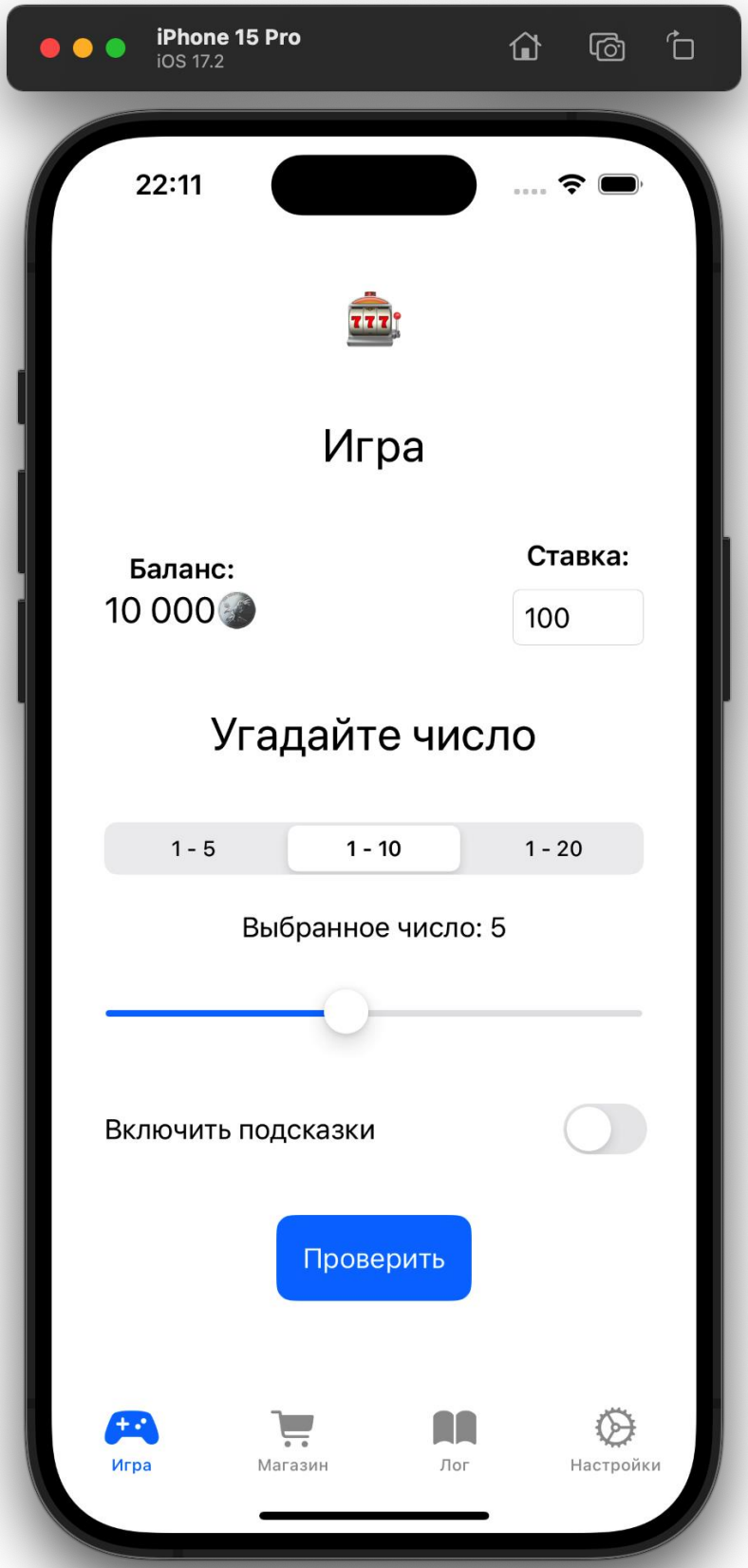

Рисунок 1 – вид приложения при запуске

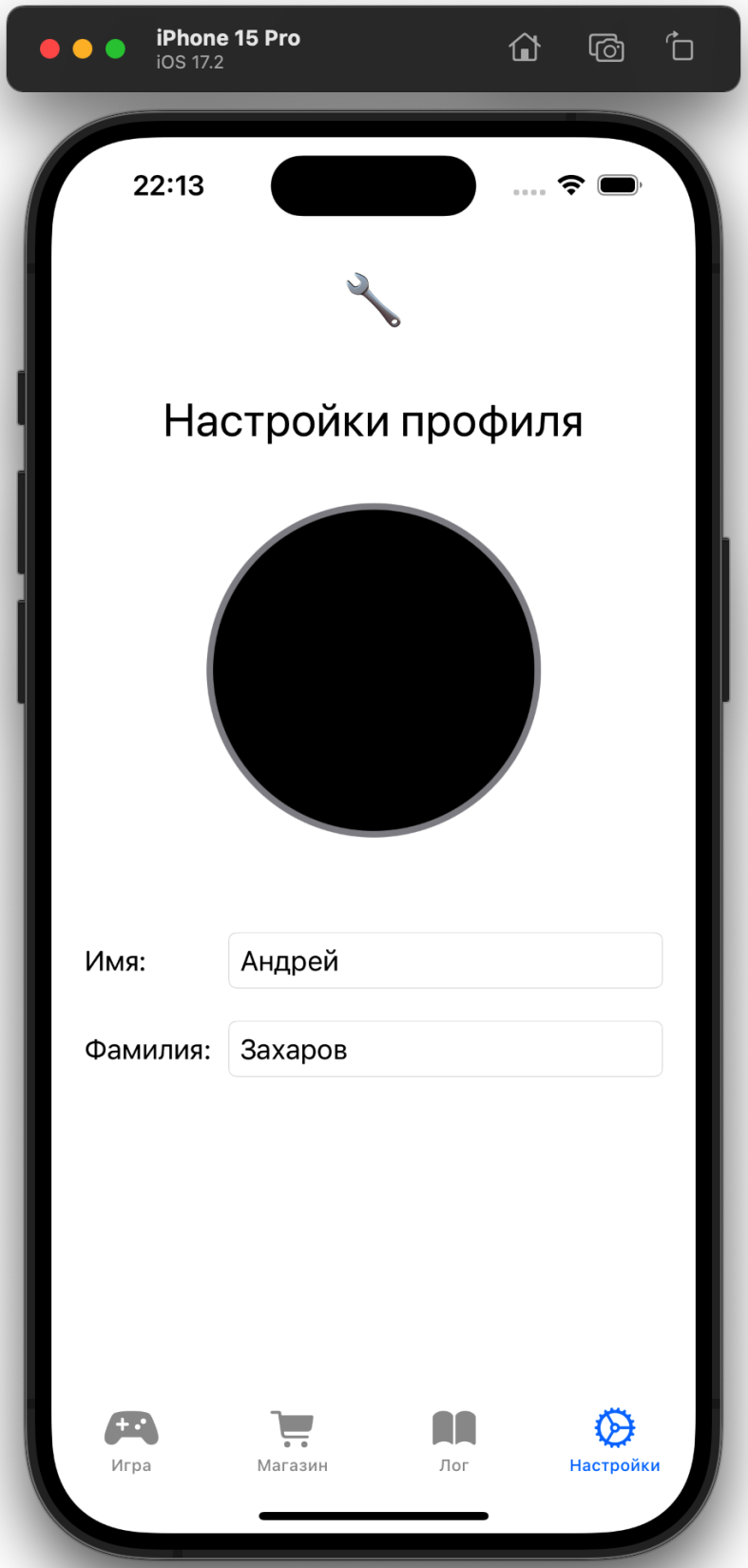

Рисунок 2 – вид добавленного окна настроек

Рисунок 3 – при нажатии на «изображение аватара» открывается «камера»

```
Torch (Flashlight) is not available on this device.
```

Рисунок 4 – уведомление в логах о том, что фонарик недоступен на эмуляторе

```
[CAMCaptureEngine] Received a session runtime error notification : Error
Domain=AVFoundationErrorDomain Code=-11800 "The operation could not be completed"
UserInfo={NSLocalizedFailureReason=An unknown error occurred (-12782),
NSLocalizedDescription=The operation could not be completed, NSUnderlyingError=0x600000d132d0
{Error Domain=NSOSStatusErrorDomain Code=-12782 "(null)"}}
[CAMCaptureEngine] Performing recovery from error: Error Domain=AVFoundationErrorDomain
Code=-11800 "The operation could not be completed" UserInfo={NSLocalizedFailureReason=An
unknown error occurred (-12782), NSLocalizedDescription=The operation could not be completed,
NSUnderlyingError=0x600000d132d0 {Error Domain=NSOSStatusErrorDomain Code=-12782 "(null)"}}
[CAMCaptureEngine] Attempting to recover from a session runtime error by restarting the
AVCaptureSession...
```

