МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп’ютерних систем

**ЗВІТ**

**з лабораторної роботи № 6**

**«Створення анімацій у застосунках Android»**

**Виконав:**

студент 3-го курсу, групи КП-83, спеціальності 121 – Інженерія програмного забезпечення

*Мар’яненко Роман Геннадійович*

**Перевірив:**

к. т. н, старший викладач

*Хайдуров Владислав Володимирович*

Київ – 2020

**ЗМІСТ**

**ВСТУП**……………………………………………………………………………….**3**

1. Завдання до лабораторної роботи..………………………………………………4

2. Короткі теоретичні відомості…………………………………………………….5

3. Програмна реалізація задачі……………………………………………………...6

4. Результати роботи програми…………………………………………………....17

**ВИСНОВКИ**………………………………………………………………………..19

**ВСТУП**

Під час лабораторної роботи ми ознайомимося із основними принципами створення анімацій у програмних застосунках під ОС Android та навчимося створювати анімації власних графічних елементів дизайну.

**Завдання до лабораторної роботи**

1. Ознайомитись із усіма теоретичними відомостями до лабораторної роботи.
2. Переглянути усі практичні завдання при роботі з графічними об’єктами та методами їх створення.
3. Створити проекти з усіма розглянутими у лабораторній роботі анімаціями, продемонструвати їх роботу. Виконати відповідні скріншоти та додати їх до звіту лабораторної роботи.
4. Створити мобільний застосунок, у якому квадрат рухається (заданий довжиною сторони у пікселях у програмі) по колу (параметри кола задаються розробником у програмі), причому центр квадрата лежить на самому колі і квадрат не обертається відносно свого центра. Кольори обрати на власний розсуд. У додатку передбачити зміну швидкості (кроку пересування) квадрата по колу.
5. Створити програмний додаток, відтворює «килим» Серпинського: <http://hpc-education.ru/files/lectures/2011/ershov/ershov_2011_slides02.pdf>. Відтворення «килиму» можна виконувати як з анімацією, так і без неї (за бажанням розробника). За додаткові бали можна реалізувати й інші (по 1 балу за кожен) клітинні автомати з даного джерела.
6. За додаткові бали (до 4 балів) створити гру «Змійка», у якій елементами робочого поля є прозорі прямокутники (з границею певного кольору). Управління «змійкою» можна виконувати за допомогою чотирьох кнопок (типу «Джойстик»). Також передбачити зміну швидкості з часом або за бажанням користувача застосунку.
7. Для бажаючих отримати ще додаткові бали (до 4 балів), необхідно оформити даний програмний додаток окремо у вигляді мініпроекту: короткий звіт (за стилем лабораторної роботи) та створити презентацію, що містить 5-10 слайдів. Дедлайн актуальності виконання мініпроекту – тиждень після дедлайну здачі звітів даної лабораторної роботи до classroom.

**Короткі теоретичні відомості**

Клас Color відповідає за колір. Кольори можна описувати чотирма числами в форматі ARGB, по одному для кожного каналу (Alpha, Red, Green, Blue).

У графіці важливу роль відіграє колір, який представлений класом Color. Клас Color містить кілька констант і методів для конвертації і вилучення колірного компонента.

Кольори можна описувати чотирма числами в форматі ARGB, по одному для кожного каналу (Alpha, Red, Green, Blue). Кожен з каналів є восьмібітних цілим числом і може приймати значення від 0 до 256. Зазвичай колір упаковують в 32-бітове ціле число. Варто зазначити, що використовувати цілі числа для квітів ефективніше, ніж екземпляри класу Color.

Клас android.graphics.Canvas (полотно) надає методи для малювання, які відображають графічні примітиви на вихідному растровому зображенні. При цьому треба спочатку підготувати кисть (клас Paint), який дозволяє вказувати, як саме графічні примітиви повинні відображатися на растровому зображенні (колір, обведення, стиль, згладжування шрифту і т.д.).

Android підтримує напівпрозорість, градієнтні заливки, округлені прямокутники і згладжування. Через обмежених ресурсів векторна графіка поки що не підтримується, замість цього використовується традиційна растрова перерисовка. Canvas працює з пікселями, тому слід дбати про конвертацію одиниць dp в px і навпаки при необхідності. Початок координат знаходиться в лівому верхньому кутку.

