**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 11 по дисциплине

**«Разработка программного обеспечения для ОС Android»**

Выполнил:

Студент группы 8И7А \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Леонов

Проверил:

Ст. преп. ОИТ ИШИТР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.А. Дорофеев

Томск 2020

# Задание

Разработайте программу, рисующую какую-либо анимированную картинку.

# Текст программы

<https://github.com/8ait/android-course>

**MainActivity.kt**

package com.leonovalexandr.lab11

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

class MainActivity : AppCompatActivity() {

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

}

}

**DrawSurface.kt**

package com.leonovalexandr.lab11

import android.content.Context

import android.graphics.Canvas

import android.graphics.Color

import android.graphics.Paint

import android.util.AttributeSet

import android.view.SurfaceHolder

import android.view.SurfaceView

import kotlinx.coroutines.GlobalScope

import kotlinx.coroutines.Job

import kotlinx.coroutines.launch

import kotlin.random.Random

class DrawSurface: SurfaceView, SurfaceHolder.Callback {

private var radius: Float = 100f

private var paint = Paint()

private lateinit var job: Job

constructor(context: Context) : this(context, null)

constructor(context: Context, attrs: AttributeSet?) : this(context, attrs, 0)

constructor(context: Context, attrs: AttributeSet?, defStyleAttr: Int) : super(context, attrs, defStyleAttr) {}

override fun surfaceChanged(holder: SurfaceHolder, format: Int, width: Int, height: Int) {

}

override fun surfaceCreated(holder: SurfaceHolder) {

paint.color = Color.WHITE

var balls = ArrayList<Ball>()

val rand = Random(0)

for (i in 1..2) {

for (j in 1..3) {

val ball = Ball()

ball.cx = width / 3 \* i.toFloat()

ball.cy = height / 4 \* j.toFloat()

ball.dx = rand.nextInt(-5, 5).toFloat()

ball.dy = rand.nextInt(-5, 5).toFloat()

ball.radius = rand.nextInt(90, 150).toFloat()

balls.add(ball)

}

}

job = GlobalScope.launch {

var canvas: Canvas?

while (true) {

canvas = holder.lockCanvas(null)

if (canvas != null) {

canvas.drawColor(Color.argb(255, 0, 192, 192))

balls.forEach{

ball -> canvas.drawCircle(ball.cx.toFloat(), ball.cy.toFloat(), ball.radius, paint)

}

holder.unlockCanvasAndPost(canvas)

}

// Перемещение шарика

balls.forEach{

ball -> ball.move(height, width)

}

for (i in 0 until balls.size) {

for (j in i + 1 until balls.size)

balls[i].ballCollision(balls[j])

}

}

}

}

override fun surfaceDestroyed(holder: SurfaceHolder) {

job.cancel()

}

init { holder.addCallback(this) }

}

**Ball.kt**

package com.leonovalexandr.lab11

import kotlin.math.absoluteValue

import kotlin.math.pow

import kotlin.math.sqrt

// Класс шарика.

class Ball {

var radius = 50f

// Координата x шара.

var cx = 0f

// Координата y шара.

var cy = 0f

// Скорость по оси x.

var dx = 10f

// Скорость по оси y.

var dy = 10f

fun move(height: Int, width: Int){

wallCollision(height, width)

cx += dx

cy += dy

}

private fun wallCollision(height: Int, width: Int) {

if (cx > width - radius || cx < radius)

dx = -dx

if (cy > height - radius || cy < radius)

dy = -dy

}

fun ballCollision(ball: Ball) {

val distance = sqrt((cx - ball.cx).toDouble().pow(2) + (cy - ball.cy).toDouble().pow(2))

if (distance < (radius + ball.radius) \* 1.05) {

val newdx = (dx.absoluteValue + ball.dx.absoluteValue)

val newdy = (dy.absoluteValue + ball.dy.absoluteValue)

val newdx1 = newdx \* (radius) / (radius + ball.radius)

val newdx2 = newdx \* (ball.radius) / (radius + ball.radius)

val newdy1 = newdy \* (radius) / (radius + ball.radius)

val newdy2 = newdy \* (ball.radius) / (radius + ball.radius)

if (dx \* ball.dx > 0){

if (dx < 0){

dx = -newdx1

ball.dx = -newdx2

} else {

dx = newdx1

ball.dx = newdx2

}

} else {

if (dx < 0) {

dx = newdx1

ball.dx = -newdx2

} else {

dx = -newdx1

ball.dx = +newdx2

}

}

if (dy \* ball.dy > 0){

if (dy < 0){

dy = -newdy1

ball.dy = -newdy2

} else {

dy = newdy1

ball.dy = newdy2

}

} else {

if (dy < 0) {

dy = newdy1

ball.dy = -newdy2

} else {

dy = -newdy1

ball.dy = +newdy2

}

}

}

}

}

**activity\_main.xml**

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

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<com.leonovalexandr.lab11.DrawSurface

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

# Результаты работы

<https://github.com/8ait/android-course/blob/main/Screenshots/lab11_vid1.mov>

# Выводы

В результате выполнения лабораторной работы я познакомился с технологиями работы анимации в приложениях Android.