

Pokračovanie

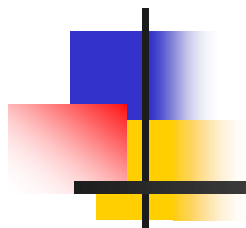
Menu

SurfaceView, Gestá

SharedPreferences

PreferenceActivity

RuntimePermissions



Peter Borovanský
KAI, I-18

MS-Teams: [2sf3ph4](#), [List](#), [github](#)

borovan 'at' ii.fmph.uniba.sk



Bolo minule

- layouts, najmä constraint layout
- ListView, ListAdapter, najmä kvôli DÚ2
- intent, intent data
- `<intent-filter />` v AndroidManifest,
- permissions
- `startActivity`, `startActivityForResult`

Option Menu

(onCreateOptionsMenu)

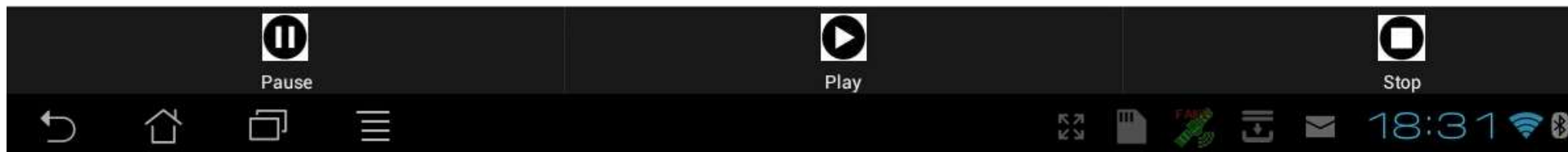
```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/pause" android:icon="@drawable/pause"
        android:title="Pause">

    </item>
    <item android:id="@+id/play" android:icon="@drawable/play"
        android:title="Play">

    </item>
    <item android:id="@+id/stop" android:icon="@drawable/stop"
        android:title="Stop">

    </item>
</menu>
```

```
override fun onCreateOptionsMenu(menu: Menu): Boolean {
    val inflater = menuInflater
    inflater.inflate(R.menu.activity_canvas, menu)
    return super.onCreateOptionsMenu(menu)
}
```

