Šolski center Novo mesto

Srednja elektro šola in tehniška gimnazija

Šegova ulica 112

8000 Novo mesto

Maturitetna seminarska naloga pri predmetu računalništvo

**IZDELAVA IGER – SPACE BIRD**

Avtor: David Gole, T4c

Mentor: dr. Albert Zorko, univ. dipl. inž.

Križišče, april 2021

**POVZETEK**

V maturitetni seminarski nalogi je predstavljena izdelava igre za operacijski sistem Android s pomočjo sodobnega programskega okolja Android Studio v programskem jeziku Java. Prednosti operacijskega sistema Android se pokažejo predvsem zaradi zelo zmogljivih prenosnih naprav, kot so pametni telefoni in tablice. V nalogi je izdelana mobilna igra Space Bird, ki temelji po igri Flappy Bird. Izdelana igra Space Bird je eno igralska igra, pri kateri se ptič poskuša izogniti parom cevi in dobiti čim več točk.

**KLJUČNE BESEDE**

* Igra
* Android
* Java
* Android Studio
* Metoda
* Razred

**ABSTRACT**

This paper presents the creation of a game for the Android operating system using the modern Android Studio programming environment in the Java programming language. The advantages of the Android operating system are shown mainly by the high-performance portable devices such as smartphones and tablets. In this paper is a mobile game Space Bird, based on the game Flappy Bird. Game Space Bird is a single-player game in which the bird tries to avoid pairs of pipes and get as many points as possible.

**KEY WORDS**

* Game
* Android
* Java
* Android Studio
* Method
* Class

**KAZALO VSEBINE**

[1 UVOD 2](#_Toc69343142)

[1.1 ANDROID STUDIO 2](#_Toc69343143)

[1.2 JAVA 3](#_Toc69343144)

[1.3 XML 6](#_Toc69343145)

[2 IZDELAVA IGRE 10](#_Toc69343146)

[2.1 NAČRTOVANJE IZDELAVE 10](#_Toc69343147)

[2.2 PROGRAMIRANJE 11](#_Toc69343148)

[2.2.1 PREDSTAVITEV RAZREDOV PROGRAMA V JAVI 11](#_Toc69343149)

[2.2.2 PROGRAMIRANJE V XML 21](#_Toc69343150)

[2.3 TESTIRANJE 24](#_Toc69343151)

[3 ZAKLJUČEK 25](#_Toc69343152)

[4 ZAHVALA 26](#_Toc69343153)

[5 VIRI IN LITERATURA 27](#_Toc69343154)

[6 PRILOGE 28](#_Toc69343155)

[7 STVARNO KAZALO 29](#_Toc69343156)

**KAZALO SLIK**

[Slika 1: Projektne datoteke v pogledu Android 3](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343129)

[Slika 2: Središče pogledov v relativnem pogledu 7](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343130)

[Slika 3: Postavitev pogledov glede na id drugega pogleda 8](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343131)

[Slika 4: Idejna skica Space Bird 10](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343132)

[Slika 5: Ptica 10](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343133)

[Slika 6: Ptica s plamenom 10](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343134)

[Slika 7: Zgornja cev 11](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343135)

[Slika 8: Spodnja cev 11](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343136)

[Slika 9: Ozadje igre(Zvezdno nebo) 11](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343137)

[Slika 10: Android Studio v pogledu urejanja XML datoteke 21](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343138)

[Slika 11: Posnetek zaslona med igranjem igre 24](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343139)

[Slika 12: Posnetek zaslona pred igranjem igre 24](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343140)

[Slika 13: Posnetek zaslona ob izgubi igre 24](https://d.docs.live.net/0b1fe88fd3a62085/Šola/Računalništvo/Izdelava%20iger%20-%20Space%20Bird.docx#_Toc69343141)

**KAZALO KODE**

[Koda 1: Lastnosti razreda MainActivity 11](#_Toc69343112)

