Igra kamen, škare papir za više igrača za Android uređaje

Čalušić, Darko

Undergraduate thesis / Završni rad

2016

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:139646

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-19**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
ELEKTROTEHNIČKI FAKULTET

Sveučilišni preddiplomski studij računarstva

IGRA KAMEN, ŠKARE PAPIR ZA ANDROID UREĐAJE

Završni rad

Darko Čalušić

Osijek, 2016.

ETFOS
ELEKTROTEHNIČKI FAKULTET OSIJEK
Sveučilište Josipa Jurja Strossmayera u Osijeku

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom studiju

Osijek, 2016.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada

Ime i prezime studenta:	Darko Čalušić
Studij, smjer:	Preddiplomski studij, računarstvo
Mat. br. studenta, godina upisa:	R-3549, 2013.
Mentor:	doc. dr. sc. Krešimir Nenadić
Sumentor:	
Naslov završnog rada:	Igra kamen, škare papir za Android uređaje
Primarna znanstvena grana rada:	Programiranje
Sekundarna znanstvena grana (ili polje) rada:	
Predložena ocjena završnog rada:	
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: Postignuti rezultati u odnosu na složenost zadatka: Jasnoća pismenog izražavanja: Razina samostalnosti:

Potpis sumentora:	Potpis mentora:

Dostaviti:

1. Studentska služba

Potpis predsjednika Odbora:

_____

Dostaviti:

1. Studentska služba

Sveučilište Josipa Jurja Strossmayera u Osijeku

ELEKTROTEHNIČKI FAKULTET OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 2016.

Ime i prezime studenta:	Darko Čalušić
Studij :	Preddiplomski studij
Mat. br. studenta, godina upisa:	R-3549, 2013.

Ovom izjavom izjavljujem da je rad pod nazivom**:**

Igra kamen, škare papir za Android uređaje

izrađen pod vodstvom mentora doc. dr. sc. Krešimira Nenadića

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1
1.1. Zadatak završnog rada 1
2. TEORIJSKI DIO 2
2.1. Razvojno okruženje Android Studio 2
2.2. Bluetooth komunikacija 3
3. IGRA KAMEN, ŠKARE PAPIR ZA ANDROID UREĐAJE 5
3.1. Ideja i osnovni koncept 5
3.2. Izrada aplikacije 6
3.3. Testiranje aplikacije 19
3.3.1. Testiranje Single-Player izbora 19
3.3.2. Testiranje Multi-Player Izbora 20
4. ZAKLJUČAK 22
LITERATURA 23
SAŽETAK 25
ABSTRACT 26
ŽIVOTOPIS 27
PRILOZI 28

1. UVOD

U današnje vrijeme mobilne aplikacije se neprestano razvijaju. Igre postaju sve veća svakodnevnica gotovo svake osobe koja koja posjeduje mobilni uređaj, međutim većina njih se igra na mreži, dok nije svatko u mogućnosti imati stalan pristup internetu. Igra Škare-Kamen-Papir je jednostavna igra koja se igra u dva ili više igrača. Pravila igre su jednostavna, škare pobjeđuju papir a gube od kamena, kamen gubi od lista a pobjeđuje škare.

Zbog malog broja aplikacija zasnovanih na bluetooth komunikaciji, igra škare-kamen-papir posjeduje tu komunikaciju. Za ovu komunikaciju su potrebna minimalno 2 uređaja. Bluetooth komunikacija se odvija pomoću tri glavne faze u kojoj razmjenjuju podatke. Prva faza je pretraga dostupnih uređaja. Druga faza je uparivanje bluetooth uređaja, točnije verificiranje da je uređaj na drugoj strani pouzdan uređaj te omogućavanje lakše daljnje komuniciranje. Zadnja faza predstavlja prijenos informacija. Aplikacija je rađena u razvojnom okruženju Android Studio.

Ovaj završni rad se sastoji od teorijskog dijela u kojemu je opisano razvojno okruženje Android Studio kao i komunikacija koja je korištena. Nakon teorijskog dijela, slijedi kronološki postupak izrade aplikacije i testiranje.