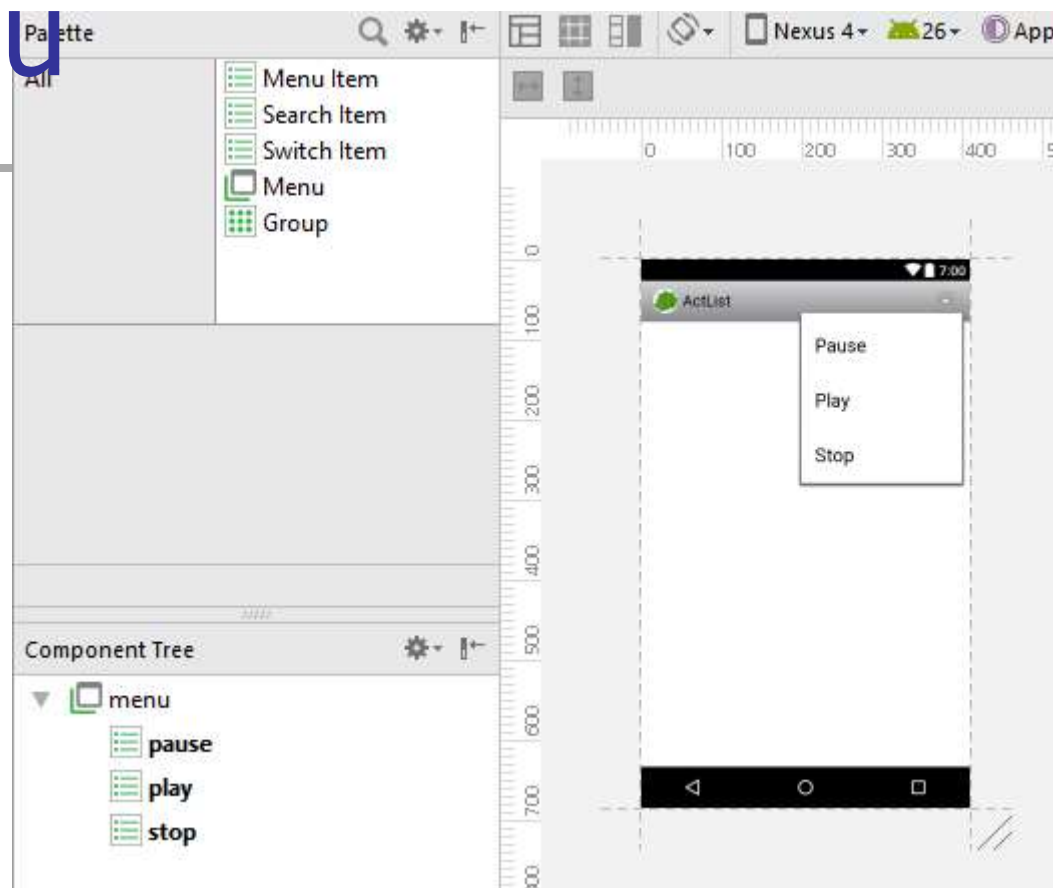


Option Menu

(onCreateOptionsMenu)

Rovnako dobre to môžete navrhovať v editore

Spôsob zobrazenia a renderovania závisí na API level zariadenia



```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/pause" android:icon="@drawable/pause" android:title="Pause"> </item>
    <item android:id="@+id/play" android:icon="@drawable/play" android:title="Play"> </item>
    <item android:id="@+id/stop" android:icon="@drawable/stop" android:title="Stop"> </item>
</menu>
```



Option Menu

```
Thread th = new Thread() {  
    fun run() {  
        while (!stopped) {  
            if (!paused) {  
                ...  
            }  
        }  
    }  
}
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.getItemId()) {  
        R.id.pause -> {  
            canvasView1?.paused = true  
            return true  
        }  
        R.id.play -> {  
            canvasView1?.paused = false  
            return true  
        }  
        R.id.stop -> {  
            canvasView1?.stopped = true  
            return true  
        }  
        else -> return super.onOptionsItemSelected(item)  
    }  
}
```

Context Menu

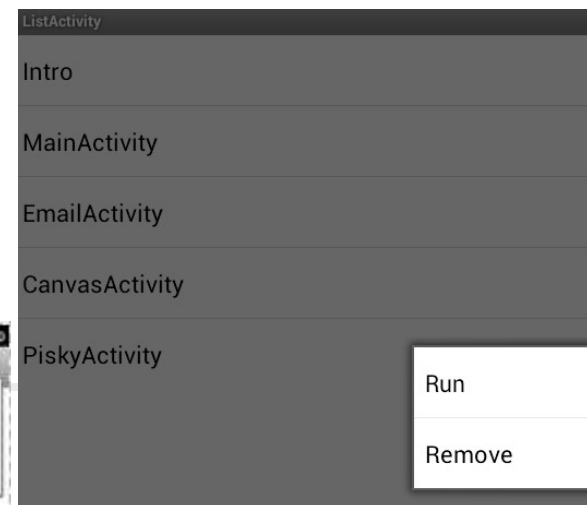
```
override fun onCreate(  
    savedInstanceState: Bundle?) { ...  
    registerForContextMenu(listView1) // rozdiel od OptionMenu  
}
```

ContextMenu (oproti OptionMenu) treba registrovať k príslušnému view

