

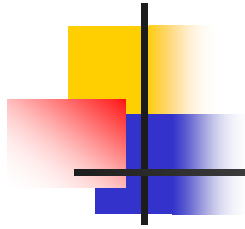


Pokračovanie

Canvas, SufaceView,
Menu, Gesta, SharedPreferences

Peter Borovanský
KAI, I-18

borovan 'at' ii.fmph.uniba.sk

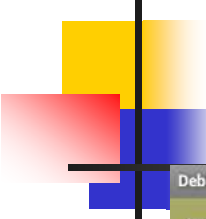


Bolo minule

- layouts, najmä constraint layout
- ListView, najmä kvôli DÚ2
- intent, intent data
- **<intent-filter />** a AndroidManifest, permissions
- **startActivity, startActivityForResult**

Domáca úloha 2

(deadline do 10.nov)



Debilníček	
zavolať svokre, že ju obdivujem...	21/11/12
vencit Harryho	21/11/12
pivo	21/11/12
syr	21/11/12
mlieko	21/11/12

Vytvorte (malú) aplikáciu zvanú Debilníček, resp. Nákupný košík:

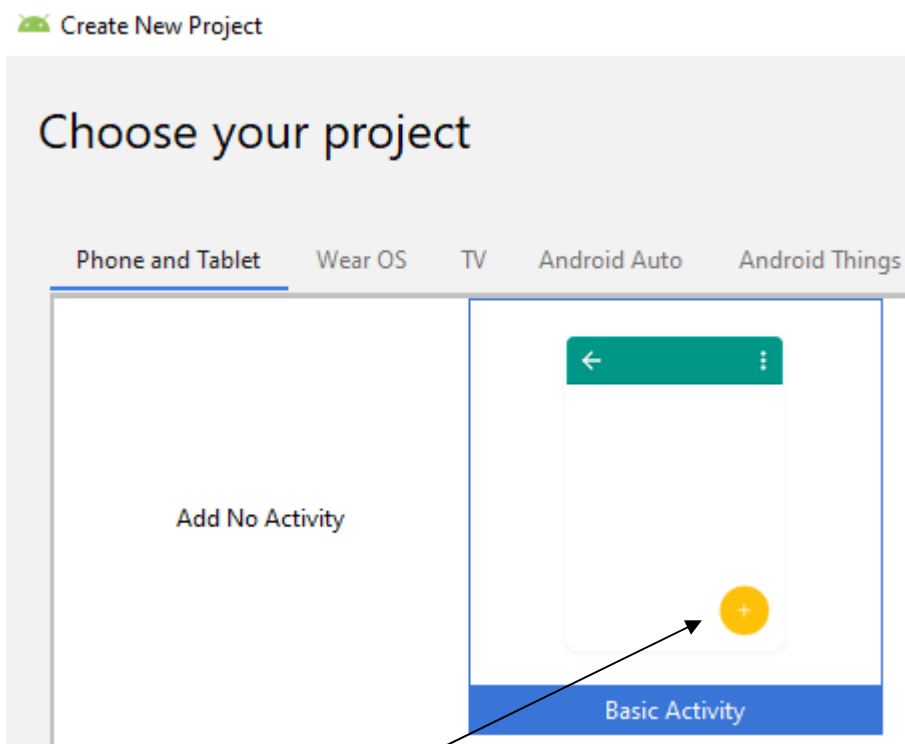
- umožní poznamenať si, veci, predmety, činnosti do tzv. ToDo listu,
- dovoľí nastaviť deadline na splnenie činnosti pomocou dátumu/času,
- ak to bude verzia nákupný košík, tak aj počet predmetov,
- umožní ich vymazať, resp. označiť za vybavené/nakúpené, resp. vymazať všetky vybavené,
- kontroluje deadline, a upozorní správou, zvukom na prešvihnutý deadline,
- pri vypnutí aplikácie si zoznam zapamätá, pri otvorení sa zoznam obnoví



Material Design

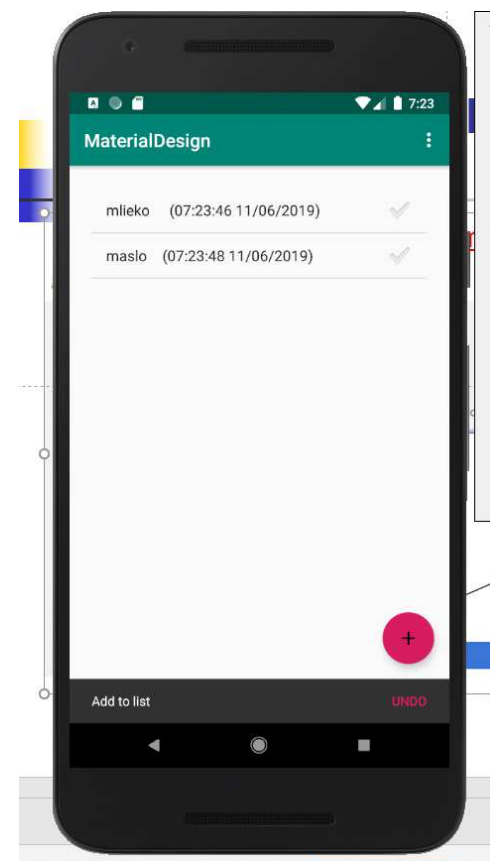
(ako začať)

- <https://developer.android.com/guide/topics/ui/look-and-feel>



Floating Action Button

`<com.google.android.material.floatingactionbutton.FloatingActionButton`



Project: MaterialDesign.zip



Bude dnes

- intent, intent data
- Canvas
 - onTouch, onDraw, invalidate, postInvalidate
- SurfaceView, GLSurfaceView
- Menu
 - OptionMenu
 - ContextMenu
- Gestá
 - štandardné
 - vlastné
- SharedPreferences
- PreferenceActivity



Domáca úloha 3 - Gameska

(deadline do 25.nov)

Cieľom domácej úlohy je, aby ste navrhli a naprogramovali pomocou triedy Canvas/SurfaceView, nejakú logickú **ALEBO** akčnú hru.

Pravidlá hodnotenia:

- **5 bodov**, ak je hra funkčná, hrateľná, ale žiaden super dojem, bez bonusových features...
- **10 bodov**, ak je radosť si zahrať, hoc aj hra môže byť jednoduchá, ale je vyšperkovaná zaujímavými fičúrkami, graficky cool, ...
- **+1 bod**, ak vaša úloha NEBUDE remake z DÚ 1 (čo ste robili pre MIT Inventor), informujte o tom v README.TXT, aby sme sa neuhľadali k smrti...,
- **-1 bod**, ak kód bude v Jave, informujte o tom v README.TXT, aj keď u tejto úlohy sa zdrojáky budú pozerat'...
- ak kód, resp. výrazná časť z neho, nebude vaše dielo, **0 bodov**.