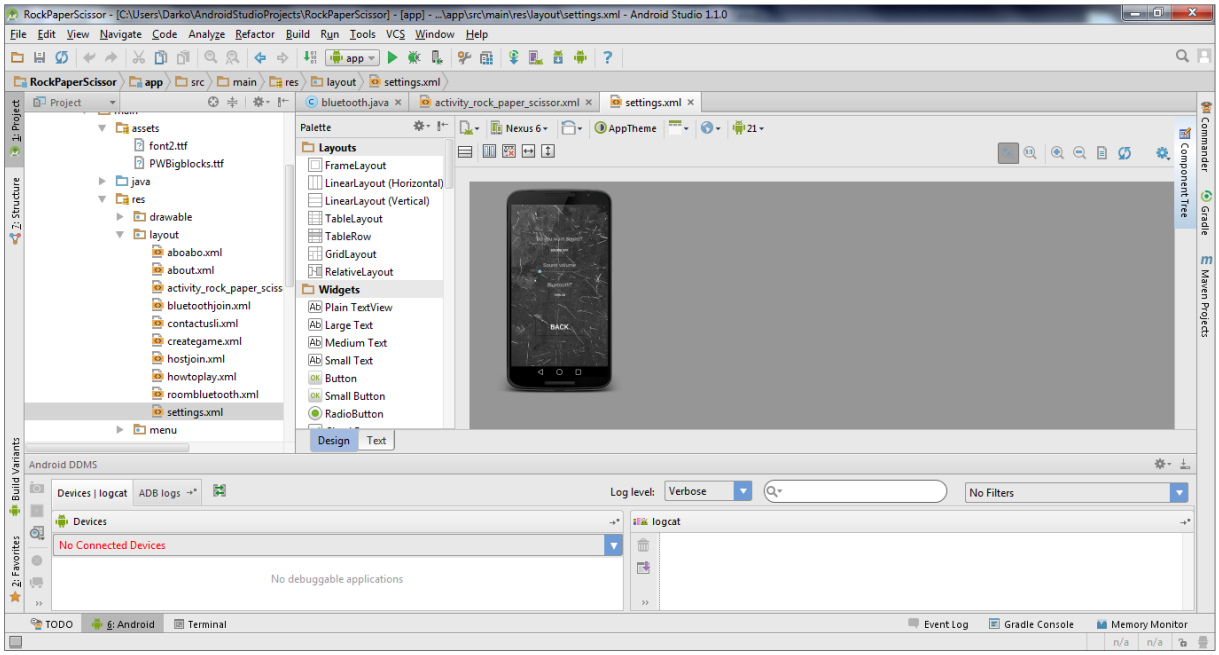
1.1. Zadatak završnog rada

Pravljenje jednostavne igre, prikazivanje Bluetooth komunikacije kao i opisivanje svih pojedinačnih koraka i omogućavanje implementiranja dijela koda u druge aplikacije.

2. TEORIJSKI DIO

2.1. Razvojno okruženje Android Studio

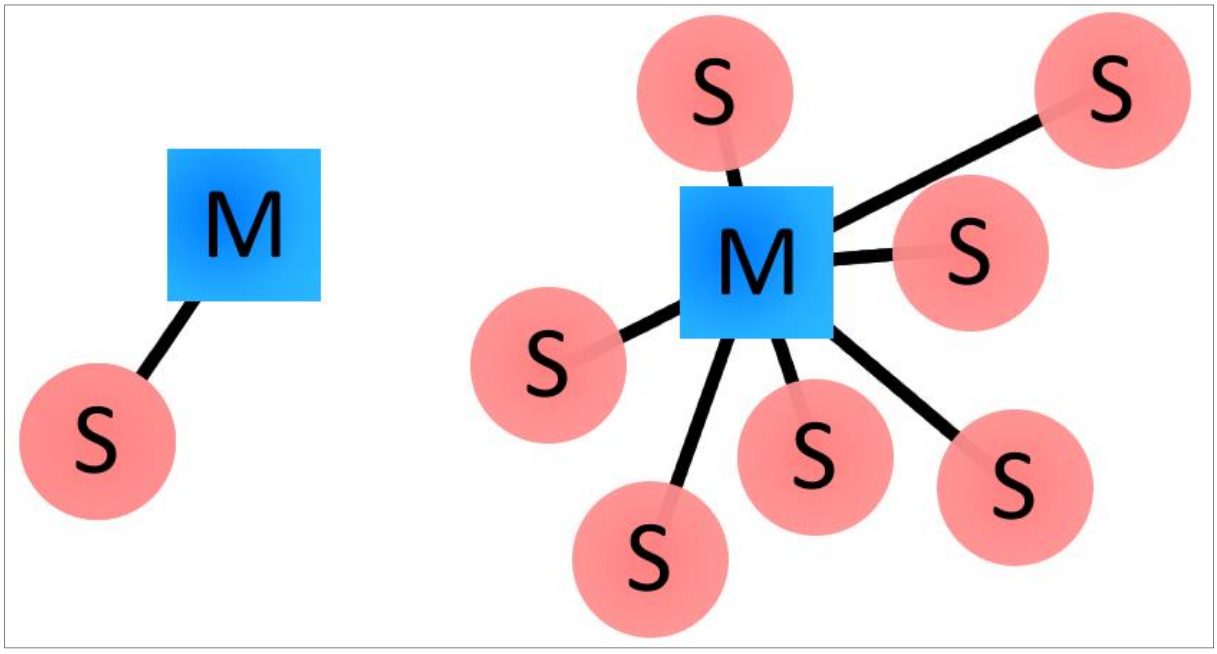
Android Studio je integralno razvojno okruženje za izradu Android aplikacija (engl. *Integrated Development Environment* – IDE). Prva inačica ovoga programa izdana u svibnju 2013. godine s inačicom 0.1., a prva stabilna u prosincu 2014. godine s inačicom 1.0. Zadnja inačica ovoga programa je izdana u inačici 2.1.3 u kolovozu 2016. godine. Inačica programa korištena za ovu aplikaciju je 2.1.3. Android Studio se bazira na JetBrains' IntelliJ Idea softveru i dostupan je za preuzimanje na Windows, Mac OS X i Linux platformama. Za pisanje aplikacija potrebno je poznavati programski jezik Java kao i XML (e*Xtensible Markup Language*) koji služi za dizajniranje izgleda aplikacije.



Sl. 2.1. *Razvojno okruženje Android Studio*

2.2. Bluetooth komunikacija

Bluetooth je serijski način bežične razmjene podataka između dva ili više uređaja. Većina današnjih modernih računala, mobitela, digitalnih kamera i audio uređaja imaju mogućnost slanja podataka pomoću bluetootha. Veza se uspostavlja putem radio valova u frekvencijskom području od 2,4 do 2,48 GHz. Zbog korištenja radio veze uređaji koji se povezuju ne moraju biti u optičkoj vidljivosti niti međusobno usmjereni a veza se može ostvariti u promjeru od otprilike 10 metara oko uređaja. Bluetooth protokol je siguran protokol, idealan za kratke udaljenosti i malu potrošnju wireless načina slanja podataka između uređaja. Bluetooth mreže se još nazivaju i pikoneti (engl. *Piconets*). Koriste poslužitelj-klijent protokol u kojima poslužitelj uređaj kontrolira komunikaciju svih drugih uređaja. Na jedan poslužitelj uređaj može biti povezano maksimalno 7 uređaja.



Sl. 2.2. *Poslužitelj-Klijent način povezivanja*

Svaki bluetooth uređaj ima jedinstvenu 48-bitnu adresu (Skraćenica BD_ADDR) koji su prezentirani pomoću 12 digitalnih hexadekadskih vrijednosti. Prva 24 bita su najznačajnija i oni prikazuju jedinstveni identifikacijski broj (engl. *Organization unique identifier* - OUI) koje definira proizvođač. Sljedeća 24 bita daju detaljniju adresu uređaja.