Отримати доступ до полотна можна через об’єкт Bitmap або компонент View. Дуже часто розробники створюють свій власний компонент, наслідуючи від View, і малюють на його полотні для реалізації своїх задумів.

**Програмна реалізація задачі**

***Завдання 3:***

***Один прямокутник що рухається:***

|  |
| --- |
| Shape.java |
| package com.example.lab6;  public class Shape {  static final int maxStrokeWidth = 50;  static final int minStrokeWidth = 5;  static final int minVel = 10;  static final int maxVel = 20;  protected int color;  protected float strokeWidth;  protected int x;  protected int y;  protected int vx;  protected int vy;  public float getStrokeWidth() {  return strokeWidth;  }  public int getColor() {  return color;  }  public int getX() {  return x;  }  public int getY() {  return y;  }  public int getVx() {  return vx;  }  public int getVy() {  return vy;  }  public void setVx(int vx) {  int vxa = Math.abs(vx);  int vxs = vx / vxa;  if (vxa >= maxVel) this.vx = vxs \* maxVel;  else if (vxa <= minVel) this.vx = vxs \* minVel;  else this.vx = vxs \* vxa;  }  public void setVy(int vy) {  int vya = Math.abs(vy);  int vys = vy / vya;  if (vya >= maxVel) this.vy = vys \* maxVel;  else if (vya <= minVel) this.vy = vys \* minVel;  else this.vy = vys \* vya;  }  public void moveX() {  this.x += this.vx;  }  public void moveY() {  this.y += this.vy;  }  public void reverseVx() {  this.vx \*= -1;  }  public void reverseVy() {  this.vy \*= -1;  }  public void setColor(int color) {  this.color = color;  }  public void setX(int x) {  this.x = x;  }  public void setY(int y) {  this.y = y;  }  public void setStrokeWidth(int strokeWidth) {  if (strokeWidth >= maxStrokeWidth) this.strokeWidth = maxStrokeWidth;  else this.strokeWidth = Math.max(strokeWidth, minStrokeWidth);  }  } |

|  |
| --- |
| Rectangle.java |
| package com.example.lab6;  import android.graphics.Color;  public class Rectangle extends Shape{  private int w;  private int h;  Rectangle() {  this.x = 0;  this.y = 0;  this.vx = minVel;  this.vy = minVel;  this.w = 100;  this.h = 100;  this.strokeWidth = minStrokeWidth;  color = Color.BLACK;  }  Rectangle(int x, int y, int vx, int vy, int w, int h, int color, float strokeWidth) {  this.x = x;  this.y = y;  this.vx = vx;  this.vy = vy;  this.w = w;  this.h = h;  this.color = color;  this.strokeWidth = strokeWidth;  }  public int getHeight() {  return h;  }  public int getWidth() {  return w;  }  public void setWidth(int w) {  this.w = w;  }  public void setHeight(int h) {  this.h = h;  }  } |

|  |
| --- |
| MovingRectangleView.java |
| package com.example.lab6;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.util.AttributeSet;  import android.view.View;  import androidx.annotation.Nullable;  public class MovingRectangleView extends View {  public MovingRectangleView(Context context) {  super(context);  init(null);  }  public MovingRectangleView(Context context, AttributeSet attrs) {  super(context, attrs);  init(attrs);  }  private void init(@Nullable AttributeSet set) {}  int x = 10;  int y = 10;  int vx = 10;  int vy = 10;  int color = Color.RED;  float strokeWidth = 12;  int w = 200;  int h = 200;  Rectangle rect = new Rectangle(x, y, vx, vy, w, h, color, strokeWidth);  @Override  protected void onDraw(Canvas canvas) {  Helper.updateRect(canvas, rect);  invalidate();  }  } |

