Pokračovanie

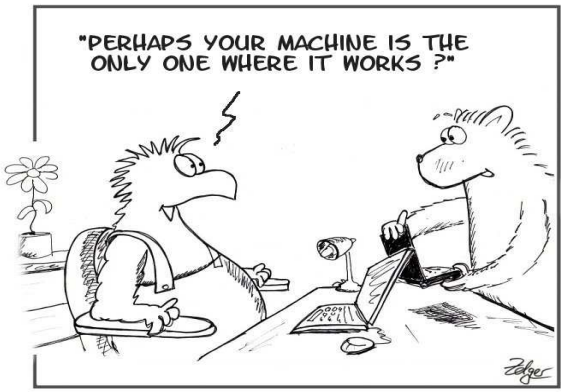
Menu
SurfaceView, Gestá
SharedPreferences
PreferenceActivity
RuntimePermissions

Peter Borovanský
KAI, I-18

MS-Teams: 2sf3ph4, List, github

borovan 'at' ii.fmph.uniba.sk

Works only on my mobile



It works on my machine

Responzívnosť:

- nebeží to len mne, na mojom zariadení
- najväčší problém je asi rozlíšenie obrazovky a to
- v kombinácii s tzv. *absolute layout*
- komponenty nemajú mať bezdôvodne fixnú veľkosť
- používať constraint/relative layout, wrap-content/match-parrent
- ak kreslím do canvasu, zistím si jeho veľkosť
- každé View má *časom-raz* .width, .height
- rozvrhnem si playground výpočtom z width, height
- v emulátore používam portrait/landscape
- nakonfigurujem si v AVD zariadenie s iným rozliškom
- aspoň jedno...

- Google hlása tendenciu penalizovať weby, ktoré nie sú prispôsobené mobilným zariadeniam

http://weblogs.asp.net/fredriknormen/stop-saying-quot-but-it-works-on-my-computer-quot

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 podtriedy 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
- aj na programovanie.

SurfaceView

(podtrieda View, nadtrieda tried ako GLSurfaceView, VideoView)

SurfaceView je typicky renderované iným vláknom pomocou triedy SurfaceHolder

```
class GamePanel(context:Context) : SurfaceView(context),
                                   SurfaceHolder.Callback {

lateinit  var thread : GameThread              // vlákno hry
init {                  // surface holder je ten, kto modifikuje SfV
   holder.addCallback(this) // holder interface vyžaduje 3 metódy
   thread = GameThread(this)
   setFocusable(true)
}
override fun surfaceCreated(holder: SurfaceHolder) {
   thread.start()                        // entry point pre SurfaceView
}
override fun surfaceChanged(holder: SurfaceHolder,
        format: Int, width: Int, height: Int) { .. }
override fun surfaceDestroyed(holder: SurfaceHolder) {
   // exit point SfV-treba zastaviť vlákno hry a počkať kým skončí
   // viď priložený projekt...
```

interface

https://developer.android.com/reference/android/view/SurfaceView.html

Project:List.zip

SurfaceView

(SurfaceView.Callback interface)

SurfaceView je typicky renderované iným vláknom pomocou triedy SurfaceHolder

```
class GamePanel(context:Context) : SurfaceView(context)
lateinit  var thread : GameThread                    // vlákno hry
holder.addCallback (object : SurfaceHolder.Callback {
    override fun surfaceCreated(holder: SurfaceHolder) {
        thread.start()
    }
    override fun surfaceChanged(holder:SurfaceHolder,format:Int,width:Int,height:Int
      ) {}
    override fun surfaceDestroyed(holder: SurfaceHolder) {
        var retry = true;
        thread.running = false;
        while (retry) {
            try {
                thread.join()
                retry = false
            } catch (e :InterruptedException) {}
    }} }
)
```

interface

https://developer.android.com/reference/android/view/SurfaceView.html

Project:List.zip

GameThread

(čo robí vlákno hry - alternatíva k invalidate)

