BUTE Department of Automation and Applied Informatics
AMORG
APPLIED MOBILE RESEARCH GROUP

Andriod alapú szoftverfejlesztés

10. Labor

Multimédia tartalom előállítása és kezelése

Tartalom

1 Felkészülés a mérésre 2
2 A szerver API bemutatása 2
3 Laborfeladatok 2
3.1 Felhasználói felület létrehozása 2
3.2 Kép feltöltése 6
3.3 Saját kamera felület létrehozása 10
3.4 Gombkezelés javítása 17
3.5 Hanglejátszás kép készítésekor 17
3.6 Képküldés fejlesztése 17
3.7 Kamera funkciók fejlesztése 18
3.8 Rajzolás továbbfejlesztése 18

1 Felkészülés a mérésre

A mérés célja, hogy bemutassa az Andriod multimédia szolgáltatásait, külön kiemelve a kamerakezelés módszereit. A mérés során egy kamerakezelő alkalmazást készítünk, melyben lehetőség lesz fénykép készítésére a külső kamera alkalmazásból, illetve beépített kamera felületen keresztül. Ezt követően a felhasználóknak lehetőségük lesz a képre rajzolni, majd pedig a képet feltölteni egy web-es galéria rendszerre.

A mérés az alábbi témákat érinti:
- Saját felületi elem készítése
- Beépített kamera alkalmazás használata
- Saját kamera felület megjelenítése
- Bináris adatok feltöltése szerverre

2 A szerver API bemutatása

A feladat megvalósításához szerver oldalon egy galéria alkalmazás áll rendelkezésre, mely az alábbi oldalon érhető el:
http://atleast.aut.bme.hu/AndroidGallery/

A galéria egy API-n keresztül lehetőséget biztosít arra, hogy új képeket tölthessünk fel POST üzenet formájában:
http://atleast.aut.bme.hu/AndroidGallery/api.php?action=uploadImage

3 Laborfeladatok

A mérés során egy kameraalkalmazást készítünk, ahol a lehetőség lesz kép készítésére a beépített kamera alkalmazás használatával, valamint saját felületről is. Sikeres képkészítés után a felhasználóknak lehetőségük lesz rárajzolni a képre, végül pedig a kép feltölthető egy galéria Web-es alkalmazásba.

3.1 Felhasználói felület létrehozása

Első lépésként tömörítsük ki a fejlesztés során használt képi erőforrásokat a .zip-ből és másoljuk az alkalmazás *res/drawable* könyvtárába.

Ezután vegyük fel a Manifest állományba a szükséges engedélyeket:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

Az alkalmazás során használt szöveges erőforrások, melyek a *res/values/strings.xml*-ben találhatók, a következők:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Kamera labor</string>
    <string name="action_settings">Settings</string>
    <string name="lblCamera">Kamerakép:</string>
    <string name="btnTakePicture">Photo</string>
    <string name="btnEffect">Negatív</string>
</resources>
```

Ezt követően hozzuk létre a felhasználói felület XML leírását, mely tartalmazni fog egy képet, amire lehet rajzolni, valamint három gombot a kamerakép készítésre, a feltöltésre és a rajzolás tisztítására.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:orientation="vertical" >

    <hu.bme.kameralabor.widget.DrawerImageView
        android:id="@+id/ivDrawer"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:scaleType="fitCenter"
        android:src="@drawable/noimage" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal" >

        <ImageButton
            android:id="@+id/imgBtnPhoto"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/ic_menu_camera" />

        <ImageButton
            android:id="@+id/imgBtnUpload"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/ic_menu_upload" />

        <ImageButton
            android:id="@+id/imgBtnDelete"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@android:drawable/ic_menu_delete" />

    </LinearLayout>
```

```
</LinearLayout>
```

