**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 №24 по дисциплине

**«Разработка приложений для мобильных устройств»**

Рисование

Выполнил:

Студент группы 8В21 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Алексейчик Ю. Д.

Проверил:

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

# Задание

Придумайте и реализуйте приложение, в котором используется рисование. На экране должно быть несколько движущихся или изменяющих свои визуальные свойства объектов: например, машинки ездят по экрану, солнце перемещается по дуге (и, возможно, сменяется луной), мыльные пузыри летают по экрану и сталкиваются друг с другом, звезды на небе случайно (но плавно) мерцают, … Удивите преподавателя (в приятном смысле!)

# Ход работы

Для выполнения задания будет создана анимация – мерцающие звёзды случайного цвета в случайных местах, появляющиеся при нажатии на кнопку “Старт”.

**SunMoonView.kt**

class SunMoonView @JvmOverloads constructor(  
 context: Context, attrs: AttributeSet? = null, defStyleAttr: Int = 0  
) : View(context, attrs, defStyleAttr) {  
  
 // кисти для рисования фона и звезд  
 private val skyPaint = Paint().*apply* **{** *isAntiAlias* = true **}** private val starPaint = Paint().*apply* **{** *isAntiAlias* = true **}** // цвет ночного неба  
 private val nightSkyColor by *lazy* **{** *resources*.getColor(R.color.*night\_sky*, null) **}** // data class для хранения параметров звезды  
 private data class Star(  
 val x: Float, // координата x  
 val y: Float, // координата y  
 val radius: Float, // радиус звезды  
 val periodMs: Long, // период мерцания в миллисекундах  
 val phase: Float, // фаза мерцания  
 val baseColor: Int // базовый цвет звезды  
 )  
  
 // список звезд  
 private var stars: List<Star> = *emptyList*()  
 // аниматор для обновления отрисовки  
 private var starsAnimator: ValueAnimator? = null  
  
 init {  
 skyPaint.*color* = nightSkyColor // установка цвета фона  
 }  
  
 // основная функция отрисовки  
 override fun onDraw(canvas: Canvas) {  
 super.onDraw(canvas)  
 // рисование фона  
 canvas.drawRect(0f, 0f, *width*.toFloat(), *height*.toFloat(), skyPaint)  
 // рисование звезд  
 drawStars(canvas)  
 }  
  
 // подготовка случайных звезд  
 private fun prepareStars() {  
 val starCount = 40  
 val rnd = *Random*(System.currentTimeMillis())  
 val list = *mutableListOf*<Star>()  
 for (i in 0 *until* starCount) {  
 // генерация случайных параметров для каждой звезды  
 val x = rnd.nextFloat() \* *width* val y = rnd.nextFloat() \* (*height* \* 0.9f)  
 val radius = rnd.nextFloat() \* (*width* \* 0.005f) + (*width* \* 0.0025f)  
 val period = 800L + rnd.nextLong(0L, 1800L)  
 val phase = rnd.nextFloat() \* 2f \* Math.*PI*.toFloat()  
  
 // генерация случайного цвета звезды  
 val r = rnd.nextInt(100, 256)  
 val g = rnd.nextInt(0, 100)  
 val b = rnd.nextInt(100, 256)  
 val color = Color.rgb(r, g, b)  
  
 list += Star(x, y, radius, period, phase, color)  
 }  
 stars = list  
 }  
  
 // запуск анимации звезд  
 fun startAnimation() {  
 prepareStars() // подготовка звезд  
 starsAnimator?.cancel() // остановка предыдущей анимации  
 starsAnimator = ValueAnimator.ofFloat(0f, 1f).*apply* **{** *duration* = 1000L  
 *repeatCount* = ValueAnimator.*INFINITE* addUpdateListener **{** invalidate() **}** // перерисовка при обновлении  
 **}** starsAnimator?.start() // запуск анимации  
 }  
  
 // отрисовка звезд с мерцанием  
 private fun drawStars(canvas: Canvas) {  
 val now = System.currentTimeMillis()  
 for (star in stars) {  
 // расчет яркости на основе времени и фазы  
 val t = now % star.periodMs  
 val angle = 2.0 \* Math.*PI* \* t / star.periodMs + star.phase  
 val brightness = (0.5 + 0.5 \* *sin*(angle)).toFloat().*coerceIn*(0f, 1f)  
 val alpha = (brightness \* 255).toInt()  
 val color = star.baseColor  
 // установка цвета с альфа-каналом для мерцания  
 starPaint.*color* = Color.argb(alpha, Color.red(color), Color.green(color), Color.blue(color))  
 canvas.drawCircle(star.x, star.y, star.radius, starPaint)  
 }  
 }  
  
 // остановка анимации  
 fun stopAnimation() {  
 starsAnimator?.cancel()  
 }  
}

Для анимации мерцания используется синусоидальная функция для плавного изменения прозрачности звезд. Каждая звезда имеет свой период и фазу мерцания, что создает плавный эффект.

Интерфейс приложения описывается в activity\_main.xml и состоит из кнопки “старт” и окна для кастомного View. В MainActivity.kt инициализируются элементы интерфейса и вызывается кастомное View – SunMoonView.kt при нажатии на кнопку.

# Результаты работы



Рисунок 1. Анимация мерцания цветных звёзд



Рисунок 2. Анимация мерцания цветных звёзд

# Выводы

В ходе выполнения лабораторной работы было разработано приложение, реализующее анимацию мерцания звёзд с использованием кастомного View. Каждая звезда имеет уникальные характеристики: цвет, размер, период мерцания, что делает анимацию более необычной и красивой.