Отчёт по индивидуальному практическому заданию №2

Цель контрольной работы.

Целью выполнения индивидуальной практической работы является закрепление материала теоретического курса. Работа с простыми графическими примитивами. Управление касаниями сенсорного экрана.

Задание

В соответствии с индивидуальным заданием создать простое игровое приложение, использующие возможности встроенной графической библиотеки Skia

Вариант 4.

То же, что и вариант-1, только шарик управляется не касаниями экрана, а наклонами устройства (использование встроенных сенсоров)

Вариант 1: Касаниями корректировать движения шарика на экране. Чтобы попасть в цель, на противоположном конце экранной области.

Исходный код (Java, xml) есть на **гит-хабе** (к сожалению ограничение в 5мб не даёт скинуть код через lms):

https://github.com/ElephantT/Android/tree/main/game_two_dimensions

Или открыть https://github.com/ElephantT/Android

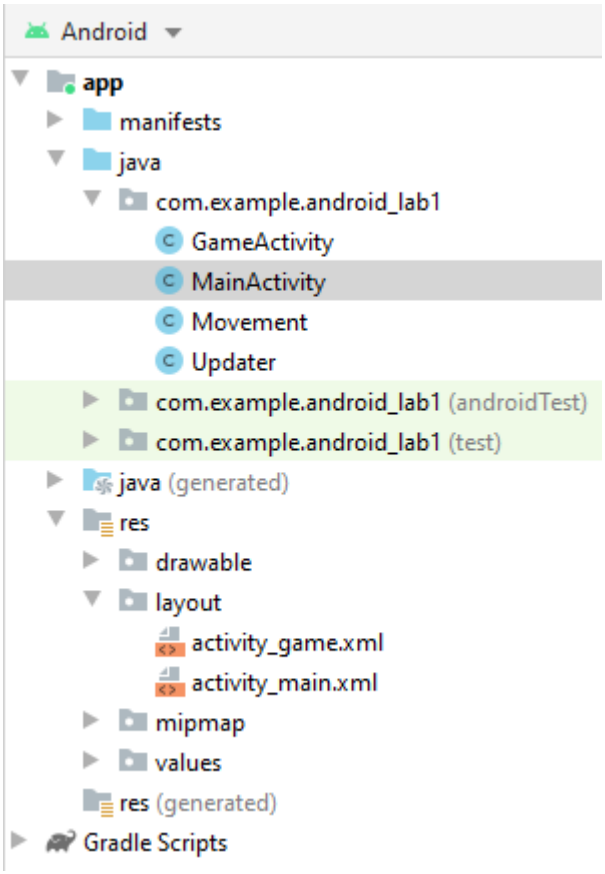
А там уже выбрать нужную лабораторную (но должна и первая ссылка работать всегда)

Видео работы (запуск приложения, успешное прохождение игры, выход из приложения):

- там же на гит-хабе (весит чуть больше 5мб, поэтому не получилось его передать так)

Изображения работы:

- будут присутствовать в отчёте: либо фотографии кода из самой AndroidStudio, либо фотографии работы приложения (лучше посмотреть видео с гит-хаба)



Структура проекта:

MainActivity - поддерживает главное меню для запуска активности игры и выхода из приложения (главная активность)

GameActivity - запускает процесс игры

Movement - класс, отвечающий за весь основной игровой функционал (работа с сенсорами, физикой в игре, стартовой позиции, условиями победы в игре и т.д.)

Updater - класс, отвечающий за обновления в игре в зависимости от того, что нам передают сенсоры

MainActivity

создание активности с обозначением ориентации

обработка результата, который приходит из игровой активности

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRequestedOrientation (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
    setContentView(R.layout.activity_main);
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data){
    if(requestCode == 1) {
        if(resultCode == RESULT_OK) {
            boolean result = data.getExtras().getBoolean( key: "win");
            if (!result) {
                Toast.makeText(getApplicationContext(),  text: "You've lost :(",
                        Toast.LENGTH_SHORT).show();
            }
            else {
                Toast.makeText(getApplicationContext(),  text: "You've just won!" ,
                        Toast.LENGTH_SHORT).show();
            }
        }
        else{
            Toast.makeText(getApplicationContext(),  text: "GAME_CANCELED",
                    Toast.LENGTH_SHORT).show();
        }
    }
    else{
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

функционал мною добавленных кнопок

```
public class MainActivity extends AppCompatActivity {
    public void Start(View V) {
        Intent intent = new Intent( packageContext: MainActivity.this, GameActivity.class);
        startActivityForResult(intent,  requestCode: 1);
    }

    public void Exit(View V) {
        finish();
    }
```

эти кнопки в xml

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
    <Button
        android:id="@+id/startButton"
        android:text="Start game"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="Start"/>
    <Button
        android:id="@+id/exitButton"
        android:text="Exit app"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="Exit" />
</LinearLayout>
```

GameActivity

создание активности, обозначение ориентации, вся внутриигровая работа будет происходить в Movement

```
public class GameActivity extends AppCompatActivity {

    Movement movementView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setRequestedOrientation (ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        movementView = new Movement( context: this,  game_activity: this);
        setContentView(movementView);
    }
}
```

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=".GameActivity">

</androidx.constraintlayout.widget.ConstraintLayout>
```

Updater

возможность запуска Updater-а для конкретного Movement

```
public class Updater extends Thread {

    private long time;
    private final int fps = 60;
    private boolean toRun = false;
    private Movement movement;
    private SurfaceHolder surface_holder;

    public Updater(Movement movement) {
        this.movement = movement;
        surface_holder = this.movement.getHolder();
    }

    public void setRunning(boolean run) {
        toRun = run;
    }
```

Простой способ реализации данного приложения через перерисовку всего canvas.