Komunikacijski proces se odvija u 3 faze:

1. Pretraga (engl. Inquiry) – Ukoliko uređaji ne posjeduju nikakve informacije o suprotnom uređaju, jedan uređaj mora započeti pretragu kako bi pronašao drugi uređaj. Uređaj koji započinje pretragu šalje upit dok mu drugi uređaj odgovara s njegovom adresom, imenom i drugim informacijama.
2. Povezivanje (engl. *Paging*) – To je proces formiranja povezivanja između dva bluetooth uređaja. Prije ovoga procesa, svaki uređaj mora znati adresu drugoga.
3. Veza (engl. *Connection*)- Kada završi proces povezivanja, uređaji ulaze u fazu prijenosa informacija.

3. IGRA KAMEN, ŠKARE PAPIR ZA ANDROID UREĐAJE

3.1. Ideja i osnovni koncept

Osnovna ideja ove igre je kreiranje klasa koje su zaslužne za bluetooth komunikaciju i kopiranjem istih omogućiti drugu aplikaciju da posjeduje bluetooth način komuniciranja. Početni zaslon ove igre treba imati kontrolu za ulazak u odabir kreiranja igre. Također posjeduje konrolu za ulazak u opcije u kojemu se može uključiti bluetooth i isključiti/podesiti glasnoća glazbe. Na sljedećem zaslonu prikazana je opcija igranja s računalom ili s prijateljem. Korisnik koji se pridružuje igri, pritiskom na ime zadanoga uparenoga uređaja započinje igru preko bluetooth komunikacije.

Glavni zaslon ove igre sastoji se od tri opcije (škare, kamen, papir). Klikom na ponuđenu opciju otvara se prozor za potvrdu ili ponovni unos odabira. Nakon potvrde, otvara se zaslon na kojemu je prikazan protivnikov odabir i poruka o rezultatu.

3.2. Izrada aplikacije

Ova aplikacija se zasniva na bluetooth tehnologiji, stoga prvi korak je unos dopuštenja korištenja Bluetooth Adaptera u Android Manifestu.

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

Sl. 3.1. *Dopuštenje korištenja adaptera u Android Manifestu*

Unošenjem BLUETOOTH_ADMIN omogućuje se korištenje naprednijih Bluetooth opcija. Zatim u klasama u kojima je potreban bluetooth uključuje se bluetooth paket.

```
import android.bluetooth.*;
```

Sl. 3.2. *Unošenje Bluetooth paketa u klasu*

Unešenim paketa bluetooth.* uključuju se svi paketi koje bluetooth API nudi (engl. *Application Programming Interfaces* – API).

Ulaskom u početni zaslon aplikacija pokreće pozadinsku glazbu, te omogućuje ulazak u kreiranje igre kao i ulazak u postavke.

```
import android.media.MediaPlayer;
```

Sl. 3.3. *Unošenje MediaPlayer paketa u klasu*

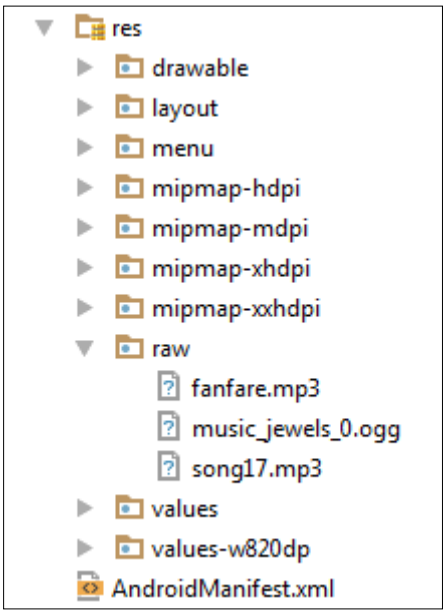
```
public MediaPlayer player ;
```

Sl.3.4. *Definiranje Mediaplayer objekta*

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_rock_paper_scissor);
    player = MediaPlayer.create(RockPaperScissor.this,R.raw.music_jewels_0);
        player.start();
        player.setLooping(true);
```

Sl. 3.5. *Uključivanje glazbe prilikom ulaska u igru (onCreate)*

Prije korištenja MediaPlayer objekta, potrebno je uvesti MediaPlayer paket na isti način na koji je uveden Bluetooth paket. Unošenje glazbe vrši se povlačenjem melodije u folder *res/raw* u aplikaciji.



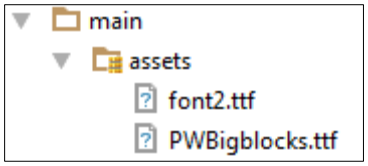
Sl. 3.6. *Umetanje glazbenih datoteka u projekt*

Ulazak u drugu klasu je omogućen objektom tipa Intent.

```
public void gotomore(View view){
    Intent intent = new Intent(RockPaperScissor.this,About.class);
    startActivity(intent);
}
public void settingss(View view){
    Intent intent11 = new Intent(RockPaperScissor.this,Settings.class);
    startActivity(intent11);
}
public void play(View view){
    Intent intentww = new Intent(RockPaperScissor.this,hostijoin.class);
    startActivity(intentww);
}
```

Sl. 3.7. *Ulazak u druge aktivnosti na početnom zaslonu*

Definiranje fontova koji je napravio korisnik se umetnu u mapu *main/assets*. Ukoliko se koriste ti fontovi, uvodi se *Typeface* paket na isti način na koji je uveden i bluetooth paket. Zatim se kreira objekt *Typeface* te mu je pridodan određeni font.



Sl. 3.8. *Umetanje fonta koji je napravio korisnik*

```
import android.graphics.Typeface;
```

Sl. 3.9. *Unošenje Typeface paketa u klasu*

```
Typeface mojfont = Typeface.createFromAsset(getAssets(), "font2.ttf");
b_play.setTypeface(mojfont);
b_settings.setTypeface(mojfont);
b_more.setTypeface(mojfont);
```

Sl. 3.10. *Kreiranje Typeface objekta i definiranje fonta za pojedino dugme*

Sl. 3.11. *Prikaz početnog zaslona*

U postavkama kako je spomenuto treba biti omogućen *Switch* koji isključuje glazbu koja se nalazi u klasi početnog zaslona. Također, treba biti omogućen klizač (engl. *Slider*) koji kontrolira glasnoću. Kada klizač dostigne svoju minimalnu vrijednost *Switch* treba promijenuti tekst sa *Sound ON* na *Sound OFF*. Zadnje dugme treba uključiti bluetooth, a u slučaju ako je uključen, treba ispisati poruku da je bluetooth uključen. Stoga se pravi nova klasa Settings.class i XML prikaz.