Figyeljük meg, hogy az *ImageButton*-ok képi erőforrásaként hogy használtuk az Android beépített ikon készletét (pl.: *@android:drawable/ic_menu_camera*). Amennyiben ipari alkalmazást fejlesztünk, mindenképp saját ikonok használata javasolt.
Továbbá figyeljük meg, hogy az XML-ben egy saját *hu.bute.daai.amorg.examples.DrawerImageView* elem is található, mely egy általunk készített *ImageView* osztály leszármazott.
Készítsük el a *DrawerImageView* osztályt, melynek feladata képek megjelenítése és hogy legyen lehetőség a képek fölé rajzolni az érintés esemény érzékelésekor. A *DrawerImageView* osztály implementációja a következő:

```
public class DrawerImageView extends ImageView {

	private ArrayList<PointF> touchPoints = new ArrayList<PointF>();

	public DrawerImageView(Context context, AttributeSet attrs) {
		super(context,attrs);
		setDrawingCacheEnabled(true);
	}

	@Override
	protected void onDraw(Canvas canvas) {
		super.onDraw(canvas);
		Paint mPaint = new Paint(Paint.ANTI_ALIAS_FLAG);
		mPaint.setColor(0x44FF0000);
		for (int i=0; i<touchPoints.size(); i++)
		{
			canvas.drawCircle(touchPoints.get(i).x,
			    touchPoints.get(i).y, 5, mPaint);
		}
	}

	@Override
	public boolean onTouchEvent(MotionEvent event) {
		if (event.getAction() == MotionEvent.ACTION_DOWN)
		{
			touchPoints.add(new PointF(event.getX(), event.getY()));
		}
		this.invalidate();
		return super.onTouchEvent(event);
	}

	public void clearDrawing()
	{
		touchPoints.clear();
		this.invalidate();
	}
}
```

Az osztály tartalmaz egy *touchPoints* nevű *PointF* tömböt, melybe új *PointF* értékeket veszünk fel az érintés esemény hatására (*onTouchEvent(...)*) függvény. Továbbá az

osztályban felüldefiniáljuk az *onDraw(Canvas c)* függvényt, melyben az ősosztály rajzoló függvényének meghívása után köröket rajzolunk a megérintett pozíciókra.
Végül az osztály tartalmaz még egy *clearDrawing()* függvényt is, mellyel töröljük az eddig kirajzolt köröket.

Végül az *Activity* kiinduló kódja a következő:

```
public class ActivityLabor9 extends Activity
{    public static final String APPTAG = "ANDLAB9";
     public static final String IMAGEPATH =
            Environment.getExternalStorageDirectory().getAbsolutePath() +
              "/tmp_image.jpg";
     public static final String MYCAMRESULT = "MYCAM";

     private final int REQUEST_CAMERA_IMAGE = 101;
     private final int REQUEST_MY_CAMERA_IMAGE = 102;
     private DrawerImageView ivDrawer;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        ivDrawer = (DrawerImageView)findViewById(R.id.ivDrawer);

        final ImageButton imgBtnDelete =
          (ImageButton)findViewById(R.id.imgBtnDelete);
        imgBtnDelete.setOnClickListener(new OnClickListener() {
                  @Override
                  public void onClick(View v) {
                         ivDrawer.clearDrawing();
                  }
           });

        final ImageButton imgBtnPhoto =
          (ImageButton)findViewById(R.id.imgBtnPhoto);
        imgBtnPhoto.setOnClickListener(new OnClickListener() {
                  @Override
                  public void onClick(View v) {
                    File imageFile = new File(IMAGEPATH);
                    Uri imageFileUri = Uri.fromFile(imageFile);
                    Intent cameraIntent =
                      new Intent(
                        android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
                    cameraIntent.putExtra(
                      android.provider.MediaStore.EXTRA_OUTPUT,
                      imageFileUri);
                    startActivityForResult(cameraIntent,
                      REQUEST_CAMERA_IMAGE);
                  }
        });
    }

     // Visszatérés a kamerától
    @Override
     protected void onActivityResult(int requestCode,
        int resultCode, Intent data) {
```

```
        if (requestCode == REQUEST_CAMERA_IMAGE) {
                if (resultCode == RESULT_OK)
                {
                        try {
                                File imageFile = new File(IMAGEPATH);
                        FileInputStream fis =
                            new FileInputStream(imageFile);
                                Bitmap img =
                                   BitmapFactory.decodeStream(fis);
                                ivDrawer.setImageBitmap(img);
                        } catch(Exception e) {}
                }
        }
    }
}
```