P.S. есть ещё вариант через обновления SurfaceHolder-а, но он более тяжёлый в реализации. Плюсы: даёт сильное преимущество по времени. Но для нашего лёгкого приложения на современных телефонах это не будет особо важно, так что в рамках данной лабораторной работы решил реализовать стандартный способ через перерисовку ВСЕХ объектов.

```
@Override
public void run() {
    Canvas canvas;
    while (toRun) {
        long cTime = System.currentTimeMillis();

        if ((cTime - time) <= (1000 / fps)) {
            canvas = null;
            try {
                canvas = surface_holder.lockCanvas( dirty: null);
                movement.updatePhysics();
                movement.onDraw(canvas);
            } finally {
                if (canvas != null) {
                    surface_holder.unlockCanvasAndPost(canvas);
                }
            }
        }
        time = cTime;
    }
}
```

Movement

информация о параметрах экран, сенсоров, updater, контекст, игровая активность, о параметрах объекта игрока и нашего пункта назначения.

```
public class Movement extends SurfaceView implements SurfaceHolder.Callback {
    private int screen_width, screen_height;
    private SensorManager sensor_manager;
    private Sensor sensor_accel;
    private Sensor sensor_magnet;

    private float[] sensor_accelerometer_values = new float[3];
    private float[] sensor_magnetic_values = new float[3];
    private float[] sensors_results = new float[3];

    private Updater updater;
    Context context;
    GameActivity game_activity;

    private int x_player_coordiante, y_player_coordinate;
    private int x_speed, y_speed;
    private int players_circle_radius;
    private Paint players_circle_paint;

    private int final_destination_width, final_destination_height;
    private int x_finals_destination_coordinate, y_finals_destination_coordinate;
    private Paint final_destination_paint;
```

Конструктор, как и требуется без какой-либо логики внутри

```
public Movement(Context context, GameActivity game_activity) {
    super(context);
    getHolder().addCallback(this);
    this.context = context;
    this.game_activity = game_activity;

    x_speed = 4;
    y_speed = 4;

    players_circle_radius = 32;
    players_circle_paint = new Paint();
    players_circle_paint.setColor(Color.BLUE);

    final_destination_width = players_circle_radius * 8;
    final_destination_height = players_circle_radius * 4;
    final_destination_paint = new Paint();
    final_destination_paint.setColor(Color.RED);

    sensor_manager = (SensorManager)context.getSystemService(SENSOR_SERVICE);
    sensor_accel = sensor_manager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
    sensor_magnet = sensor_manager.getDefaultSensor(Sensor.TYPE_MAGNETIC_FIELD);
}
```

обработчик для сенсора и работа с его результатами

```
private SensorEventListener sensor_checker = new SensorEventListener() {
    // get sensor's results
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
            System.arraycopy(event.values, srcPos: 0, sensor_accelerometer_values, destPos: 0, length: 3);
        } else if (event.sensor.getType() == Sensor.TYPE_MAGNETIC_FIELD) {
            System.arraycopy(event.values, srcPos: 0, sensor_magnetic_values, destPos: 0, length: 3);
        }
    }
};
void getDeviceOrientation() {
    // transform sensor results to degrees for easier operations on them
    float[] temp = new float[9];
    SensorManager.getRotationMatrix(temp, I: null, sensor_accelerometer_values,
            sensor_magnetic_values);
    SensorManager.getOrientation(temp, sensors_results);
    sensors_results[0] = (float) Math.toDegrees(sensors_results[0]);
    sensors_results[1] = (float) Math.toDegrees(sensors_results[1]);
    sensors_results[2] = (float) Math.toDegrees(sensors_results[2]);
}
```

Остальной функционал, не нуждающийся в дополнительных пояснениях:

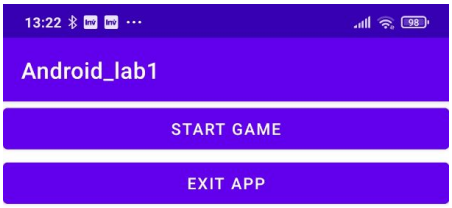
```
public void surfaceCreated(SurfaceHolder holder) {
    // start
    Rect surface_frame = holder.getSurfaceFrame();
    screen_width = surface_frame.width();
    screen_height = surface_frame.height();

    x_player_coordiante = screen_width / 2;
    y_player_coordinate = players_circle_radius;
    setFinalDestinationParameters();

    updater = new Updater( movement: this);
    updater.setRunning(true);
    updater.start();
}

protected void setFinalDestinationParameters() {
    Random random = new Random();
    x_finals_destination_coordinate = random.nextInt( bound: screen_width -
            2 * final_destination_width);
    x_finals_destination_coordinate += final_destination_width;
    y_finals_destination_coordinate = screen_height - final_destination_height * 4;
}
```

