МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

СХІДНОУКРАЇНСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

імені ВОЛОДИМИРА ДАЛЯ

Звіт

з лабораторної роботи № 4, 5

з теми: «**Разработка игрового приложения**»

Виконав студент:

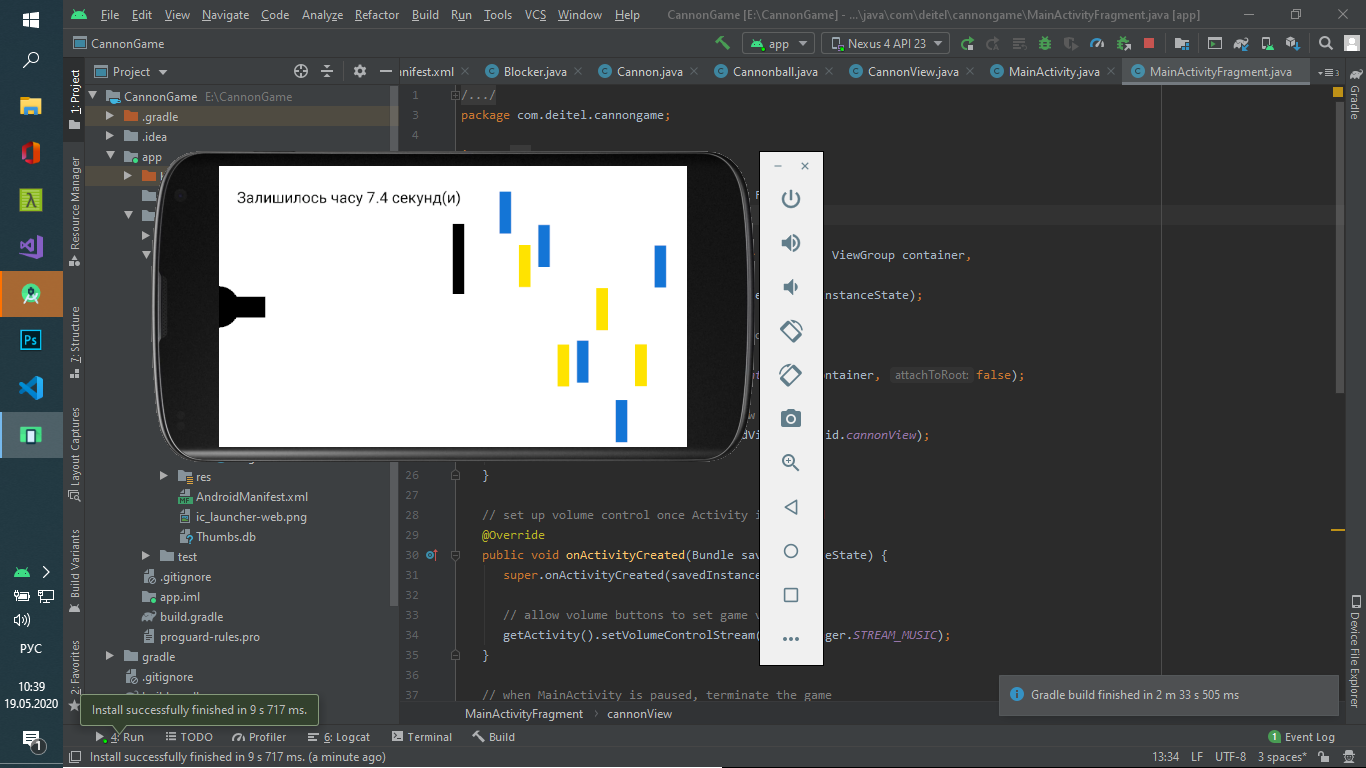
Група КН-18д

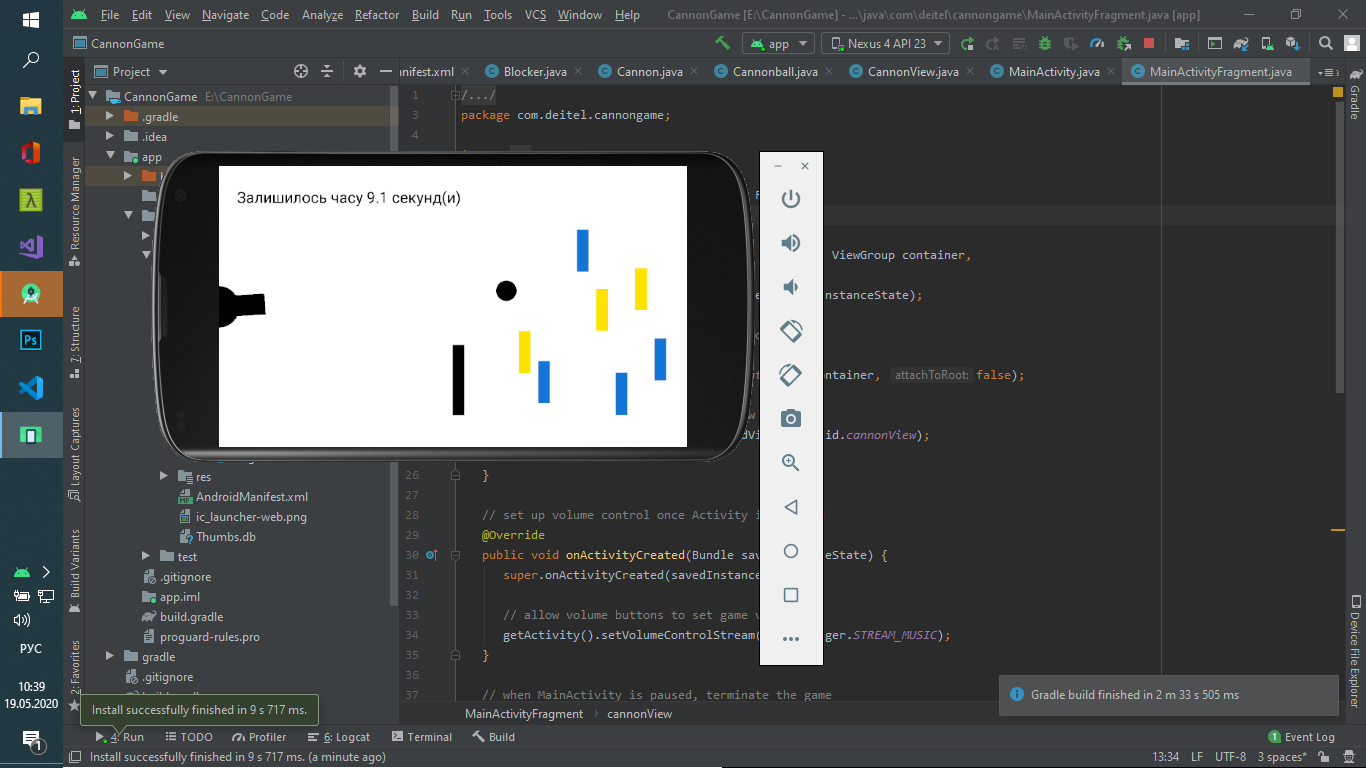
Скороход С.Г.