Az *Activity* az elején tartalmaz néhány konstans, melyekre a későbbiekben még szükségünk lesz. Ezt követően az *onCreate(...)* függvényben megvalósítjuk a képtisztítás és a fotózás gomb eseménykezelőit. A fotózás gomb eseménykezelője meghívja a beépített kameraalkalmazást és képkészítés után a felüldefiniált *onActivityResult(...)* függvényben megkapjuk a képet, amelyet megjelenítünk a saját *DrawerImageView*-nkon.

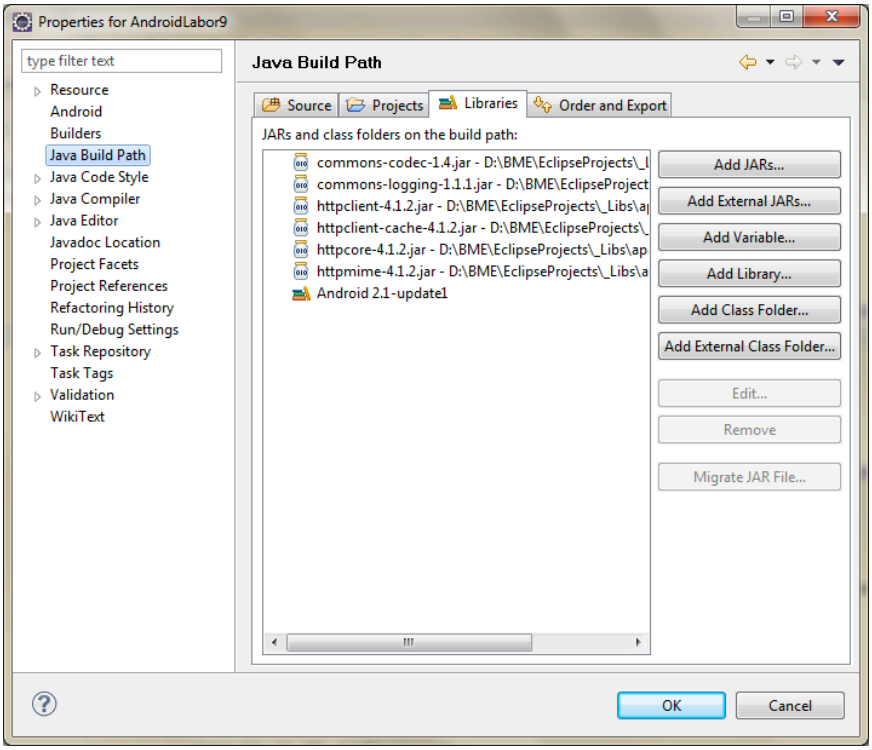
Activity felület

3.2 Kép feltöltése

Következő lépésként valósítsuk meg a képfeltöltés funkciót. Képek feltöltéséhez a szerver oldal POST-ként várja a képet, ahol az állomány nevét „img”-ként kell megadni: http://atleast.aut.bme.hu/AndroidGallery/api.php?action=uploadImage

A feltöltéshez a Apache *HttpPost* osztályát fogjuk alkalmazni a *HttpGet*-hez hasonlóan. Azonban az Android nem tartalmazza a legújabb Apache könyvtárat, ezért azt ki kell csomagolnunk a .zip állományban lévő "libs" mappából és be kell állítanunk a projektnek külső könyvtárként.

