# Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

**Национальный исследовательский**

**Томский политехнический университет**

Инженерная школа информационных технологий и робототехники Отделение информационных технологий

Отчет по лабораторной работе №26 по дисциплине

**«Мобильная разработка. Хранение данных и сетевое взаимодействие»**

Камера

Выполнил:

Студент группы 8B21 П. А. Жердева

Проверил:

Ст. преп. ОИТ ИШИТР В. А. Дорофеев

# Задание

Разработайте приложение для работы с камерой. Приложение должно быть основано на библиотеке CameraX. В нём должен быть реализован следующий функционал:

* Определение HTML-цвета точки в центре снимка. В центре экрана нужно разместить визуальный объект – например, круг или пересечение линий – который показывает пользователю примерное место размещения этого пикселя. Когда пользователь перемещает камеру, приложение постоянно анализирует цвет центрального пикселя и выводит его HTML-код на экран.
* При нажатии кнопки съемки осуществляется захват и сохранение текущего кадра в галерею телефона.

Приложение может выглядеть примерно таким образом:

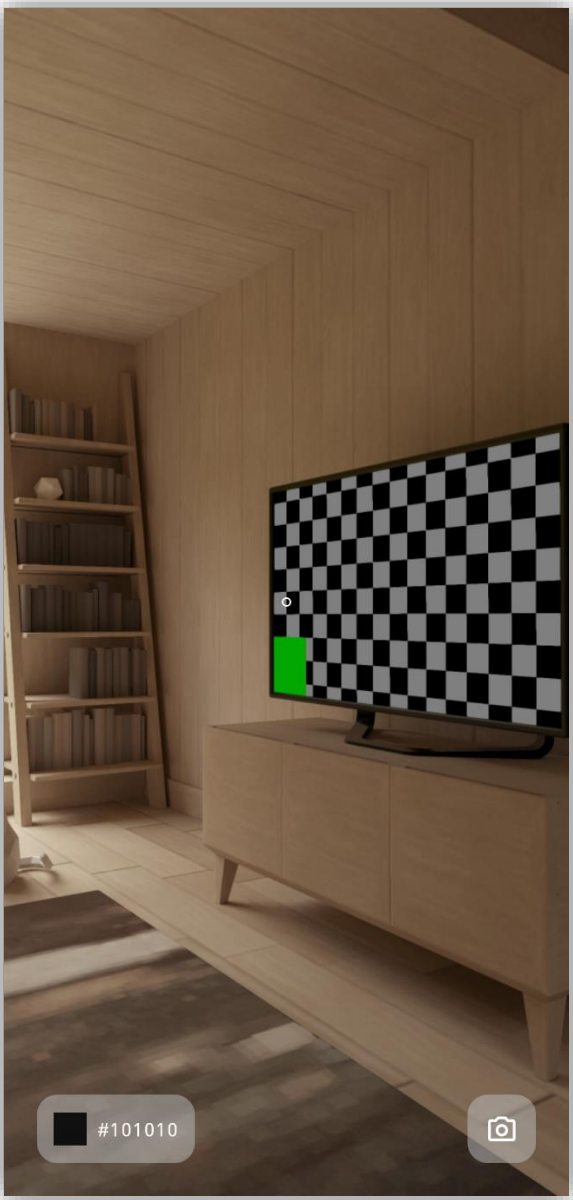


Рис.1. Примерный внешний вид приложения

Чтобы сделать круглый «прицел», можно добавить XML-файл crosshair.xml в раздел res → drawable со следующим содержимым:

<?xml version="1.0" encoding="utf-8"?>

<shape xmlns:android=<http://schemas.android.com/apk/res/android> android:shape="oval">

<solid android:color="#00000000"/>

<stroke android:color="@color/white" android:width="1dp"/>

<size android:width="7dp"android:height="7dp"/>

</shape>

Здесь у элемента shape задаётся форма oval, а вложенные элементы задают параметры: solid – цвет (в данном случае прозрачный), stroke – обводку, size – размер. Затем в разметку активности можно добавить элемент ImageView с такими параметрами:

<ImageView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:src="@drawable/crosshair"

android:layout\_gravity="center"/>

Это разместит элемент-«прицел» по центру экрана

**Ход выполнения**

Для работы было добавлено несколько разрешений в манифест:

1. <uses-feature android:name="android.hardware.camera" android:required="true" /> - приложение требует наличие камеры
2. <uses-permission android:name="android.permission.CAMERA" /> разрешение на использование камеры
3. <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" android:maxSdkVersion="28" /> - разрешение на сохранение фото во внешнее хранилище (для старых устройств)
4. <uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" /> - разрешение на доступ к изображениям, находящимся в медиахранилище (включая недавно снятые фото) (для новых устройств).