```
public void jelukljs(View view) {
    if (switch1.isChecked()) {
        AudioManager aManager=(AudioManager)getSystemService(AUDIO_SERVICE);
        aManager.setStreamMute(AudioManager.STREAM_MUSIC,false );
    } else {
        AudioManager aManager=(AudioManager)getSystemService(AUDIO_SERVICE);
        aManager.setStreamMute(AudioManager.STREAM_MUSIC, true);
    }
}
```

Sl. 3.12. *Definiranje Switch dugmeta u klasi Settings.class*

```
<ToggleButton
    android:layout_width="wrap_content"
    android:layout_height="60dp"
    android:textOn="Sound On"
    android:textOff="Sound Off"
    android:background="#00ffffff"
    android:onClick="jelukljs"
    android:layout_marginBottom="20dp"
    android:id="@+id/switch1"
    android:checked="false" />
```

Sl 3.13. *XML kod za Switch*

Klizač koji kontrolira glasnoću definiran je funkcijom *„initControls()“*. Na početku te funkcije definiran je klizač koji se nalazi u XML-u. Zatim se podešava maksimalna glasnoća koja se kontrolira klizačem. To se definira naredbom *„seekb.setMax()“*. Naredbom *„SetOnSeekBarChangeListener()“* podešava se glasnoća od minimalnog do maximalnog koja je zadana naredbom *„setMax()“*.

```
private void initControls()
{
    try
    {
        seekb = (SeekBar)findViewById(R.id.seekb);
        seekb.setMax(audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC));
        seekb.setProgress(audioManager.getStreamVolume(AudioManager.STREAM_MUSIC));
        seekb.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onStopTrackingTouch(SeekBar arg0) {
            }

            @Override
            public void onStartTrackingTouch(SeekBar arg0) {
            }

            @Override
            public void onProgressChanged(SeekBar arg0, int progress, boolean arg2) {
                audioManager.setStreamVolume(AudioManager.STREAM_MUSIC,progress, 0);
            }
        });
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}
```

3.14. *Definiranje SeekBar-a u klasi Settings.class*

```
<SeekBar
    android:layout_width="200dp"
    android:layout_height="wrap_content"
    android:max="100"
    android:id="@+id/seekb"
    android:layout_marginBottom="30dp"/>
```

3.15. *XML kod za SeekBar*

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    switch (keyCode) {
        case KeyEvent.KEYCODE_VOLUME_UP:
            audioManager.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                    AudioManager.ADJUST_RAISE, AudioManager.FLAG_SHOW_UI);
            return true;
        case KeyEvent.KEYCODE_VOLUME_DOWN:
            audioManager.adjustStreamVolume(AudioManager.STREAM_MUSIC,
                    AudioManager.ADJUST_LOWER, AudioManager.FLAG_SHOW_UI);
            return true;
        default:
            return false;
    }
}
```

Sl. 3.16. *Definiranje glasnoće pomoću tipki na uređaju*

Prilikom pritiska na dugme *„Turn ON“* bluetooth, pojavljuje se skočni prozor s upitom paljenja bluetootha. Ukoliko je bluetooth upaljen, treba biti ispisana poruka *„Bluetooth is allready turned ON“*. Na početku je potrebno definirati novi BluetoothAdapter. To se radi pomoću metode *„getDefaultAdapter()“*.

```
public BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
```

Sl. 3.17. *Definiranje Bluetooth Adaptera*

```
public void jelukljb(View view) {

    if (mBluetoothAdapter != null) {
        if (!mBluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivity(enableBtIntent);
        }
        else {
            Context popup = getApplicationContext();
            CharSequence text = "Bluetooth is already turned ON!";
            int duration = Toast.LENGTH_SHORT;

            Toast toast = Toast.makeText(popup, text, duration);
            toast.show();
        }
    }
  }
```

Sl. 3.18. *Funkcija paljenja bluetooth-a*

Klikom na dugme „*Play*“ otvara se prozor odabira načina igranja. Prva opcija je igranje protiv kompijutera. Ulaskom u taj način igranja, prikazana su tri odabira. Klikom na bilo koji od tih odabira otvara se prozor potvrde određenog unosa. Ukoliko korisnik nije zadovoljan odabirom, to radi pritiskom na dugme „*Try Again!*“. U suprotnome potvrđuje unos i ispisuje se opcija odabira računala, opcija odabira korisnika i poruka o rezultatu.

