Петрозаводский государственный университет
Институт математики и информационных технологий
Кафедра информатики и математического обеспечения

Направление подготовки бакалавриата

09.03.04 — Программная инженерия

Отчет по проекту

Разработка игрового приложения

Упрощенная Версия ритм игры "Guitar Hero

Выполнил:
студента 2 курса группы 22207

О. В. Лисицкий _____
подпись

Петрозаводск — 2022

Содержание

Введение

Цель проекта: Создание упрощенного аналога игрового процесса ритм игры "Guitar Hero"на д языке программирования java для получения дополнительного опыта разработки игровых приложени.

Задачи проекта:

1. задача 1: используя android studio, подготовить прототип проекта;
2. задача 2: Создать структуру игрового пространства, с которым будет происходить взаимодействие пользователя;
3. задача 3: Организовать способы взаимодействия пользователя с объектами игры - функционал обработки ввода от пользователя, вывода изображения на экран, интерфейс пользователя;
4. задача 4: Произвести тестирование работоспособности игрового процесса, запустив получившееся приложение.

Рис. 1: Демонстрация игрового процесса "Guitar Hero"

1 Требования к приложению

1)Игровое пространство должно представлять собой совокупность нескольких рядов, вдоль которых движутся обьекты-ноты. Ноты начинают свое движение за пределемаи экрана;

2)Каждый ряд на одном из своих концов должен иметь кнопку, активирующаяся по нажатию пользователем соответсвующей клавиши клавиатурыж

4)Если в момент нажатия кнопки в непосредственной близости к ней находился объект-нота, то этот объект пропадает с зоны видимости, количество успешных нажатий должно отбражаться на экране в течении всей игры;

5)В случае, если нота была пропущена, объект нота также покидает видимую область игрового простравнства. Игра должна фиксировать пропуски нот. Пропускание нот должно приводить к окончанию игры с соответсвующим сообщение от игры;

6)Игра должна иметь кнопки начала новой игровой сессии.

2 Проектирование приложения

модули и функции приложения.

activity main.xml - проекта;

MainActivity.java- код игры;

timer - таймер, за счет которого реализуется анимация;

button(-1,-2,-3,-4) - игровые кнопки;

addview - кнопка старта новой сессии;

makeTrack - вспомогалтельная функция начала новой сессии, расставляет ноты за полем;

Randomize - функция перетасовки ушедших вниз нот;

addView - создание новой ноты на поле.