```
override fun onCreateContextMenu(menu: ContextMenu?, v: View?,  
    menuInfo: ContextMenu.ContextMenuInfo? ) {  
    getMenuInflater().inflate(R.menu.list_menu, menu)  
}
```

// v je View, na ktoré bolo spáchané ContextMenu Action

```
override fun onContextItemSelected(item: MenuItem): Boolean {  
    val info = item.getMenuInfo() as AdapterContextMenuInfo  
    val className = actList.get(info.id.toInt())  
    when (item.getItemId()) {  
        R.id.remove -> {  
            actList.removeAt(info.id.toInt())  
            la.notifyDataSetChanged()  
            return true  
        }  
    }
```





invalidate() vs. postInvalidate()

(sumár poznatkov)

vo **View**, ak chceme modifikovať obsah, používame:

- `view.invalidate()` v **GUI vlákne**, t.j. v event handleroch `onKey`, `onTouch`
- `view.postInvalidate()` v iných (**non-GUI**) vláknach, ktoré chcú `view` modifikovať, alternatíva `Activity.runOnUiThread` (z minulej prednášky)

toto však nenastane hneď (podobne, ako `Event Dispatch Thread` vo `JavaFx`)
nastane to po `VSYNC` (vertical synchronization), 40 fps ~ každých 25 ms

Všetky `View` sú kreslené v jednom `GUI vlákne`. Preto, ak

- chceme lepšie kontrolovať renderovanie (veľa) objektov, resp.
 - renderovanie objektov trvá dlho
- používame triedu **SurfaceView**. To je však náročnejšie
- na `cpu`
 - programovanie.