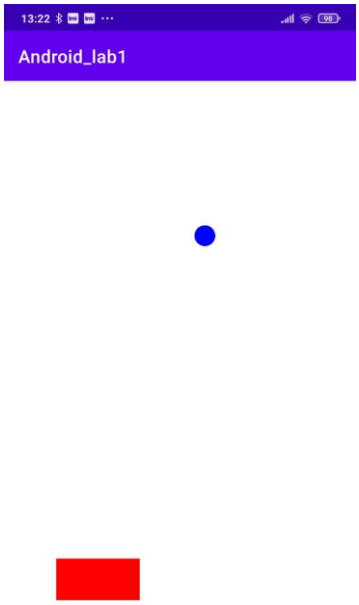
```
public void surfaceChanged(SurfaceHolder holder, int format, int width, int height) {
    // we will do everything in special class Updater
}

public void surfaceDestroyed(SurfaceHolder holder) {
    sensor_manager.unregisterListener(sensor_checker);
    updater.setRunning(false);
    while (true) {
        try {
            updater.join();
            break;
        } catch (InterruptedException e) {
        }
    }
}
@Override
protected void onDraw(Canvas canvas) {
    canvas.drawColor(Color.WHITE);
    canvas.drawCircle(x_player_coordiante, y_player_coordinate, players_circle_radius,
            players_circle_paint);
    canvas.drawRect( left: x_finals_destination_coordinate - (int)(final_destination_width / 2.0),
             top: y_finals_destination_coordinate + (int)(final_destination_height / 2.0),
             right: x_finals_destination_coordinate + (int)(final_destination_width / 2.0),
             bottom: y_finals_destination_coordinate - (int)(final_destination_height / 2.0),
            final_destination_paint);
}
```

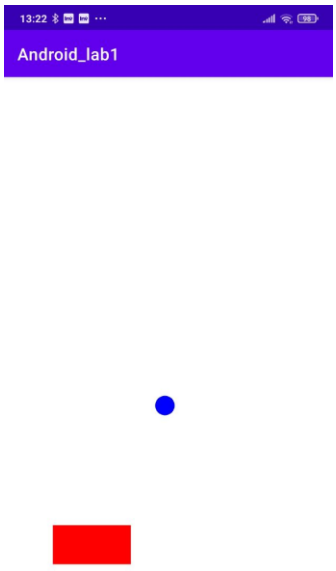
```
public void updatePhysics() {
    sensor_manager.registerListener(sensor_checker, sensor_accel, SensorManager.SENSOR_DELAY_NORMAL);
    sensor_manager.registerListener(sensor_checker, sensor_magnet, SensorManager.SENSOR_DELAY_NORMAL);
    getDeviceOrientation();

    if (sensors_results[2] > 0) {
        x_player_coordiante += x_speed;
    }
    if (sensors_results[2] < 0) {
        x_player_coordiante -= x_speed;
    }
    if (sensors_results[1] < 0) {
        y_player_coordinate += y_speed;
    }
    if (sensors_results[1] > 0) {
        y_player_coordinate -= y_speed;
    }

    if (x_player_coordiante >= x_finals_destination_coordinate - final_destination_width / 2 &&
        x_player_coordiante <= x_finals_destination_coordinate + final_destination_width / 2 &&
        y_player_coordinate >= y_finals_destination_coordinate - final_destination_height / 2 &&
        y_player_coordinate <= y_finals_destination_coordinate + final_destination_height / 2) {
        exitSystem( flag: true);
    }
```

```
    if (y_player_coordinate - players_circle_radius < 0 ||
            y_player_coordinate + players_circle_radius > screen_height) {
        exitSystem( flag: false);
    }
    if (x_player_coordiante - players_circle_radius < 0 ||
            x_player_coordiante + players_circle_radius > screen_width) {
        exitSystem( flag: false);
    }
}

protected void exitSystem(boolean flag) {
    updater.setRunning(false);
    Intent data = new Intent();
    data.putExtra( name: "win", flag);
    game_activity.setResult(RESULT_OK,data);
    game_activity.finish();
}
```

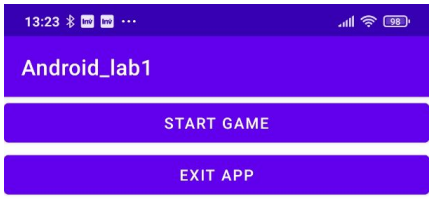
Скриншоты процесса (видео на гит хабе (ссылка в начале) лучше передаёт весь процесс):

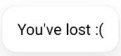



13:22
Android_lab1


13:22
Android_lab1


13:23
Android_lab1


13:23
Android_lab1
START GAME
EXIT APP


You've lost :(

Александр Дубейковский, студент 893551