Ezt követően állítsuk be a külső könyvtárakat (Project Properties->Java Build Path->Libararies->Add External JARs):



Külső osztálykönyvtárak beállítása

Miután ezzel megvagyunk, ismét az *AsyncTask* módszert fogjuk használni a képfeltöltés megvalósítására egy saját *AsyncTask* osztállyal:

```
public class AsyncTaskUploadImage extends AsyncTask<String, Void, String> {
    public interface UploadCompleteListener {
            public void onTaskComplete(String aResult);
            public void onError(String aError);
    }

    private Context context = null;
    private ProgressDialog progressDialog = null;
    private byte[] imageToUpload;
    private UploadCompleteListener listener;
    private String error = null;

    public AsyncTaskUploadImage(Context context,
      byte[] aImageToUpload, UploadCompleteListener aListener) {
        this.context = context;
```

```
        imageToUpload = aImageToUpload;
        listener = aListener;
    }

    @Override
    protected void onPreExecute() {
           progressDialog = new ProgressDialog(this.context);
        progressDialog.setMessage("Kérem várjon...");
        progressDialog.show();
    }

    @Override
    protected String doInBackground(String... params) {
           String result = "";
        HttpClient httpclient = new DefaultHttpClient();
        InputStream is = null;
        HttpPost httpPost = new HttpPost(params[0]);
        MultipartEntity reqEntity =
          new MultipartEntity(HttpMultipartMode.BROWSER_COMPATIBLE);
        ByteArrayBody bab = new ByteArrayBody(imageToUpload, "tmp.jpg");
         reqEntity.addPart("img", bab);
         httpPost.setEntity(reqEntity);
        HttpResponse response;
        try {
            response = httpclient.execute(httpPost);
            if (response.getStatusLine().getStatusCode() ==
              HttpStatus.SC_OK)
            {
                   HttpEntity entity = response.getEntity();
                   if (entity != null) {

                         BufferedReader reader = new BufferedReader(
                           new InputStreamReader(entity.getContent(),
                           "UTF-8"));
                         StringBuilder sb = new StringBuilder();
                         String line;
                         while ((line = reader.readLine()) != null) {
                                sb.append(line);
                         }

                         result = sb.toString();
                   }
                   else
                         error = "HttpEntity is empty";
            }
        } catch (Exception e) {
           error = "Error: "+e.getMessage();
        } finally {
           if (is != null)
           {
                  try {
                         is.close();
                         } catch (IOException e) {
                                e.printStackTrace();
                         }
           }
        }
        return result;
    }

    @Override
```

```
    protected void onPostExecute(String result) {
        progressDialog.dismiss();
        if (error != null) {
           listener.onError(error);
        }
        else {
           listener.onTaskComplete(result);
        }
    }
}
```

Az *AsyncTask* osztály a konstruktorában veszi át a feltöltendő képet. Figyeljük meg a kódban, hogy hogy van összeállítva a *HttpPost* üzenet kép feltöltéshez egy *MultipartEntity* megadásával.

Implementáljuk az *Activity*-ben az *UploadCompleteListener*-t:

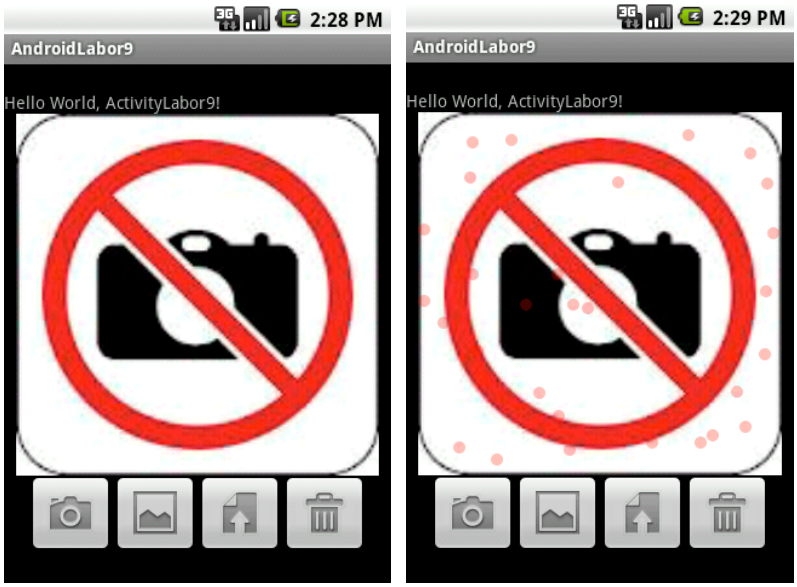
```
@Override
public void onTaskComplete(String aResult) {
    Toast.makeText(this, aResult, Toast.LENGTH_LONG).show();
}

@Override
public void onError(String aError) {
    Toast.makeText(this, aError, Toast.LENGTH_LONG).show();
}
```

Adjuk meg az *onCreate(...)* függvényben a képfeltöltés gomb eseménykezelőjét:

```
final ImageButton imgBtnUpload =
  (ImageButton)findViewById(R.id.imgBtnUpload);
imgBtnUpload.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
           ivDrawer.invalidate();
           ivDrawer.refreshDrawableState();
           Bitmap bm = ivDrawer.getDrawingCache();
           ByteArrayOutputStream baos = new ByteArrayOutputStream();
           bm.compress(Bitmap.CompressFormat.JPEG, 70, baos);
           byte[] b = baos.toByteArray();
           AsyncTaskUploadImage taskUpload =
             new AsyncTaskUploadImage(ActivityLabor9.this,
             b, ActivityLabor9.this);
           taskUpload.execute("http://atleast.aut.bme.hu/AndroidGallery/"+
             "api.php?action=uploadImage");
    }
});
```

Figyeljük meg, hogy kép feltöltésekor hogyan kérjük le a *DrawerImageView*-től a cache által tárolt képet. Ez a *getDrawingCache()* függvény csak akkor működik, ha az *ImageView setDrawingCacheEnabled(true)* függvényét meghívtuk.



Kép feltöltési és rajzolási felület

3.3 Saját kamera felület létrehozása

Következő feladatunk egy saját kamera felület létrehozása. Ehhez elsőként egy saját Preview osztályt kell létrehoznunk, amely a SurfaceView-ból származik le és implementálja a SurfaceHolder.Callback interface-t.

```
public class MyPreview extends SurfaceView implements
SurfaceHolder.Callback {

    // Log tag
    public static final String TAG = MyPreview.class.getSimpleName();

    // State
    private SurfaceHolder mHolder;
    private Camera mCamera;

    public MyPreview(Context context, AttributeSet attrs, int defStyle) {
           super(context, attrs, defStyle);
           init();
    }

    public MyPreview(Context context, AttributeSet attrs) {
           super(context, attrs);
           init();
    }

    public MyPreview(Context context) {
           super(context);
           init();
    }

    private void init() {
           mHolder = getHolder();
```

```
            mHolder.addCallback(this);
      }

      public void setCamera(Camera camera) {
            mCamera = camera;

            try {
                  mCamera.setPreviewDisplay(mHolder);
                  mCamera.startPreview();
            } catch (Exception e) {
                  e.printStackTrace();
                  Log.d(TAG, "Failed to start camera preview!");
            }
      }

      @Override
      public void surfaceCreated(SurfaceHolder holder) {
            // Not used
      }

      @Override
      public void surfaceDestroyed(SurfaceHolder holder) {
            // Not used
      }

      @Override
      public void surfaceChanged(SurfaceHolder holder, int format, int w, int
h) {

            if (mHolder.getSurface() == null || mCamera == null) {
                  return;
            }

            try {
                  mCamera.stopPreview();
            } catch (Exception e) {
                  e.printStackTrace();
                  Log.d(TAG,   "Tried   to   stop   a   non-existing   camera
preview!");
            }

            try {
                  mCamera.setPreviewDisplay(mHolder);
                  mCamera.startPreview();
            } catch (Exception e) {
                  e.printStackTrace();
                  Log.d(TAG, "Failed to restart camera preview!");
            }
      }
}
```