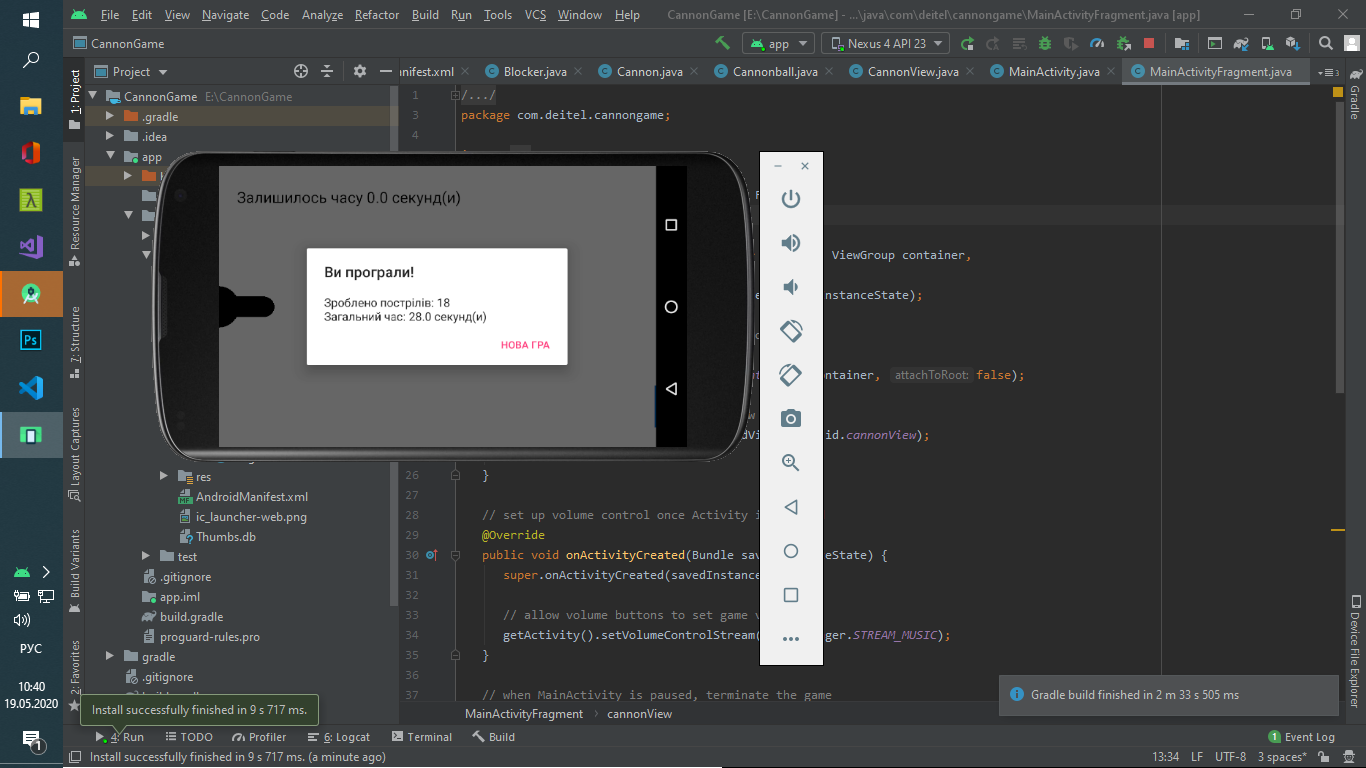
Дата 10.05.2020

**м. Северодонецьк**

2020

****

****

****

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>  
<fragment  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity"  
 android:id="@+id/fragment"  
 android:name="com.deitel.cannongame.MainActivityFragment"  
 tools:layout="@layout/fragment\_main">  
  
</fragment>

**fragment\_main.xml**

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context="com.deitel.cannongame.MainActivityFragment">  
  
  
 <com.deitel.cannongame.CannonView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:id="@+id/cannonView"/>  
</FrameLayout>

**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 package="com.deitel.cannongame" >  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:supportsRtl="true"  
 android:theme="@style/AppTheme" >  
 <activity android:name=".MainActivity"  
 android:screenOrientation = "landscape">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
</manifest>

**Blocker.java**

package com.deitel.cannongame;  
  
public class Blocker extends GameElement {  
 private int missPenalty;   
  
 public Blocker(CannonView view, int color, int missPenalty, int x,  
 int y, int width, int length, float velocityY) {  
 super(view, color, CannonView.*BLOCKER\_SOUND\_ID*, x, y, width, length,  
 velocityY);  
 this.missPenalty = missPenalty;  
 }  
  
  
 public int getMissPenalty() {  
 return missPenalty;  
 }  
}

**Cannon.java**

package com.deitel.cannongame;  
  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Point;  
  
public class Cannon {  
 private int baseRadius;   
 private int barrelLength;   
 private Point barrelEnd = new Point();  
 private double barrelAngle;   
 private Cannonball cannonball;   
 private Paint paint = new Paint();  
 private CannonView view;   
  
 // constructor  
 public Cannon(CannonView view, int baseRadius, int barrelLength,  
 int barrelWidth) {  
 this.view = view;  
 this.baseRadius = baseRadius;  
 this.barrelLength = barrelLength;  
 paint.setStrokeWidth(barrelWidth);   
 paint.setColor(Color.*BLACK*);   
 align(Math.*PI* / 2);   
 }  
  
  
 public void align(double barrelAngle) {  
 this.barrelAngle = barrelAngle;  
 barrelEnd.x = (int) (barrelLength \* Math.*sin*(barrelAngle));  
 barrelEnd.y = (int) (-barrelLength \* Math.*cos*(barrelAngle)) +  
 view.getScreenHeight() / 2;  
 }  
  
  
 public void fireCannonball() {  
 int velocityX = (int) (CannonView.*CANNONBALL\_SPEED\_PERCENT* \*  
 view.getScreenWidth() \* Math.*sin*(barrelAngle));  
  