3 Реализация приложения

```
1  package com.example.mproject;
2
3  import androidx.appcompat.app.AppCompatActivity;
4  import androidx.constraintlayout.widget.ConstraintLayout;
5  import androidx.constraintlayout.widget.ConstraintSet;
6
7  import android.os.Bundle;
8  import android.widget.ImageView;
9  import android.graphics.Color;
10 import android.view.View;
11 import android.widget.Button;
12 import android.os.CountDownTimer;
13 import android.widget.TextView;
14
15 import java.sql.Time;
16 import java.util.Random;
17
18 public class MainActivity extends AppCompatActivity {
19     CountDownTimer timer;
20     Button addview;
21     Button button1;
22     Button button2;
23     Button button3;
24     Button button4;
25     TextView textView;
26     ConstraintLayout layout;
27     Random r = new Random();
28     int speed = 1;
29     int score = 0;
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34         // initialising layout
35         addview = findViewById(R.id.addiview);
36         button1 = findViewById(R.id.button1);
37         button2 = findViewById(R.id.button2);
38         button3 = findViewById(R.id.button3);
```

```
39          button4 = findViewById(R.id.button4);
40          textView = findViewById(R.id.textView);
41          layout = findViewById(R.id.layout);
42          timer = new CountDownTimer(42690,10) {
43              @Override
44              public void onTick(long l) {
45                  int count=0;
46                  for(int b = 6; b < layout.getChildCount(); b++) {
47                      layout.getChildAt(b).setY(layout.getChildAt(b).getY() + 10 * speed*((42690-l)/
48                      if(layout.getChildAt(b).getY() > 2600) {
49                          count++;
50                      }
51                      if(layout.getChildAt(b).getY() > 2500 && layout.getChildAt(b).getVisibility()
52                          timer.cancel();
53                          timer.onFinish();
54                      }
55                  }
56                  if(count == layout.getChildCount()-6)
57                      Randomize();
58                  textView.setText(String.valueOf(score));
59
60
61              }
62              @Override
63              public void onFinish() {
64                  layout.removeViews(6,layout.getChildCount() - 6);
65                  addview.setVisibility(View.VISIBLE);
66              }
67          };
68          button1.setOnClickListener(new View.OnClickListener() {
69              @Override
70              public void onClick(View v) {
71                  for(int i=6; i < layout.getChildCount(); i++)
72                      if(Math.abs(layout.getChildAt(i).getY() - button1.getY()) < 125 &&
73                              Math.abs(layout.getChildAt(i).getX() - button1.getX()) < 25) {
74                          score++;
75                          layout.getChildAt(i).setVisibility(View.GONE);
76                          return;
77                      }
78                  if(layout.getChildCount() != 6){
```

```
                timer.cancel();
                timer.onFinish();
            }
        }
    });
    button2.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            for(int i=6; i < layout.getChildCount(); i++)
                if(Math.abs(layout.getChildAt(i).getY() - button2.getY()) < 125 &&
                        Math.abs(layout.getChildAt(i).getX() - button2.getX()) < 25) {
                    score++;
                    layout.getChildAt(i).setVisibility(View.GONE);
                    return;
                }
            if(layout.getChildCount() != 6){
                timer.cancel();
                timer.onFinish();
            }
        }
    });
    button3.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            for(int i=6; i < layout.getChildCount(); i++)
                if(Math.abs(layout.getChildAt(i).getY() - button3.getY()) < 125 &&
                        Math.abs(layout.getChildAt(i).getX() - button3.getX()) < 25) {
                    score++;
                    layout.getChildAt(i).setVisibility(View.GONE);
                    return;
                }
            if(layout.getChildCount() != 6){
                timer.cancel();
                timer.onFinish();
            }
        }
    });
    button4.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
```

```
                for(int i=6; i < layout.getChildCount(); i++)
                    if(Math.abs(layout.getChildAt(i).getY() - button4.getY()) < 125 &&
                            Math.abs(layout.getChildAt(i).getX() - button4.getX()) < 25) {
                        score++;
                        layout.getChildAt(i).setVisibility(View.GONE);
                        return;
                    }
                if(layout.getChildCount() != 6) {
                    timer.cancel();
                    timer.onFinish();
                }
            }
        });
        // we will click on the add view button
        addview.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                score = 0;
                addview.setVisibility(View.GONE);
                makeTrack();
                timer.start();
            }
        });

    }
    public void makeTrack() {
   for(int i =0; i < 60; i++){
            ImageView imageView = new ImageView(MainActivity.this);
            // setting the image in the layout
            imageView.setImageResource(R.drawable.brick);
            // calling addview with width and height
            addvieW(imageView, 250,100, layout.getChildCount());
        }
    }

    public void Randomize() {
        for(int i = 6; i < layout.getChildCount(); i++) {
            layout.getChildAt(i).setVisibility(View.VISIBLE);
            layout.getChildAt(i).setX(r.nextInt(4) * 250 + 20);
            layout.getChildAt(i).setY(-250*i);
```

```
159             }
160         }
161
162         private void addvieW(ImageView imageView, int width, int height, int i) {
163             ConstraintLayout.LayoutParams params = new ConstraintLayout.LayoutParams(width, height);
164             imageView.setLayoutParams(params);
165             imageView.setX(r.nextInt(4) * (250+16) + 15);
166             if(i==6)
167                 imageView.setY(-250);
168             else
169             imageView.setY(layout.getChildAt(i-1).getY()-250);
170             // adding the image in layout
171             layout.addView(imageView);
172             i++;
173         }
174     }
175     }
```

Интерфейс окна приложения организован довольно просто. В течение игры игроку представленны игровые кнопки, счет очков и падающие ноты. В начале и конце появляется кнопка старта, ноты исчезают.

