Министерство цифрового развития, связи и массовых коммуникаций
Российской Федерации Сибирский Государственный Университет
Телекоммуникаций и Информатики СибГУТИ

Кафедра прикладной математики и кибернетики

Лабораторная работа №2
по дисциплине “Программирование мобильных устройств”
Игра “Жуки”

Выполнил:
Студент группы ИП-916
Меньщиков Д.А.

Работу проверила:
Павлова У. В.

Новосибирск, 2022

Задание:

Создайте игру "ЖУК". Жуки бегают по экрану. Игроку предлагается при помощи touchScreen-а уничтожить как можно большее число жуков. Обработка отдельного жука должна производиться в отдельном потоке. За каждый промах игроку начисляется штраф. Предусмотреть несколько видов насекомых. Попадание и промах должны иметь звуковое сопровождение. По окончанию игры выводятся результаты.

Выполнение лабораторной работы:

Лабораторная работа была выполнена в среде android studio на языке Java.

Реализация:

MainActivity.java

```
protected void onCreate(Bundle savedInstanceState)
```

Метод для создания сцены для игры, и инициализации игрового класса.

GameClass.java

```
public GameClass(Context context)
```

В данном методе реализовано создание фона, а так же вывод счета и результата выигрыша, или проигрыша.

```
protected void onDraw(Canvas canvas)
```

Метод для отрисовки текста на экране.

GameLogic.java

```
private void CheckLive()
```

Метод для проверки состояния жизни юнита.

```
void drawUnit(Canvas canvas)
```

Метод отрисовки юнита.

```
private void createUnit()
```

Метод создания юнита.

```
private void handleUnit(Unit unit)
```

Метод определяющий движение созданных юнитов.

```
void touchEvent(float x, float y)
```

Метод позволяющий производить нажатие на юнита.

```
void update()
```

Метод проверки состояния юнита.

Интерфейс программы:

Листинг:

MainActivity.java

```
package org.o7planning.lab3;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;

import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity {
    private GameClass gameClass;
    private Handler handler;
    private final static int interval = 1000 / 60;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        gameClass = new GameClass(this);
        setContentView(gameClass);


        handler = new Handler();
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        gameClass.invalidate();
                    }
                });
            }
        }, 0, interval);
    }
}
```

GameClass.java

```
package org.o7planning.lab3;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.os.Handler;
import android.view.WindowManager;

import java.util.Timer;
import java.util.TimerTask;

public class MainActivity extends AppCompatActivity {
    private GameClass gameClass;
    private Handler handler;
```

```
    private final static int interval = 1000 / 60;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this.getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
WindowManager.LayoutParams.FLAG_FULLSCREEN);

        gameClass = new GameClass(this);
        setContentView(gameClass);


        handler = new Handler();
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        gameClass.invalidate();
                    }
                });
            }
        }, 0, interval);
    }
}
```