Opcija računala urađena je generiranjem nasumičnoga broja od 1-10. Određene kombinacije brojeva su prirdužene određenom odabiru. Zatim aplikacija u odnosu na odabir računala i korisnikov unos određuje rezultat.

```
private String getComputerChoice() {
    Random r = new Random();
    num = r.nextInt(10 - 1) + 1;
    return codeToWord(num);
}


private String codeToWord(int num) {
    if (num == 1 || num == 4 || num == 7) {
        cpu_choice = "" + 1;
        cpu_select.setText("ROCK!!");
    } else if (num == 2 || num == 5 || num == 8) {
        cpu_choice = "" + 2;
        cpu_select.setText("PAPER!!");
    } else if (num == 3 || num == 6 || num == 9) {
        cpu_choice = "" + 3;
        cpu_select.setText("SCISSORS!!");
    }
    return cpu_choice;
}
```

Sl. 3.19. *Odabir opcije računala*

Izračunavanje rezultata urađeno je posebnom funkcijom *computeWinner().* Svakom odabiru pridružen je broj 1,2 ili 3, ovisno o odabiru kamen, papir ili škare. Ukoliko je odabrana opcija „1 – kamen“ od strane korisnika, a od strane računala generiran je broj „1- kamen“ funkcija će izbaciti poruku da je izjednačeno.

```
private void computeWinner(String player_choice_number, String cpu_choice) {

    String player2 = za_handlanje.isSinglePlayer ? "Computer" : "Your Friend";

    if (player_choice_number.equals("1") && cpu_choice.equals("2")) {
        winner.setText(player2);
    } else if (player_choice_number.equals("1") && cpu_choice.equals("3")) {
        winner.setText("You");
    } else if (player_choice_number.equals("2") && cpu_choice.equals("1")) {
        winner.setText("You");
    } else if (player_choice_number.equals("2") && cpu_choice.equals("3")) {
        winner.setText(player2);
    } else if (player_choice_number.equals("3") && cpu_choice.equals("1")) {
        winner.setText(player2);
    } else if (player_choice_number.equals("3") && cpu_choice.equals("2")) {
        winner.setText("You");
    } else if (player_choice_number.equals("3") && cpu_choice.equals("3")) {
        winner.setText("Its a TIE! No Result!");
    } else if (player_choice_number.equals("2") && cpu_choice.equals("2")) {
        winner.setText("Its a TIE! No Result!");
    } else if (player_choice_number.equals("1") && cpu_choice.equals("1")) {
        winner.setText("Its a TIE! No Result!");
    }
    flag=false;
}
```

Sl. 3.20. *Funkcija computeWinner()*

Nakon potvrde toga zaslona, otvara se zaslon sa upitima ponavljanja igre, početka nove ili izlaska iz igre.

Ulaskom u zaslon za kreiranje igre preko bluetooth komunikacije, otvara se upit uključivanja bluetootha. Na tome zaslonu obje osobe trebaju biti kako bi pokrenule igru. Osoba koja se pridružuje igri, klikom na dugme „*Search for opponents*“ izlista popis uparenih uređaja, te klikom na ime protivnikovog uređaja započinje igru. Ukoliko protivnik nije na popisu uparenih uređaja, klikom na dugme „*Scan for devices*“ ispisuje sve pronađene bluetooth uređaje. Zatim klikom na ime zadanog traženog uređaja, obojici korisnika se ispisuje potvrda o uparivanju. Nakon što su uređaji upareni, spomenutim postupkom se započinje igra.

Za komunikaciju između bluetooth uređaja potrebno je definirati već spomenuti „*BluetoothAdapter*“. To je ulazno-izlazna točka za svu bluetooth komunikaciju. Korištenjem adaptera možemo prikazati ostale bluetooth uređaje, uparene uređaje, započeti komunikaciju korištenjem jedinstvene *MAC* adrese i kreirati *BluetoothServerSocket* za kontroliranje komunikacije u odnosu na ostale uređaje. „*BluetoothDevice*“ predstavlja udaljeni uređaj. Koristi se za slanje zahtjeva kroz „*BluetoothSocket*“ kako bi saznali traženo ime, adresu, klasu kao i stanje uređaja. „*BluetoothSocket*“ predstavlja tip povezivanja koja dozvoljava aplikaciji da razmijeni podatke sa drugim Bluetooth uređajem koristeći dvije funkcije; „*InputStream*“ i „*OutputStream*“. „*BluetoothServerSocket*“ predstavlja server koji „sluša“ obje strane za određenim zahtjevima. Kako bi se spojila dva uređaja, jedan mora napraviti „*server socket*“ koristeći određenu klasu. Kada bluetooth uređaj pošalje poziv za povezivanje, „*BluetoothServerSocket*“ će vratiti „*BluetoothSocket*“ kada je povezivanje omogućeno. Nakon toga uređaji mogu nesmetano komunicirati.

```
private final BroadcastReceiver mReceiver = new BroadcastReceiver() {
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();

        if (BluetoothDevice.ACTION_FOUND.equals(action)) {
            BluetoothDevice device = intent.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);
            if (device.getBondState() != BluetoothDevice.BOND_BONDED) {
                mNewDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
            }
        } else if (BluetoothAdapter.ACTION_DISCOVERY_FINISHED.equals(action)) {
            setProgressBarIndeterminateVisibility(false);
            setTitle("select a device to connect");
            if (mNewDevicesArrayAdapter.getCount() == 0) {
                String noDevices = "No devices found".toString();
                mNewDevicesArrayAdapter.add(noDevices);
            }
        }
    }
};
```

Sl. 3.21. *Definiranje BroadcastReciever-a*

Definiranjem funkcije *onRecieve()* omogućava se aplikaciji traženje udaljenoga uređaja (kao što je napomenuto u tekstu iznad – *BluetoothDevice-a*). Intentom se dobije tražena MAC adresa uređaja, što se u nastavku uspoređuje sa popisom uparenih uređaja. Ukoliko uređaj nije u popisu uparenih uređaja, nakon završetka rada *BluetoothAdaptera*, uređaj se dodaje na popis neuparenih uređaja. Klikom na jedan od uparenih uređaja, počinje proces povezivanja.