PhotoActivity

(data z intentu)

Princíp intent-`startActivityForResult` spolu s `onActivityResult` ešte raz:

```
val takePictureIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE);  
    // test, či existuje komponent, ktorý spracuje intent  
if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
    Toast.makeText(this@PhotoActivity,  
        "smile ... taking picture", Toast.LENGTH_LONG).show();  
    startActivityForResult(takePictureIntent, REQUEST_IMAGE_CAPTURE);  
} else {  
    Toast.makeText(this@PhotoActivity,  
        "sorry ... no picture", Toast.LENGTH_LONG).show();  
}  
  
private val REQUEST_IMAGE_CAPTURE = 690
```



PhotoActivity

(data z intentu)

V callback `onActivityResult` získavame z intentu data a interpretujeme ako bitmapu, teda odfotený obrázok:

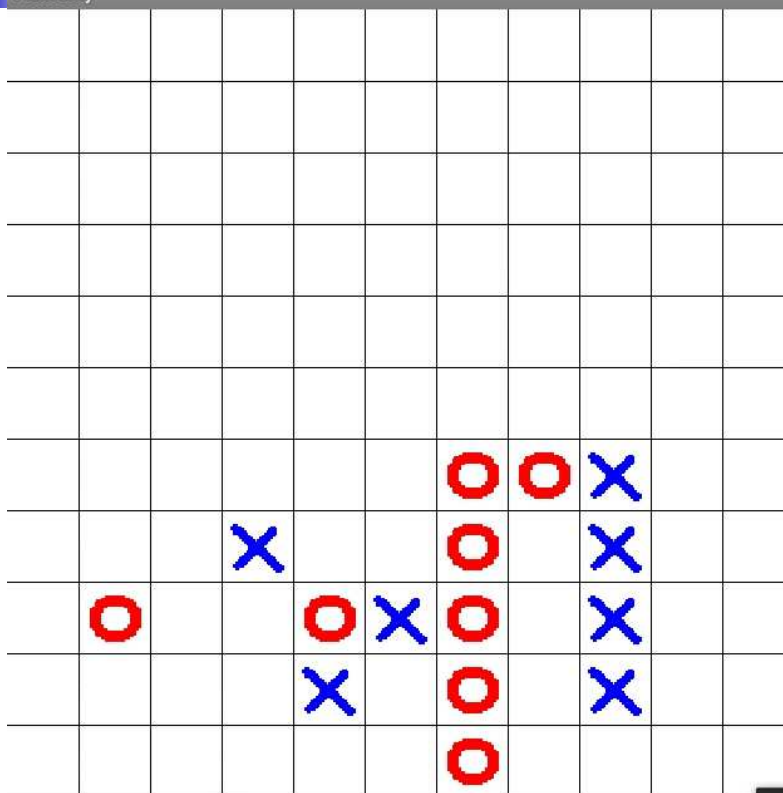
```
private val REQUEST_IMAGE_CAPTURE = 690

override fun onActivityResult(
    requestCode: Int,
    resultCode: Int,
    data: Intent?) {
    if (requestCode == REQUEST_IMAGE_CAPTURE &&
        resultCode == RESULT_OK) {
        Toast.makeText(this@PhotoActivity, "thanks ...",
            Toast.LENGTH_LONG).show()
        val extras = data?.getExtras() // elvis operátor
        val imageBitmap = extras?.get("data") as Bitmap
        pictureImageView.setImageBitmap(imageBitmap)
    }
}
```


Piškvorky

(logická hra v canvase)

Piškvorky



Príklad logickej hry (hlavolamu), kde

- na pozadí **nebeží žiadne vlákno**,
- teda nehýbu sa postavičky,
- nemusíme pravidelne prekreslovať plochu, aby sme navodili ilúziu

☐ hry

☐ simulácie

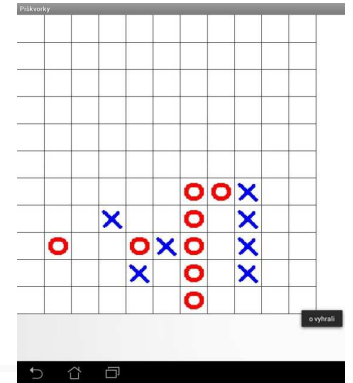
Na príklade Piškvorky ilustrujeme, že View má metódy:

- onTouch
- onDraw



onTouch vo View

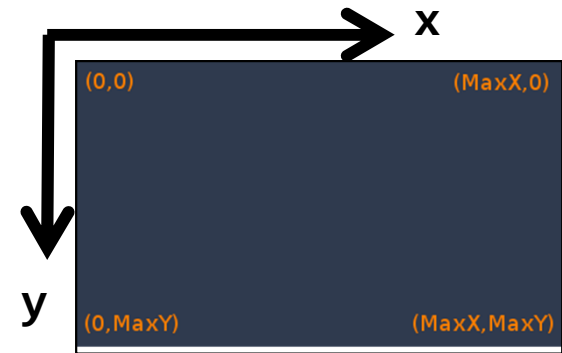
(onTouchEvent)



```
class PiskyView(context: Context, attrs: AttributeSet) :  
    View(context) {    // Piškvorky sú podtrieda View  
                        // načítanie bitmapy obrázkov, postavičiek  
  
    o_img = resources.getDrawable(R.drawable.o).toBitmap()  
    x_img = resources.getDrawable(R.drawable.x).toBitmap()  
  
    override fun onTouchEvent(e: MotionEvent): Boolean {  
        if (e.action == MotionEvent.ACTION_DOWN) {  
            val iX = (e.x / cellSize).toInt()           // transformácia  
            val iY = (e.y / cellSize).toInt()           // pixlov na bunku  
            if (iX >= SIZE || iY >= SIZE) return true   // mimo hraciu dosku  
            if (playGround[iY][iX] == -1) {             // voľné políčko ?  
                playGround[iY][iX] = onTurn            // polož značku hráča  
                onTurn = 1 - onTurn                    // na ťahu, a ide súper  
                invalidate()                            // toto nakoniec prekreslí view  
                val winner = check(iX, iY)              // vyhodnotenie víťazov...  
                if (winner != -1)  
                    Toast.makeText(context, "x vyhrali", Toast.LENGTH_LONG).show()  
            } else
```

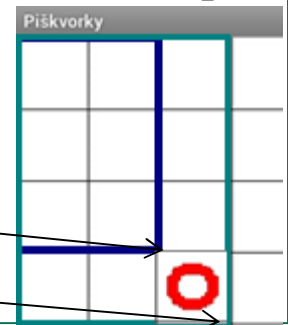
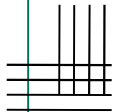
onDraw vo View

(kreslenie do Canvas)



```
override protected fun onDraw(canvas: Canvas) { // paint()
    minSize = Math.min(getWidth(), getHeight()) - 2
    cellSize = minSize / SIZE

    canvas.drawColor(Color.WHITE)
    val p = Paint()
    p.setColor(Color.BLACK)
    p.setStrokeWidth(1F)
    for (i in 1..SIZE) {
        canvas.drawLine(i*cellSize, 0F, i*cellSize, minSize, p)
        canvas.drawLine(0F, i*cellSize, minSize, i*cellSize, p)
    }
    for (y in 0 until SIZE) {
        for (x in 0 until SIZE) {
            canvas.drawBitmap(o_img, srcRect,
                destRect,
                p);
        }
    }
}
```



Project:List.zip



Maľovátko

(MotionEvent actions)

Finger
paint

```
private val mPath: Path // android.graphics.Path
override protected fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    canvas.drawPath(mPath, mPaint)
}

override fun onTouchEvent(event: MotionEvent): Boolean {
    val x = event.x
    val y = event.y
    when (event.action) {
        MotionEvent.ACTION_DOWN -> {
            startTouch(x, y) invalidate() }
        MotionEvent.ACTION_MOVE -> {
            moveTouch(x, y) invalidate() }
        MotionEvent.ACTION_UP -> {
            upTouch() invalidate() }
    }
    return true
}
```



Maľovátko

(bezier vs. linear - nebezier)

Finger
Paint

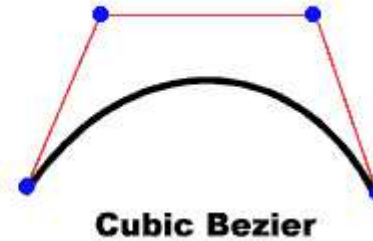
```
private fun startTouch(x: Float, y: Float) {  
    mPath.moveTo(x, y)  
    lastX = x  
    lastY = y  
}  
  
private val TOLERANCE = 5f  
private fun moveTouch(x: Float, y: Float) {  
    val dx = Math.abs(x - lastX)  
    val dy = Math.abs(y - lastY)  
    if (dx >= TOLERANCE || dy >= TOLERANCE) {  
  
        mPath.quadTo(lastX, lastY, (x+lastX)/2, (y+lastY)/2)  
  
        mPath.lineTo(x, y);  
  
        lastX = x  
        lastY = y  
    }  
}
```