[Koda 2: Metoda onCreate() v razredu MainActivity 12](#_Toc69343113)

[Koda 3: Razred Konstante 13](#_Toc69343114)

[Koda 4: Lastnosti razreda OsnovniRazred 13](#_Toc69343115)

[Koda 5: Konstruktor razreda OsnovniRazred 13](#_Toc69343116)

[Koda 6: Osnovne metode razreda OsnovniRazred 14](#_Toc69343117)

[Koda 7: Lastnosti in konstruktor razreda Ptica 14](#_Toc69343118)

[Koda 8: Metode draw(), gravitacija() in setArrBm() v razredu Ptica 15](#_Toc69343119)

[Koda 9: Animacije ptice v razredu Ptica 16](#_Toc69343120)

[Koda 10: Lastnosti in konstruktor razreda Cevi 16](#_Toc69343121)

[Koda 11: Metode draw(), nakljucnoY() in setBm() v razredu Cevi 17](#_Toc69343122)

[Koda 12: Lastnosti in konstruktor razreda UprizoritevIgre 18](#_Toc69343123)

[Koda 13: Metoda ustvariCev() v razredu UprizoritevIgre 18](#_Toc69343124)

[Koda 14: Metoda ustvariPtico() v razredu UprizoritevIgre 19](#_Toc69343125)

[Koda 15: Metoda draw() v razredu UprizoritevIgre 20](#_Toc69343126)

[Koda 16: Metode onTouchEvent(), isStart(), setStart() in reset() v razredu UprizoritevIgre 20](#_Toc69343127)

[Koda 17: Datoteka activity\_main.xml 23](#_Toc69343128)

# UVOD

Mobilna igra je video igra, ki se običajno igra na mobilnem telefonu. V preteklosti se izraz nanaša na vse igre, ki se igrajo na kateri koli prenosni napravi , vključno z mobilnim telefonom. Najzgodnejša znana igra na mobilnem telefonu je bila različica Tetris-a na napravi Hagenuk MT-2000 iz leta 1994. (1)

Mobilna igra Space Bird temelji na sorodni igri z naslovom Flappy Bird, ki jo je maja 2013 razvil vietnamski programer Dong Nguyen. Obe igri imata enak način igranja. Igralec te igre nadzoruje ptico, ki se vztrajno premika v desno. Igralec je zadolžen za navigacijo ptice in se poskusi izogniti parom cevi, ki imajo enako velike reže, nameščene na naključni višini. Ptica se samodejno spusti in dvigne šele, ko igralec tapne na zaslon na dotik . Vsak uspešen prehod skozi par cevi igralcu podeli eno točko. Trčenje s cevjo ali umik ptice iz zaslona konča igro.

Telefoni, ki so danes na tržišču, imajo različne operacijske sisteme, kar pri razvoju iger vnaša še dodaten izziv. […] Prednost operacijskega sistema Android je ta, da je odprtokoden ter omogoča lažje in cenejše razvijanje aplikacij, pri čemer je tudi bolj priljubljena platforma med izdelovalci. Slabost pa, da naprave pridejo v več različnih velikosti in s tem otežijo izdelavo igre za določene android naprave. (2)

Naša igra bo ustvarjena za operacijski sistem Android. Razvoj igre bo izveden s programskim okoljem Android Studio in sicer v programskem jeziku Java. Vse potrebne slike, ki jih bom potreboval za izdelavo igre bom izdelal v programu Windows 3D Slikar in Windows Slikar. Namen maturitetne seminarske naloge je prikaz sodobnega ustvarjanja 2D igre za operacijski sistem Android. Teza naloge je, da je možno izdelati igro za pametni telefon s pomočjo programskih orodij, ki so namenjena za razvoj iger.