Рисунок 5 – уведомление о камере в логах

Листинг:

LabaApp.swift

```
//
//  LabaApp.swift
//  Laba
//
//  Created by Андрей Захаров on 28.02.2024.
//

import SwiftUI

@main
struct LabaApp: App {
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}
```

ContentView.swift

```
//
//  ContentView.swift
//  Laba
```

```
//
//  Created by Андрей Захаров on 28.02.2024.
//

import SwiftUI
import AVFoundation

struct ContentView: View {
    @State private var userGuess: Double = 5
    @State private var randomNumber = Int.random(in: 1...10)
    @State private var showAlert = false
    @State private var alertTitle = ""
    @State private var isHintEnabled: Bool = false
    @State private var numberRange = "1 - 10"
    let numberRanges = ["1 - 5", "1 - 10", "1 - 20"]
    @State private var balance: Int = 10000
    @State private var betAmount: Int = 100
    @State private var logRecords: [String] = []

    var body: some View {
        TabView {
            // Вкладка игрового экрана
            gameView()
                .tabItem {
                    Label("Игра", systemImage: "gamecontroller")
                }
                .tag(1)

            // Вкладка магазина
            ShopView(balance: $balance, logRecords: $logRecords)
                .tabItem {
                    Label("Магазин", systemImage: "cart")
                }
                .tag(2)

            // Вкладка лога операций
            LogView(logs: logRecords)
                .tabItem {
                    Label("Лог", systemImage: "book")
                }
                .tag(3)

            SettingsView() // Предполагается, что вы создадите эту вью
                .tabItem {
                    Label("Настройки", systemImage: "gear")
                }
                .tag(4)

        }
    }
```

```
func gameView() -> some View {
    NavigationView {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Игра")
                .font(.title)
                .padding()

            HStack {
                VStack {
                    Text("Баланс:")
                        .font(.headline)
                    Text("\(balance)□")
                        .font(.title2)
                }
                Spacer()
                VStack {
                    Text("Ставка:")
                        .font(.headline)
                    TextField("100", value: $betAmount, format: .number)
                        .textFieldStyle(RoundedBorderTextFieldStyle())
                        .keyboardType(.numberPad)
                        .frame(width: 80)
                }
            }
            .padding()
            Spacer(minLength: 10)

            Text("Угадайте число")
                .font(.title)
            Spacer(minLength: 20)

            Picker("Выберите диапазон чисел", selection: $numberRange) {
                ForEach(numberRanges, id: \.self) {
                    Text($0)
                }
            }
            .pickerStyle(SegmentedPickerStyle())
            .padding()
            .onChange(of: numberRange) { _ in
                updateRandomNumber()
            }
            Text("Выбранное число: \(Int(userGuess))")

            Slider(value: $userGuess, in: 1...CGFloat(getUpperRangeLimit()), step: 1)
                .padding()

            Toggle("Включить подсказки", isOn: $isHintEnabled)
                .padding()
```

```
            Spacer(minLength: 20)

            Button("Проверить") {
                checkGuess()
            }
            .padding()
            .background(Color.blue)
            .foregroundColor(.white)
            .cornerRadius(10)
            Spacer(minLength: 20)

                .alert(isPresented: $showAlert) {
                    Alert(title: Text(alertTitle), dismissButton: .default(Text("OK")))
                }
            Spacer(minLength: 20)
        }
        .padding()
    }
}

func updateRandomNumber() {
    let limit = getUpperRangeLimit()
    randomNumber = Int.random(in: 1...limit)
    userGuess = min(userGuess, Double(limit))
}

func getUpperRangeLimit() -> Int {
    switch numberRange {
    case "1 - 20":
        return 20
    case "1 - 5":
        return 5
    default: // "1 - 10"
        return 10
    }
}

func checkGuess() {
    if betAmount > balance {
        alertTitle = "Ваша ставка превышает баланс!"
        showAlert = true
        return
    }

    balance -= betAmount
    let guess = Int(userGuess)

    if guess == randomNumber {
        var winMultiplier = 1.0
        switch numberRange {
        case "1 - 5":
```

```
            winMultiplier = 2.0
        case "1 - 10":
            winMultiplier = 5.0
        case "1 - 20":
            winMultiplier = 10.0
        default:
            break
        }

        let winAmount = Int(Double(betAmount) * winMultiplier)
        balance += winAmount
        alertTitle = "Правильно! Вы угадали число!"
        logRecords.append("✅Выигрыш: \(winAmount)□ в режиме \(numberRange)")
        flashLight() // Вызов функции мигания фонарика

        updateRandomNumber()

    } else {
        if isHintEnabled {
            if guess < randomNumber {
                alertTitle = "Слишком мало! Попробуйте число побольше."
            } else {
                alertTitle = "Слишком много! Попробуйте число поменьше."
            }
        } else {
            alertTitle = "Не угадали! Попробуйте еще раз."
        }
        logRecords.append("❌Проигрыш: \(betAmount)□ в режиме \(numberRange)")
    }

    showAlert = true
}
func flashLight() {
    guard let device = AVCaptureDevice.default(for: .video), device.hasTorch else {
        NSLog("Torch (Flashlight) is not available on this device.")
        return
    }
    do {
        try device.lockForConfiguration()
        for _ in 1...3 {
            NSLog("Torch (Flashlight) state changing.")
            device.torchMode = .on
            device.torchMode = .off
            Thread.sleep(forTimeInterval: 0.1)
        }
        device.unlockForConfiguration()
    } catch {
        NSLog("Error occurred while trying to access torch (flashlight):
\(error.localizedDescription)")
    }
}
```

```
}

// Preview section
struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

ShopView.swift