Slide 13

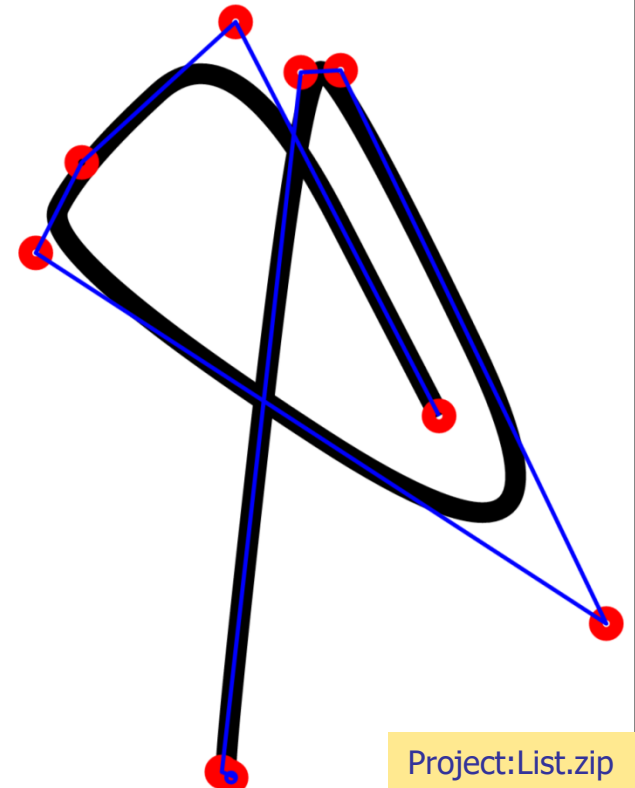
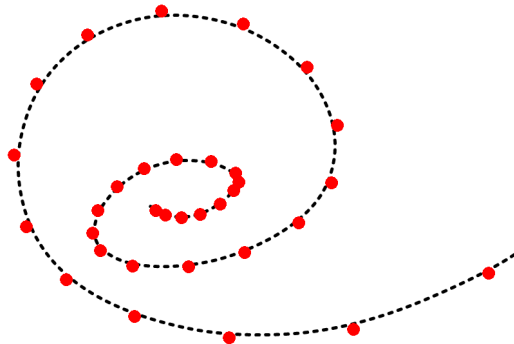
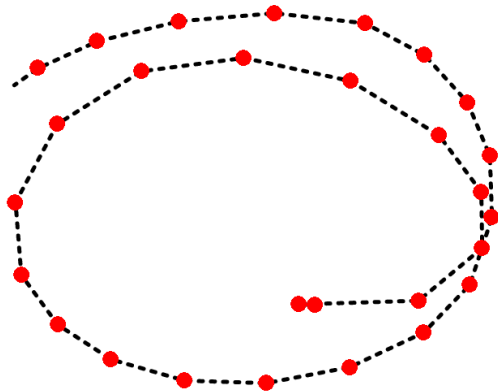
PB1

Peter Borovansky, 11/6/2019

Bézier



- `lineTo(x,y)`
- `quadTo(controlX, controlY, x, y)`
- `cubeTo(controlX1, controlY1, controlX2, controlY2, x, y)`



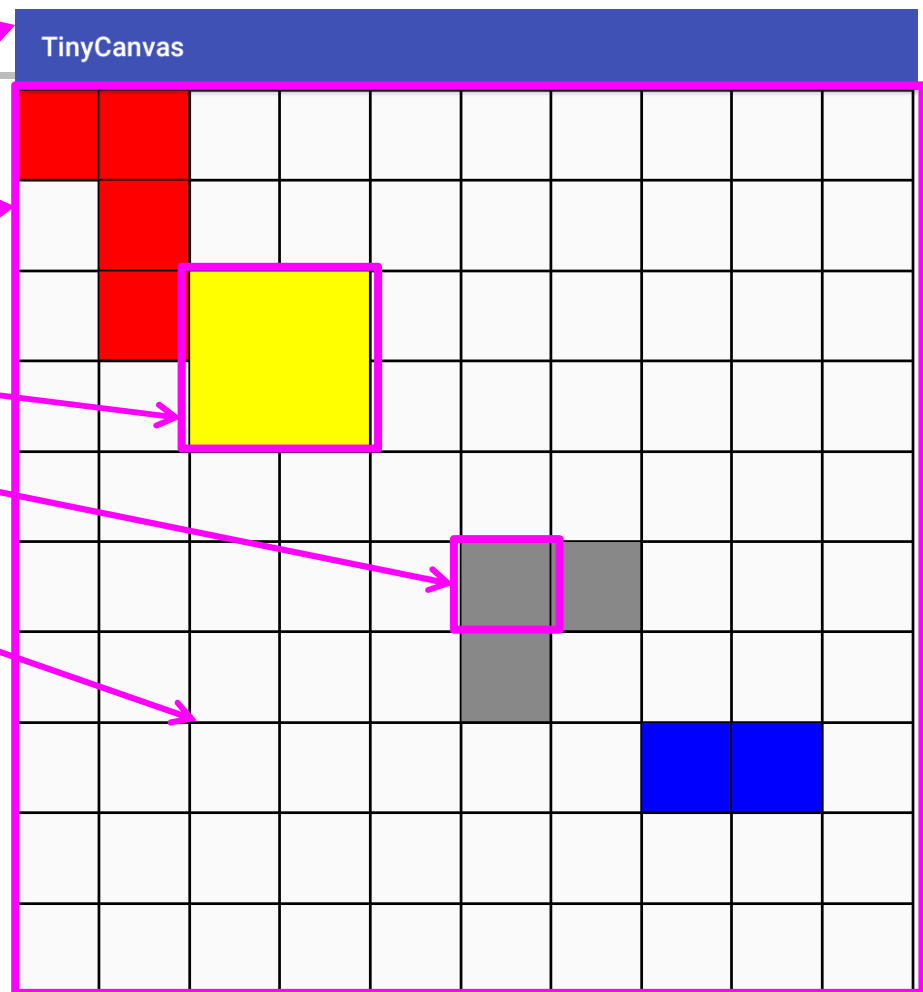
Hierarchia

Objektov/tried:

- **Canvas**
- **Scena**
- **Tvar**
- **Stvorcek**
- **Mreza**

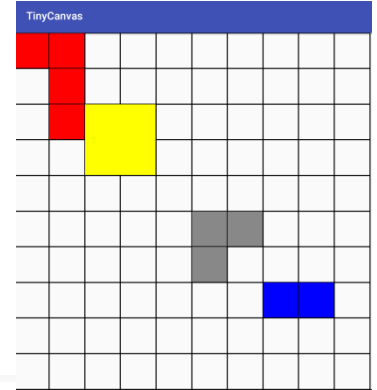
Každý reaguje na:

- **onTouch ()**
- **onDraw ()**



Hierarchia

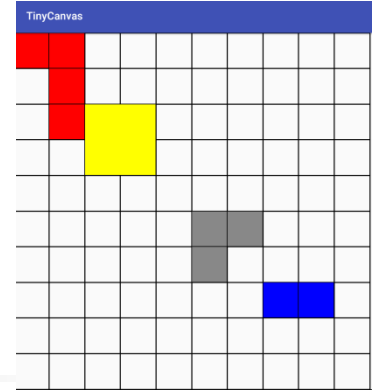
(Tvar - Shape)



```
class Tvar(private val stvoceky: List<Stvorcek>) { // tvar je zoznam
    fun onDraw() { // vykresli tvar - vykresli každý štvorček
        for (stvorcek in stvoceky) stvorcek.onDraw()
    }
    fun onTouched(motionEvent: MotionEvent): Boolean {
        if (isIn(motionEvent)) { // bol tvar zasiahnutý eventom ?
            var reDraw = false // oznám všetkým prekresli sa
            for (stvorcek in stvoceky)
                reDraw = reDraw or stvorcek.onTouched(motionEvent)
            return reDraw // true, ak treba invalidate()
        } else
            return false
    }
    private fun isIn(motionEvent: MotionEvent): Boolean {
        var isIn = false // ak niektorý zo štvorčekov bol
        for (stvorcek in stvoceky) // zasiahnutý, tak Tvar bol tiež
            isIn = isIn or stvorcek.isIn(motionEvent)
        return isIn
    }
}
```

