Федеральное агентство связи

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет

телекоммуникаций и информатики»

Лабораторная работа №3

«Жуки»

Выполнили: студенты 3 курса

ИВТ, гр. ИП-713

Трусов К.В.

Михеев Н.А.

Новосибирск, 2020 г.

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.

Решение поставленной задачи

Для реализации программы были реализованы 3 различных класса:

- MainActivity – в нем происходит инициализация всей программы, создание handler и timer по которому происходит обновление состояния игры.
- Bug – класс который является базовым классом для жуков, в нем определяется стартовое положение, скорость, точка отправления жука.
- BugView – класс занимающийся прорисовкой жуков и счета на Canvas, математикой скорости жука в зависимости от удаленности точки следования, регистрация попаданий и обновления счета.

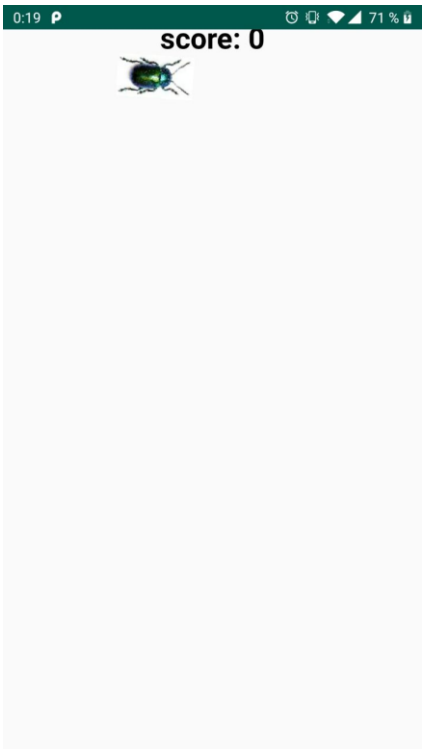

Рис.1 – стартовое поле с жучком

Рис.2 – состояние приложения после непродолжительной игры

Листинг программы

Класс MainActivity:

```
public class MainActivity extends AppCompatActivity {
    private BugView view;
    private Handler handler;
    private final static int interval = 50;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        view = new BugView(this);
        setContentView(view);
        handler = new Handler();
        Timer timer = new Timer();
        timer.schedule(new TimerTask() {
            @Override
            public void run() {
                handler.post(new Runnable() {
                    @Override
                    public void run() {
                        view.invalidate();
                    }
```

```
            });
        }
    }, 0, interval);
  }
}
```

Класс Bug:

```
public class Bug {
  public Bitmap texture;
  public Float x, y, stepX, stepY, destX, destY;
  public Integer p;
  private Point point;
  public Bug() {
    x = 0f;
    y = 0f;
    p = 0;
    destX = 0f;
    destY = 0f;
    p = 0;
  }
  public boolean checkCol(float Sx, float Sy) {
    return true;
  }
}
```

Класс BugView:

```
public class BugView extends View {
  private Bitmap background;
  private Bug bug = new Bug();
  private Paint score;
  private Matrix matrix;
  private boolean reached;
  private Integer sc = 0;
  public BugView(Context context) {
    super(context);
    bug.texture = BitmapFactory.decodeResource(context.getResources(), R.drawable.bug); // Change this shit later;
    //background = BitmapFactory.decodeResource(getResources(), R.drawable.ic_launcher_background);// Change this shit later;
    score = new Paint();
    score.setColor(Color.BLACK);
```

```
    score.setTextAlign(Paint.Align.CENTER);
    score.setTextSize(75);
    score.setTypeface(Typeface.DEFAULT_BOLD);
    score.setAntiAlias(true);
    matrix = new Matrix();
    reached = true;
    DisplayMetrics metrics = getResources().getDisplayMetrics();
    //metrics.
}

@Override
protected void onDraw(Canvas canvas) {
    super.onDraw(canvas);
    //canvas.drawBitmap(background, 0, 0, null);
    canvas.drawText("score: " + sc, getWidth()/(float)2, 50, score);
    //matrix.postScale(3.0f, 3.0f); // Масштабируем
    if (reached) {
        bug.destX = (float) Math.random() * getWidth();
        bug.destY = (float) Math.random() * getHeight();
        bug.stepX = (bug.destX - bug.x) / 50;
        bug.stepY = (bug.destY - bug.y) / 50;
        Integer tp;
        if (bug.x <= bug.destX && bug.y >= bug.destY)
            tp = (int)Math.floor(Math.toDegrees(Math.atan(Math.abs(bug.x - bug.destX)/Math.abs(bug.y - bug.destY))));
        else if (bug.x <= bug.destX && bug.y <= bug.destY)
            tp = 90 + (int)Math.floor(Math.toDegrees(Math.atan(Math.abs(bug.y - bug.destY)/Math.abs(bug.x - bug.destX))));
        else if (bug.x >= bug.destX && bug.y <= bug.destY)
            tp = 180 + (int)Math.floor(Math.toDegrees(Math.atan(Math.abs(bug.x - bug.destX)/Math.abs(bug.y - bug.destY))));
        else
            tp = 270 + (int)Math.floor(Math.toDegrees(Math.atan(Math.abs(bug.y - bug.destY)/Math.abs(bug.x - bug.destX))));
        matrix.preRotate(tp - bug.p, bug.texture.getWidth() / 2, bug.texture.getHeight() / 2);
        bug.p = tp;
        reached = false;
    }
    else {
        if (Math.abs(bug.x - bug.destX) < 0.1 &&
                Math.abs(bug.y - bug.destY) < 0.1)
            reached = true;
```

```
        matrix.postTranslate(bug.stepX, bug.stepY);
        bug.x += bug.stepX;
        bug.y += bug.stepY;
    }
    canvas.drawBitmap(bug.texture, matrix,null);
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    if (event.getAction() == MotionEvent.ACTION_DOWN) {
        if (Math.abs(bug.x - event.getX() + 30) < 40 &&
                Math.abs(bug.y - event.getY() + 30) < 60) {
            matrix.setRotate(0, bug.texture.getWidth() / 2, bug.texture.getHeight() / 2);
            matrix.reset();
            bug.p = 0;
            sc += 10;
            reached = true;
            float ty, tx;
            int temp = (int)Math.floor(Math.random() * 4);
            switch (temp) {
                case 0:
                    ty = (float)Math.random() * getHeight();
                    bug.x = 0f;
                    bug.y = ty;
                break;
                case 1:
                    ty = (float)Math.random() * getHeight();
                    bug.x = (float)getWidth();
                    bug.y = ty;
                break;
                case 2:
                    tx = (float)Math.random() * getWidth();
                    bug.x = tx;
                    bug.y = 0f;
                break;
                case 3:
                    tx = (float)Math.random() * getWidth();
```

```
                    bug.x = tx;
                    bug.y = (float)getHeight();
                  break;
               }
               matrix.postTranslate(bug.x, bug.y);
            }


         }
         return true;
      }
   }
```