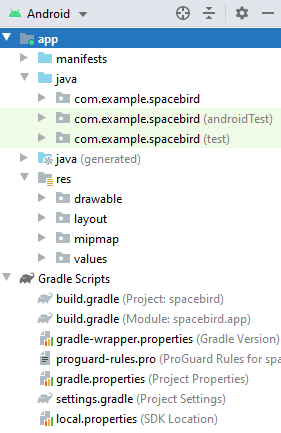
## ANDROID STUDIO

Android Studio je uradno integrirano razvojno okolje (IDE) za Googlov operacijski sistem Android. […] Android Studio je na voljo za prenos v operacijskih sistemih Windows , macOS in Linux ali kot naročninska storitev leta 2020. Je nadomestek za Eclipse Android Development Tools (E-ADT) kot primarni IDE za razvoj izvornih aplikacij za Android. (3)

Kotlin je 7. maja 2019 zamenjal Javo kot Googlov najljubši jezik za razvoj aplikacij za Android. Java je še vedno podprta, prav tako C ++ . (3)

Vsak projekt v programu Android Studio vsebuje enega ali več modulov z datotekami izvorne kode in datotekami virov. Vrste modulov vključujejo:

* Moduli aplikacij za Android
* Moduli knjižnic
* Moduli Google App Engine (4)



Slika : Projektne datoteke v pogledu Android

Android Studio privzeto prikaže vaše projektne datoteke v pogledu projekta Android, kot prikazuje Slika 1. Ta prikaz je organiziran po modulih za hiter dostop do ključnih izvornih datotek vašega projekta.

Vse gradbene datoteke so vidne na najvišji ravni pod **Gradle Scripts** in vsak modul aplikacije vsebuje naslednje mape:

* **manifests** (Vsebuje AndroidManifest.xml datoteko)
* **java** (Vsebuje datoteke izvorne kode Java, vključno s preskusno kodo JUnit.)
* **res** (Vsebuje vse nekodirane vire, kot so postavitve XML, nizi uporabniškega vmesnika in bitne slike.)

## JAVA

Java je v skoraj desetletnem obstoju doživela veličasten razvoj. Pridobila je zaupanje številnih programerjev, poleg tega pa predstavlja danes znanje programiranja v javi eno najboljših priporočil za zaposlitev. (5)

Veliko je slišati očitkov o Javi, da je prepočasna za programiranje iger, da ima puščanje pomnilnika, da je na previsoki ravni in da je njena namestitev prezahtevna.

Velja povedati, da mislim, da so skoraj vsi ti ugovori v bistvu napačni. Java ima približno enako hitrost kot C ++. Puščanju pomnilnika se je mogoče izogniti z dobrimi tehnikami programiranja, kot je profiliranje. Da, Java je na visoki ravni, vendar ponuja več neposrednega dostopa do grafične strojne opreme in zunanjih naprav. Namestitev ni nočna mora, če uporabljate dostojno namestitveno programsko opremo. Vse več je odličnih, zabavnih iger v Javi in ogromno podpore. (6)

Samo programiranje v razvojnem programskem okolju Android Studio je skoraj identično kot na platformi Eclipse IDE. Programski jezik je enak, je pa potrebno več znanja za pravilno delovanje aplikacije oziroma igre saj morajo skupno delovati različni programski jeziki, kot XML in Java. Za delujoč rezultat so potrebni različni razredi in različne metode. Spodaj je naštetih nekaj metod za povezovanje XML z Javo.

**setContentView()**

public void setContentView(int layoutResID)

Metoda omogoča določitev željene postavitve ob določenem trenutku in uporabi poljubno XML datoteko, ki predstavlja izgled in postavitev (7). V tej igri ta metoda uporabi activity\_main.xml.