Hierarchia

(Stvorcek)



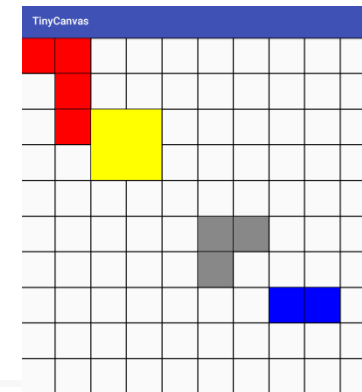
```
fun onDraw() {
    val r = Rect(x+1, y+1, x+sizeX-1, y+sizeY-1);
    CanvasView.c!!.drawRect(r, p);
}

fun onTouched(event: MotionEvent): Boolean {
    int action = event.getAction();
    if (action == MotionEvent.ACTION_DOWN) {
        ... START ...
    }
    else if (action == MotionEvent.ACTION_UP ||
        action == MotionEvent.ACTION_CANCEL) {
        ... END ...
    }
    else if (action == MotionEvent.ACTION_MOVE) {
        ... MOVE ...
    }
}

fun isIn(event: MotionEvent): Boolean {
    return x <= event.getX() && event.getX() <= x + sizeX
        &&
        y <= event.getY() && event.getY() <= y + sizeY;
}
```

Hierarchia

(top level Canvas)



```
override fun onTouch(view: View,
                      motionEvent: MotionEvent): Boolean {
    if (scena.onTouched(motionEvent));
    invalidate()
    return true
}

override fun onDraw(canvas: Canvas) {
    super.onDraw(canvas)
    c = canvas
    m?.onDraw()
    scena.onDraw()
}
```

Objektov/tried:

- Canvas
- Scena
- Tvar
- Stvorcek
- Mreza

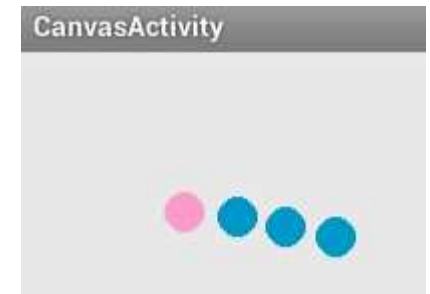
reagujú na:

- onTouch(event)
 - onDraw()
- príp.
- isIn(event)

Vláknó (Thread) vo View

(dynamická hra v canvase, simulácia cez thread)


```
class CanvasView(context: Context, attrs: AttributeSet) :  
    View(context), View.OnTouchListener, View.OnKeyListener {  
    var touchX = 100f; var touchY = 100f           // interface  
    var ballX = 200f;   var ballY = 200f  
  
    init {  
        setOnTouchListener(this) setOnKeyListener(this)  
        val th = object : Thread() {  
            override fun run() {                    // život vlákna  
                while (!stopped) {  
                    if (!paused) {                  // simulácia  
                        ballX += (touchX-ballX)/touches/50  
                        ballY += (touchY-ballY)/touches/50  
                        touchX = (ballX+50*touchX[i])/51  
                        touchY = (ballY+50*touchY[i])/51  
                        try {                          // pozdržanie  
                            Thread.sleep(100)  
                            postInvalidate()          // prekreslenie v GUI vlákne  
                        } catch (e: InterruptedException) {  
                            th.start()                // spustenie vlákna  
                        }  
                    }  
                }  
            }  
        }  
    }  
}
```





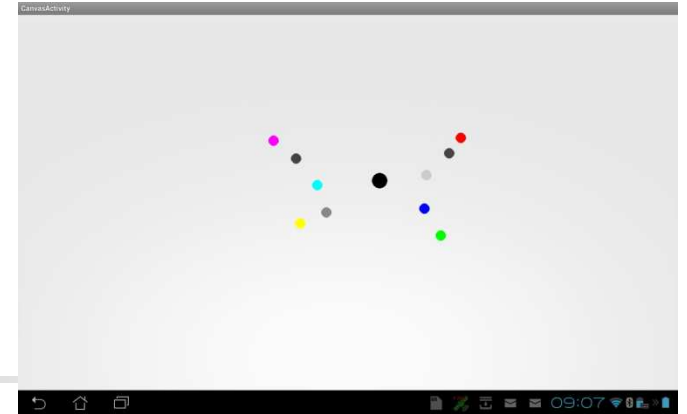
onDraw vo View

```
override protected fun onDraw(canvas: Canvas?) {  
    super.onDraw(canvas)  
  
    if (canvas != null) {  
        val p = Paint()  
  
        for (i in 0 until touches) {  
            p.setColor(colors[i])  
            // tu už canvas nemôže byť null  
            canvas!!.drawCircle(touchX[i], touchY[i], 10F, p)  
            canvas.drawCircle(touchX[i], touchY[i], 10F, p)  
        }  
        p.setColor(Color.BLACK)  
        // tu už canvas nemôže byť null  
        canvas!!.drawCircle(ballX, ballY, 15F, p)  
        canvas.drawCircle(ballX, ballY, 15F, p)  
    } else  
        Log.d("Canvas", "null")  
}
```



onDraw vo View

(MultiTouch)



MotionEvent poskytuje

- pointerCount – počet bodov reakcie z eventu
- [getX(i), getY(i)] – body reakcie
- typ reakcie (ACTION_DOWN,...)

```
override fun onTouch(v: View, event: MotionEvent): Boolean {  
    Log.d("Canvas", "counts:" + event.pointerCount)  
    val maskedAction = event.actionMasked  
    if (maskedAction == MotionEvent.ACTION_DOWN ||  
        maskedAction == MotionEvent.ACTION_POINTER_DOWN) {  
        touches = event.pointerCount  
        for (i in 0 until event.pointerCount) {  
            Log.d("Canvas", "X:" + event.getX(i))  
            Log.d("Canvas", "Y:" + event.getY(i))  
            touchX[i] = event.getX(i)  
            touchY[i] = event.getY(i)  
        }  
        return true // event handled  
    }  
}
```

Žiadne dva prsty
sa nedotknú naraz



onKey vo View

Vstup z klávesnice, ak by sme nejakú mali...

```
override fun onKeyDown(arg0: View, arg1: Int, arg2: KeyEvent):  
    Boolean {  
    val rnd = Random()  
    when (arg1) {  
        KeyEvent.KEYCODE_DPAD_LEFT -> ballX -= rnd.nextInt(50)  
        KeyEvent.KEYCODE_DPAD_RIGHT -> ballX += rnd.nextInt(50)  
        KeyEvent.KEYCODE_DPAD_UP -> ballY -= rnd.nextInt(50)  
        KeyEvent.KEYCODE_DPAD_DOWN -> ballY += rnd.nextInt(50)  
        KeyEvent.KEYCODE_SPACE -> {  
            ballX += rnd.nextInt(100) - 50  
            ballY += rnd.nextInt(100) - 50  
        }  
        else -> return false // event handled unhandled  
    }  
    invalidate()  
    return true // event handled  
}
```