|  |
| --- |
| Helper.js |
| package com.example.lab6;  import android.graphics.Canvas;  import android.graphics.Color;  import android.graphics.Paint;  import android.view.View;  import java.util.Random;  public class Helper {  public static int randInt(int from, int to) {  Random rand = new Random();  int r = Math.abs(rand.nextInt());  return from + r % (to - from + 1);  }  public static void updateRect(Canvas canvas, Rectangle rect) {  Paint paint = new Paint();  paint.setColor(rect.getColor());  paint.setStyle(Paint.Style.STROKE);  paint.setStrokeWidth(rect.getStrokeWidth());  canvas.drawRect(rect.getX(), rect.getY(), rect.getWidth() + rect.getX(),  rect.getHeight() + rect.getY(), paint);  rect.moveX();  rect.moveY();  boolean isHorCollision = rect.getX() < 0 || rect.getX() > canvas.getWidth() - rect.getWidth();  boolean isVerCollision = rect.getY() < 0 || rect.getY() > canvas.getHeight() - rect.getHeight();  if (isHorCollision || isVerCollision) {  if (isHorCollision) rect.reverseVx();  if (isVerCollision) rect.reverseVy();  int r = Helper.randInt(0, 255);  int g = Helper.randInt(0, 255);  int b = Helper.randInt(0, 255);  rect.setColor(Color.rgb(r, g, b));  rect.setStrokeWidth(Helper.randInt(Rectangle.minStrokeWidth, Rectangle.maxStrokeWidth));  }  }  } |

***Декілька прямокутників що рухається:***

|  |
| --- |
| IndependentRectanglesView.java |
| package com.example.lab6;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.util.AttributeSet;  import android.view.View;  import androidx.annotation.Nullable;  public class IndependentRectanglesView extends View {  public IndependentRectanglesView(Context context) {  super(context);  initRects();  init(null);  }  public IndependentRectanglesView(Context context, AttributeSet attrs) {  super(context, attrs);  initRects();  init(attrs);  }  private void init(@Nullable AttributeSet set) {}  int rectQuantity = 10;  int color = Color.RED;  float strokeWidth = 12;  int w = 300;  int h = 200;  Rectangle[] rects;  private void initRects() {  rects = new Rectangle[rectQuantity];  for(int i = 0; i < rectQuantity; i++) {  int x = Helper.randInt(0, this.getWidth() - w);  int y = Helper.randInt(0, this.getHeight() - h);  int vx = Helper.randInt(Rectangle.minVel, Rectangle.maxVel);  int vy = Helper.randInt(Rectangle.minVel, Rectangle.maxVel);  rects[i] = new Rectangle(x, y, vx, vy, w, h, color, strokeWidth);  }  }  @Override  protected void onDraw(Canvas canvas) {  for (Rectangle rect : rects) {  Helper.updateRect(canvas, rect);  }  invalidate();  }  } |

***Прямокутників що рухаються синхронно, з колом всередині:***

|  |
| --- |
| Circle.java |
| package com.example.lab6;  import android.graphics.Color;  public class Circle extends Shape {  private int r;  Circle() {  this.x = 0;  this.y = 0;  this.vx = minVel;  this.vy = minVel;  this.r = 50;  this.strokeWidth = minStrokeWidth;  color = Color.BLACK;  }  Circle(int x, int y, int vx, int vy, int r, int color, float strokeWidth) {  this.x = x;  this.y = y;  this.vx = vx;  this.vy = vy;  this.r = r;  this.color = color;  this.strokeWidth = strokeWidth;  }  public int getRadius() {  return r;  }  public void setRadius(int w) {  this.r = r;  }  } |

***Завдання 4:***

|  |
| --- |
| CirclingSquareView.java |
| package com.example.lab6;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.graphics.Paint;  import android.util.AttributeSet;  import android.view.View;  import androidx.annotation.Nullable;  public class CirclingSquareView extends View {  public CirclingSquareView(Context context) {  super(context);  init(null);  }  public CirclingSquareView(Context context, AttributeSet attrs) {  super(context, attrs);  init(attrs);  }  private void init(@Nullable AttributeSet set) {}  int a = 100;  double pos = 0;  double velocity = 0.03;  Rectangle rect = new Rectangle(0, 0, 0, 0, a, a, Color.YELLOW, 0);  @Override  protected void onDraw(Canvas canvas) {  Paint paint = new Paint();  int cenX = getWidth() / 2;  int cenY = getHeight() / 2;  int r = (int)(cenX \* 0.8);  pos += velocity;  double x = cenX + r \* Math.cos(pos) - rect.getWidth() / 2;  double y = cenY + r \* Math.sin(pos) - rect.getHeight() / 2;  System.out.println(pos);  paint.setColor(Color.rgb(230, 230, 230));  paint.setStyle(Paint.Style.STROKE);  paint.setStrokeWidth(12);  canvas.drawCircle(cenX, cenY, r, paint);  paint.setStyle(Paint.Style.FILL);  paint.setColor(rect.getColor());  canvas.drawRect((float)x, (float)(y),  (float)(x + rect.getWidth()), (float)(y + rect.getHeight()), paint);  invalidate();  }  } |