SurfaceView

(podtrieda View, nadtrieda ako GLSurfaceView, VideoView)

SurfaceView je typicky renderované iným vláknom pomocou SurfaceHolder.Callback

```
class GamePanel(context:Context) : SurfaceView(context),  
    SurfaceHolder.Callback {  
  
    lateinit var thread : GameThread                // vlákno hry  
    init {  
        getHolder().addCallback(this) //kto implementuje SurfaceHolder  
        thread = GameThread(this)  
        setFocusable(true)  
    }  
  
    override fun surfaceCreated(holder: SurfaceHolder?) {  
        thread.start()                // entry point pre SurfaceView  
    }  
  
    override fun surfaceDestroyed(holder: SurfaceHolder?) {  
        // exit point SfV-treba zastaviť vlákno hry a počkať kým skončí  
        // vid' priložený projekt...    }  
}
```


GameThread

(čo robí vlákno hry - alternatíva k invalidate)

```
class GameThread(val gamePanel: GamePanel) : Thread() {  
    // zapamätáme v konštruktore GameTread  
    override fun run() { // hlavný cyklus vlákna, hry, simulácie  
        val surfaceHolder = gamePanel.holder  
        while (running) {  
            try {  
                canvas = surfaceHolder.lockCanvas()  
                synchronized (surfaceHolder) {  
                    for (pika in gamePanel.pikaList)  
                        pika.update(gamePanel.getWidth(),  
                                    gamePanel.getHeight())  
                    gamePanel.showPika(canvas) // draw  
                    running = gamePanel.killed < gamePanel.pika.length  
                }  
                try {Thread.sleep(FRAME_PERIOD - elapsedTime)} catch (e) {}  
            }  
        } finally {  
            surfaceHolder.unlockCanvasAndPost(canvas)  
        }  
    }  
}
```

vlákno
nemusí
byť jediné

elapsedTime

Frame per second

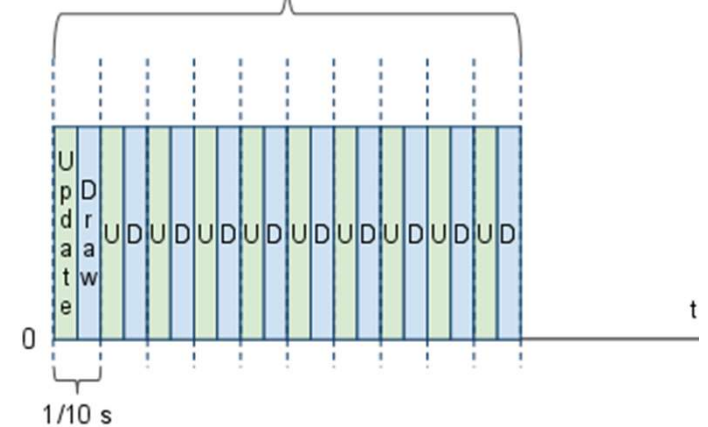
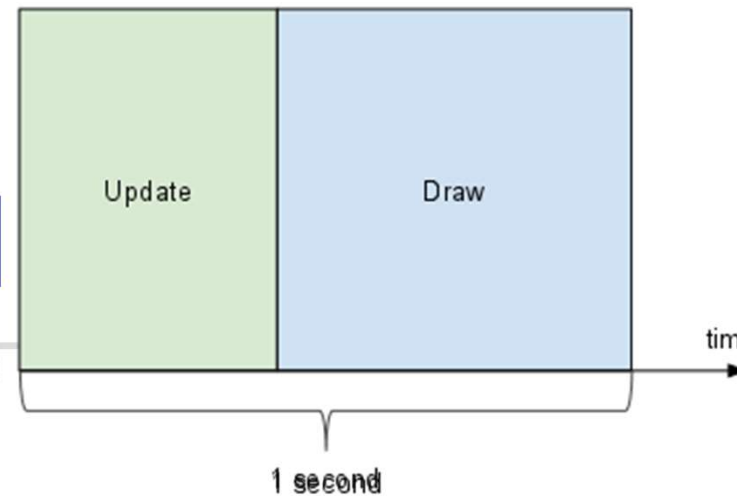
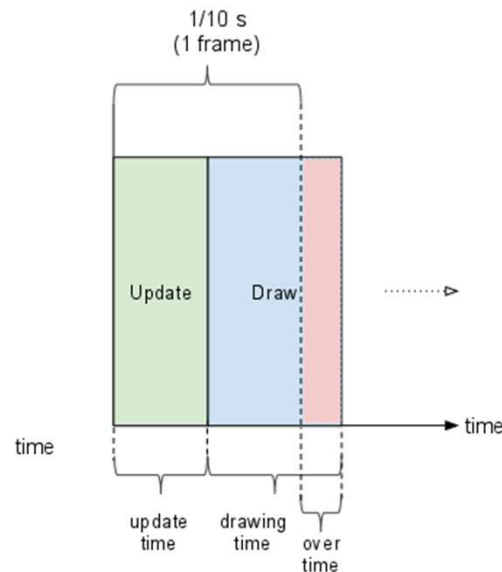
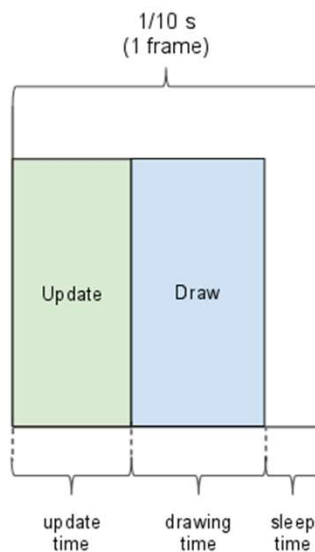
- 1 Frame per Second