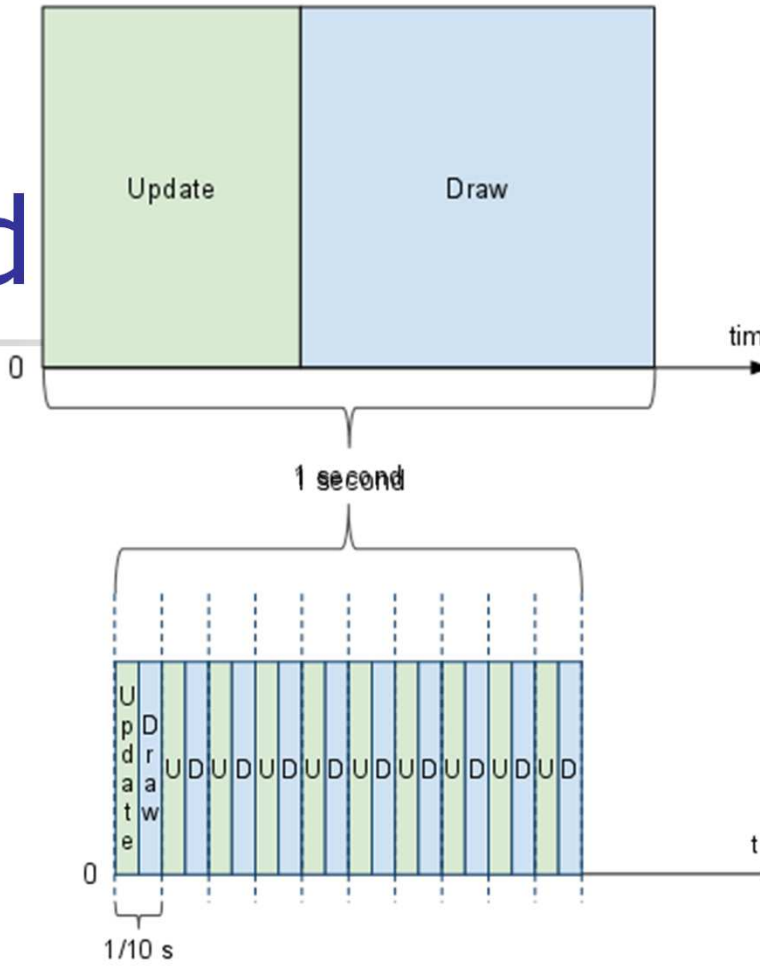
```
class GameThread(val gamePanel: GamePanel) : Thread() {
                              // zapamätáme v konštruktore GameTread
override fun run() {  // hlavný cyklus vlákna, hry, simulácie
   val surfaceHolder = gamePanel.holder
   while (running) {   // kým beží hra
        try {
            canvas = 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 (){}
            }
    } finally {
            surfaceHolder.unlockCanvasAndPost(canvas)
```