**Как работает приложение:**

1. Показывает изображение с камеры в реальном времени.
2. Отображает «прицел» в центре экрана.
3. Постоянно определяет цвет пикселя в центре кадра и показывает его HTML-код (#RRGGBB).
4. Позволяет сделать фото и сохранить его в галерею телефона.

Таблица 1. Структура проекта

|  |  |
| --- | --- |
| **Файл** | **Назначение** |
| **MainActivity.kt** | Основная активность приложения — управляет камерой, анализирует кадры, сохраняет фото. |
| **activity\_main.xml** | Разметка интерфейса: окно камеры, прицел, панель с цветом и кнопка съёмки. |
| **crosshair.xml** | Графика прицела — белый контурный кружок по центру экрана. |
| **panel\_bg.xml** | Фон для панели, где отображается текущий цвет и его HEX-код. |

**MainActivity.kt -** главный класс приложения, наследуется от AppCompatActivity.

Основные элементы интерфейса:

* PreviewView — окно, в котором показывается изображение с камеры.
* View colorPreview — квадрат, который заполняется текущим цветом.
* TextView colorHexText — текст с HTML-кодом цвета (#RRGGBB).
* ImageButton captureButton — кнопка для снимка.

Проверка и запрос разрешения камеры:

При запуске приложение проверяет, есть ли разрешение CAMERA.  
Если нет — запрашивает его через requestPermissionLauncher.  
После получения — запускает камеру (startCamera()).

Настройка камеры (startCamera()):

1. Получается экземпляр ProcessCameraProvider — это контроллер камеры.
2. Настраивается:
   * Preview — показывает изображение в PreviewView.
   * ImageCapture — отвечает за съёмку и сохранение кадров.
   * ImageAnalysis — анализирует каждый кадр в реальном времени.
3. В ImageAnalysis задаётся метод processImage(imageProxy), который обрабатывает изображение.

Анализ изображения (processImage()):

Каждый кадр из камеры приходит в формате YUV\_420\_888.  
Метод:

1. Получает центральные координаты кадра (centerX, centerY).
2. Извлекает значения Y, U, V для этого пикселя.
3. Преобразует их в RGB по формулам:
4. r = y + 1.370705 \* v
5. g = y - 0.337633 \* u - 0.698001 \* v
6. b = y + 1.732446 \* u
7. Приводит значения к диапазону [0, 255].
8. Создаёт цвет Color.rgb(r, g, b) и формирует строку #RRGGBB.
9. Обновляет интерфейс:
   * квадрат (colorPreview) заливается этим цветом;
   * текст (colorHexText) показывает его HEX-код.

Съёмка фотографии (takePhoto()):

1. Создаёт файл в каталоге DCIM/ с текущей датой.
2. Вызывает imageCapture.takePicture().
3. После сохранения показывает Toast с именем файла.

**activity\_main.xml -** главный интерфейс приложения

* PreviewView — заполняет весь экран (камера).
* ImageView с @drawable/crosshair — белый кружок-прицел по центру.
* LinearLayout (colorInfoPanel) — панель внизу, содержит:
* Квадрат с текущим цветом (View colorPreview).
* Текст с HTML-кодом цвета (TextView colorHexText).
* ImageButton capture — кнопка для фотографирования, расположена снизу.