Chceli by sme viac, napr. 10 fps

$FRAME_PERIOD = 1000 / 10 // 10 \text{ fps}$

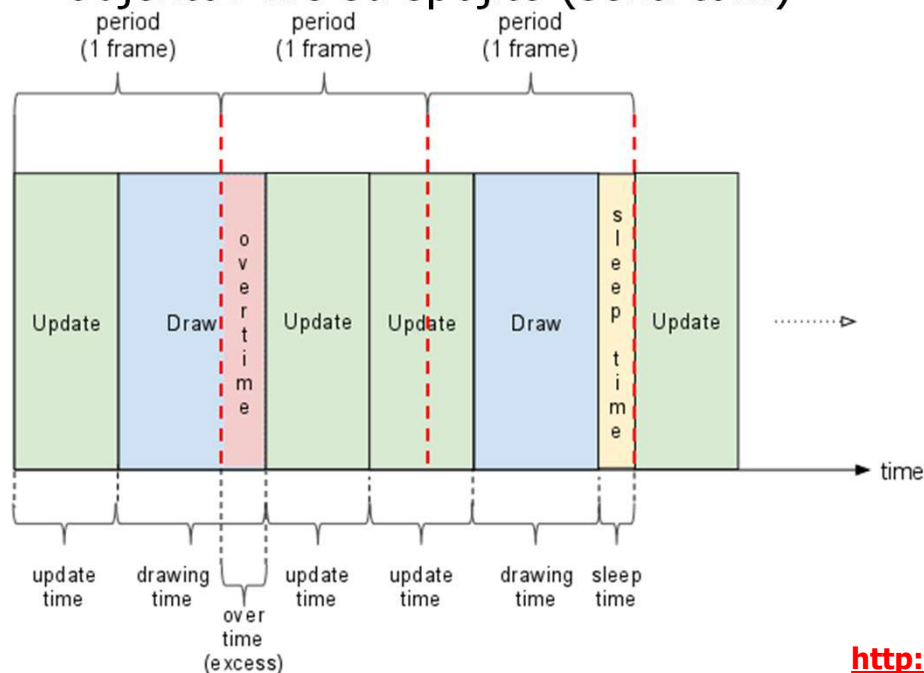
Môže sa nám stat', že to

stihneme alebo nestihneme



Čo ak nestíhame vykreslovať

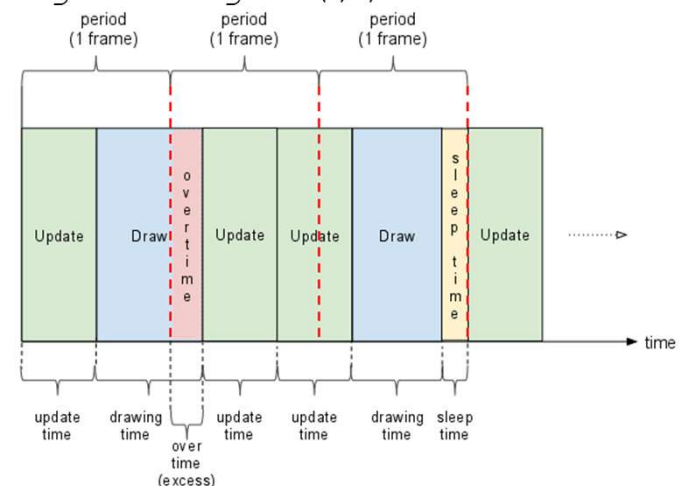
- ak nestíhame vykreslovať, nemali by sme zmenšiť rýchlosť hry,
- rýchlosť hry nie je rýchlosť vykreslovania,
- radšej niektoré prekreslenia scény vynecháme, sústredíme sa na update stavu hry,
- výsledkom je hra, ktorá sa nespomaľuje kvôli vykreslovaniu, ale pohyby objektov nie sú spojité (seká to...)



`FRAME_PERIOD = 1000/50; //50 fps`

Preskočíme pár vykreslování

```
if (elapsedTime <= FRAME_PERIOD) { // lepší případ, stíháme  
    try { // počkáme zbyšný čas  
        Thread.sleep(FRAME_PERIOD - elapsedTime)  
    } catch (InterruptedException e) {}  
}  
  
while (elapsedTime > FRAME_PERIOD) { // nestíháme  
    for (pika in gamePanel.pikaList)  
        pika.update(r.getWidth(), r.getHeight())  
    elapsedTime -= FRAME_PERIOD  
    skippedInPeriod++  
}  
framesInPeriod++
```





GLSurfaceView

- OpenGL renderer
- detaily v kóde pre tých, čo sú 3D...





Gestá

(štandardné)

```
class GesturesActivity : AppCompatActivity(),  
    GestureDetector.OnGestureListener,  
    GestureDetector.OnDoubleTapListener {  
    lateinit var gDetector: GestureDetectorCompat
```

```
GestDetector.OnDoubleTapListener:
```

```
override fun onDoubleTap(event: MotionEvent): Boolean  
override fun onDoubleTapEvent(event: MotionEvent): Boolean  
override fun onSingleTapConfirmed(event: MotionEvent): Boolean
```

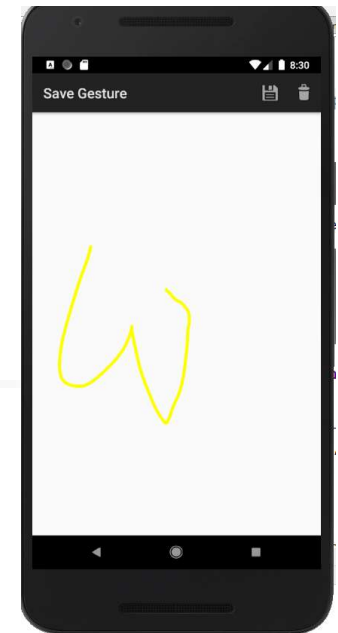
```
GestDetector.OnGestureListener:
```

```
override fun onDown(event: MotionEvent): Boolean  
override fun onFling(event1: MotionEvent, event2: MotionEvent,  
    velocityX: Float, velocityY: Float): Boolean  
override fun onLongPress(event: MotionEvent)  
override fun onScroll(e1: MotionEvent, e2: MotionEvent,  
    distanceX: Float, distanceY: Float): Boolean  
override fun onShowPress(event: MotionEvent)  
override fun onSingleTapUp(event: MotionEvent): Boolean
```