 int velocityY = (int) (CannonView.*CANNONBALL\_SPEED\_PERCENT* \*  
 view.getScreenWidth() \* -Math.*cos*(barrelAngle));  
  
 int radius = (int) (view.getScreenHeight() \*  
 CannonView.*CANNONBALL\_RADIUS\_PERCENT*);  
  
 cannonball = new Cannonball(view, Color.*BLACK*,  
 CannonView.*CANNON\_SOUND\_ID*, -radius,  
 view.getScreenHeight() / 2 - radius, radius, velocityX,  
 velocityY);  
  
 cannonball.playSound();  
 }  
 public void draw(Canvas canvas) {  
  
 canvas.drawLine(0, view.getScreenHeight() / 2, barrelEnd.x,  
 barrelEnd.y, paint);  
  
 canvas.drawCircle(0, (int) view.getScreenHeight() / 2,  
 (int) baseRadius, paint);  
 }  
  
 public Cannonball getCannonball() {  
 return cannonball;  
 }  
  
 public void removeCannonball() {  
 cannonball = null;  
 }  
}

**Cannonball.java**

package com.deitel.cannongame;  
  
import android.graphics.Canvas;  
import android.graphics.Rect;  
  
public class Cannonball extends GameElement {  
 private float velocityX;  
 private boolean onScreen;  
  
 public Cannonball(CannonView view, int color, int soundId, int x,  
 int y, int radius, float velocityX, float velocityY) {  
 super(view, color, soundId, x, y,  
 2 \* radius, 2 \* radius, velocityY);  
 this.velocityX = velocityX;  
 onScreen = true;  
 }  
  
 private int getRadius() {  
 return (shape.right - shape.left) / 2;  
 }  
  
 public boolean collidesWith(GameElement element) {  
 return (Rect.*intersects*(shape, element.shape) && velocityX > 0);  
 }  
  
 public boolean isOnScreen() {  
 return onScreen;  
 }  
  
 public void reverseVelocityX() {  
 velocityX \*= -1;  
 }  
  
 @Override  
 public void update(double interval) {  
 super.update(interval);   
  
 shape.offset((int) (velocityX \* interval), 0);  
  
 if (shape.top < 0 || shape.left < 0 ||  
 shape.bottom > view.getScreenHeight() ||  
 shape.right > view.getScreenWidth())  
 onScreen = false;   
 }  
  
 @Override  
 public void draw(Canvas canvas) {  
 canvas.drawCircle(shape.left + getRadius(),  
 shape.top + getRadius(), getRadius(), paint);  
 }  
}

**CannonView.java**

package com.deitel.cannongame;  
  
import android.app.Activity;  
import android.app.AlertDialog;  
import android.app.Dialog;  
import android.app.DialogFragment;  
import android.content.Context;  
import android.content.DialogInterface;  
import android.graphics.Canvas;  
import android.graphics.Color;  
import android.graphics.Paint;  
import android.graphics.Point;  
import android.media.AudioAttributes;  
import android.media.SoundPool;  
import android.os.Build;  
import android.os.Bundle;  
import android.util.AttributeSet;  
import android.util.Log;  
import android.util.SparseIntArray;  
import android.view.MotionEvent;  
import android.view.SurfaceHolder;  
import android.view.SurfaceView;  
import android.view.View;  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class CannonView extends SurfaceView  
 implements SurfaceHolder.Callback {  
  
 private static final String *TAG* = "CannonView";  
  
  
 public static final int *MISS\_PENALTY* = 2;  
 public static final int *HIT\_REWARD* = 3;  
  
  
 public static final double *CANNON\_BASE\_RADIUS\_PERCENT* = 3.0 / 40;  
 public static final double *CANNON\_BARREL\_WIDTH\_PERCENT* = 3.0 / 40;  
 public static final double *CANNON\_BARREL\_LENGTH\_PERCENT* = 1.0 / 10;  
  
  
 public static final double *CANNONBALL\_RADIUS\_PERCENT* = 3.0 / 80;  
 public static final double *CANNONBALL\_SPEED\_PERCENT* = 3.0 / 2;  
  