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)
}
```

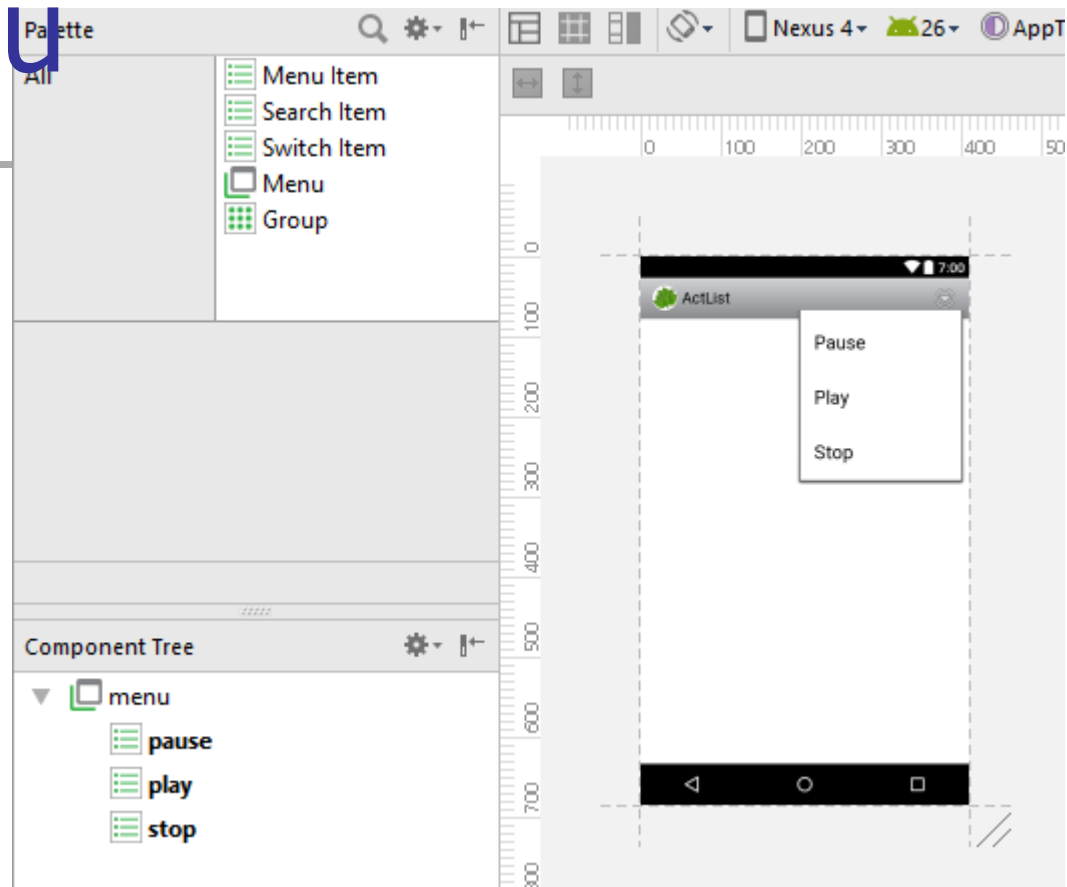



Option Menu

(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) {  
                ...  
            }  
        }  
    }  
}
```

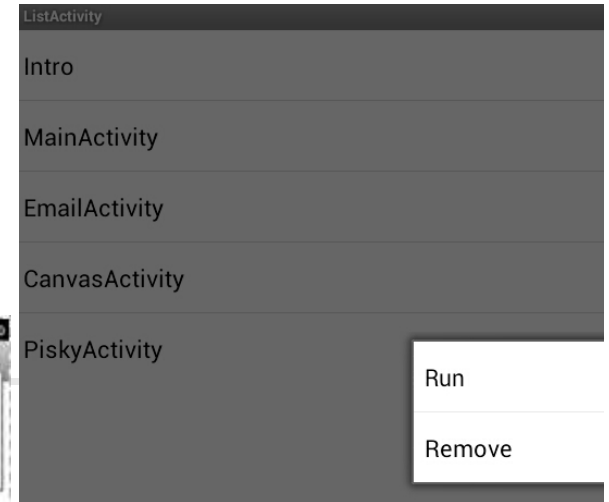
```
override fun onOptionsItemSelected(item: MenuItem): Boolean {  
    when (item.getItemId()) {  
        R.id.pause -> {  
            canvasView1.paused = true  
            return true  
        }  
        R.id.play -> {  
            canvasView1.paused = false  
            return true  
    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

Context Menu

```
Override fun onCreate(  
    savedInstanceState: Bundle?) { ...  
    registerForContextMenu(listView1); // na rozdiel od  
} 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)  
}
```

```
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  
        // viď 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 (i in gamePanel.pikaList.indices)  
                        gamePanel.pika[i].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

Frame per second

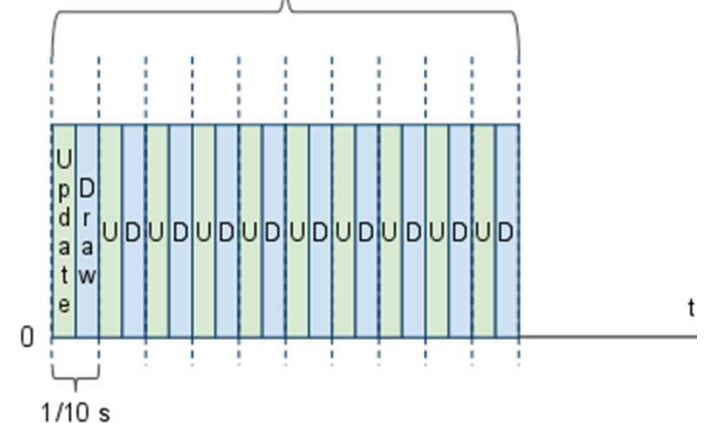
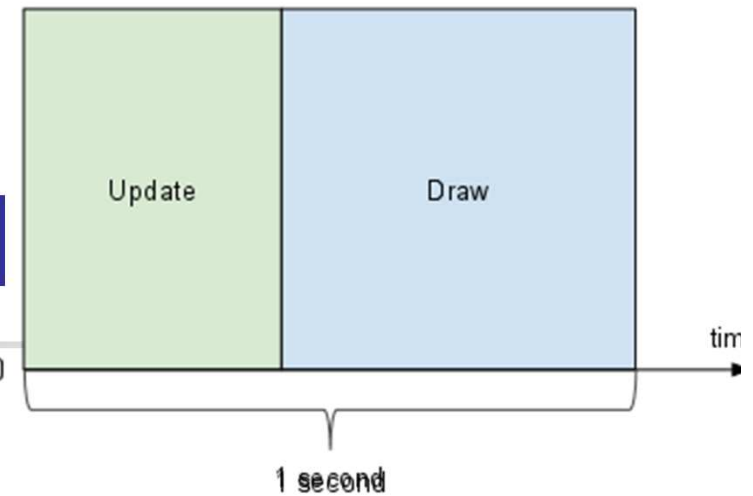
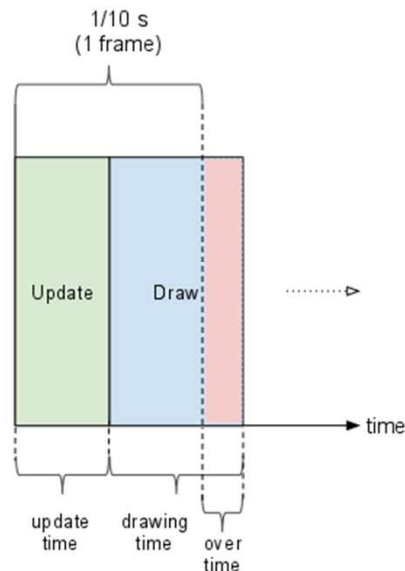
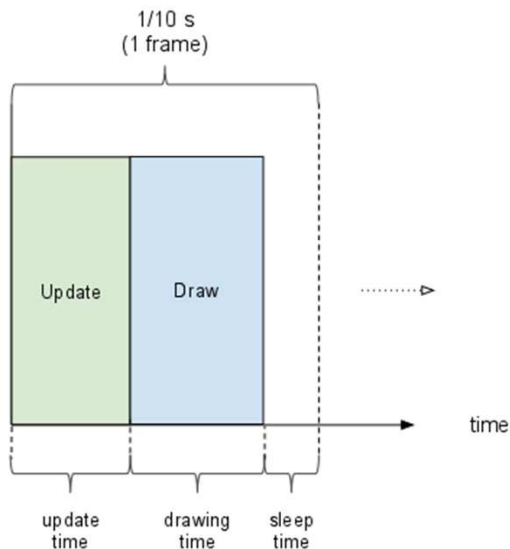
- 1 Frame per Second