Gestá

(vlastné – definované)

```
class GesturesActivity : AppCompatActivity(),
    OnGesturePerformedListener {
    lateinit var gLibrary: GestureLibrary
    ...
    gLibrary = GestureLibraries.fromRawResource(this,
        R.raw.gestures2 // tento súbor si
                        // vyrobíte v Gesture Editore, uložíte do raw/
    if (gLibrary.load() == false) {
        finish()
    }
    gOverlay.addOnGesturePerformedListener {
        overlay: GestureOverlayView, gesture: Gesture ->
        val predictions = gLibrary.recognize(gesture)
        predictions?.let {
            if (it.size > 0 && it[0].score > 1.0) {
                val action = it[0].name
                Toast.makeText(this, action, Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```





Ako uložiť dáta/nastavenia

(lokálne/na server)

- SharedPreferences - umožní uložiť dvojice (kľúč, hodnota) pre hodnoty typu int, boolean, string, float, ... a poskytuje metódy
 - [get|put][Boolean|Float|String|Long|Int]
- Súbory – ukladá do internej resp. externej pamäte zariadenia
- Databáza – sqlite (<http://www.sqlite.org/>) - open-source, sql-standard, malá a ľahko použiteľná DB vo vašom zariadení

- Vlastný server – protokol najčastejšie http-https

príde neskôr...

~~■ najčastejšie (v bakalárkach) AMP – Apache-MySQL-PHP~~ **OLD STYLE**

- Cloudový server - poskytuje nejaké SDK pre našu platformu
 - www.parse.com – iOS, Android, JS, Unity, PHP, Xamarin, Arduino, ...
 - [Firebase API](#) – iOS, Android, C++
 - [Google datastore API](#) – iOS, Android, JS, PHP, ...

Kľúče si nejako pomenujeme:
`LOGIN_ENTRY_KEY = "Login"`
`SUCCLOGS_ENTRY_KEY = "SUCC"`

SharedPreferences

(nič jednoduchšie...)

LoginActivity si pamätá login a passwd, v prípade úspešného prihlásenia, a tiež počet úspešných a neúspešných prihlásení

```
settings=PreferenceManager.getDefaultSharedPreferences(this)  
    getPreferences(Activity.MODE_PRIVATE) // alebo  
    getSharedPreferences("seti", Activity.MODE_PRIVATE)  
    ...MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE
```

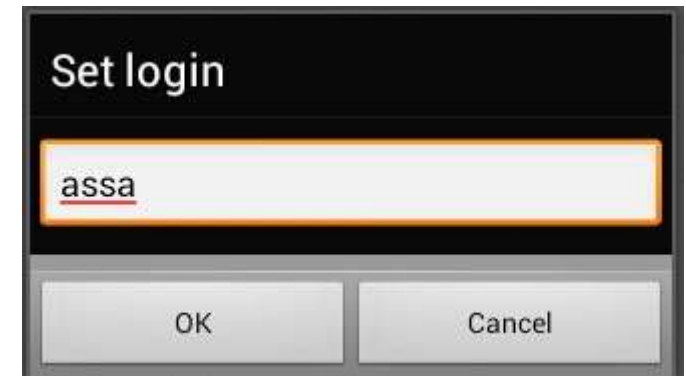
Načítanie:

```
settings.getString(LOGIN_ENTRY_KEY, "") //"" default hodnota  
settings.getInt(SUCCLOGS_ENTRY_KEY, 0) //0 ak sa nenachádza
```

Uloženie:

```
settings.edit() {  
    putString(LOGIN_ENTRY_KEY, "")  
    putString(PASSWORD_ENTRY_KEY, "")  
    remove(SUCCLOGS_ENTRY_KEY)  
    remove(FAILEDLOGS_ENTRY_KEY)  
}
```

PreferenceActivity



```
public class MyPreferenceActivity extends PreferenceActivity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState) //res/xml.setting.xml  
        addPreferencesFromResource(R.xml.settings)
```

```
<PreferenceCategory  
    android:title="@string/pref_login_pass_profile" >
```

```
<EditTextPreference
```

```
    android:title="@Set login"
```

```
    android:summary= "Set your email-login"
```

```
    android:key="prefLogin"/>
```

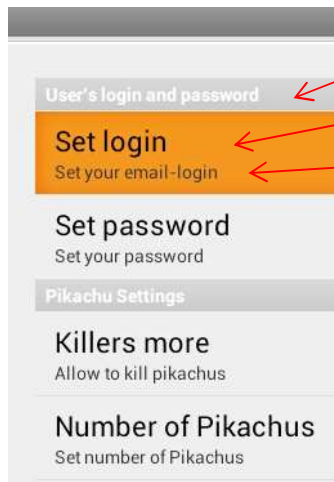
```
<EditTextPreference
```

```
    android:title="@string/pref_pass"
```

```
    android:summary="@string/pref_pass_summary"
```

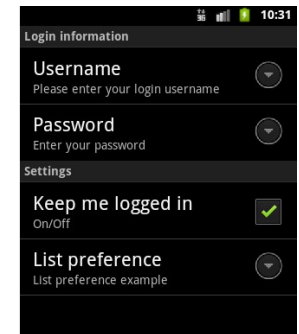
```
    android:inputType="textPassword"
```

```
    android:key="prefPass"/>
```



PreferenceCategories

(xml)