vlákno nemusí byť jediné

elapsedTime

Project:List.zip

Frame per second



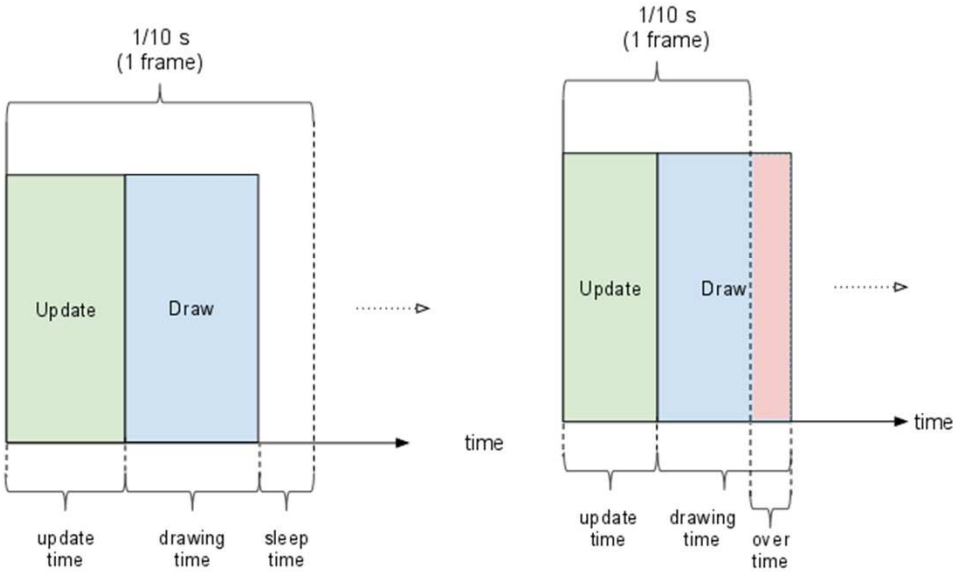
- 1 Frame per Second

Chceli by sme viac, napr. 10 fps

```
FRAME_PERIOD = 1000 / 10 //10 fps
```

Môže sa nám stať, že to

stihneme alebo nestihneme

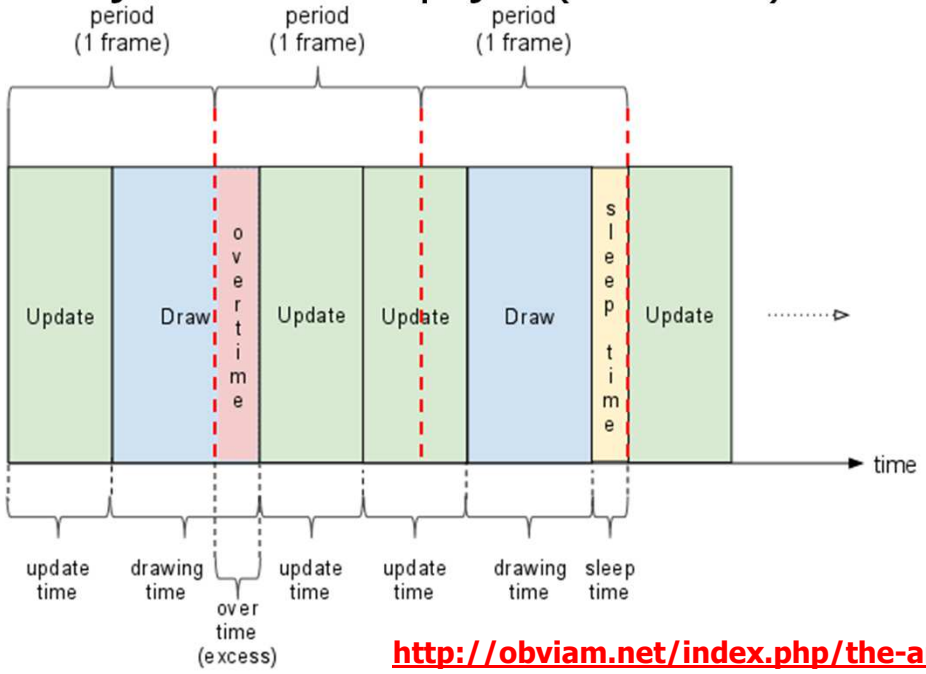


http://obviam.net/index.php/the-android-game-loop/

Project:List.zip

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 nespomaluje kvôli vykreslovaniu, ale pohyby objektov nie sú spojité (seká to...)



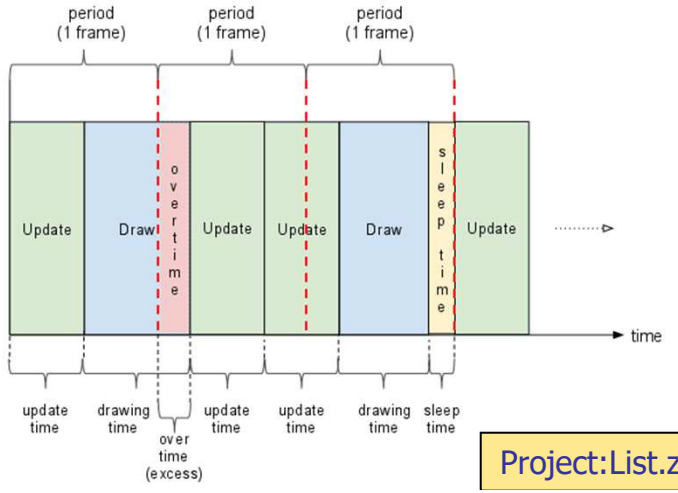
http://obviam.net/index.php/the-android-game-loop/

Project:List.zip

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

Preskočíme pár vykreslovaní

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



http://obviam.net/index.php/the-android-game-loop/

Project:List.zip

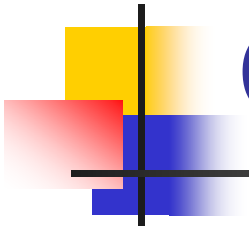
DU-3

- programujte vašu obľúbenú hru, idea je dynamickú, nie logickú
- navrhnite si triedy pre všetky objekty vo vašej hre
- každý nech má metódu update() a event. aj draw(canvas), onDraw()

```
class InvadersView(context: Context, private val size: Point)
: SurfaceView(context), Runnable {

private fun update(fps: Long) {
private fun draw() { … }
override fun run() { … }
```