GameLogic.java

```
package org.o7planning.lab3;

import android.graphics.BitmapFactory;
import android.graphics.Canvas;
import android.media.MediaPlayer;
import android.view.View;

import java.util.ArrayList;
import java.util.concurrent.ThreadLocalRandom;

class GameLogic {
    public View view;
    private ArrayList<Unit> unitList;
    private int unitCounter;
    int point;

    private void CheckLive() {
        ArrayList<Unit> newUnitList = new ArrayList<>(5);
        for(Unit unit : unitList) {
            if (unit.unitAlive) {
                newUnitList.add(unit);
            }
        }
        unitList = newUnitList;
    }

    void drawUnit(Canvas canvas) {
        for(Unit unit : unitList) {
            canvas.drawBitmap(unit.unitTexture, unit.matrix, null);
        }
    }
```

```
    private void createUnit() {
        Unit unit = new Unit();
        switch(ThreadLocalRandom.current().nextInt(0, 3)) {
            case 0:
                unit.unitTexture =
BitmapFactory.decodeResource(view.getResources(), R.drawable.bugorange);
                break;
            case 1:
                unit.unitTexture =
BitmapFactory.decodeResource(view.getResources(), R.drawable.bugred);
                break;
            case 2:
                unit.unitTexture =
BitmapFactory.decodeResource(view.getResources(), R.drawable.bugcyan);
        }
        unitList.add(unit);
        unit.matrix.setRotate(0, unit.unitTexture.getWidth() / 10,
unit.unitTexture.getHeight() / 10);
        unit.matrix.reset();
        unit.p = 0;
        unit.isRun = false;
        float ty, tx;
        int temp = (int) Math.floor(Math.random() * 4);
        switch (temp) {
            case 0:
                ty = (float) Math.random() * view.getHeight();
                unit.x = 0f;
                unit.y = ty;
                break;
            case 1:
                ty = (float) Math.random() * view.getHeight();
                unit.x = (float) view.getWidth();
                unit.y = ty;
                break;
            case 2:
                tx = (float) Math.random() * view.getWidth();
                unit.x = tx;
                unit.y = 0f;
                break;
            case 3:
                tx = (float) Math.random() * view.getWidth();
                unit.x = tx;
                unit.y = (float) view.getHeight();
                break;
        }
        unit.matrix.postTranslate(unit.x, unit.y);
        }

        private void handleUnit(Unit unit) {
        if(!unit.isRun) {
            unit.destX = (float) Math.random() * view.getWidth();
            unit.destY = (float) Math.random() * view.getHeight();
            unit.stepX = (unit.destX - unit.x) / 50;
            unit.stepY = (unit.destY - unit.y) / 50;
            Integer tp;
            if (unit.x <= unit.destX && unit.y >= unit.destY)
                tp = (int)
Math.floor(Math.toDegrees(Math.atan(Math.abs(unit.x - unit.destX) /
Math.abs(unit.y - unit.destY))));
            else if (unit.x <= unit.destX && unit.y <= unit.destY)
                tp = 90 + (int)
Math.floor(Math.toDegrees(Math.atan(Math.abs(unit.y - unit.destY) /
Math.abs(unit.x - unit.destX))));
            else if (unit.x >= unit.destX && unit.y <= unit.destY)
```

```
                tp = 180 + (int)
Math.floor(Math.toDegrees(Math.atan(Math.abs(unit.x - unit.destX) /
Math.abs(unit.y - unit.destY))));
            else
                tp = 270 + (int)
Math.floor(Math.toDegrees(Math.atan(Math.abs(unit.y - unit.destY) /
Math.abs(unit.x - unit.destX))));
            unit.matrix.preRotate(tp - unit.p, unit.unitTexture.getWidth() /
2, unit.unitTexture.getHeight() / 2);
            unit.p = tp;
            unit.isRun = true;
        } else {
            if (Math.abs(unit.x - unit.destX) < 0.1 && Math.abs(unit.y -
unit.destY) < 0.1) {
                unit.isRun = false;
            }
            unit.matrix.postTranslate(unit.stepX, unit.stepY);
            unit.x += unit.stepX;
            unit.y += unit.stepY;
        }
    }

    void touchEvent(float x, float y) {
        boolean touch = false;
        for(Unit unit : unitList) {
            if(Math.abs(unit.x - x + 140) < 144
                    && Math.abs(unit.y - y + 140) < 156) {
                //MediaPlayer hit = MediaPlayer.create(view.getContext(),
R.raw.hit);
                //hit.setOnCompletionListener(mediaPlayer ->
mediaPlayer.release());
                //hit.setOnPreparedListener(mediaPlayer ->
mediaPlayer.start());
                MediaPlayer Win = MediaPlayer.create(view.getContext(),
R.raw.hit);
                Win.setOnCompletionListener(mediaPlayer ->
mediaPlayer.release());
                Win.setOnPreparedListener(mediaPlayer ->
mediaPlayer.start());
                unit.unitNotAlive();
                point++;
                touch = true;
                break;
            }
        }
        if(!touch) {
            point--;
        }
    }

    GameLogic(int unitCounter, View view) {
        point = 0;
        this.view = view;
        this.unitCounter = unitCounter;
        unitList = new ArrayList<>(5);
    }

    void update() {
        CheckLive();
        while (unitList.size() < unitCounter) {
            createUnit();
        }
        for(final Unit unit : unitList) {
            new Thread(new Runnable() {
```

```
                @Override
                public void run() {
                    handleUnit(unit);
                }
            }).start();
        }
    }
}
```

Unit.java

```
package org.o7planning.lab3;

import android.graphics.Bitmap;
import android.graphics.Matrix;

public class Unit {
    boolean isRun;
    Matrix matrix;
    boolean unitAlive;
    Bitmap unitTexture;
    Float x;
    Float y;
    Float stepX;
    Float stepY;
    Float destX;
    Float destY;
    Integer p;

    Unit() {
        matrix = new Matrix();
        x = 0f;
        y = 0f;
        p = 0;
        destX = 0f;
        destY = 0f;
        unitAlive = true;
    }

    void unitNotAlive() {
        unitAlive = false;
    }
}
```