Chceli by sme viac, napr. 10 fps

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

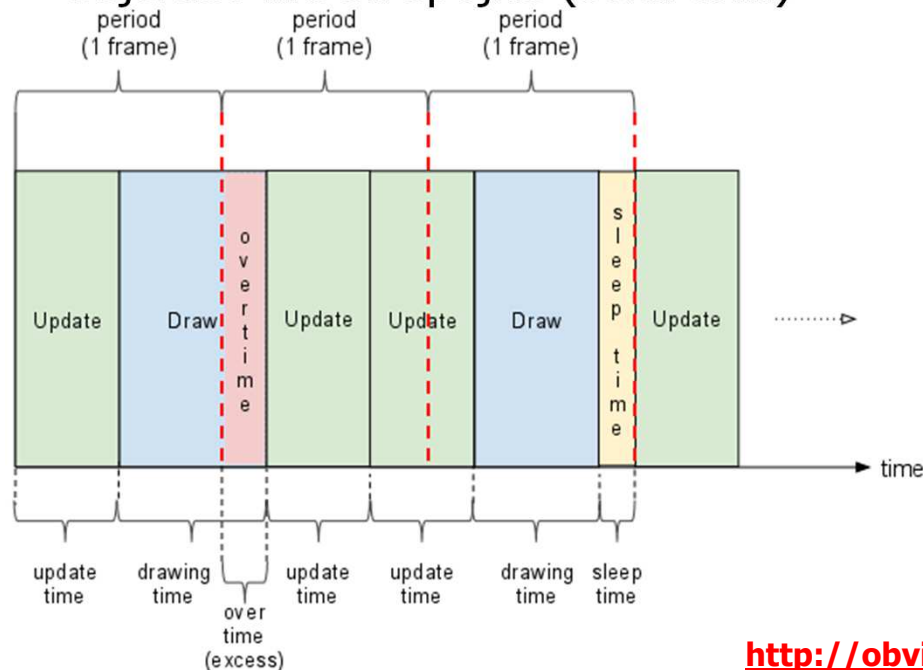
Môže sa nám stat', že to

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



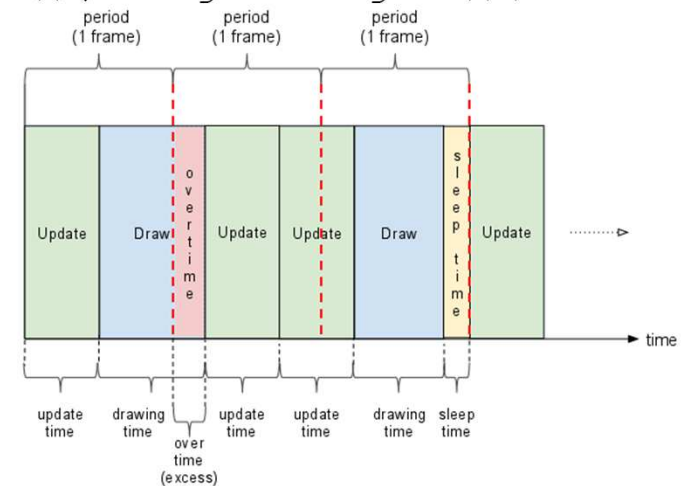
`FRAME_PERIOD = 1000/50; //50 fps`

Preskočíme pár vykreslování

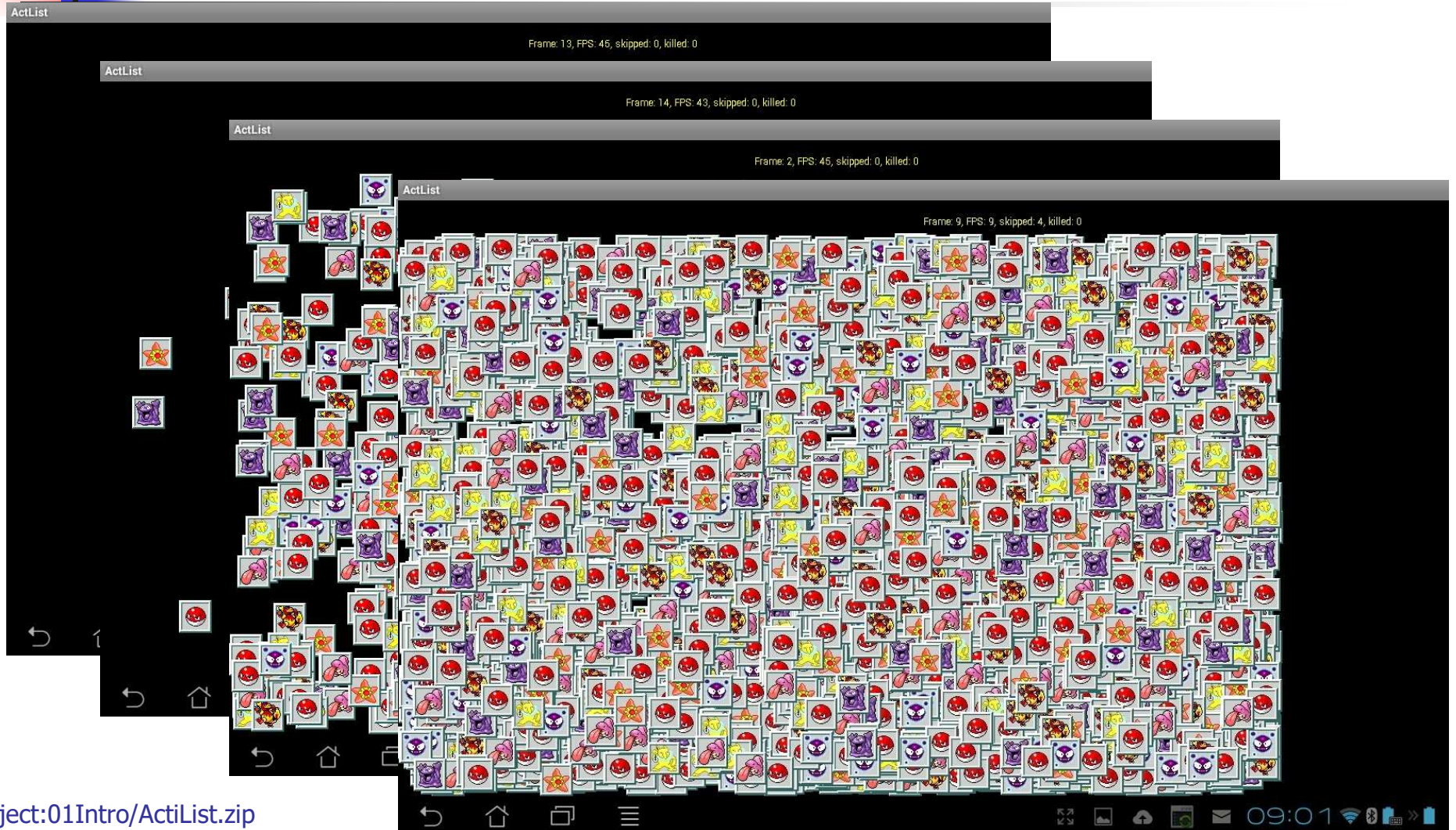
```
if (elapsedTime <= FRAME_PERIOD) { // lepší případ, stíhame
    try {                               // počkáme zvyšný čas
        Thread.sleep(FRAME_PERIOD - elapsedTime)
    } catch (InterruptedException e) {}
}

while (elapsedTime > FRAME_PERIOD) { // nestíhame
    for (int i = 0; i < r.pika.length; i++)
        r.pika[i].update(r.getWidth(), r.getHeight())
    elapsedTime -= FRAME_PERIOD
    skippedInPeriod++
}

framesInPeriod++
```