http://gamecodeschool.com/kotlin/coding-a-space-invaders-game-in-kotlin/



Game run thread

```
override fun run() {
    var fps: Long = 0                                    // frame rate
     while (playing) {
         val startFrameTime = System.currentTimeMillis()      // current time
         if (!paused) {
             update(fps)
         }
         draw()
                                      // calculate the fps rate this frame
         val timeThisFrame = System.currentTimeMillis() - startFrameTime
         if (timeThisFrame >= 1) {
             fps = 1000 / timeThisFrame
         }

                                      // Play a sound based on the menace level
         if (!paused && ((startFrameTime - lastMenaceTime) > menaceInterval))
             menacePlayer()
     }
}
```

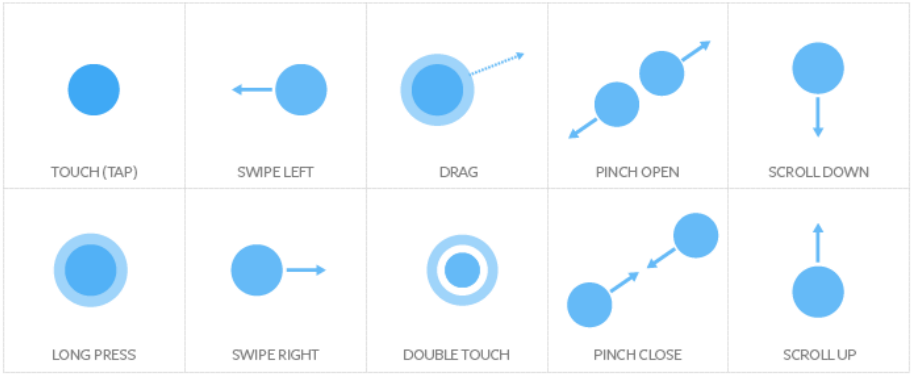
http://gamecodeschool.com/kotlin/coding-a-space-invaders-game-in-kotlin/

GLSurfaceView

- GLSurfaceView je podtrieda SurfaceView
- openGL renderer
- detaily v kóde pre tých, čo sú 3D...

Project:List.zip

Gestá
(štandardné)



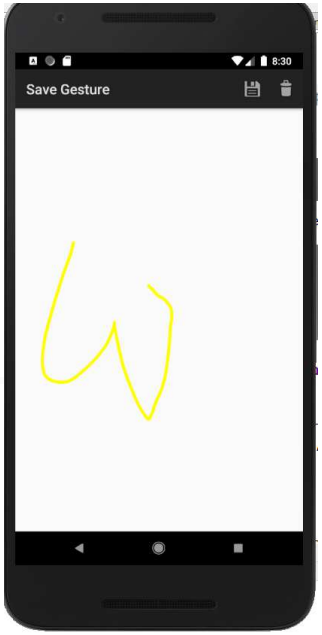

```
class GesturesActivity : AppCompatActivity(),
    GestureDetector.OnGestureListener,
    GestureDetector.OnDoubleTapListener {
    lateinit var gDetector: GestureDetectorCompat

interface GestureDetector.OnDoubleTapListener:
•   override fun onDoubleTap(event: MotionEvent): Boolean
•   override fun onDoubleTapEvent(event: MotionEvent): Boolean
•   override fun onSingleTapConfirmed(event: MotionEvent): Boolean

GestureDetector.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
```

Project:List.zip

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, vložíte do raw
if (gLibrary.load() == false) {
    finish()
}
gOverlay.addOnGesturePerformedListener {
  overlay:GestureOverlayView, gesture:Gesture ->
    val predictions:List<Prediction>= 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()
      }
    }
}
```

Project:GestureBuilder.zip

Ako uložiť dáta/nastavenia
(lokálne/na server)

- SharedPreferences - umožní uložiť dvojice (kľúč, hodnota) pre hodnoty typu int, boolean, string, float, … a podskytuje 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

príde neskôr...

- Vlastný server – protokol najčastejšie http-https
  - ~~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_WRITABLE
```

Načítanie:

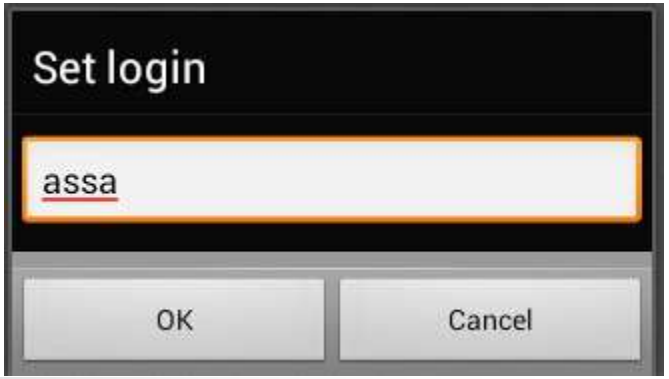
```
settings.getString(LOGIN_ENTRY_KEY, "")//"" default hodnota
settings.getInt(SUCCLOGS_ENTRY_KEY, 0) //0 ak sa nenachádza
```

Uloženie:

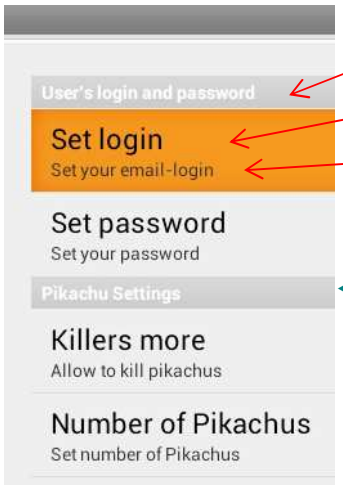
```
settings.edit() {
    putString(LOGIN_ENTRY_KEY, "")
    putString(PASSWORD_ENTRY_KEY, "")
    remove(SUCCLOGS_ENTRY_KEY)
    remove(FAILEDLOGS_ENTRY_KEY)
}
```

Project:List.zip

PreferenceActivity



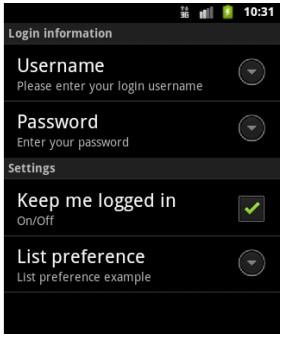

```
public class MyPreferenceActivity extends PreferenceActivity {
public void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState)    //res/xml.setting.xml
   addPreferencesFromResource(R.xml.settings)
<PreferenceCategory
   android:title="@string/pref_login_pass_profile" >
      <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"/>
```

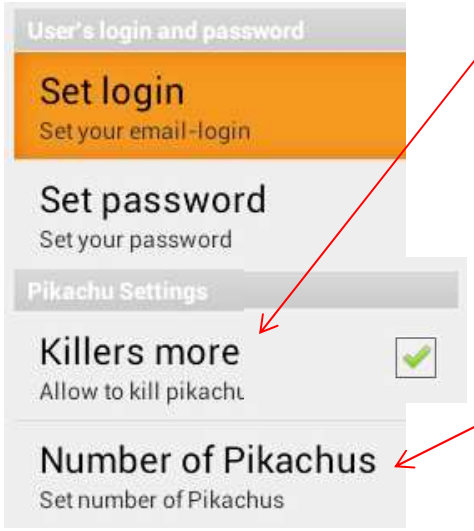


Project:List.zip

PreferenceCategories

(xml)