**Текст программы**

**MainActivity.kt:**

package ru.polinazherdeva.lr26  
  
import android.Manifest  
import android.content.pm.PackageManager  
import android.graphics.Color  
import android.os.Bundle  
import android.os.Environment  
import android.view.View  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.appcompat.app.AppCompatActivity  
import androidx.camera.core.\*  
import androidx.camera.lifecycle.ProcessCameraProvider  
import androidx.camera.view.PreviewView  
import androidx.core.content.ContextCompat  
import android.widget.\*  
import java.io.File  
import java.nio.ByteBuffer  
import java.text.SimpleDateFormat  
import java.util.\*  
  
class MainActivity : AppCompatActivity() {  
  
 *// --- UI элементы ---* private lateinit var previewView: PreviewView *// Элемент для отображения камеры* private lateinit var colorPreview: View *// Прямоугольник, показывающий текущий цвет* private lateinit var colorHexText: TextView *// TextView для HTML-кода цвета* private lateinit var captureButton: ImageButton *// Кнопка для съемки фото* private var imageCapture: ImageCapture? = null *// Объект CameraX для фотосъемки  
  
 // --- Разрешение на камеру ---* private val requestPermissionLauncher =  
 registerForActivityResult(ActivityResultContracts.RequestPermission()) **{** isGranted **->** *// Если пользователь дал разрешение* if (isGranted) {  
 startCamera() *// Запускаем камеру* } else {  
 Toast.makeText(this, "Разрешение на камеру отклонено", Toast.*LENGTH\_LONG*).show()  
 }  
 **}** override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*) *// Связываем с разметкой  
  
 // Находим View по ID* previewView = findViewById(R.id.*previewView*)  
 colorPreview = findViewById(R.id.*colorPreview*)  
 colorHexText = findViewById(R.id.*colorHexText*)  
 captureButton = findViewById(R.id.*capture*)  
  
 *// Проверка разрешения на камеру* if (ContextCompat.checkSelfPermission(  
 this,  
 Manifest.permission.*CAMERA* ) == PackageManager.*PERMISSION\_GRANTED* ) {  
 startCamera() *// Разрешение уже есть — запускаем камеру* } else {  
 *// Разрешения нет — запрашиваем* requestPermissionLauncher.launch(Manifest.permission.*CAMERA*)  
 }  
  
 *// Обработка нажатия кнопки съемки* captureButton.setOnClickListener **{** takePhoto()  
 **}** }  
  
 *// --- Метод инициализации камеры ---* private fun startCamera() {  
 val cameraProviderFuture = ProcessCameraProvider.getInstance(this)  
  
 *// Ждем получения CameraProvider* cameraProviderFuture.addListener(**{** val cameraProvider = cameraProviderFuture.get()  
  
 *// Создаем предварительный просмотр камеры* val preview = Preview.Builder()  
 .build()  
 .*also* **{  
 it**.*surfaceProvider* = previewView.*surfaceProvider // Подключаем PreviewView* **}** val cameraSelector = CameraSelector.*DEFAULT\_BACK\_CAMERA // Используем заднюю камеру  
  
 // Настройка фотосъемки* imageCapture = ImageCapture.Builder()  
 .setTargetRotation(previewView.*display*.*rotation*) *// Учитываем поворот экрана* .build()  
  
 *// Настройка анализа изображения (для считывания цвета)* val imageAnalysis = ImageAnalysis.Builder()  
 .setBackpressureStrategy(ImageAnalysis.*STRATEGY\_KEEP\_ONLY\_LATEST*) *// Берем только последний кадр* .setOutputImageFormat(ImageAnalysis.*OUTPUT\_IMAGE\_FORMAT\_YUV\_420\_888*) *// Формат YUV* .build()  
  
 *// Подключаем анализатор изображений* imageAnalysis.setAnalyzer(ContextCompat.getMainExecutor(this)) **{** imageProxy **->** processImage(imageProxy) *// Обработка кадра* **}** try {  
 cameraProvider.unbindAll() *// Отвязываем все предыдущие use cases* cameraProvider.bindToLifecycle(  
 this, cameraSelector, preview, imageCapture, imageAnalysis  
 ) *// Привязываем камеру к жизненному циклу Activity* } catch (exc: Exception) {  
 Toast.makeText(this, "Ошибка инициализации камеры", Toast.*LENGTH\_SHORT*).show()  
 }  
 **}**, ContextCompat.getMainExecutor(this))  
 }  
  