```
//
//  ShopView.swift
//  Laba
//
//  Created by Андрей Захаров on 07.03.2024.
//

import SwiftUI

struct ShopView: View {
    @Binding var balance: Int
    @Binding var logRecords: [String]
    @State private var showAlert = false
    @State private var alertTitle = ""

    struct Product {
        var emoji: String
        var price: Int
        var count: Int
    }

    @State private var products: [Product] = [
        Product(emoji: "□", price: 50, count: 0),
        Product(emoji: "□", price: 100, count: 0),
        Product(emoji: "□", price: 5000, count: 0),
        Product(emoji: "□", price: 10000, count: 0)
    ]

    var body: some View {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Магазин")
                .font(.title)
                .padding()
            Text("Баланс: \(balance)□")
                .font(.title2)
                .padding()
```

```
            // Список товаров
            ForEach($products.indices, id: \.self) { index in
                HStack {
                    Text(products[index].emoji)
                        .font(.largeTitle)
                        .frame(width: 50, alignment: .leading)
                    Text("\(products[index].price)□")
                        .frame(width: 100, alignment: .leading)
                    Button("Купить") {
                        buyProduct(index: index)
                    }
                    .padding(8)
                    .background(Color.blue)
                    .foregroundColor(.white)
                    .cornerRadius(8)
                    Spacer()
                    Text("x\(products[index].count)")
                }
                .padding()
            }

            Spacer()
        }
        .alert(isPresented: $showAlert) {
            Alert(title: Text(alertTitle), dismissButton: .default(Text("OK")))
        }
    }

    func buyProduct(index: Int) {
        if products[index].price <= balance {
            balance -= products[index].price
            products[index].count += 1
            logRecords.append("□ Покупка: \(products[index].emoji) за
\(products[index].price)□")
        } else {
            alertTitle = "Недостаточно средств для покупки!"
            showAlert = true
        }
    }
}

struct ShopView_Previews: PreviewProvider {
    @State static var tempBalance = 10000
    @State static var tempLogRecords: [String] = []

    static var previews: some View {
        ShopView(balance: $tempBalance, logRecords: $tempLogRecords)
    }
}
```

LogView.swift

```
//
//  LogView.swift
//  Laba
//
//  Created by Андрей Захаров on 07.03.2024.
//

import SwiftUI

struct LogView: View {
    var logs: [String] // Массив строк с записями лога

    var body: some View {
        VStack {
            Text("□")
                .font(.largeTitle)
                .padding()
            Text("Лог операций")
                .font(.title)
                .padding()