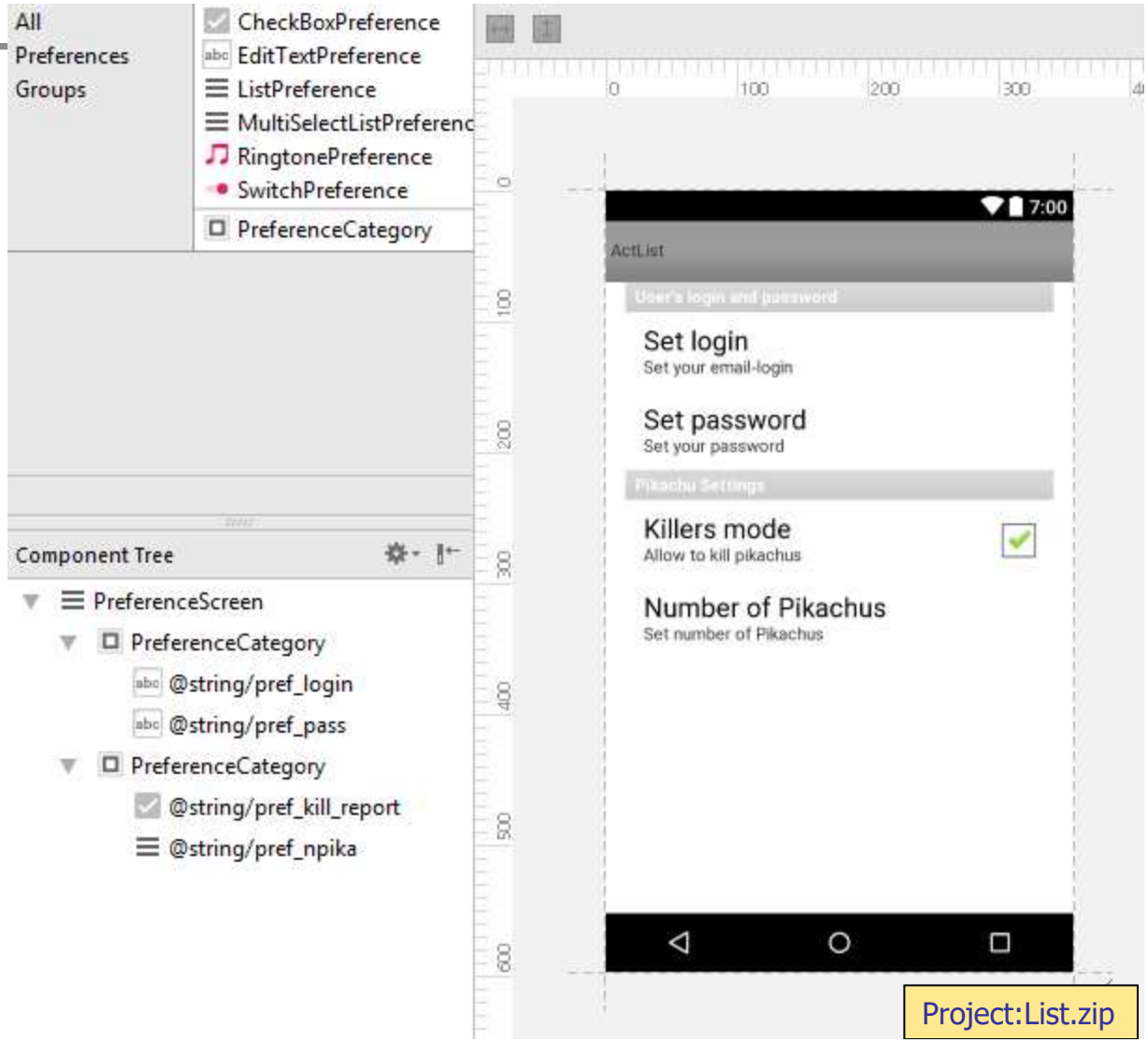

```
<PreferenceCategory android:title= "Pikachu settings" >
    <CheckBoxPreference
        android:defaultValue="true"
        android:key="prefKill"
        android:summary="Allow to kill pikachus"
        android:title="@Killers mode" >
</CheckBoxPreference>

    <ListPreference
        android:key="prefCount"
        android:entries="@array/pikaCount"
        android:summary="Set number of Pikachus"
        android:entryValues="@array/pikaValues"
        android:title="Number of Pikachus" />
</PreferenceCategory>
```

Project:List.zip

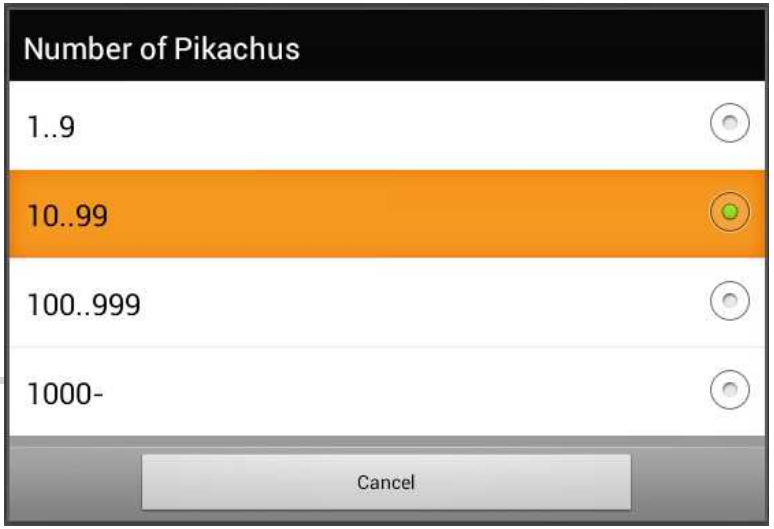
PreferenceCategories

(editor)