***Завдання 5:***

|  |
| --- |
| SerpinskiCarpetView.java |
| package com.example.lab6;  import android.content.Context;  import android.graphics.Canvas;  import android.graphics.Color;  import android.graphics.Path;  import android.graphics.Paint;  import android.util.AttributeSet;  import android.view.View;  import androidx.annotation.Nullable;  public class SerpinskiCarpetView extends View {  public SerpinskiCarpetView(Context context) {  super(context);  init(null);  }  public SerpinskiCarpetView(Context context, AttributeSet attrs) {  super(context, attrs);  init(attrs);  }  int level = 6;  private void init(@Nullable AttributeSet set) {}  @Override  protected void onDraw(Canvas canvas) {  int size = (int)(0.9 \* getWidth());  int cenX = getWidth() / 2;  int cenY = getHeight() / 2;  Paint paint = new Paint();  paint.setColor(Color.BLACK);  drawSerpinskiCarpet(canvas, paint, level, cenX, cenY, size);  invalidate();  }  public void drawTriangle(Canvas canvas, Paint paint, int x, int y, int width) {  int halfWidth = width / 2;  Path path = new Path();  path.moveTo(x, y - halfWidth);  path.lineTo(x - halfWidth, y + halfWidth);  path.lineTo(x + halfWidth, y + halfWidth);  path.close();  canvas.drawPath(path, paint);  }  public void drawSerpinskiCarpet(Canvas canvas, Paint paint, int level, int cx, int cy, int size) {  if (level <= 1) {  drawTriangle(canvas, paint, cx, cy, size);  } else {  int dxy = size / 4;  drawSerpinskiCarpet(canvas, paint, level - 1, cx, cy - dxy, size / 2);  drawSerpinskiCarpet(canvas, paint, level - 1, cx - dxy, cy + dxy, size / 2);  drawSerpinskiCarpet(canvas, paint, level - 1, cx + dxy, cy + dxy, size / 2);  }  }  } |