Kada se povezuju dva uređaja, jedan od njih se mora ponašati kao poslužitelj tako što omogućava otvoren *BluetoothServerSocket*. Svrha toga predstavlja čekanje nadolazećega zahtjeva, i kad je isti prihvaćen, omogućava se odgovarajući stvoreni *BluetoothSocket*. Pozivanjem funkcije *listenUsingRfcommWithServiceRecord(String, UUID)* korisnik zahtjeva određeni, unaprijed definirani jedinstveni broj aplikacije (*Universally Unique Identifier*). Taj broj je jedinstven za svaku aplikaciju. Ukoliko uređaj koji se želi priključiti na poslužitelj nema isti UUID, aplikacija ne omogućava daljnju komunikaciju uređajima. Kada se jedinstveni brojevi podudaraju kod obje strane, tada server odgovara pozivom na *accept()*.

```
private static final UUID MY_UUID_SECURE = UUID.fromString("fa87c0d0-afac-11de-8a39-0800200c9a66");
```

Sl. 3.22. *Definiranje UUID-a*

Komunikacija između servera i hosta u Bluetooth tehnologiji se omogućava preko *Thread*-ova. Poslužitelj strana posjeduje *AcceptThread* dok klijent strana posjeduje klasu *ConnectThread*.

```
private class AcceptThread extends Thread {
    // The local server socket
    private final BluetoothServerSocket mmServerSocket;
    private String mSocketType;

    public AcceptThread(boolean secure) {
        BluetoothServerSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Create a new listening server socket
        try {
            if (secure) {
                tmp = mAdapter.listenUsingRfcommWithServiceRecord(NAME_SECURE, MY_UUID_SECURE);
            } else {
                tmp = mAdapter.listenUsingInsecureRfcommWithServiceRecord(NAME_INSECURE, MY_UUID_INSECURE);
            }
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "listen() failed", e);
        }
        mmServerSocket = tmp;
    }

    public void run() {...}

    public void cancel() {...}
}
```

Sl. 3.23. *Klasa AcceptThread*

Dva su načina povezivanja bluetooth servera i klijenta; *secure* i *insecure*. *Secure* način zahtijeva takav oblik komunikacije da se nakon svakog slanja od suprotne strane primi potvrda za uspješno primljen paket. Kod *insecure* načina rada, nakon uspostave veze, podatci se neprestano šalju bez ikakve potvrde što omogućava trećoj strani (*hakerima*) da jednostavno

uđu u vezu. Takva vrsta veze se u novim funkcijama bluetootha ne može upotrebljavati i izbjegava se. Ova aplikacija je bazirana na *secure* tipu razmjene podataka.

```
    public ConnectThread(BluetoothDevice device, boolean secure) {
        mmDevice = device;
        BluetoothSocket tmp = null;
        mSocketType = secure ? "Secure" : "Insecure";

        // Get a BluetoothSocket for a connection with the
        // given BluetoothDevice
        try {
            if (secure) {
                tmp = device.createRfcommSocketToServiceRecord(MY_UUID_SECURE);
            } else {
                tmp = device.createInsecureRfcommSocketToServiceRecord(MY_UUID_INSECURE);
            }
        } catch (IOException e) {
            Log.e(TAG, "Socket Type: " + mSocketType + "create() failed", e);
        }
        mmSocket = tmp;
    }

    public void run() {...}

    public void cancel() {...}
}
```

Sl. 3.24. *Klasa ConnectThread*

Nakon što su uređaji povezani, između njih je stanje *.isconnected(),* i time je omogućeno neometano slanje podataka. Klikom na jednu od ponuđenih opcija, drugoj strani se šalje opcija odabira prve osobe. Ta iteracija je određena vremenom od 60 sekundi, točnije unutar toga razdoblja, drugi uređaj mora klikuti jednu od navedenih. U suprotnom se prekida komunikacija između ta dva uređaja i oba igrača se automatski vraćaju na zaslon izbora igranja protiv računala ili prijatelja. Ukoliko je druga osoba kliknula na jednu od opcija, spomenuta funkcija *computeWinner()* izračunava rezultat i isti se ispisuje na ekranima igrača.

```
public void decideMultiPlayerWinner() {

    new CountDownTimer(60000,1000) {

        @Override
        public void onTick(long l) {
            if (za_handlanje.inComingMessage != null && !flag) {
                String player2Choice = decodePlayerChoice(za_handlanje.inComingMessage);
                codeToWord(Integer.parseInt(player2Choice));
                computeWinner(player_choice_number, player2Choice);
                records.setEnabled(true);
                za_handlanje.inComingMessage = null;
                flag=true;
                this.cancel();
            }
        }

            @Override
            public void onFinish () {

            }
    }.start();

}
```

Sl. 3.25. *Kreiranje Timera*

```
private String getPlayerChoice() {
        player_choice = sig_sure.playerChoiceS3.toString();

    player_select = (TextView) findViewById(R.id.player_selected_ans);
    player_choice_number = decodePlayerChoice(player_choice);
    if (player_choice_number.equals("1")) {
        player_select.setText("ROCK!!");
    } else if (player_choice_number.equals("2")) {
        player_select.setText("PAPER!!");
    } else if (player_choice_number.equals("3")) {
        player_select.setText("SCISSORS!!");
    }
    return player_choice_number;
}
```

Sl. 3.26. *Funkcija za odabir ponuđenih opcija*

Na tome zaslonu, prikazani su upiti želi li korisnik ponovno igrati, kreirati novu igru i izlaći iz aplikacije. Sve tri opcije su urađene pomoću objekta tipa *Intent* čime se odlazi u odgovarajuću traženu klasu.

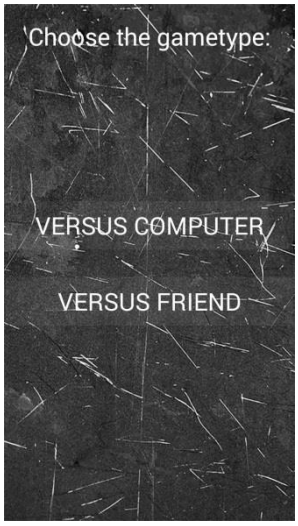
```
public void nextActivity(View v) {
    Intent i = new Intent(winner_is_rez.this, play_again.class);
    startActivity(i);
}
```

Sl. 3.27. *Kreiranje objekta tipa Intent za opciju ponovnog igranja*

3.3. Testiranje aplikacije

3.3.1. Testiranje Single-Player izbora

Klikom na dugme play otvara se izbor igranja (Slika 1.). Nakon toga ponuđen je *Rock*, *Paper* ili *Scissors* izbor (Slika 2.).