  
 public static final double *TARGET\_WIDTH\_PERCENT* = 1.0 / 40;  
 public static final double *TARGET\_LENGTH\_PERCENT* = 3.0 / 20;  
 public static final double *TARGET\_FIRST\_X\_PERCENT* = 3.0 / 5;  
 public static final double *TARGET\_SPACING\_PERCENT* = 1.0 / 60;  
 public static final double *TARGET\_PIECES* = 9;  
 public static final double *TARGET\_MIN\_SPEED\_PERCENT* = 3.0 / 4;  
 public static final double *TARGET\_MAX\_SPEED\_PERCENT* = 6.0 / 4;  
  
  
 public static final double *BLOCKER\_WIDTH\_PERCENT* = 1.0 / 40;  
 public static final double *BLOCKER\_LENGTH\_PERCENT* = 1.0 / 4;  
 public static final double *BLOCKER\_X\_PERCENT* = 1.0 / 2;  
 public static final double *BLOCKER\_SPEED\_PERCENT* = 1.0;  
  
  
 public static final double *TEXT\_SIZE\_PERCENT* = 1.0 / 18;  
  
 private CannonThread cannonThread;  
 private Activity activity;  
 private boolean dialogIsDisplayed = false;  
  
 private Cannon cannon;  
 private Blocker blocker;  
 private ArrayList<Target> targets;  
  
  
 private int screenWidth;  
 private int screenHeight;  
  
 private boolean gameOver;  
 private double timeLeft;  
 private int shotsFired;  
 private double totalElapsedTime;  
  
  
 public static final int *TARGET\_SOUND\_ID* = 0;  
 public static final int *CANNON\_SOUND\_ID* = 1;  
 public static final int *BLOCKER\_SOUND\_ID* = 2;  
 private SoundPool soundPool;  
 private SparseIntArray soundMap;  
  
 private Paint textPaint;   
 private Paint backgroundPaint;   
 public CannonView(Context context, AttributeSet attrs) {  
 super(context, attrs);   
 activity = (Activity) context;   
  
 getHolder().addCallback(this);  
  
 AudioAttributes.Builder attrBuilder = new AudioAttributes.Builder();  
 attrBuilder.setUsage(AudioAttributes.*USAGE\_GAME*);  
  
 SoundPool.Builder builder = new SoundPool.Builder();  
 builder.setMaxStreams(1);  
 builder.setAudioAttributes(attrBuilder.build());  
 soundPool = builder.build();  
  
 soundMap = new SparseIntArray(3);   
 soundMap.put(*TARGET\_SOUND\_ID*,  
 soundPool.load(context, R.raw.*target\_hit*, 1));  
 soundMap.put(*CANNON\_SOUND\_ID*,  
 soundPool.load(context, R.raw.*cannon\_fire*, 1));  
 soundMap.put(*BLOCKER\_SOUND\_ID*,  
 soundPool.load(context, R.raw.*blocker\_hit*, 1));  
  
 textPaint = new Paint();  
 backgroundPaint = new Paint();  
 backgroundPaint.setColor(Color.*WHITE*);  
 }  
  
 @Override  
 protected void onSizeChanged(int w, int h, int oldw, int oldh) {  
 super.onSizeChanged(w, h, oldw, oldh);  
  
 screenWidth = w;   
 screenHeight = h;   
  
 textPaint.setTextSize((int) (*TEXT\_SIZE\_PERCENT* \* screenHeight));  
 textPaint.setAntiAlias(true);   
 }  
  
 public int getScreenWidth() {  
 return screenWidth;  
 }  
  
 public int getScreenHeight() {  
 return screenHeight;  
 }  
  
 public void playSound(int soundId) {  
 soundPool.play(soundMap.get(soundId), 1, 1, 1, 0, 1f);  
 }  
  
  
 public void newGame() {  
  
 cannon = new Cannon(this,  
 (int) (*CANNON\_BASE\_RADIUS\_PERCENT* \* screenHeight),  
 (int) (*CANNON\_BARREL\_LENGTH\_PERCENT* \* screenWidth),  
 (int) (*CANNON\_BARREL\_WIDTH\_PERCENT* \* screenHeight));  
  