**Результати роботи програми**

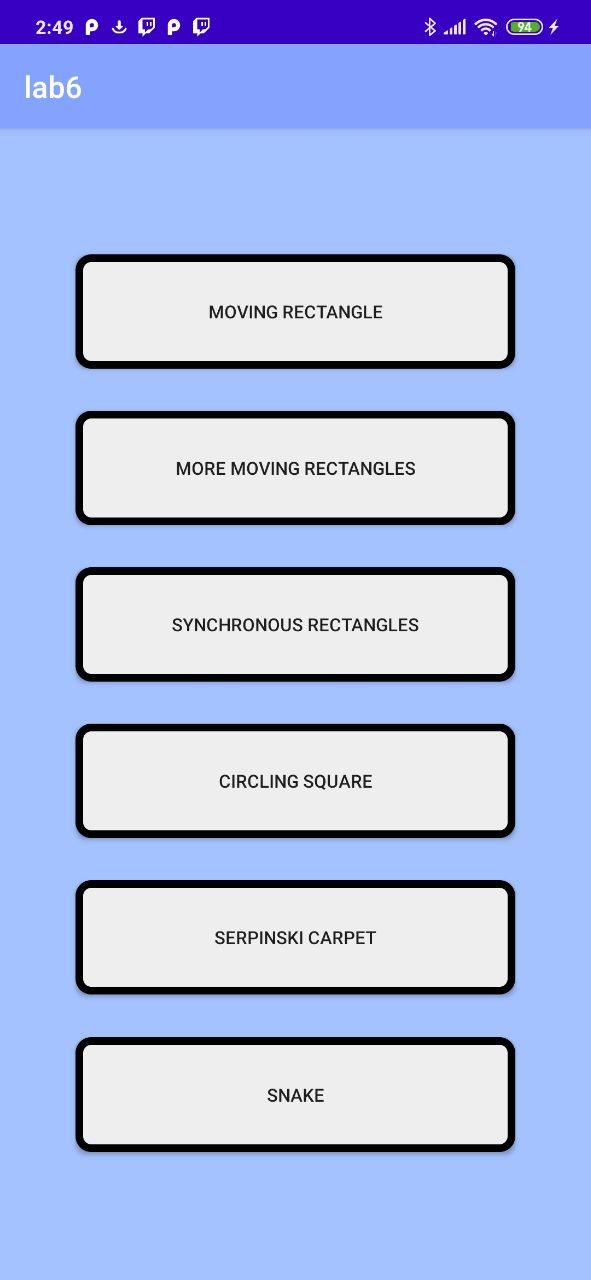
****

Рис 1.1, зовнішній вигляд головного меню

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рис 1.2, прямокутник що рухається | Рис 1.3, прямокутник що рухається | Рис 1.4, прямокутник що рухається |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рис 1.5, незалежні прямокутники, що рухаються. 2 прямокутника | Рис 1.6, незалежні прямокутники, що рухаються. 15 прямокутника | Рис 1.7, незалежні прямокутники, що рухаються. 100 прямокутника |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рис 1.8, прямокутники що рухаються синхронно та мають коло всередині | Рис 1.9, прямокутники що рухаються синхронно та мають коло всередині | Рис 1.10, прямокутники що рухаються синхронно та мають коло всередині |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Рис 1.11, квадрат що обертається по колу | Рис 1.12, квадрат що обертається по колу | Рис 1.13, квадрат що обертається по колу |

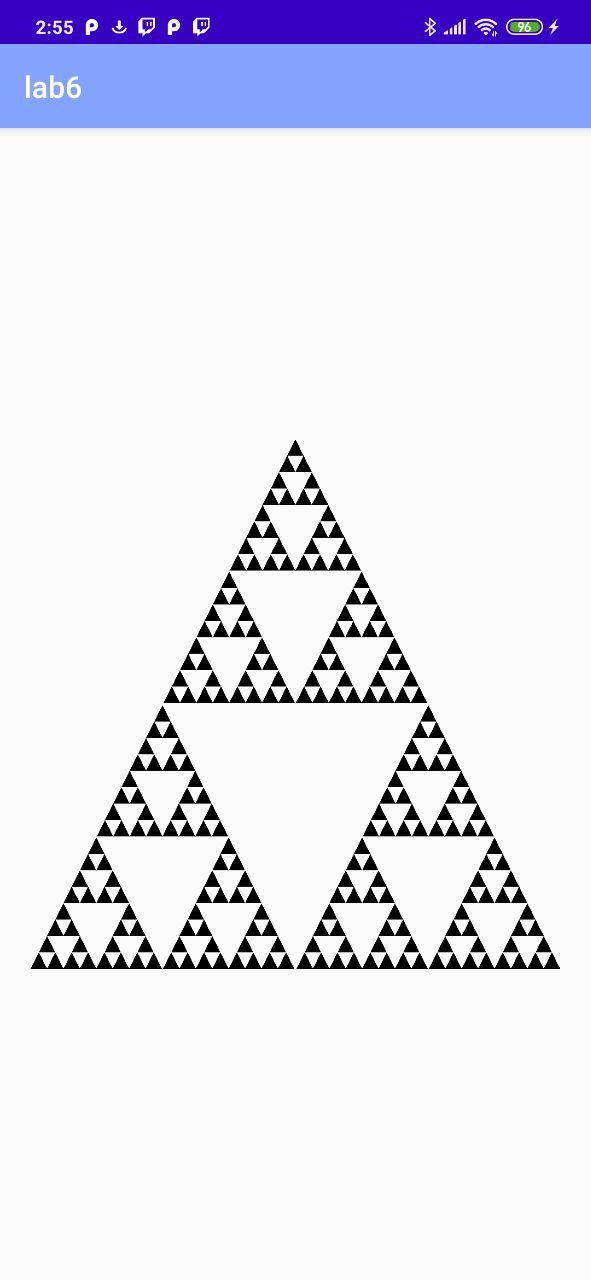


Рис 1.14, килим Серпинського

**ВИСНОВКИ**

Під час лабораторної роботи ми ознайомимилися із основними принципами створення анімацій у програмних застосунках під ОС Android та навчилися створювати анімації власних графічних елементів дизайну.