```
<PreferenceCategory android:title= "Pikachu settings" >
```

```
<CheckBoxPreference
```

```
    android:defaultValue="true"
```

```
    android:key="prefKill"
```

```
    android:summary="Allow to kill pikachus"
```

```
    android:title="@Killers mode" >
```

```
</CheckBoxPreference>
```

```
<ListPreference
```

```
    android:key="prefCount"
```

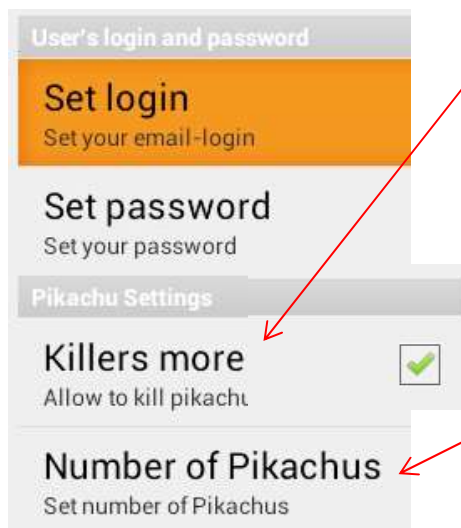
```
    android:entries="@array/pikaCount"
```

```
    android:summary="Set number of Pikachus"
```

```
    android:entryValues="@array/pikaValues"
```

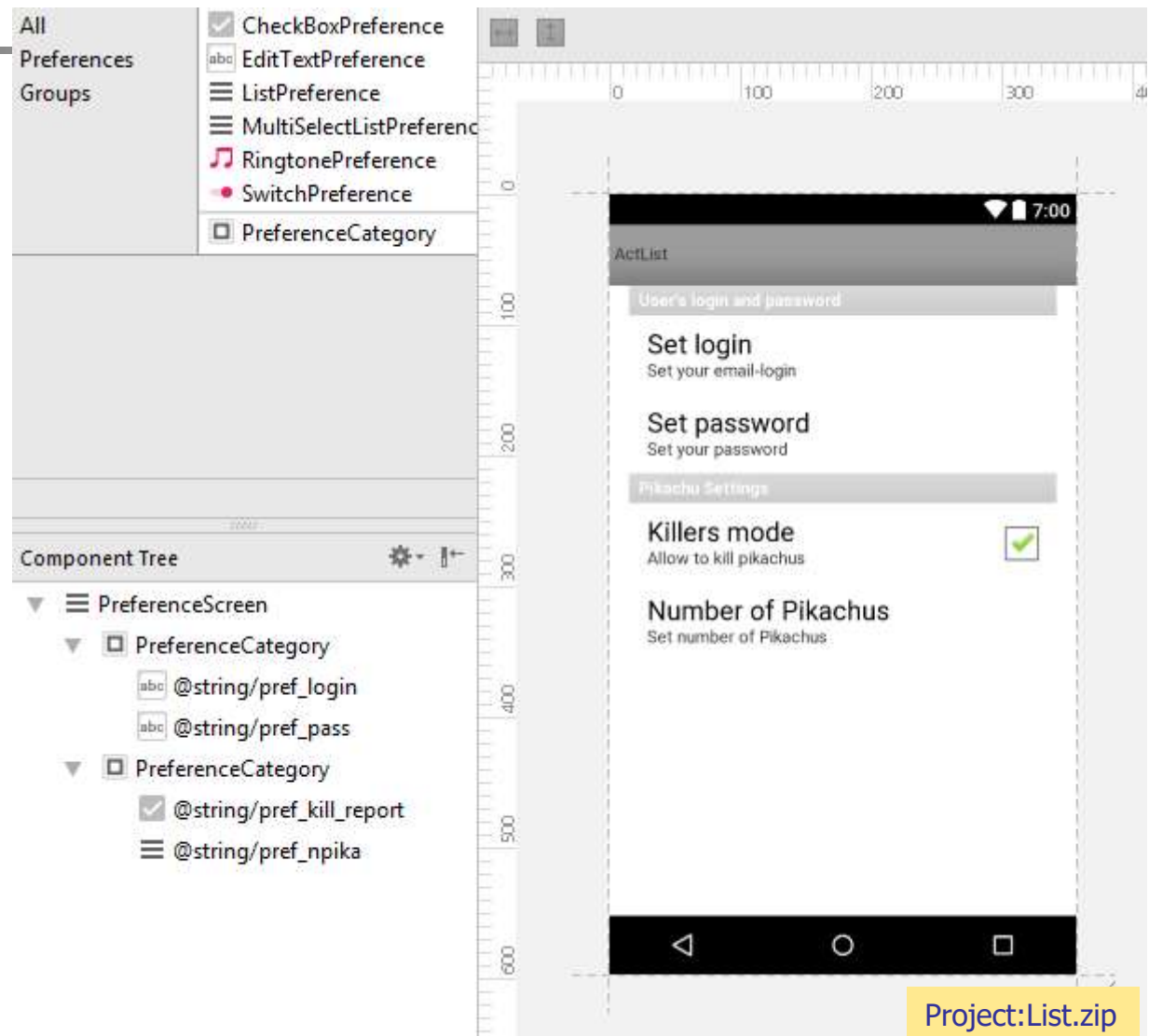
```
    android:title="Number of Pikachus" />
```