 Random random = new Random();  
 targets = new ArrayList<>();   
  
 int targetX = (int) (*TARGET\_FIRST\_X\_PERCENT* \* screenWidth);  
  
 int targetY = (int) ((0.5 - *TARGET\_LENGTH\_PERCENT* / 2) \*  
 screenHeight);  
  
  
 for (int n = 0; n < *TARGET\_PIECES*; n++) {  
  
 double velocity = screenHeight \* (random.nextDouble() \*  
 (*TARGET\_MAX\_SPEED\_PERCENT* - *TARGET\_MIN\_SPEED\_PERCENT*) +  
 *TARGET\_MIN\_SPEED\_PERCENT*);  
  
 int color = (n % 2 == 0) ?  
 getResources().getColor(R.color.*dark*) :  
 getResources().getColor(R.color.*light*);  
  
  
 velocity \*= -1;   
  
 targets.add(new Target(this, color, *HIT\_REWARD*, targetX, targetY,  
 (int) (*TARGET\_WIDTH\_PERCENT* \* screenWidth),  
 (int) (*TARGET\_LENGTH\_PERCENT* \* screenHeight),  
 (int) velocity));  
  
 targetX += (*TARGET\_WIDTH\_PERCENT* + *TARGET\_SPACING\_PERCENT*) \*  
 screenWidth;  
 }  
 blocker = new Blocker(this, Color.*BLACK*, *MISS\_PENALTY*,  
 (int) (*BLOCKER\_X\_PERCENT* \* screenWidth),  
 (int) ((0.5 - *BLOCKER\_LENGTH\_PERCENT* / 2) \* screenHeight),  
 (int) (*BLOCKER\_WIDTH\_PERCENT* \* screenWidth),  
 (int) (*BLOCKER\_LENGTH\_PERCENT* \* screenHeight),  
 (float) (*BLOCKER\_SPEED\_PERCENT* \* screenHeight));  
  
 timeLeft = 10;   
  
 shotsFired = 0;   
 totalElapsedTime = 0.0;   
 if (gameOver) {   
 gameOver = false;   
 cannonThread = new CannonThread(getHolder());  
 cannonThread.start();  
 }  
  
 hideSystemBars();  
 }  
  
 private void updatePositions(double elapsedTimeMS) {  
 double interval = elapsedTimeMS / 1000.0;   
  
 if (cannon.getCannonball() != null)  
 cannon.getCannonball().update(interval);  
  
 blocker.update(interval);   
  
 for (GameElement target : targets)  
 target.update(interval);   
  
 timeLeft -= interval;   
  
 if (timeLeft <= 0) {  
 timeLeft = 0.0;  
 gameOver = true;   
 cannonThread.setRunning(false);   
 showGameOverDialog(R.string.*lose*);   
 }  
  
 if (targets.isEmpty()) {  
 cannonThread.setRunning(false);   
 showGameOverDialog(R.string.*win*);   
 gameOver = true;  
 }  
 }  
  
  
 public void alignAndFireCannonball(MotionEvent event) {  
 Point touchPoint = new Point((int) event.getX(),  
 (int) event.getY());  
  
 double centerMinusY = (screenHeight / 2 - touchPoint.y);  
  
 double angle = 0;   
  
 angle = Math.*atan2*(touchPoint.x, centerMinusY);  
  
 cannon.align(angle);  
  
 if (cannon.getCannonball() == null ||  
 !cannon.getCannonball().isOnScreen()) {  
 cannon.fireCannonball();  
 ++shotsFired;  
 }  
 }  
  
 private void showGameOverDialog(final int messageId) {  
  
 final DialogFragment gameResult =  
 new DialogFragment() {  
  
 @Override  
 public Dialog onCreateDialog(Bundle bundle) {  
  
 AlertDialog.Builder builder =  
 new AlertDialog.Builder(getActivity());  
 builder.setTitle(getResources().getString(messageId));  
  
 builder.setMessage(getResources().getString(  
 R.string.*results\_format*, shotsFired, totalElapsedTime));  
 builder.setPositiveButton(R.string.*reset\_game*,  
 new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog,  
 int which) {  
 dialogIsDisplayed = false;  
 newGame();  
 }  
 }  
 );  
  
 return builder.create();  
 }  
 };  
  
 activity.runOnUiThread(  
 new Runnable() {  
 public void run() {  
 showSystemBars();  
 dialogIsDisplayed = true;  
 gameResult.setCancelable(false);   
 gameResult.show(activity.getFragmentManager(), "results");  
 }  
 }  
 );  
 }  
  
  
 public void drawGameElements(Canvas canvas) {  
  
 canvas.drawRect(0, 0, canvas.getWidth(), canvas.getHeight(),  
 backgroundPaint);  
  
 canvas.drawText(getResources().getString(  
 R.string.*time\_remaining\_format*, timeLeft), 50, 100, textPaint);  
  
 cannon.draw(canvas);   
  