            // Проверяем, есть ли записи в логе
            if logs.isEmpty {
                Text("Записей в логе нет")
                    .padding()
            } else {
                List(logs.reversed(), id: \.self) { log in
                    Text(log)
                }
            }
        }
    }
}

// Предпросмотр
struct LogView_Previews: PreviewProvider {
    static var previews: some View {
        // Создание примера данных для предпросмотра
        let sampleLogs = ["Покупка: □ за 50□", "Выигрыш: 500 в режиме 1 - 10"]

        // Возвращаем LogView с примерными данными
        LogView(logs: sampleLogs)
    }
}
```

SettingsView.swift

```
//
//  SettingsView.swift
//  Laba
//
//  Created by Андрей Захаров on 07.03.2024.
//

import SwiftUI
import AVFoundation

struct SettingsView: View {
    @State private var showingImagePicker = false
    @State private var inputImage: UIImage?
    @State private var avatarImage: Image?
    @State var firstName = "Андрей"
    @State var lastName = "Захаров"

    var body: some View {
        ScrollView {
            VStack(spacing: 20) {
                Text("□")
                    .font(.largeTitle)
                    .padding(.top, 20)
                Text("Настройки профиля")
                    .font(.title)
                    .padding()

                avatarSection
                Spacer(minLength: 20)

                Group {
                    HStack {
                        Text("Имя:")
                            .frame(width: 80, alignment: .leading)
                        TextField("Введите имя", text: $firstName)
                            .textFieldStyle(RoundedBorderTextFieldStyle())
                    }

                    HStack {
                        Text("Фамилия:")
                            .frame(width: 80, alignment: .leading)
                        TextField("Введите фамилию", text: $lastName)
                            .textFieldStyle(RoundedBorderTextFieldStyle())
                    }
                }
                .frame(maxWidth: .infinity)
                .padding([.leading, .trailing], 20)
            }
            .padding(.bottom, 20)
        }
```

```
        .sheet(isPresented: $showingImagePicker, onDismiss: loadImage) {
            ImagePicker(image: self.$inputImage)
        }
    }

    var avatarSection: some View {
        ZStack {
            if let avatarImage = avatarImage {
                avatarImage
                    .resizable()
                    .scaledToFit()
                    .frame(width: 200, height: 200)
                    .clipShape(Circle())
                    .overlay(Circle().stroke(Color.gray, lineWidth: 4))
            } else {
                Circle()
                    .frame(width: 200, height: 200)
                    .overlay(Circle().stroke(Color.gray, lineWidth: 4))
            }
        }
        .onTapGesture {
            self.showingImagePicker = true
        }
    }

    func loadImage() {
        guard let inputImage = inputImage else { return }
        avatarImage = Image(uiImage: inputImage)
    }
}


struct ImagePicker: UIViewControllerRepresentable {
    @Environment(\.presentationMode) var presentationMode
    @Binding var image: UIImage?

    func makeUIViewController(context: Context) -> UIImagePickerController {
        let picker = UIImagePickerController()
        if UIImagePickerController.isSourceTypeAvailable(.camera) {
            picker.sourceType = .camera
        } else {
            // Обработка случая, когда камера недоступна
            print("Камера недоступна на данном устройстве")
        }
        picker.delegate = context.coordinator
        return picker
    }

    func updateUIViewController(_ uiViewController: UIImagePickerController, context:
Context) {}
```

```
    func makeCoordinator() -> Coordinator {
        Coordinator(self)
    }

    class Coordinator: NSObject, UINavigationControllerDelegate,
UIImagePickerControllerDelegate {
        let parent: ImagePicker

        init(_ parent: ImagePicker) {
            self.parent = parent
        }

        func imagePickerController(_ picker: UIImagePickerController,
didFinishPickingMediaWithInfo info: [UIImagePickerController.InfoKey : Any]) {
            print("Камера активирована")
            if let uiImage = info[.originalImage] as? UIImage {
                parent.image = uiImage
                print("Изображение выбрано")
            }
            parent.presentationMode.wrappedValue.dismiss()
        }

        func imagePickerControllerDidCancel(_ picker: UIImagePickerController) {
            print("Выбор изображения отменён")
            parent.presentationMode.wrappedValue.dismiss()
        }
    }

}

struct SettingsView_Previews: PreviewProvider {
    static var previews: some View {
        SettingsView()
    }
}
```