A saját kamerakép megjelenítésért egy új *Activity* lesz felelős, amely sikeres fényképezés után egy ideiglenes file-ba elmenti a képet és visszalép a fő *Activity*-hez. Figyeljük meg, hogy az *onResume()* callback metódusban elkérjük, az *onPause()* callback metódusban elengedjük a kamera erőforrást. A kamera elkérést a rendszertől egy külön szálon, *AsyncTask* segítségével oldjuk meg, mivel ez egy hosszan tartó, blokkoló műveletnek számít.

```
public class CameraActivity extends Activity {

    // State
    private Camera camera;

    // UI
    private ViewGroup previewLayout;

    @Override
    public void onCreate(Bundle savedInstanceState) {
           super.onCreate(savedInstanceState);
           setContentView(R.layout.activity_camera);

           Button           btnTakePicture           =           (Button)
findViewById(R.id.btnTakePicture);
           btnTakePicture.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                         // get an image from the camera
                         camera.takePicture(null, null, mPicture);
                  }
           });

           ToggleButton         btnEffect         =         (ToggleButton)
findViewById(R.id.btnEffect);
           btnEffect.setOnCheckedChangeListener(new
OnCheckedChangeListener() {
                  @Override
                  public void onCheckedChanged(CompoundButton buttonView,
                                boolean isChecked) {
                         Camera.Parameters           parameters           =
camera.getParameters();

                         if (isChecked) {

    parameters.setColorEffect(Parameters.EFFECT_NEGATIVE);
                         } else {

    parameters.setColorEffect(Parameters.EFFECT_NONE);
                         }

                         camera.setParameters(parameters);
                  }
           });
           btnEffect.setOnClickListener(new View.OnClickListener() {
                  @Override
                  public void onClick(View v) {
                         Camera.Parameters           parameters           =
camera.getParameters();

    parameters.setColorEffect(Parameters.EFFECT_NEGATIVE);
                         camera.setParameters(parameters);
                  }
           });

           previewLayout = (ViewGroup) findViewById(R.id.camera_preview);
    }

    @Override
```

```
    protected void onResume() {
        super.onResume();

        CameraTask task = new CameraTask();
        task.execute();
    }

    @Override
    protected void onPause() {
        if (camera != null) {
            camera.stopPreview();
            camera.release();
        }

        super.onPause();
    }

    private PictureCallback mPicture = new PictureCallback() {
        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            String result = "";
            try {
                File tmpImage = new File(MainActivity.IMAGEPATH);
                if (tmpImage.exists())
                    tmpImage.delete();
                tmpImage.createNewFile();
                FileOutputStream buf = new
FileOutputStream(tmpImage);
                buf.write(data);
                buf.flush();
                buf.close();
                result = "OK";
            } catch (Exception e) {
                result = "ERROR: " + e.getMessage();
            }

            Intent resultIntent = new Intent();
            resultIntent.putExtra(MainActivity.MYCAMRESULT, result);
            setResult(MainActivity.RESULT_OK, resultIntent);
            finish();
        }
    };