 if (cannon.getCannonball() != null &&  
 cannon.getCannonball().isOnScreen())  
 cannon.getCannonball().draw(canvas);  
  
 blocker.draw(canvas);   
  
 for (GameElement target : targets)  
 target.draw(canvas);  
 }  
  
 public void testForCollisions() {  
  
 if (cannon.getCannonball() != null &&  
 cannon.getCannonball().isOnScreen()) {  
 for (int n = 0; n < targets.size(); n++) {  
 if (cannon.getCannonball().collidesWith(targets.get(n))) {  
 targets.get(n).playSound();  
  
 timeLeft += targets.get(n).getHitReward();  
  
 cannon.removeCannonball();  
 targets.remove(n);

--n;   
 break;  
 }  
 }  
 }  
 else {   
 cannon.removeCannonball();  
 }  
  
 if (cannon.getCannonball() != null &&  
 cannon.getCannonball().collidesWith(blocker)) {  
 blocker.playSound(); // play Blocker hit sound  
  
 cannon.getCannonball().reverseVelocityX();  
  
 timeLeft -= blocker.getMissPenalty();  
 }  
 }  
  
  
 public void stopGame() {  
 if (cannonThread != null)  
 cannonThread.setRunning(false);   
 }  
  
  
 public void releaseResources() {  
 soundPool.release();  
 soundPool = null;  
 }  
  
  
 @Override  
 public void surfaceChanged(SurfaceHolder holder, int format,  
 int width, int height) { }  
  
   
 @Override  
 public void surfaceCreated(SurfaceHolder holder) {  
 if (!dialogIsDisplayed) {  
 newGame();  
 cannonThread = new CannonThread(holder);   
 cannonThread.setRunning(true);   
 cannonThread.start();  
 }  
 }  
  
  
 @Override  
 public void surfaceDestroyed(SurfaceHolder holder) {  
 boolean retry = true;  
 cannonThread.setRunning(false);   
  
 while (retry) {  
 try {  
 cannonThread.join();  
 retry = false;  
 }  
 catch (InterruptedException e) {  
 Log.*e*(*TAG*, "Thread interrupted", e);  
 }  
 }  
 }  
  
 @Override  
 public boolean onTouchEvent(MotionEvent e) {  
 int action = e.getAction();  
  
  
 if (action == MotionEvent.*ACTION\_DOWN* ||  
 action == MotionEvent.*ACTION\_MOVE*) {  
  
 alignAndFireCannonball(e);  
 }  
  
 return true;  
 }  
  
 private class CannonThread extends Thread {  
 private SurfaceHolder surfaceHolder;   
 private boolean threadIsRunning = true;   
  
  
 public CannonThread(SurfaceHolder holder) {  
 surfaceHolder = holder;  
 setName("CannonThread");  
 }  
  
  
 public void setRunning(boolean running) {  
 threadIsRunning = running;  
 }  
  
 @Override  
 public void run() {  
 Canvas canvas = null;   
 long previousFrameTime = System.*currentTimeMillis*();  
  
 while (threadIsRunning) {  
 try {  
  
 canvas = surfaceHolder.lockCanvas(null);  
  
 synchronized(surfaceHolder) {  
 long currentTime = System.*currentTimeMillis*();  
 double elapsedTimeMS = currentTime - previousFrameTime;  
 totalElapsedTime += elapsedTimeMS / 1000.0;  
 updatePositions(elapsedTimeMS);   
 testForCollisions();  
 drawGameElements(canvas);   
 previousFrameTime = currentTime;   
 }  
 }  
 finally {  
  
 if (canvas != null)  
 surfaceHolder.unlockCanvasAndPost(canvas);  
 }  
 }  
 }  
 }  
  
  
 private void hideSystemBars() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*)  
 setSystemUiVisibility(  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE* |  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_HIDE\_NAVIGATION* |  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN* |  
 View.*SYSTEM\_UI\_FLAG\_HIDE\_NAVIGATION* |  
 View.*SYSTEM\_UI\_FLAG\_FULLSCREEN* |  
 View.*SYSTEM\_UI\_FLAG\_IMMERSIVE*);  
 }  
  
  
 private void showSystemBars() {  
 if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*)  
 setSystemUiVisibility(  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_STABLE* |  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_HIDE\_NAVIGATION* |  
 View.*SYSTEM\_UI\_FLAG\_LAYOUT\_FULLSCREEN*);  
 }  
}