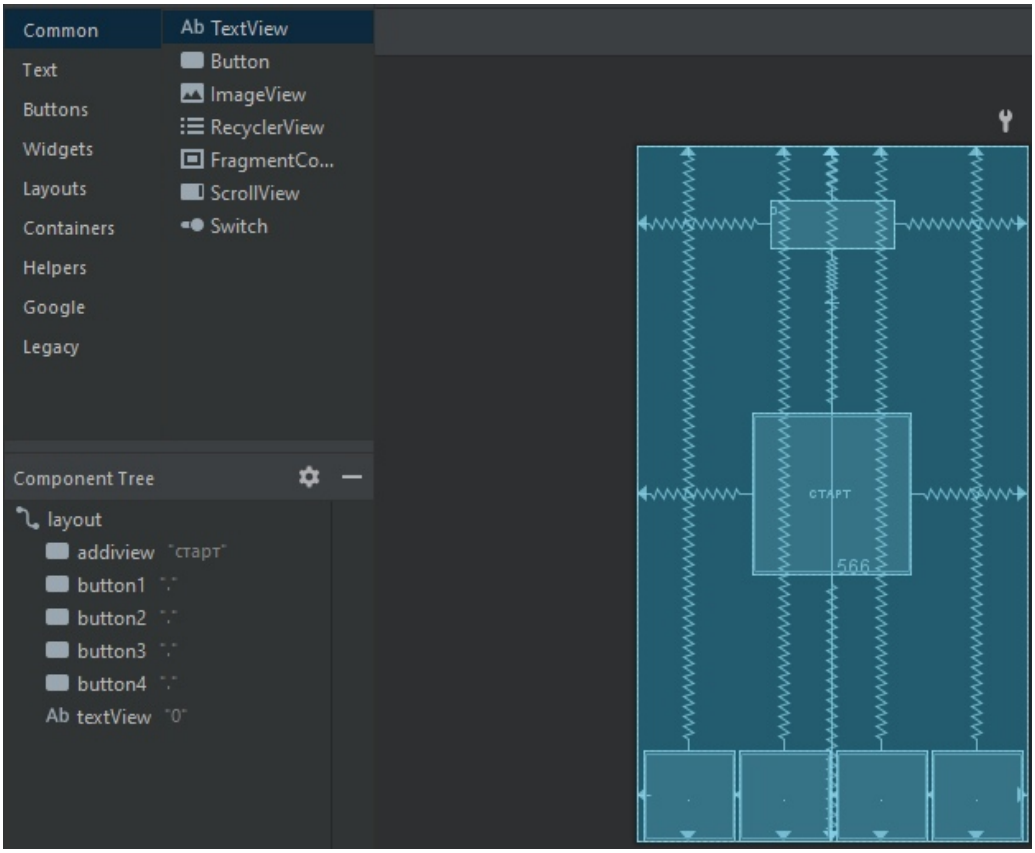

Рис. 2: Схематическое представление интерфейса приложения"

Код интерфейса:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.an
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:baselineAligned="false"
    tools:context=".MainActivity">

    <!--Adding the Button in the layout-->
```

```
<Button
    android:id="@+id/addiview"
    android:layout_width="167dp"
    android:layout_height="170dp"
    android:background="@color/purple_500"
    android:text="старт"
    android:textColor="@android:color/white"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/button1"
    android:layout_width="250px"
    android:layout_height="250px"
    android:text="."
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button2"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<Button
    android:id="@+id/button2"
    android:layout_width="250px"
    android:layout_height="250px"
    android:text="."
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button3"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/button1"
```