Sl. 3.28. *Slika 1.*

Sl. 3.29. *Slika 2.*

Nakon izbora otvara se mogućnost potvrde unosa, ili ponovnog odabira (*Slika 3.*). Klikom na *Confirm* potvrđuje se unos i ispisuje rezultat (*Slika 4.*). Poslije potvrde rezultata ponuđene su opcije ponovnoga igranja, početka nove igre i izlaska iz aplikacije (*Slika 5*).

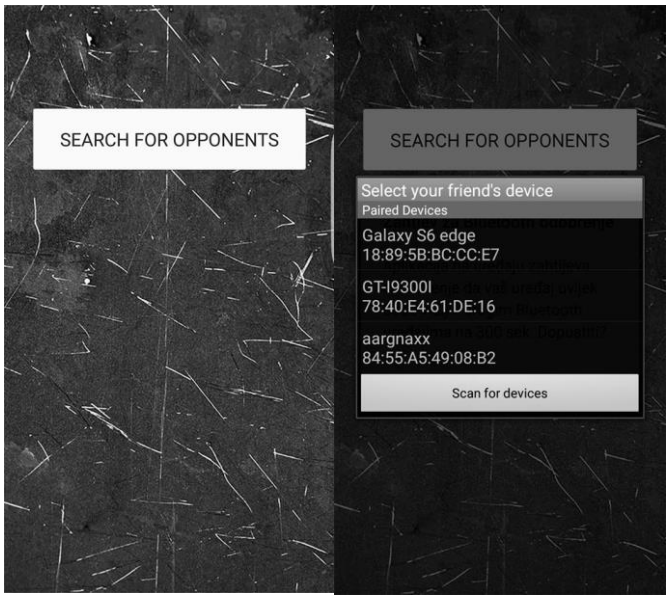
Sl. 3.30 *Slika 3.*

Sl. 3.31. *Slika 4.*

Sl. 3.32. *Slika 5.*

3.3.2. Testiranje Multi-Player Izbora

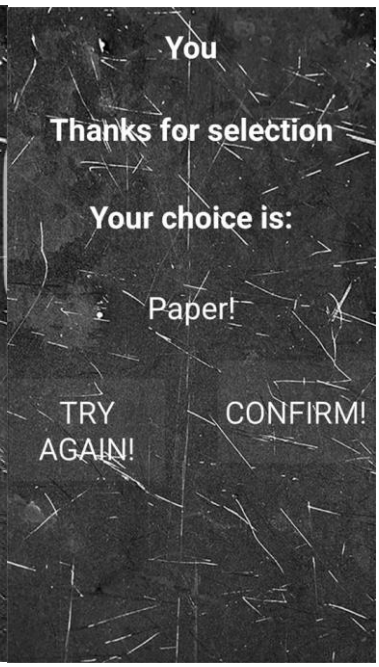
Prilikom pritiska na dugme *VERSUS FRIEND* otvara se prozor traženja uređaja. Kako bi se klijent mogao povezati sa poslužiteljem, oba trebaju biti u istome zaslonu. Pritiskom na dugme *Search for opponents* otvara se zaslon uparenih uređaja. Odabirom na zadani uređaj počinje postupak povezivanja i oba započinu igru tako što se postavlja zaslon odabira opcija.



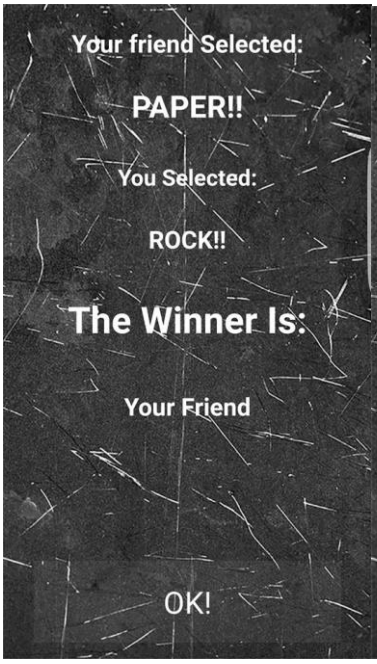
Sl. 3.33 *Slika 6.* ***Sl. 3.34.*** *Slika 7.*

Sl. 3.35. *Slika 8.* ***Sl. 3.36.*** *Slika 9.*

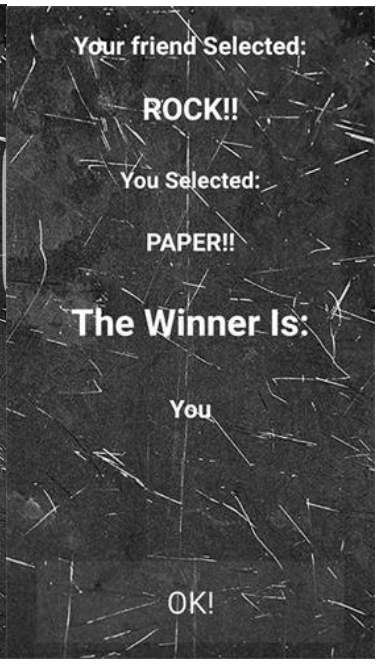
Sl. 3.37 *Slika 10.*



Sl. 3.38. *Slika 11.*



Sl. 3.39 *Slika 12.*



Sl. 3.40. *Slika 13.*

Kada oba igrača pritisnu dugme *OK!* otvara se već spomenuti prozor (***Sl. 3.32.*** *Slika 5.)*, i odabirom na jednu od ponuđenih stvari izvršavaju zadanu funkciju.

