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

Andriod alapú szoftverfejlesztés

9. Labor

Hálózati kapcsolatok kezelése Android platformon

Tartalom

1 Felkészülés a mérésre

A mérés célja a hálózati kommunikáció, azon belül is főként a http kommunikáció alapjainak bemutatása. A mérés során egy multiplayer labirintus játékhoz fogunk mobil klienst fejleszteni. A kliens segítségével irányíthatjuk a labirintusban egy bábut, továbbá lehetőség lesz üzenetek küldésére is.

A mérés az alábbi témákat érinti:
- Hálózati kommunikáció
- Aszinkron hívások
- Animációk

2 Aszinkron hívások Android platformon

Android platformon a hálózati kommunikáció tipikusan új szálon történik, hogy a felhasználói felületet ne blokkoljuk. A hálózatról érkező választ azonban általában a felhasználói felületen jelenítjük meg valamilyen módon, de a platform nem engedi, hogy más szálból a UI-t módosítsuk. Az alkalmazás indításakor a rendszer létrehoz egy úgynevezett main szálat (UI szál) és csak ebben történhet a felület frissítése.
A platform által biztosított eszközök ezen helyzetek kezelésére:
- *Activity.runOnUiThread(Runnable)*
- *View.post(Runnable)*
- *View.postDelayed(Runnable, long)*
- *Handler*
- *AsyncTask* (http://developer.android.com/reference/android/os/AsyncTask.html)

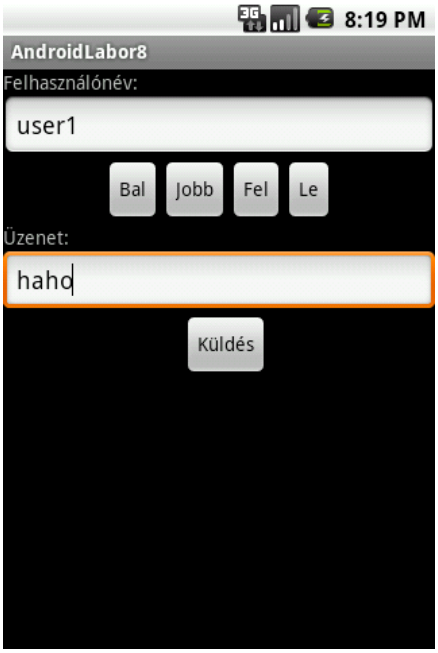
3 Laborfeladatok

A következőkben egy olyan Android alkalmazást készítünk, mely egy tulajdonképpen egy kliensalkalmazás egy multiplayer labirintus játékhoz. A labirintust egy JavaFX alkalmazás jeleníti meg, amelyet a projektorról láthatunk kivetítve. A játék szabályai egyszerűek, a játékosunkat a készülékről négy gomb segítségével (bal, jobb, fel, le) irányíthatjuk, továbbá lehetőség van még üzenetküldésre. Az első lépésünk során kerül rá az új játékos a játéktérre egy véletlen pozícióra. Ha egy játékos a másikra lép, akkor pontot kap!
A labor során használt szerver oldali megoldás és a JavaFX megjelenítő felület jelenleg prototípus, ezért előre is elnézést kérünk az esetleges kellemetlenségekért.
A feladatok megoldása során a hálózati kommunikációra és az aszinkron hívásokra több módszert is mutatunk.
A feladat megoldásához szükséges nagyobb kódrészek egy *patch.txt*-ből másolhatók, melyet a laborvezető ad meg.