Project:List.zip

ListPreferences



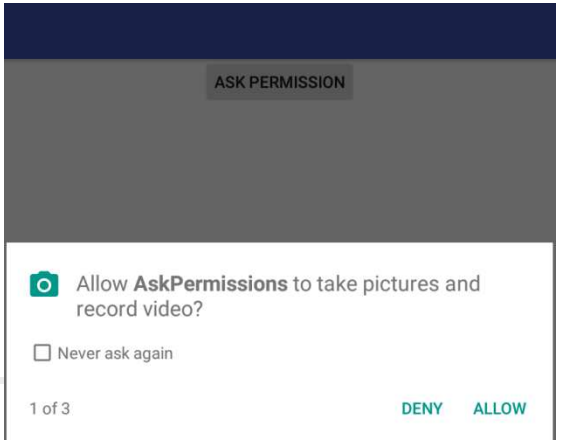

```
<resources>
<string-array name="pikaCount">
        <item name="1">1..9</item>
        <item name="10">10..99</item>
        <item name="100">100..999</item>
        <item name="1000">1000-</item>
    </string-array>
<string-array name="pikaValues">
        <item name="1">5</item>
        <item name="10">50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

</resources>
```

```
try {
  PIKATCHUS = Integer.parseInt(
    settings.getString("prefCount", "5000"))
} catch (e : Exception) {
   PIKATCHUS = 5000
}
```

Project:List.zip

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 inštalácií, 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" /
```

Project:RunTimePermissions.zip

Úrovne povolení

Normal Permissions –

nízka úroveň narušenia súkromia

- ACCESS_NETWORK_STATE
- CHANGE_NETWORK_STATE
- ACCESS_WIFI_STATE
- CHANGE_WIFI_STATE
- CHANGE_WIFI_MULTICAST_STATE
- BLUETOOTH
- BLUETOOTH_ADMIN
- INTERNET
- SET_ALARM
- SET_WALLPAPER
- VIBRATE
- WAKE_LOCK

Signature Permissions –

appka musí byť podpísaná autoritou

- BIND_ACCESSIBILITY_SERVICE
- BIND_NFC_SERVICE
- BIND_TV_INPUT
- BIND_WALLPAPER
- READ/WRITE_VOICEMAIL
- WRITE_SETTINGS

Dangerous Permissions –

appka musí explicitne žiadať povolenie

- READ/WRITE_CALENDAR
- CAMERA
- READ/WRITE_CALL_LOG
- READ/WRITE_CONTACTS
- GET_ACCOUNTS
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- SEND/RECEIVE_SMS

Permissions do Manifest.xml

(ak API >=23)

Okrem tohoto:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
a veľmi skoro budeme potrebovať ...
<uses-permission android:name="android.permission.INTERNET"/>
```

treba v kóde dynamicky žiadať o povolenie (zjednodušený kód):

test verzie API,

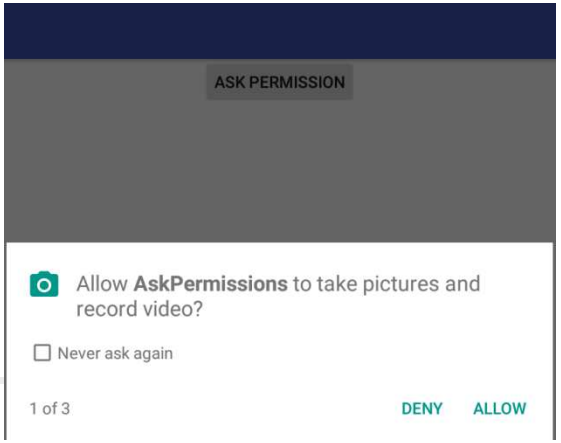
test či je permission schválená... ak nie, vyrobím zoznam permissions

```
if (android.os.Build.VERSION.SDK_INT >= 23) {
   if (getApplicationContext().checkSelfPermission(permission) !=
         PackageManager.PERMISSION_GRANTED)
            permissionsList.add(permission)
}
```

... a následne požiadať o povolenia:

```
requestPermissions(permissionsList.toArray()),
        REQUEST_CODE_ASK_MULTIPLE_PERMISSIONS)
```

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")
```

Project:RunTimePermissions.zip