    private class CameraTask extends AsyncTask<Void, Void, Camera> {

        private static final int MAX_RETRY_COUNT = 3;

        @Override
        protected Camera doInBackground(Void... params) {
            Camera result = null;
            int trialCount = 0;

            while (result == null && trialCount < MAX_RETRY_COUNT) {
                result = Camera.open(0);
                trialCount++;
            }

            return result;
        }

        @Override
```

```
            protected void onPostExecute(Camera result) {
                  super.onPostExecute(result);

                  if (result != null) {
                        camera = result;

                        // Lekerdezzuk a lehetseges preview mereteket
                        List<Size>          supportedPreviewSizes          =
camera.getParameters()
                                    .getSupportedPreviewSizes();

                        //   Kiszorjuk   azokat,   amik   tul   nagyok   a
rendelkezesre allo
                        // merethez kepest
                        int availableWidth = previewLayout.getWidth();
                        int availableHeight = previewLayout.getHeight();

                        Iterator<Size>                iter                =
supportedPreviewSizes.iterator();
                        while (iter.hasNext()) {
                              Size current = iter.next();
                              if (current.width > availableWidth
                                          ||       current.height       >
availableHeight) {
                                    iter.remove();
                              }
                        }

                        // A maradeknak vesszuk a maximumat
                        Size selectedPreviewSize = Collections.max(
                                    supportedPreviewSizes,             new
Comparator<Camera.Size>() {
                                          @Override
                                          public  int  compare(Size  lhs,
Size rhs) {
                                                return    (lhs.width    *
lhs.height) < (rhs.width * rhs.height) ? -1
                                                            : 1;
                                          }
                                    });

                        //  Letrehozzuk  a  Preview  View-t  a  kivalasztott
meretekkel es
                        // hozzaadjuk a layout-hoz
                        previewLayout.removeAllViews();

                        MyPreview          myPreview          =          new
MyPreview(CameraActivity.this);
                        RelativeLayout.LayoutParams      lp      =      new
RelativeLayout.LayoutParams(
                                    selectedPreviewSize.width,
selectedPreviewSize.height);
                        myPreview.setLayoutParams(lp);

                        previewLayout.addView(myPreview);

                        // Bellitjuk a kameranak is a kivalasztott preview
felbontast
                        // majd odaadjuk a Preview nezetunknek
                        camera.getParameters().setPreviewSize(
```

```
                                        selectedPreviewSize.width,
selectedPreviewSize.height);
                        myPreview.setCamera(camera);
                    } else {
                        Toast.makeText(CameraActivity.this,
                                        "Nem      sikerult      a      kamera
inicializalasa!",
                                        Toast.LENGTH_SHORT).show();
                    }
                }

        }
}
```