 *// --- Метод обработки кадра для получения цвета центра ---* private fun processImage(imageProxy: ImageProxy) {  
 try {  
 *// Получаем буферы Y, U и V (формат YUV\_420\_888)* val yPlane = imageProxy.*planes*[0].*buffer* val uPlane = imageProxy.*planes*[1].*buffer* val vPlane = imageProxy.*planes*[2].*buffer  
  
 // Страйды и пиксельный шаг для корректного доступа к данным* val yRowStride = imageProxy.*planes*[0].*rowStride* val uvRowStride = imageProxy.*planes*[1].*rowStride* val uvPixelStride = imageProxy.*planes*[1].*pixelStride* val width = imageProxy.*width* val height = imageProxy.*height  
  
 // Центр кадра* val centerX = width / 2  
 val centerY = height / 2  
  
 *// Индексы центрального пикселя в буферах* val yIndex = centerY \* yRowStride + centerX  
 val uvIndex = (centerY / 2) \* uvRowStride + (centerX / 2) \* uvPixelStride  
  
 *// Считываем YUV значения* val y = (yPlane.get(yIndex).toInt() and 0xFF)  
 val u = (uPlane.get(uvIndex).toInt() and 0xFF) - 128  
 val v = (vPlane.get(uvIndex).toInt() and 0xFF) - 128  
  
 *// Конвертация YUV -> RGB* var r = (y + 1.370705f \* v).toInt()  
 var g = (y - 0.337633f \* u - 0.698001f \* v).toInt()  
 var b = (y + 1.732446f \* u).toInt()  
  
 *// Ограничиваем значения RGB в диапазоне 0–255* r = r.*coerceIn*(0, 255)  
 g = g.*coerceIn*(0, 255)  
 b = b.*coerceIn*(0, 255)  
  
 *// Формируем цвет и HTML-код* val colorInt = Color.rgb(r, g, b)  
 val hex = String.*format*("#%02X%02X%02X", r, g, b)  
  
 *// Обновляем UI на главном потоке* runOnUiThread **{** colorPreview.setBackgroundColor(colorInt) *// Меняем цвет квадрата* colorHexText.*text* = hex *// Выводим HTML-код* **}** } catch (e: Exception) {  
 e.printStackTrace()  
 } finally {  
 imageProxy.close() *// Обязательно закрываем ImageProxy* }  
 }  
  
 *// --- Метод фотосъемки ---* private fun takePhoto() {  
 val imageCapture = imageCapture ?: return *// Проверяем, что камера инициализирована  
  
 // Создаем файл для сохранения фото* val photoFile = File(  
 Environment.getExternalStoragePublicDirectory(Environment.*DIRECTORY\_DCIM*), *// Каталог DCIM* SimpleDateFormat("yyyyMMdd\_HHmmss", Locale.*US*)  
 .format(System.currentTimeMillis()) + ".jpg" *// Имя файла по дате* )  
  
 *// Опции сохранения фото* val outputOptions = ImageCapture.OutputFileOptions.Builder(photoFile).build()  
  
 *// Запускаем съемку* imageCapture.takePicture(  
 outputOptions,  
 ContextCompat.getMainExecutor(this), *// Главный поток* object : ImageCapture.OnImageSavedCallback {  
 override fun onError(exc: ImageCaptureException) {  
 Toast.makeText(*applicationContext*, "Ошибка сохранения фото", Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
  
 override fun onImageSaved(output: ImageCapture.OutputFileResults) {  
 Toast.makeText(*applicationContext*, "Фото сохранено: ${photoFile.*name*}", Toast.*LENGTH\_SHORT*)  
 .show()  
 }  
 }  
 )  
 }  
}

**crosshair.xml:**

*<?*xml version="1.0" encoding="utf-8"*?>*<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="oval">  
 <solid android:color="#00000000" />  
 <stroke android:color="@color/white" android:width="1dp" />  
 <size android:width="20dp" android:height="20dp" />  
</shape>

**panel\_bg.xml:**

<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <solid android:color="#88CCCCCC" />  
 <corners android:radius="12dp" />  
</shape>

**activity\_main.xml:**

<FrameLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
  
 *<!-- Камера -->* <androidx.camera.view.PreviewView  
 android:id="@+id/previewView"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent" />  
  
 *<!-- Прицел -->* <ImageView  
 android:src="@drawable/crosshair"  
 android:layout\_width="24dp"  
 android:layout\_height="24dp"  
 android:layout\_gravity="center" />  
  
