Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Комсомольский-на-Амуре государственный университет»

Факультет компьютерных технологий

Кафедра ПУРИС

Лабораторная работа №7

по дисциплине «Программирование мобильных устройств»

Студент группы 0ВТб-1 Н.Д. Малышев

Преподаватель В.А. Тихомиров

2023

1. **Разработка простейшего приложения**

По заданию необходимо, чтобы воспроизводила анимацию, приведен примерный вид экрана (рисунок 1.1) в момент работы программы Начало работы программы должно происходить по щелчку мыши на экране. Остановка программы – повторный щелчок.

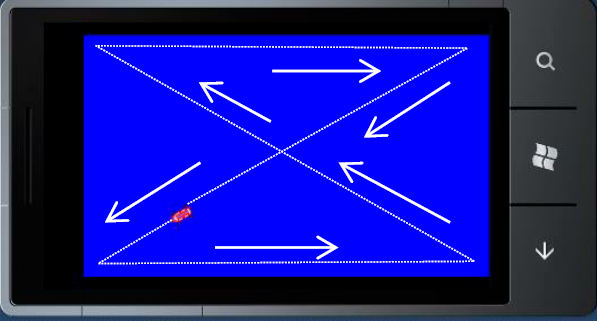


Рисунок 1.1 – Пример вид приложения

В классе MainActivity (листинг 1.1) создается метод onCreate() в котором вызывается метод setContentView(), который устанавливает View для активности, созданную с помощью конструктора MoveRaceCar.

Устанавливается ориентация экрана на LANDSCAPE.

Конструктов MoveRaceCar, который наследуется от класса View. Он используется для создания анимации движения машинки по заранее заданному пути.

В конструкторе класса инициализируются переменные и метод init() для установки значений по умолчанию. В методе init() создаются объекты mPaint, mPath, mBitmap, mPathMeasure, mMatrix, которые используются для отрисовки пути, машинки и ее перемещения.

В методе onDraw() происходит отрисовка пути, определение перемещения машинки и ее поворота. Если машинка не дошла до конца пути, то она перемещается по пути с помощью метода getPosTan() класса PathMeasure и поворачивается в направлении движения. Если машинка дошла до конца пути, то перемещение начинается сначала.

В методе onTouchEvent() происходит обработка касания экрана. Если произошло касание, то устанавливается значение переменной mStep, которая отвечает за длину шага машинки. Если mStep равна 40, то значение устанавливается в 0, а если она равна 0, то значение устанавливается в 40.

В целом, данный класс используется для создания анимации движения машинки по пути с возможностью управления длиной шага.

Листинг 1.1 – MainActivity

package ru.lab\_7;  
  
import android.content.ContentValues;  
import android.content.Context;  
import android.content.Intent;  
import android.content.pm.ActivityInfo;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.CornerPathEffect;  
import android.graphics.Matrix;  
import android.graphics.Paint;  
import android.graphics.Path;  
import android.graphics.PathMeasure;  
import android.graphics.Rect;  
import android.os.Bundle;  
import android.util.AttributeSet;  
import android.view.MotionEvent;  
import android.view.View;  
  
  
import androidx.appcompat.app.AppCompatActivity;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 public void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(new MoveRaceCar(this));  
 setRequestedOrientation(ActivityInfo.SCREEN\_ORIENTATION\_LANDSCAPE);  
 }  
}

Листинг 1.2 – MoveRaceCar

package ru.lab\_7;  
  
import android.content.Context;  
import android.graphics.Bitmap;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.CornerPathEffect;  
import android.graphics.Matrix;  
import android.graphics.Paint;  
import android.graphics.Path;  
import android.graphics.PathMeasure;  
import android.util.AttributeSet;  
import android.view.MotionEvent;  
import android.view.View;  
  
  
public class MoveRaceCar extends View {  
 private Paint mPaint;  
 private Path mPath;  
 private Bitmap mBitmap;  
 private PathMeasure mPathMeasure;  
 private Matrix mMatrix;  
 private int mOffsetX, mOffsetY;  
 private float mPathLength;  
 private float mStep; // distance each step  
 private float mDistance; // distance moved  
 private float[] mPosition;  
 private float[] mTan;  
 private float mCurX, mCurY;  
 private float mCurAngle; // current angle  
 private float mTargetAngle; // target angle  
 private float mStepAngle; // angle each step  
  
 public MoveRaceCar(Context context) {  
 super(context);  
 init();  
 }  
  
 public MoveRaceCar(Context context, AttributeSet attrs) {  
 super(context, attrs);  
 init();  
 }  
  
 public MoveRaceCar(Context context, AttributeSet attrs, int defStyle) {  
 super(context, attrs, defStyle);  
 init();  
 }  
  