A játéktér JavaFX felülete

3.1 A felhasználói felület elkészítése

Első lépésként készítsük el az alkalmazás felhasználói felületét XML erőforrásból. A felületen helyezzünk el két *EditText*-et, egyet a felhasználónév bekéréséhez, egyet pedig üzenetküldéshez. Emellett legyen összesen 5 gomb, négy gomb az irányításhoz, egy pedig az üzenetküldéshez.
Legegyszerűbb esetben például a következő módon rendezhetjük el ezeket a vezérlőket:



Felhasználói felület

Az ehhez megfelelő XML állomány a következő:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/bgLayout"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/usernamehead"
         />
     <EditText
            android:id="@+id/editTextUserName"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"/>

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

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

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

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

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

     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/messagehead"
         />
     <EditText
            android:id="@+id/editTextMessage"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"/>
     <LinearLayout
```

```
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center">
           <Button
                  android:text="@string/send"
                  android:id="@+id/buttonSendMessage"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"/>
    </LinearLayout>
</LinearLayout>
```

A felület tartalmaz több szöveges konstanst is, ezért töltsük fel a *values* könyvtárban lévő *strings.xml* állományunkat a következő értékekkel:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">AndroidLabor8</string>
    <string name="usernamehead">Felhasználónév:</string>
    <string name="messagehead">Üzenet:</string>
    <string name="left">Bal</string>
    <string name="right">Jobb</string>
    <string name="up">Fel</string>
    <string name="down">Le</string>
    <string name="send">Küldés</string>
    <string name="empty_user">Üres felhasználónév!</string>
    <string name="empty_user_or_message">
        Üres felhasználónév vagy jelszó!</string>
</resources>
```

Töröljük a res/menu könyvtárat és tartalmát, mivel az alkalmazásban nem fogunk menüt használni. Illetve az *Activity*-ben a menü felfújás kódrészt is töröljük.

Végül az *Activity*-ben kérjük el a referenciát a vezérlőkre, mivel ezekre a későbbiekben szükség lesz.

```
public class ActivityAndroidLabor8 extends Activity
{
     private LinearLayout bgLayout;
     private EditText etUserName;
     private Button btnLeft;
     private Button btnRight;
     private Button btnUp;
     private Button btnDown;

     private EditText etMessage;
     private Button btnSendMessage;

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

```
        bgLayout = (LinearLayout)findViewById(R.id.bgLayout);

        etUserName = (EditText)findViewById(R.id.editTextUserName);
        btnLeft = (Button)findViewById(R.id.buttonLeft);
        btnRight = (Button)findViewById(R.id.buttonRight);
        btnUp = (Button)findViewById(R.id.buttonUp);
        btnDown = (Button)findViewById(R.id.buttonDown);

        etMessage = (EditText)findViewById(R.id.editTextMessage);
        btnSendMessage = (Button)findViewById(R.id.buttonSendMessage);
    }
}
```

3.2 A szerver oldali API bemutatása

A szerver egy PHP alapú oldal, amely HTTP GET kérésekben várja a lépéseket és az üzeneteket. Ezeket eltárolja egy adatbázisban, amelyet egy PHP oldalon tesz elérhetővé a megjelenítésért felelős JavaFX alkalmazás számára. A Java FX alkalmazás ezt a PHP oldalt pollozza relatív kis időközönként és kapott válaszok alapján frissíti a felhasználói felületét. A szerver alap címe az alábbi oldalon érhető el:
avalon.aut.bme.hu/~tyrael/labyrinthwar/

Ezen belül kell majd a megfelelő PHP állományokat meghívni az előre definiált GET paraméterekkel.

A PHPk-től hiba esetén mindig „*ERROR*”-al kezdődő üzenetet kapunk.

3.2.1 Játékos mozgatása

A játékos mozgatásához a moveuser.php-t kell meghívni, amely két paramétert vár:

- *username*: felhasználónév (ne felejtsük URL encode-olni!)
- *step*: lépés típusa (1: bal, 2: jobb, 3: fel, 4: le)

Például:
http://avalon.aut.bme.hu/~tyrael/labyrinthwar/moveuser.php?username=user1&step=1

3.2.2 Üzenet feltöltése

Üzenet feltöltéséhez a writemessage.php-t kell hívni,amely szintén két paramétert vár:

- *username*: felhasználónév (ne felejtsük URL encode-olni!)
- *message*: üzenet (ne felejtsük URL encode-olni!)

Például:
http://avalon.aut.bme.hu/~tyrael/labyrinthwar/writemessage.php?username=user1&message=uzenet

3.3 Kommunikáció a szerver oldallal

Következő feladatunk a szerver oldali kommunikációt biztosító osztály megvalósítása, mely végrehajtja a HTTP GET hívásokat és egy interface-n keresztül jelez a hívó fél számára.

3.3.1 Általános HTTP kezelő készítése

Hozzunk létre egy általános célú, HTTP GET kommunikációra használható osztályt (lehet *AsyncTask* is, de későbbi órákon elő fog még fordulni; *HttpClient* helyett lehet *HttpUrlConnection*-t is használni, ami Android 2.3 felett jobban támogatott):