 *<!-- Плашка с цветом и кодом -->* <LinearLayout  
 android:id="@+id/colorInfoPanel"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom|center\_horizontal"  
 android:layout\_marginBottom="100dp"  
 android:orientation="horizontal"  
 android:background="@drawable/panel\_bg"  
 android:padding="8dp"  
 android:elevation="6dp">  
  
 *<!-- Квадрат цвета -->* <View  
 android:id="@+id/colorPreview"  
 android:layout\_width="40dp"  
 android:layout\_height="40dp"  
 android:layout\_marginEnd="12dp"  
 android:background="@android:color/black" />  
  
 *<!-- HEX-код -->* <TextView  
 android:id="@+id/colorHexText"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="#FFFFFF"  
 android:textSize="18sp"  
 android:textColor="@android:color/white"  
 android:fontFamily="monospace"  
 android:layout\_gravity="center\_vertical" />  
 </LinearLayout>  
  
 *<!-- Кнопка съемки -->* <ImageButton  
 android:id="@+id/capture"  
 android:layout\_width="64dp"  
 android:layout\_height="64dp"  
 android:layout\_gravity="bottom|center\_horizontal"  
 android:layout\_marginBottom="24dp"  
 android:background="@android:color/transparent"  
 android:src="@android:drawable/ic\_menu\_camera"  
 android:contentDescription="@string/capture\_button\_desc" />  
  
</FrameLayout>

**Результат работы:**

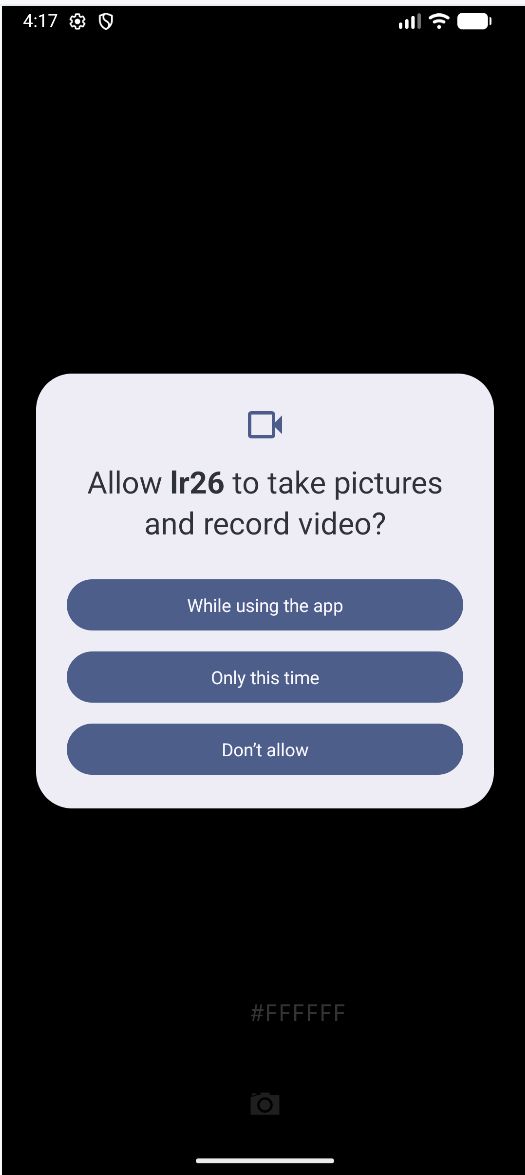
****

Рис. 2. Запрос разрешения на использование камеры



Рис. 3. Вывод цвета и его hex-кода предмета при наведении «прицела» на него

# Вывод

В ходе лабораторной работы было создано приложение с использованием CameraX, которое отображает превью камеры, определяет цвет центрального пикселя и показывает его HTML-код, а также позволяет делать и сохранять фотографии. Реализованы основные компоненты CameraX: Preview, ImageAnalysis и ImageCapture. Приложение демонстрирует работу с разрешениями камеры и сохранением файлов, обработку изображений в реальном времени и взаимодействие UI с камерой. Работа закрепила навыки работы с камерой на Android и анализом пиксельных данных.