```
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<Button
    android:id="@+id/button3"
    android:layout_width="250px"
    android:layout_height="250px"
    android:text="."
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toStartOf="@+id/button4"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/button2"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<Button
    android:id="@+id/button4"
    android:layout_width="250px"
    android:layout_height="250px"
    android:text="."
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.5"
    app:layout_constraintStart_toEndOf="@+id/button3"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="1.0" />

<TextView
    android:id="@+id/textView"
    android:layout_width="130dp"
    android:layout_height="50dp"
    android:layout_marginBottom="566dp"
    android:text="0"
    app:layout_constraintBottom_toBottomOf="parent"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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

Внутри каждого ряда есть свой собственный объект-нота и свой таймер. Это позволяет довольно просто настроить скорость передвижения на каждом ряду индивидуально, если в этом возникнет необходимость. У игровой зоны есть также и свой собственный таймер, обновляющийся по константе. Он не обходим для проверки на достижение игрой своего конца. Конец игры происходит по вызову сигнала, отстанавливающим все таймеры после трех пропусков нот. Вызов сигнала рестарта игры ставит все ноты за пределы экрана, зануляет счет и количество пропусков, активирует таймеры заново.

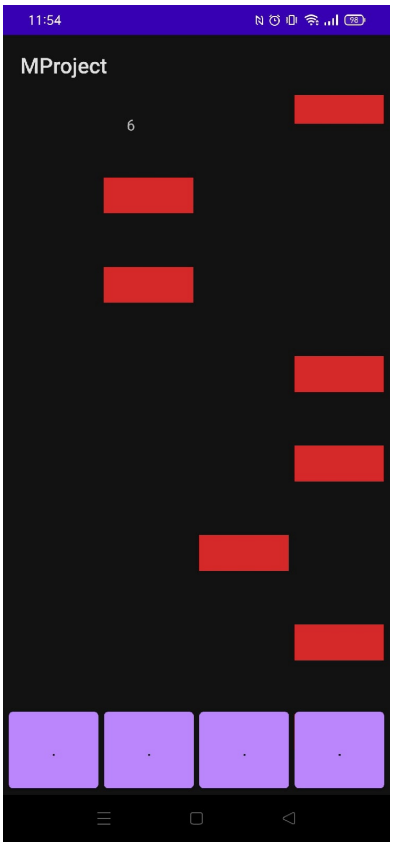

Рис. 3: Итоговое приложение: процессе игры

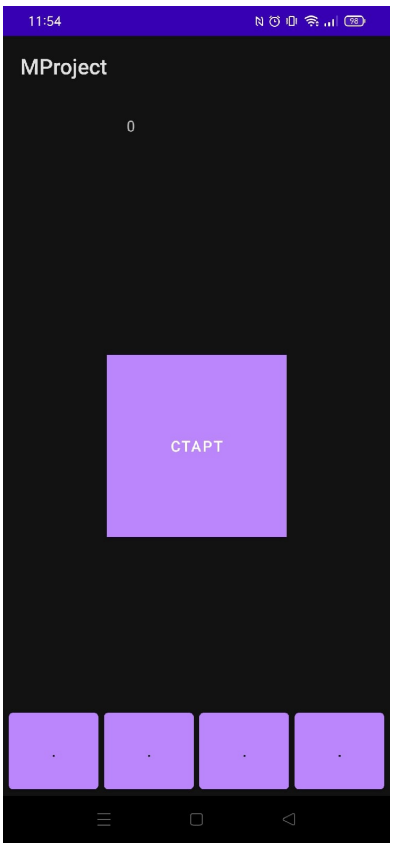

Рис. 4: Итоговое приложение: игра окончена или начало игры

Заключение

Было полученно игровое приложение с достаточно схожим игровым процессом таковому у "Guitar Hero". У программы есть игровое пространство с которым можно взаимодействовать и по которому движутся объекты-ноты. Имеется кнопка имеется кнопка начала новой сессии. Программа написана на java и работает, согласно поставленным требованиям. Были выполнены все поставленные цели и задачи работы.