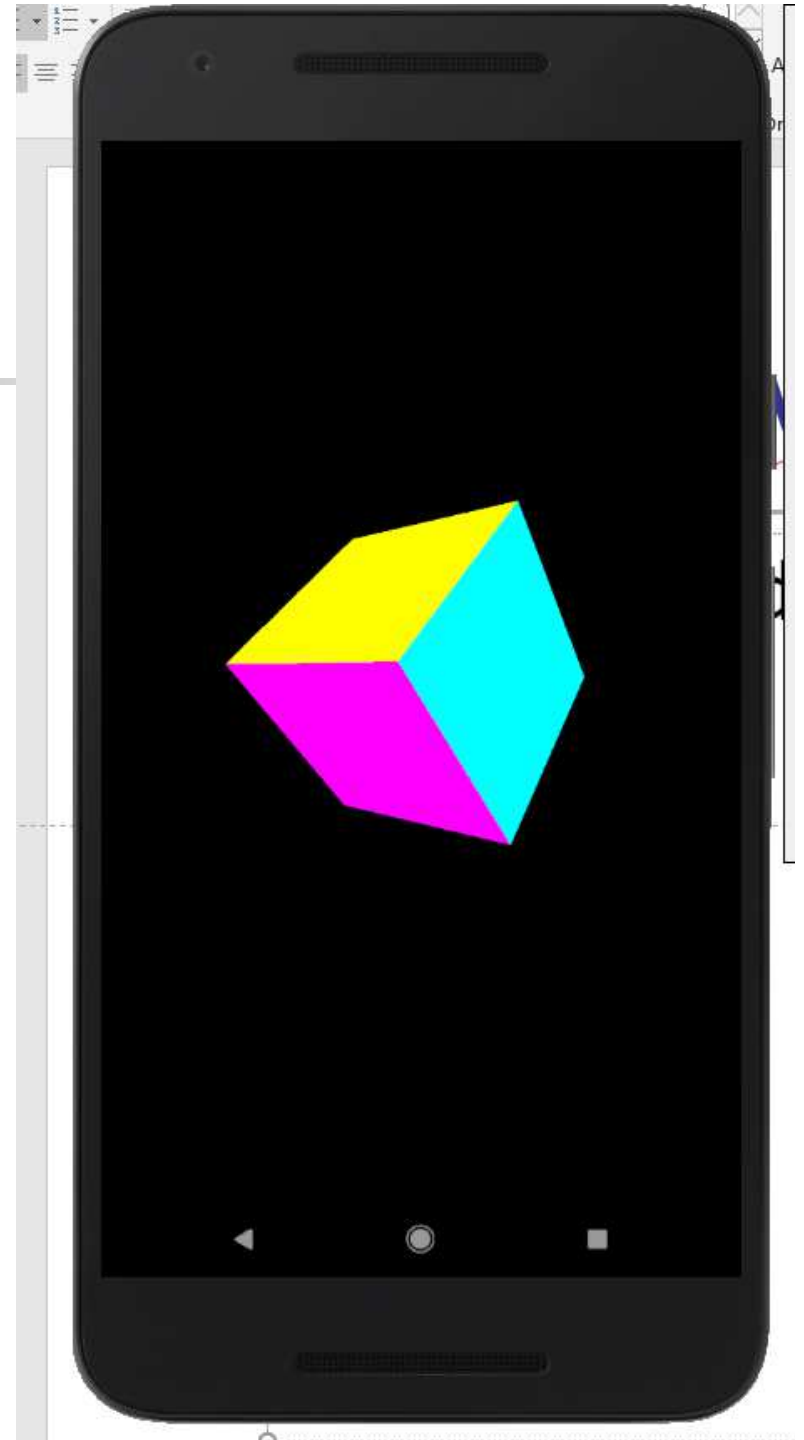
5, 50, 500, 5000 Pikachu





GLSurfaceView

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





Gestá

(štandardné)

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

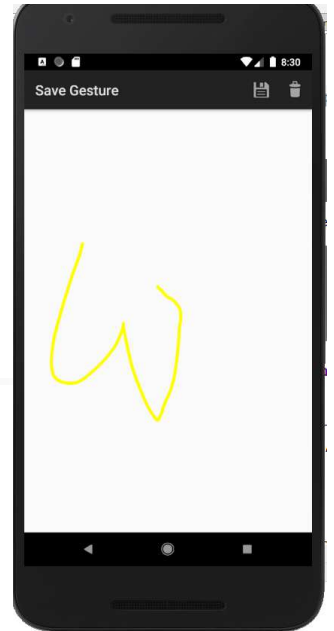
    GestureDetector.OnDoubleTapListener:
    override fun onDoubleTap(event: MotionEvent): Boolean
    override fun onDoubleTapEvent(event: MotionEvent): Boolean
    override fun onSingleTapConfirmed(event: MotionEvent): Boolean

    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
```

Gestá

(vlastné)

```
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 nejakو 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
```

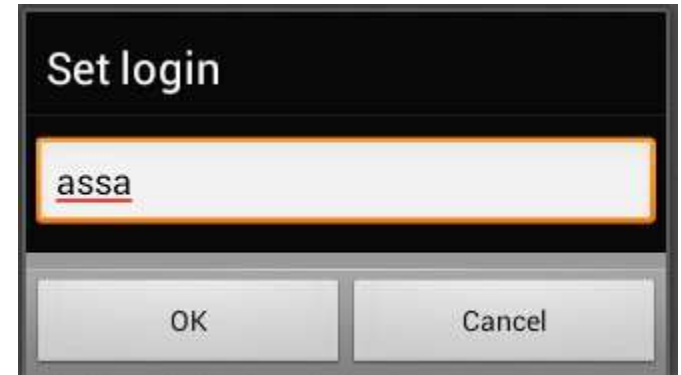
Načítanie:

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

Uloženie:

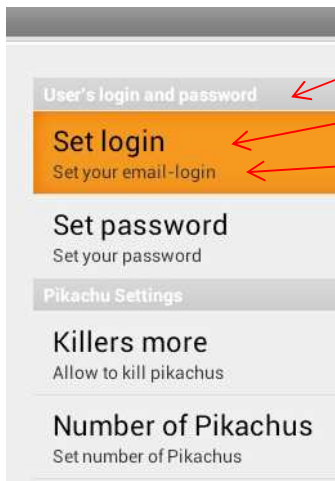
```
settings.edit() {  
    putString(LOGIN_ENTRY_KEY, "")  
    putString(PASSWORD_ENTRY_KEY, "")  
    remove(SUCCLOGS_ENTRY_KEY)  
    remove(FAILEDLOGS_ENTRY_KEY)  
}
```

PreferenceActivity



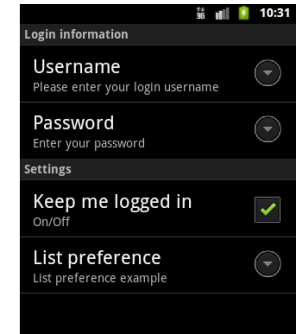
```
public class MyPreferenceActivity extends PreferenceActivity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        addPreferencesFromResource(R.xml.settings);  
    }  
}
```

```
<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:key="prefPass"/>  
</PreferenceCategory>
```



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"
```

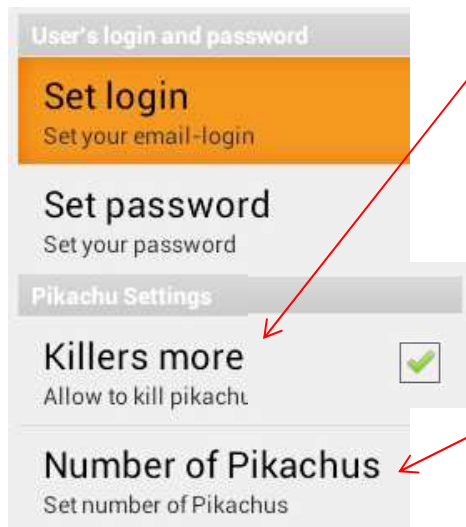
```
    android:entries="@array/pikaCount"
```

