ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Лабораторная работа №6

«Живые обои»

Выполнил: студент группы ИП-712

Алексеев С.В.

Работу принял: ассистент кафедры ПМиК

Павлова У.В.

Новосибирск 2020

Оглавление

[Цель работы 2](#_Toc40596455)

[Описание программы и некоторых функций 3](#_Toc40596456)

[Прорисовка 4](#_Toc40596457)

[Листинг 5](#_Toc40596458)

[mywallpaper.xml 7](#_Toc40596459)

[Каталог ресурсов: 8](#_Toc40596460)

[Скрины. 9](#_Toc40596461)

# Цель работы

Разработать приложение Живые обои

# Описание программы и некоторых функций

Приложение - “Живые обои” для рабочего стола, которое показывает заряд батареи.

Cоздаем файл настроек:

<wallpaper

xmlns:android="http://schemas.android.com/apk/res/android"

android:thumbnail="@mipmap/ic\_launcher"

/>

Добавляем в manifest сервис:

<service

android:name=".MyActivity"

android:enabled="true"

android:label="Wallpaper Example Title" название обоев

android:permission="android.permission.BIND\_WALLPAPER" >

<intent-filter>

<action

android:name="android.service.wallpaper.WallpaperService" >

</action>

</intent-filter>

<meta-data

android:name="android.service.wallpaper"

android:resource="@xml/mywallpaper" файл настроек

>

</meta-data>

</service>

public class MyActivity extends WallpaperService – создаем новый класс сервиса

private final Handler handler = new Handler(); - Handler – ставит потоки в очередь на выполнение

private final Runnable drawRunner = new Runnable() { - drawRunner сам поток.

@Override

public void run() {

draw(); - метод прорисовки обоев.

}

};

private boolean visible = true; - обои видны или нет.

public void onCreate(SurfaceHolder surfaceHolder) – вызывается при создании обоев.

if (visible) – если обои видны.

{

handler.post(drawRunner); - поставить прорисовку в очередь.

}

else{

handler.removeCallbacks(drawRunner); - убрать прорисовку из очереди.

}

public void onOffsetsChanged(float xOffset, float yOffset, float xStep, float, yStep, int xPixels, int yPixels)- Вызывается при смене рабочих столов.

# Прорисовка

final SurfaceHolder holder = getSurfaceHolder(); - получаем полотно рабочего стола

Canvas canvas = null;

canvas = holder.lockCanvas(); -Захват полотна(только наш процесс рисует)

if (canvas != null) - если удалось захватить , рисуем на нем, что хотим

holder.unlockCanvasAndPost(canvas); - Разблокируем полотно и выводим его на экран

handler.removeCallbacks(drawRunner); - убрать прорисовку из очереди

handler.postDelayed(drawRunner, 20); - поставить прорисовку в очередь через 70 мс.

Листинг

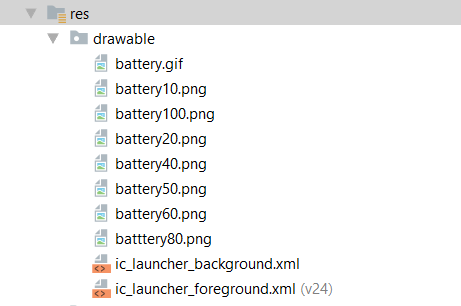
MyActivity.java

package ru.example.livewallpaper;  
  
import android.content.Intent;  
import android.content.IntentFilter;  
import android.graphics.BitmapFactory;  
import android.graphics.Canvas;  
import android.os.Handler;  
import android.service.wallpaper.WallpaperService;  
import android.view.SurfaceHolder;  
  
public class MyActivity extends WallpaperService {  
  
 public void onCreate()  
 {  
 super.onCreate();  
 }  
 public void onDestroy()  
 {  
 super.onDestroy();  
 }  
 public Engine onCreateEngine()  
 {  
 return new MyWallpaperEngine();  
 }  
  
 class MyWallpaperEngine extends Engine  
 {  
  
 private final Handler handler = new Handler();  
  
 private final Runnable drawRunner = new Runnable() {  
 @Override  
 public void run() {  
 draw();  
 }  
 };  
 private boolean visible = true;  
  
 MyWallpaperEngine()  
 {  
 }  
 public void onCreate(SurfaceHolder surfaceHolder)  
 {  
 super.onCreate(surfaceHolder);  
 }  
 @Override  
 public void onVisibilityChanged(boolean visible)  
 {  
 this.visible = visible;  
  
 if (visible)  
 {  
 handler.post(drawRunner);  
 }  
 else{  
 handler.removeCallbacks(drawRunner);  
 }  
 }  
  
 @Override  
 public void onSurfaceDestroyed(SurfaceHolder holder)  
 {  
 super.onSurfaceDestroyed(holder);  
 this.visible = false;  
 handler.removeCallbacks(drawRunner);  
 }  
  
 public void onOffsetsChanged(float xOffset, float yOffset, float xStep, float  
 yStep, int xPixels, int yPixels)  
 {  
 handler.post(drawRunner);  
 }  
  
 public void draw() {  
 final SurfaceHolder holder = getSurfaceHolder();  
 Canvas canvas = null;  
  
 try  
 {  
 canvas = holder.lockCanvas();  
 if (canvas != null) {  
  
 IntentFilter ifilter = new IntentFilter(Intent.*ACTION\_BATTERY\_CHANGED*);  
 Intent batteryStatus = registerReceiver(null, ifilter);  
 int status = batteryStatus.getIntExtra("level", -1);  
  
  
 if (status > 80) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery100*), 0, 0, null);  
 }  
 else {  
 if (status <= 80 && status > 60 ) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*batttery80*), 0, 0, null);  
 }  
 else {  
 if (status <= 60 && status > 50 ) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery60*), 0, 0, null);  
 }  
 else {  
 if (status <= 50 && status > 40) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery50*), 0, 0, null);  
 }  
 else {  
 if (status <= 40 && status > 20) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery40*), 0, 0, null);  
 }  
 else {  
 if (status <= 20 && status > 10) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery20*), 0, 0, null);  
 }  
 else {  
 if (status <= 10) {  
 canvas.drawBitmap(BitmapFactory.*decodeResource*(getResources(), R.drawable.*battery10*), 0, 0, null);  
 }  
 }  
 }  
 }  
 }  
 }  
 }  
  
 }  
 }  
 finally  
 {  
 if (canvas != null)  
 holder.unlockCanvasAndPost(canvas);  
 }  
 handler.removeCallbacks(drawRunner);  
 if (visible)  
 {  
 handler.postDelayed(drawRunner, 20);  
 }  
 }  
 }  
}

# mywallpaper.xml

<wallpaper  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:thumbnail="@mipmap/ic\_launcher"  
 />

# Каталог ресурсов:



# Скрины.

На рабочем столе меняется цвет, исходя из уровня заряда телефона