4. ZAKLJUČAK

Znanje koje je bilo potrebno za izradu ove aplikacije je usvojeno samostalno. Java programski jezik je usvojen preko knjige *Java FOR DUMMIES,* preko određenih internet članaka i stranice *Stackoverflow.com.* Bluetooth komunikacija je idealna komunikacija za energetski-intenzivne iteracije s drugim korisnicima. U android verziji 4.3 (*API Level 18*) predstavljena je podrška *Bluetooth Low Energy*. Time je zadano da komunikacija između bluetooh uređaja zahtjeva jako malu potrošnju energije što je čini idealnom za prijenos datoteka i komunikaciju.

Ideja ove aplikacije i završnog rada bila je implementacija određenih klasa u druge aplikacije kako bi posjedovale ovu vrstu komunikacije. Zadane klase u aplikaciji Rock Paper Scissors koje su ključne za komunikaciju i u kojima su definirane metode za podržavanje su *DefiniranjeBTADSC.java, DefiniranjeThreadova.java i KlasaTratenjaUredjaja.java.* Aplikacija je napravljena u razvojnom okruženju Android Studio.

Posebne zahvale kolegi Kristijanu Štefančiću koji mi je uveliko pomogao pri rješavanju određenih problema.

LITERATURA

[1] Wikipedija ,Bluetooth komunikacija (Bluetooth) (Stranica posjećena 22.06.2016) https://hr.wikipedia.org/wiki/Bluetooth

[2] Osnove Bluetootha (Bluetooth basics) (Stranica posjećena 22.06.2016) https://learn.sparkfun.com/tutorials/bluetooth-basics

[3] Wikipedija, Android Studio (Stranica posjećena 22.06.2016) https://en.wikipedia.org/wiki/Android_Studio

[4] Bluetooth Low Energy (Stranica posjećena 24.06.2016) https://developer.android.com/guide/topics/connectivity/bluetooth-le.html

[5] Slanje Intenta pomoću Bluetooth komunikacije (Stranica posjećena 28.06.2016) http://stackoverflow.com/questions/16755732/share-intent-with-bluetooth-option-only

[6] Bluetooth Chat (Stranica posjećena 13.07.2016) https://developer.android.com/guide/topics/connectivity/bluetooth.html#TheBasics

[7] Bluetooth komunikacija (Stranica posjećena 15.09.2016) http://stackoverflow.com/questions/19897628/need-to-handle-uncaught-exception-and-send-log-file

[8] Slanje String tipa podatka korištenjem Bluetooth komunikacije (Stranica posjećena 12.09.2016) http://stackoverflow.com/questions/22899475/android-sample-bluetooth-code-to-send-a-simple-string-via-bluetooth

[9] Bluetooth tutorial (Stranica posjećena 23.06.2016) http://www.tutorialspoint.com/android/android_bluetooth.htm

[10] Uparivanje Bluetooth uređaja (Stranica posjećena 28.06.2016) http://stackoverflow.com/questions/14228289/android-pair-devices-via-bluetooth-programmatically

[11] Bluetooth Demo Aplikacija (Stranica posjećena 17.08.2016) http://www.coderzheaven.com/2016/03/10/bluetooth-demo-in-android-listing-paired-devies-find-other-online-bt-devices-connect-to-a-bt-device-send-data-from-and-to-bt-devices-chat/

[12] Barry Burd, Java FOR DUMMIES

[13] Mark L. Murphy, The Busy Coders (Guide to Android Development)

[14] Infiniteskills - Beginners Java Programming Video Tutorial

[15] Infiniteskills – Advanced Java Programming Video Tutorial

SAŽETAK

Ova aplikacija je napravljena u svrhu zabave. Ukoliko osobe koje žele igrati imaju ovu aplikaciju instaliranu na uređaju, mogu to učiniti jako lako. Igra se bazira na Bluetooth komunikaciji što jedan uređaj čini poslužiteljem a drugi uređaj klijentom. Prednost ovakve aplikacije je u tome što uređaji ne moraju imati pristup internetu. Zbog nezahtijevnog grafičkog sučelja, svi android uređaji koji posjeduju Bluetooth adapter mogu instalirati igru. Još jedna prednost je jednostavnost. Praćenjem određenih koraka koji su prikazani na zaslonu možemo jednostavno, brzo i efikasno igrati ovu igru. Implementacijom spomenutih klasa se jednostavno mogu napraviti programi u kojima je Bluetooth povezivost poželjna.

Ključne riječi: Bluetooth, Android, Igra Škare-Kamen-Papir

ABSTRACT

Rock, Paper, Scissors Android Game for Multiple Players

This application was made for fun. If people playing it have this application instaled on their device, they can do it very easily. Game is based on bluetooth comunication that makes one device server and other client. Advantage of this application is not needing to have internet acces. Becouse of undemanding graphical interface, all android devices that have bluetooth adapter can instal the game. Another advantage is its simplicity. By tracking certain steps that are showcased on screen we can quickly and efficiently play this game. Implementing mantioned classes can easily make programs in witch bluetooth connection is desirable.

Keywords: Bluetooth, Android, Game Rock-Paper-Scissors

ŽIVOTOPIS

Darko Čalušić rođen je 25. ožujka 1995. godine u Ozimici, Žepče ,Bosna i Hercegovina. Stanuje u Osijeku, na adresi Frana Krste Frankopana 5. Završio je drugu godinu preddiplomskoga studija računarstva na Fakultetu eletrotehnike, računarstva i informacijskih tehnologija. Srednju školu pohađao je u Katoličkom školskom centru u Žepču kao tenhičar za mehatroniku. Osim brojnih rezultata na općinskim natjecanjima u znanju, posebno se istaknuo u glazbenim natjecanjima. Posjeduje glazbeni studio i bio je član glazbenog rock banda 7 godina. Od tehničkog znanja koje posjeduje, ističe znanje u Pascalu, C, C++, C#, Javi, pravljenju Wordpress stranica i poznavanje rada s bazama podataka u MySQL-u.

Potpis:

PRILOZI

DVD

-Android Studio projekt

-Rad u .docx i .pdf formatu