```
public class HttpManager {

    public interface HttpManagerListener {
           public void responseArrived(String aResponse);
           public void errorOccured(String aError);
    }

    public static final int EMESSAGE_LEFT = 1;
    public static final int EMESSAGE_RIGHT = 2;
    public static final int EMESSAGE_UP = 3;
    public static final int EMESSAGE_DOWN = 4;

    private HttpManagerListener listener;

    public HttpManager(HttpManagerListener aListener)
    {
           listener = aListener;
    }

    public void execute(String url)
    {
        HttpClient httpclient = new DefaultHttpClient();
        InputStream is = null;
        HttpGet httpget = new HttpGet(url);
        HttpResponse response;
        try {
            response = httpclient.execute(httpget);
            if (response.getStatusLine().getStatusCode() ==
                HttpStatus.SC_OK)
            {
                   HttpEntity entity = response.getEntity();
                   if (entity != null) {
                       is = entity.getContent();
                       ByteArrayOutputStream bos =
                           new ByteArrayOutputStream();
                               int inChar;
                               while ((inChar = is.read()) != -1)
                               {
                                      bos.write(inChar);
                               }

                               listener.responseArrived(bos.toString());
                   }
                   else
                         listener.errorOccured("HttpEntity is empty");
```

```
            }
        } catch (Exception e) {
           listener.errorOccured(e.getMessage());
        } finally {
           if (is != null)
           {
                  try {
                         is.close();
                  } catch (IOException e) {
                         e.printStackTrace();
                  }
           }
        }
    }
}
```

Az osztály elején egy *interface* található, amelyet majd az *Activity*-ben implementálunk és azon keresztül fogunk értesülni a hálózati kérés eredményéről.
Ezt követően négy konstans található, melyekkel majd a lépés irányát fogjuk meghatározni. Az osztály fő részét az *execute(...)* függvény jelenti, amely a paraméterül kapott URL-t meghívja és a választ a *listener*-en keresztül jelzi.

3.3.2 A HTTP kezelő használata az Activity-ben

Az osztály használatához az *Activity*-ben vegyünk fel egy változót és egy konstanst. Előbbi a *HttpManager* osztály példánya lesz, utóbbi pedig a szerver oldal alapcíme:

```
private HttpManager httpManager = new HttpManager(this);
private final String BASE_URL =
    "http://avalon.aut.bme.hu/~tyrael/labyrinthwar/";
```

Ezt követően implementáljuk a ***HttpManagerListener***-t az *Activity*-ben, majd valósítsuk meg a szükséges függvényeket, illetve készítsünk még egy *showMessage(...)* függvényt *Toast* üzenet megjelenítéshez:

```
public void responseArrived(final String aMessage)
{
     runOnUiThread(new Runnable() {
            @Override
            public void run() {
            showMessage(aMessage);
            }
     });
}

public void errorOccured(final String aError)
{
     runOnUiThread(new Runnable() {
            @Override
            public void run() {
```

```
            showMessage(aError);
            }
      });
}

public void showMessage(String aMessage)
{
      Toast.makeText(this, aMessage, Toast.LENGTH_SHORT).show();
}
```

Vegyük észre, hogy a válasz feldolgozása a *runOnUiThread(...)* hívással már a UI szálon fog megtörténni, a kommunikációs szál helyett.
Következő lépésként készítsünk egy függvényt, mely összeállítja a játékos mozgatásához tartozó HTTP kérést:

```
public void handleMoveMessage(int aStepCode)
{
      if (etUserName.getText().toString().equals(""))
      {
            showMessage(getString(R.string.empty_user));
            return;
      }

      final StringBuilder sb = new StringBuilder(BASE_URL);
      sb.append("moveuser.php?username=");
      sb.append(URLEncoder.encode(etUserName.getText().toString()));
      sb.append("&step=");
      sb.append(aStepCode);

      new Thread() {
            public void run()
            {
                  httpManager.execute(sb.toString());
            }
      }.start();
}
```

Az URL összeállításakor jól látható, hogy a felhasználónév megadásához URL encode-olást alkalmazunk, továbbá vegyük észre, hogy a *handleMoveMessage(...)* függvényben a *HttpManager execute(...)* függvénye új szálban van meghívva, így a hívás nem fogja blokkolni a felhasználói felületet.
Végül a mozgatásért felelős gombok eseménykezelőit valósítsuk meg az *Activity onCreate(...)* függvényében a gombok referenciájának elkérése után:

```
btnLeft.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
            if (etUserName.isEnabled())
                  etUserName.setEnabled(false);
            handleMoveMessage(HttpManager.EMESSAGE_LEFT);
      }
});
btnRight.setOnClickListener(new OnClickListener() {
```

```
	@Override
	public void onClick(View v) {
		if (etUserName.isEnabled())
			etUserName.setEnabled(false);
		handleMoveMessage(HttpManager.EMESSAGE_RIGHT);
	}
});
btnUp.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		if (etUserName.isEnabled())
			etUserName.setEnabled(false);
		handleMoveMessage(HttpManager.EMESSAGE_UP);
	}
});
btnDown.setOnClickListener(new OnClickListener() {
	@Override
	public void onClick(View v) {
		if (etUserName.isEnabled())
			etUserName.setEnabled(false);
		handleMoveMessage(HttpManager.EMESSAGE_DOWN);
	}
});
```

A gombok eseménykezelőjének elején az egyszerűség kedvéért a felhasználónév bevitelére szolgáló mezőt letiltjuk, hogy első lépés után ne tudja változtatni a játékos a nevét.
Próbáljuk ki, hogy a játékosunkkal tudunk-e lépkedni a labirintuson!

3.4 Üzenetek küldése

A játékos lépésének kezelése után következő feladatunk, hogy a játék során üzeneteket is tudjunk feltölteni a szerverre a 3.2.2-ben megadott módon.
Ehhez hozzunk létre egy függvényt az *Activity*-ben a *handleMoveMessage(...)* függvényhez hasonlóan:

```
public void handleWriteCommentMessage()
{
	if (etUserName.getText().toString().equals("") ||
	   etMessage.getText().equals(""))
	{
		showMessage(getString(R.string.empty_user_or_message));
		return;
	}

	final StringBuilder sb = new StringBuilder(BASE_URL);
	sb.append("writemessage.php?username=");
	sb.append(URLEncoder.encode(etUserName.getText().toString()));
	sb.append("&message=");
	sb.append(URLEncoder.encode(etMessage.getText().toString()));

	new Thread() {
		public void run()
		{
```

```
                httpManager.execute(sb.toString());
            }
        }.start();
    }
```

Továbbá az *Activity onCreate(...)* függvényében állítsuk be az üzenetküldés gomb eseménykezelőjét:

```
btnSendMessage.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
        handleWriteCommentMessage();
    }
});
```

3.5 Animációk és hibás lépés kezelése

A következőkben az alkalmazás felhasználói élményét növeljük kicsit egyszerű animációkkal.

3.5.1 Hibás lépés kezelése

Az alkalmazás élvezetesebbé tételéhez valósítsuk meg, hogy ha hibás lépést hajt végre a játékos, vagy falra próbál lépni, akkor egy rövid időre változzon a háttér pirosra.
Ehhez a *responseArrived(...)* függvényt írjuk át úgy, hogy ha a válasz „ERROR”-al kezdődik, akkor legyen piros a háttér és egyben indítsunk is el egy *TimerTask*-ot, amely rövid idő múlva visszaállítja azt feketére:

```
public void responseArrived(final String aMessage)
{
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            if (aMessage.startsWith("ERROR"))
            {
            ((LinearLayout)findViewById(R.id.bgLayout)).
                setBackgroundColor(Color.RED);
            TimerTaskResetBackground ttbg =
                new TimerTaskResetBackground();
            timer.schedule(ttbg, 300);
            }

            showMessage(aMessage);
            }
    });
}
```

A *TimerTask* használatához szükség lesz egy *timer* objektumra, melyet az *Activity* elején vegyünk fel:

```
private Timer timer = new Timer();
```

A visszaállításért felelős *TimerTaskResetBackground* implementációja, melyet az egyszerűség kedvéért az *Activity*-ben is felvehetünk belső osztályként:

```
private class TimerTaskResetBackground extends TimerTask
{
     @Override
     public void run() {
            runOnUiThread(new Runnable() {
                  @Override
                  public void run() {
                        bgLayout.setBackgroundColor(Color.BLACK);
                  }
            });
     }
}
```

3.5.2 Lépés gomb animáció

Valósítsuk meg, hogy a bal/jobb/fel/le gombokat megnyomva a gomb besüllyedjen és kiemelkedjen rövid idő alatt egy animáció segítségével.
Az animációt vegyük fel erőforrásként az *anim* könyvtárba *pushanim.xml* néven:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <scale
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromXScale="1.0"
        android:toXScale="0.7"
        android:fromYScale="1.0"
        android:toYScale="0.7"
        android:pivotX="50%"
        android:pivotY="50%"
        android:duration="100" />

    <scale
        android:interpolator="@android:anim/accelerate_interpolator"
        android:fromXScale="0.7"
        android:toXScale="1.0"
        android:fromYScale="0.7"
        android:toYScale="1.0"
        android:pivotX="50%"
        android:pivotY="50%"
        android:startOffset="100"
        android:duration="100" />
</set>
```

Állítsuk be a négy gomb eseménykezelőjében, hogy játsszák le ezt az animációt:

```
Animation showAnim = AnimationUtils.loadAnimation(
     ActivityAndroidLabor8.this,
     R.anim.pushanim);
// btnLeft helyett a megfelelő gombok referenciája kell a többi esetben
btnLeft.startAnimation(showAnim);
```

3.5.3 Üzenetküldés animáció

A lépés gombokhoz hasonlóan valósítsuk meg, hogy üzenetküldéskor a szövegmező halványodjon el, majd erősödjön vissza egy rövid idő alatt.
Az animációt vegyük fel erőforrásként az *anim* könyvtárba *fadeanim.xml* néven:

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
    <alpha
           android:fromAlpha="1.0"
           android:toAlpha="0.5"
           android:duration="600"/>
     <alpha
           android:fromAlpha="0.5"
           android:toAlpha="1.0"
           android:startOffset="600"
           android:duration="600"/>
</set>
```

Az üzenet küldése gomb eseménykezelőjében pedig játsszuk le az animációt:

```
Animation showAnim = AnimationUtils.loadAnimation(
     ActivityAndroidLabor8.this,
     R.anim.fadeanim);
etMessage.startAnimation(showAnim);
```

Végül próbáljuk ki az alkalmazást működés közben:

A játéktér

3.6 Válaszidő kijelzése

Egészítsük ki az alkalmazást úgy, hogy a felhasználói felületen megjelenítsük a szerverrel való kommunikáció során tapasztalt átlagos válaszidőt (üzenet küldése és válasz megérkezése közti idő).

3.7 WiFi állapot kijelzése - Bonus

Egészítsük ki az alkalmazást úgy, hogy a *WiFi* állapotát és a hálózat nevét megjelenítsük a felhasználói felületen. A hálózat nevét a *WifiManager.EXTRA_BSSID*-al tudjuk lekérdezni az előadáson ismertetett példa megfelelő kiegészítésével.
Segítség:

```
WifiManager wifiManager = (WifiManager) getSystemService(WIFI_SERVICE);
WifiInfo wifiInfo = wifiManager.getConnectionInfo();
Log.d("wifiInfo", wifiInfo.toString());
Log.d("SSID",wifiInfo.getSSID());
```

Manifest engedély

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
```