**GameElement.java**

package com.deitel.cannongame;  
  
import android.graphics.Canvas;  
import android.graphics.Paint;  
import android.graphics.Rect;  
  
public class GameElement {  
 protected CannonView view;   
 protected Paint paint = new Paint();  
 protected Rect shape;   
 private float velocityY;   
 private int soundId;   
  
 public GameElement(CannonView view, int color, int soundId, int x,  
 int y, int width, int length, float velocityY) {  
 this.view = view;  
 paint.setColor(color);  
 shape = new Rect(x, y, x + width, y + length);   
 this.soundId = soundId;  
 this.velocityY = velocityY;  
 }  
  
 public void update(double interval) {  
 shape.offset(0, (int) (velocityY \* interval));  
  
 if (shape.top < 0 && velocityY < 0 ||  
 shape.bottom > view.getScreenHeight() && velocityY > 0)  
 velocityY \*= -1;   
 }  
  
  
 public void draw(Canvas canvas) {  
 canvas.drawRect(shape, paint);  
 }  
  
 public void playSound() {  
 view.playSound(soundId);  
 }  
}

**MainActivity.java**

package com.deitel.cannongame;  
  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
 }  
}

**MainActivityFragment.java**

package com.deitel.cannongame;  
  
import android.media.AudioManager;  
import android.os.Bundle;  
import android.support.v4.app.Fragment;  
import android.view.LayoutInflater;  
import android.view.View;  
import android.view.ViewGroup;  
  
public class MainActivityFragment extends Fragment {  
 private CannonView cannonView;  
 @Override  
 public View onCreateView(LayoutInflater inflater, ViewGroup container,  
 Bundle savedInstanceState) {  
 super.onCreateView(inflater, container, savedInstanceState);  
  
  
 View view =  
 inflater.inflate(R.layout.*fragment\_main*, container, false);  
  
 cannonView = (CannonView) view.findViewById(R.id.*cannonView*);  
 return view;  
 }  
  
  
 @Override  
 public void onActivityCreated(Bundle savedInstanceState) {  
 super.onActivityCreated(savedInstanceState);  
  
 getActivity().setVolumeControlStream(AudioManager.*STREAM\_MUSIC*);  
 }  
  
  
 @Override  
 public void onPause() {  
 super.onPause();  
 cannonView.stopGame();  
 }  
  
  
 @Override  
 public void onDestroy() {  
 super.onDestroy();  
 cannonView.releaseResources();  
 }  
}

**Target.java**

package com.deitel.cannongame;  
  
public class Target extends GameElement {  
 private int hitReward;   
  
 // constructor  
 public Target(CannonView view, int color, int hitReward, int x, int y,  
 int width, int length, float velocityY) {  
 super(view, color, CannonView.*TARGET\_SOUND\_ID*, x, y, width, length,  
 velocityY);  
 this.hitReward = hitReward;  
 }  
  
 public int getHitReward() {  
 return hitReward;  
 }  
}

**strings.xml**

<resources>  
 <string name="app\_name">Гра "Гармата"</string>  
 <string name="results\_format">Зроблено пострілів: %1$d\nЗагальний час: %2$.1f секунд(и)</string>  
 <string name="reset\_game">Нова гра</string>  
 <string name="win">Ви виграли!</string>  
 <string name="lose">Ви програли!</string>  
 <string name="time\_remaining\_format">Залишилось часу %.1f секунд(и)</string>  
   
</resources>

**colors.xml**

<?xml version="1.0" encoding="utf-8"?>  
<resources>  
 <color name="colorPrimary">#3F51B5</color>  
 <color name="colorPrimaryDark">#303F9F</color>  
 <color name="colorAccent">#FF4081</color>  
 <color name="dark">#1976D2</color>  
 <color name="light">#FFE100</color>  
</resources>

**styles.xml**

<resources>  
  
 <style name="AppTheme" parent="Theme.AppCompat.Light">  
 <item name="colorPrimary">@color/colorPrimary</item>  
 <item name="colorPrimaryDark">@color/colorPrimaryDark</item>  
 <item name="colorAccent">@color/colorAccent</item>  
 <item name="windowNoTitle">true</item>  
 <item name="windowActionBar">false</item>  
 <item name="android:windowFullscreen">true</item>  
 </style>  
  
</resources>

**Висновок:** Навчився розробляти прості ігрові додатки для мобільної платформи Android.