A saját kamera felület XML felülete (*layoutcamera.xml*):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:orientation="vertical" >

    <RelativeLayout
        android:id="@+id/camera_preview"
        android:layout_width="match_parent"
        android:layout_height="0dip"
        android:layout_weight="1"
        android:gravity="center" />

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:orientation="horizontal" >

        <Button
            android:id="@+id/btnTakePicture"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/btnTakePicture" />

        <ToggleButton
            android:id="@+id/btnEffect"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textOn="@string/btnEffect"
            android:textOff="@string/btnEffect" />
    </LinearLayout>

</LinearLayout>
```

Ezt követően egészítsük ki az eredeti *Activity* felhasználói felületét (*main.xml*) egy új kép készítés gombbal:

```
<ImageButton
     android:id="@+id/imgBtnOwnPhoto"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:src="@android:drawable/ic_menu_gallery"
/>
```

Az eredeti *Activity onCreate(...)*-jébe pedig vegyük fel az új eseményt, amely elindítja a saját kamera *Activity*-nket:

```
final ImageButton imgBtnOwnPhoto =
  (ImageButton)findViewById(R.id.imgBtnOwnPhoto);
imgBtnOwnPhoto.setOnClickListener(new OnClickListener() {
     @Override
     public void onClick(View v) {
            Intent i = new Intent(
              getApplicationContext(), ActivityMyCamera.class);
            startActivityForResult(i, REQUEST_MY_CAMERA_IMAGE);
     }
});
```

Ezen felül az eredeti *Activity onActivityResult()* függvényét egészítsük ki, hogy kezelje a saját *Activity*-ből való visszatérést:

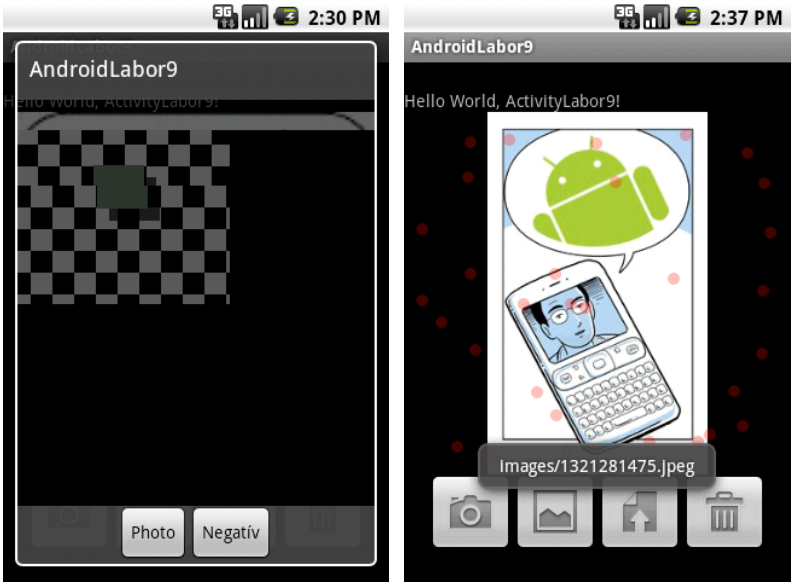
```
else if (requestCode == REQUEST_MY_CAMERA_IMAGE) {
     if (resultCode == RESULT_OK)
     {
            String camResult = data.getExtras().getString(MYCAMRESULT);

            if (camResult.startsWith("OK"))
            {
                  try {
                        File imageFile = new File(IMAGEPATH);
                        FileInputStream fis =
                            new FileInputStream(imageFile);
                        Bitmap img = BitmapFactory.decodeStream(fis);
                        // TODO a DrawableCache nem frissül így még
                        ivDrawer.setImageDrawable(new
                              BitmapDrawable(getResources(), img));
                        ivDrawer.invalidate();
                        ivDrawer.refreshDrawableState();
                  } catch(Exception e) {}
            }
            else
            {
                  Toast.makeText(this,
                    camResult, Toast.LENGTH_LONG).show();
            }
     }
}
```

Vegyük fel az új *Activity*-t a *Manifest* állományba és állítsuk be, hogy az kizárólag *landscape* nézetben jelenjen meg. Ezt könnyítésképp állítjuk be, mivel így nem kell

foglalkoznunk azzal, hogy megfelelően elforgatva jelenítsük meg a kamerából érkező képet a felületen.

```
<activity
    android:name=".CameraActivity"
    android:screenOrientation="landscape"
    android:label="@string/app_name" >
</activity>
```



Saját kamera felület

3.4 Gombkezelés javítása

Valósítsa meg, hogy a feltöltés gomb csak akkor működjön, ha van már kamerával rögzített kép.

3.5 Hanglejátszás kép készítésekor

A saját kamerakép nézetet egészítse ki úgy, hogy a rendszer játssza le az alapértelmezett *Notification* hangot kép készítésekor (*ShutterCallback*, előadás anyaga).

3.6 Képküldés fejlesztése

Valósítsa meg, hogy legyen lehetőség külön a tiszta kamerakép feltöltésére az átrajzolt kamerakép feltöltése mellett.

3.7 Kamera funkciók fejlesztése

Biztosítson extra funkciókat a saját kamera nézeten, például tegye lehetővé, hogy több szín effekt közül is választhassanak a felhasználók, illetve bármikor vissza tudjanak állni az alap beállításokra.

3.8 Rajzolás továbbfejlesztése

Bővítse az alkalmazás képre rajzolás funkcióit tetszőlegesen. Például:

- Vonalak rajzolás a lenyomás és a felengedés helye között.
- Színválasztás
- Alakzat kiválasztása
- Stb.