```
</PreferenceCategory>
```



PreferenceCategories

(editor)



Project:List.zip

ListPreferences

```
<resources>
  <string-array name="pikaCount">
    <item name="1">1..9</item>
    <item name="10">10..99</item>
    <item name="100">100..999</item>
    <item name="1000">1000-</item>
  </string-array>
  <string-array name="pikaValues">
    <item name="1">5</item>
    <item name="10">50</item>
    <item name="100">500</item>
    <item name="1000">5000</item>
  </string-array>
</resources>
```

Number of Pokachus

1..9	<input type="radio"/>
10..99	<input checked="" type="radio"/>
100..999	<input type="radio"/>
1000-	<input type="radio"/>

Cancel

Runtime Permissions

Povolenia sú:

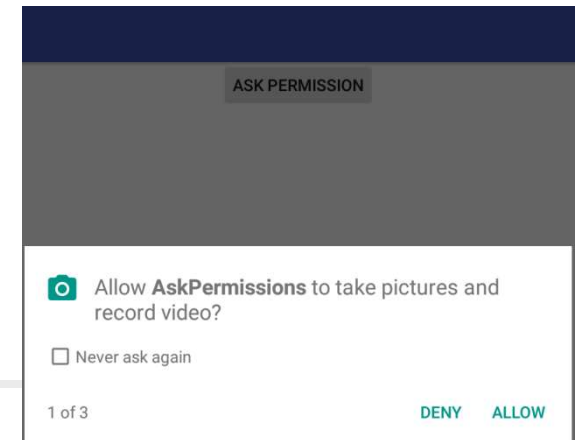
- neohrozujú vaše privátne dáta (INTERNET, BLUETOOTH, ACCESS_WIFI)
- nebezpečné (ACCESS_FINE_LOCATION, [READ/WRITE]_CONTACTS)

Ak máte Android ≤ 5.1 || target SDK < 23 , `<uses-permissions` v Manifest.xml, Povolenia sa získavajú staticky pri inštalácii, ak užívateľ odmietne, neinštaluje sa.

Inak (Android ≥ 6.0 || target SDK ≥ 23) aplikácia môže žiadať počas behu. Ak užívateľ odmietne, aplikácia beží ďalej.

Aj dynamické permissions píšete do AndroidManifest.xml

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission-sdk-23 android:name="android.permission.READ_CONTACTS" />
<uses-permission-sdk-23 android:name="android.permission.WRITE_CONTACTS" />
<uses-permission-sdk-23 android:name="android.permission.ACCESS_FINE_LOCATION" />
```



Runtime Permissions

```
val RUNTIME_PERMISSION_REQUEST_CODE = 777
val perms = arrayOf(
    Manifest.permission.WRITE_CONTACTS,
    Manifest.permission.CAMERA,
    Manifest.permission.ACCESS_FINE_LOCATION ... )
if (getApplicationContext().checkSelfPermission(
    Manifest.permission.READ_CONTACTS) !=
    PackageManager.PERMISSION_GRANTED) {
    requestPermissions(perms, RUNTIME_PERMISSION_REQUEST_CODE)
}

override fun onRequestPermissionsResult( requestCode: Int,
    permissions: Array<String>, grantResults: IntArray) {
    when (requestCode) {
        RUNTIME_PERMISSION_REQUEST_CODE -> {
            for (i in grantResults.indices) {
                if (grantResults[i]==PackageManager.PERMISSION_GRANTED) {
                    Log.d("Permissions", "GRANTED")
                } else { // denied
                    Log.d("Permissions", "DENIED")
                }
            }
        }
    }
}
```