 private void init() {  
 mPaint = new Paint();  
 mPaint.setAntiAlias(true);  
 mPaint.setColor(Color.DKGRAY);  
 mPaint.setStrokeWidth(100);  
 mPaint.setStyle(Paint.Style.STROKE);  
  
 mBitmap = BitmapFactory.decodeResource(getResources(), R.drawable.car1);  
  
 mOffsetX = mBitmap.getWidth() / 2;  
 mOffsetY = mBitmap.getHeight() / 2;  
  
 mPath = new Path();  
 float radius = 40.0f;  
 CornerPathEffect mCornerPathEffect = new CornerPathEffect(radius);  
 mPaint.setPathEffect(mCornerPathEffect);  
  
 mPath.moveTo(100, 100);  
 mPath.lineTo(2100, 100);  
 mPath.lineTo(100, 900);  
 mPath.lineTo(2100, 900);  
  
 mPath.close();  
  
 mPathMeasure = new PathMeasure(mPath, false);  
 mPathLength = mPathMeasure.getLength();  
  
 mStep = 0;  
 mDistance = 0;  
 mCurX = 0;  
 mCurY = 0;  
  
 mStepAngle = 20;  
 mCurAngle = 0;  
 mTargetAngle = 0;  
  
 mPosition = new float[2];  
 mTan = new float[2];  
  
 mMatrix = new Matrix();  
  
 }  
  
 @Override  
 protected void onDraw(Canvas canvas) {  
 super.onDraw(canvas);  
  
 canvas.drawPath(mPath, mPaint);  
 mMatrix.reset();  
  
 if ((mTargetAngle - mCurAngle) > mStepAngle) {  
 mCurAngle += mStepAngle;  
 mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);  
 mMatrix.postTranslate(mCurX, mCurY);  
 canvas.drawBitmap(mBitmap, mMatrix, null);  
  
  
 } else if ((mCurAngle - mTargetAngle) > mStepAngle) {  
 mCurAngle -= mStepAngle;  
 mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);  
 mMatrix.postTranslate(mCurX, mCurY);  
 canvas.drawBitmap(mBitmap, mMatrix, null);  
  
  
 } else {  
 mCurAngle = mTargetAngle;  
 if (mDistance < mPathLength) {  
 mPathMeasure.getPosTan(mDistance, mPosition, mTan);  
  
 mTargetAngle = (float) (Math.atan2(mTan[1], mTan[0]) \* 180.0 / Math.PI);  
 mMatrix.postRotate(mCurAngle, mOffsetX, mOffsetY);  
  
 mCurX = mPosition[0] - mOffsetX;  
 mCurY = mPosition[1] - mOffsetY;  
 mMatrix.postTranslate(mCurX, mCurY);  
  
 canvas.drawBitmap(mBitmap, mMatrix, null);  
  
 mDistance += mStep;  
 } else {  
 mDistance = 0;  
 }  
 }  
  
 invalidate();  
 }  
  
 @Override  
 public boolean onTouchEvent(MotionEvent event) {  
  
 int action = event.getAction();  
  
 switch (action) {  
 case MotionEvent.ACTION\_DOWN:  
  
 if (mStep == 40){  
 mStep = 0;  
 } else if (mStep == 0) {  
 mStep = 40;  
 }  
  
 break;  
 }  
 return true;  
 }  
}

Результаты работа представлены на рисунке 1.2 – 1.3.

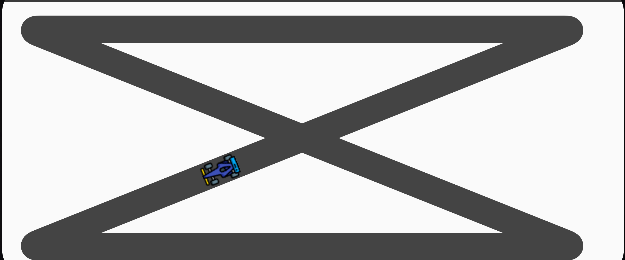


Рисунок 1.2 – Пример работы приложения

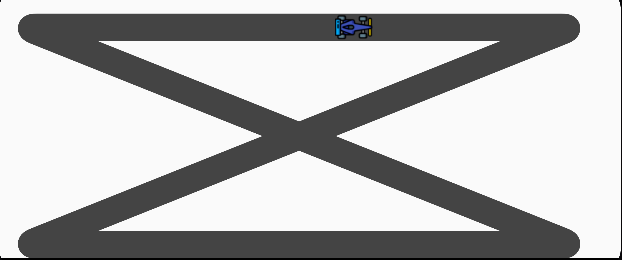


Рисунок 1.3 – Пример работы приложения

# Список использованных источников

1. РД ФГБОУ ВО «КнАГТУ» 013-2016. Текстовые студенческие работы. Правила оформления. – Введ. 2016-03-10. – Комсомольск-на-Амуре: ФГБОУ ВО «КнАГТУ», 2016. – 55 с.