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

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

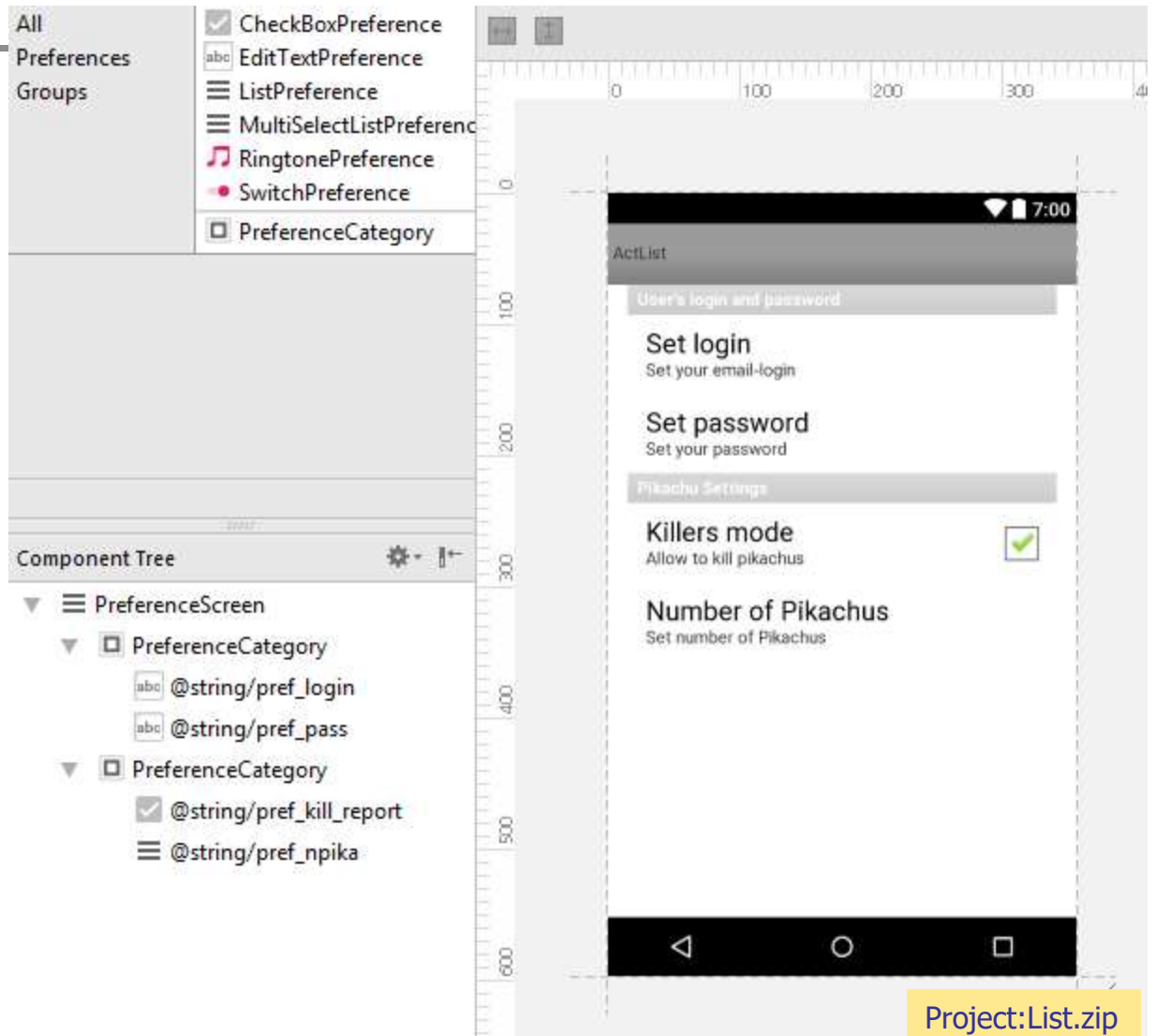
```
    android:title="Number of Pikachus" />
```

```
</PreferenceCategory>
```



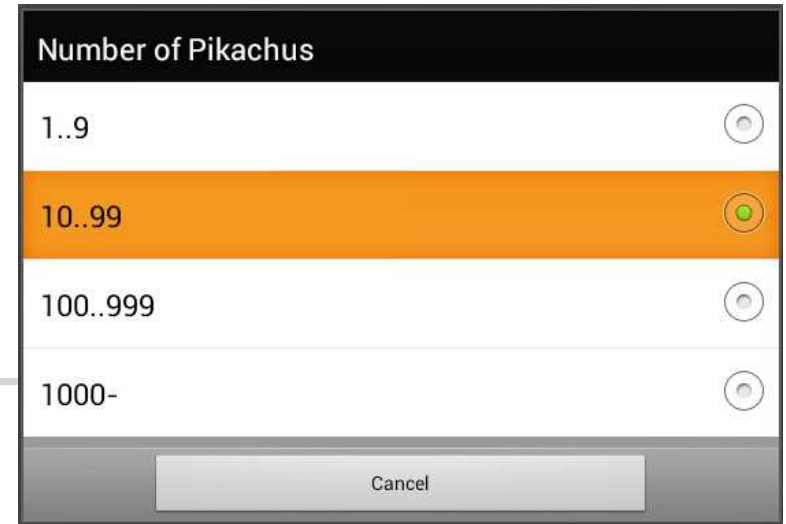
PreferenceCategories

(editor)



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

Cancel

Prémie miesto úloh

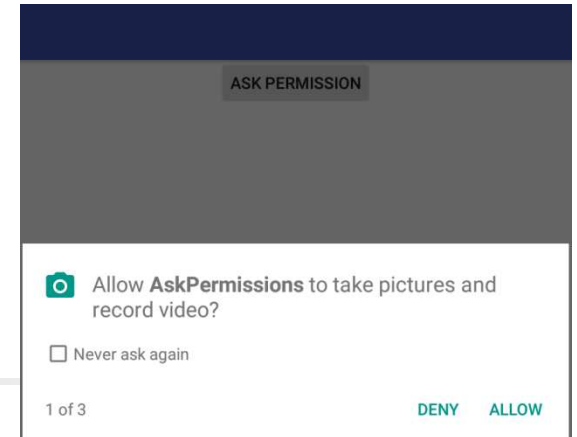
(a budú ďalšie)



- Dorobiť do hier PacMan či Tetris ovládanie pomocou akcelelačného a orientačného senzora zariadenia
- PacMan (2D = pohyby vľavo, vpravo, hore, dole) = 5 bodov (clonuj)
- Arcanoid = Naprogramujte dnes už hernú klasiku, hru Arkanoid, ešte z čias Commodore 64. Pre inšpiráciu, resp. ak netušíte o akú hru ide, sa inšpirujte z google play, akýkoľvek arkanoid... Podmienkou je vaša vlastná implementácia.
- Kalkulačka v 7mičkovej sústave (classic)
- Díško – tringelt advisor (clonuj)
- Maľovátko (clonuj existujúci projekt)
- InstantApp (absolute challenge)
- MasterDetail (fragmenty)



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

```
final int RUNTIME_PERMISSION_REQUEST_CODE = 999;
final String[] perms = new String[] {
    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);
}

public void onRequestPermissionsResult(int requestCode,
    String permissions[], int[] grantResults) {
    switch (requestCode) {
        case RUNTIME_PERMISSION_REQUEST_CODE: {
            for(int i = 0; i < grantResults.length; i++) {
                Log.d("Permissions", grantResults[i] ==
                    PackageManager.PERMISSION_GRANTED ? "GRANTED": "DENIED");
            }
        }
    }
}
```