**findViewById()**

public final T findViewById(int id)

Poišče pogled, ki ga je identificiral android: id atribut XML, ki je bil obdelan v onCreate(Bundle). Parameter id pomeni ID za iskanje, T pa pogled z danim ID-jem, če ga najdemo ali vrednost null, če ga ne najdemo. (7)

**setOnClickListener()**

public void setOnClickListener(View.OnClickListener l)

Metoda omogoča vezavo vmesnika za poslušanje oziroma OnClickListener() na vmesniški element, npr. gumb (angl. Button). Gnezdena metoda onClick()nato ob pritisku na določeni vmesniški element izvede kodo, ki je znotraj metode. (7)

Spodaj so še podani razredi Handler, Runnable in Bitmap, kateri vsebujejo nekatere zelo priročne metode za sistematično ponavljanje programa in kreiranje in urejanje bitnih slik katere sem tudi uporabil.

**Razred Handler**

Handler omogoča pošiljanje in obdelavo Message in Runnable objektov, povezane z nitjo MessageQueue. Vsak primerek Handler-ja je povezan z eno nitjo in čakalno vrsto sporočil te niti. Ko se ustvari nov Handler, je ta vezan na Looper. V to čakalno vrsto sporočil Looper bo dostavil sporočila in izvedljive datoteke ter jih izvedel v tej Looperjevi niti (8). Iz razreda Handler sem uporabil eno metodo in sicer postDelayed(), ki zagotavlja ponavljajoče se zaganjanje in osveževanje programa po prej določenem času.

postDelayed()

public final boolean postDelayed (Runnable r, long delayMillis)

Povzroči, da se Runnable r doda v čakalno vrsto sporočil in se zažene po poteku določenega časa. Izvajalec se bo zagnal v niti, na katero je ta upravljalnik pritrjen. Časovna osnova je SystemClock.uptimeMillis(). Čas, ki je preživet v globokem spanju, bo še dodatno zakasnil izvedbo (8).

**Razred Runnable**

Java Runnable je vmesnik, ki se uporablja za izvajanje kode v sočasni niti. To je vmesnik, ki ga izvaja kateri koli razred, če želimo, da se primerki tega razreda izvajajo z nitjo. Vmesnik, ki ga je mogoče zagnati, ima nedoločeno metodo run() z void kot vrnitveno vrsto in ne zajema nobenih argumentov. (9)

run()

public abstract void run()

Ta metoda ne upošteva argumentov. Ko se objekt razreda, ki izvaja razred Runnable, uporablja za ustvarjanje niti, se v niti, ki se izvaja ločeno, prikliče metoda run(). V tej metodi imamo kodo, ki jo želimo izvesti v sočasni niti. Pri tej metodi lahko uporabljamo spremenljivke, ustvarimo primerke razredov in izvedemo dejanje na enak način kot glavna nit. Nit ostane do vrnitve te metode. Metoda run() vzpostavi vstopno točko v novo nit. (9)

**Razred Bitmap**

Bitmap je digitalna slika sestavljena iz matrike pik. Pri 100-odstotni gledanosti vsaka pika ustreza posamezni slikovni piki na zaslonu. Na standardni slikovni sliki lahko vsaki piki dodelimo drugačno barvo. (10)

createBitmap()

public static Bitmap createBitmap(Bitmap source, int x, int y, int width, int height)

Vrne bitno sliko iz podane podmnožice izvorne bitne slike. Nova bitna slika je lahko isti predmet kot vir ali pa je bila narejena kopija. Inicializira se z enako gostoto in barvnim prostorom kot prvotna bitna slika. (10)

createScaledBitmap()

public static Bitmap createScaledBitmap(Bitmap src, int dstWidth, int dstHeight, boolean filter)

Če moramo spremeniti velikost Bitmap, lahko pokličemo createScaledBitmap() metodo za spreminjanje velikosti bitne slike na želeno širino in višino. (10)

Zelo pomembni sta metodi invalidate(), ki razveljavi pogled, ter metoda intersect(), ki preveri presečišče dveh pravokotnikov.

**invalidate()**

public void invalidate()

Funkcija invalidate() razveljavi celoten pogled (angl. View). Če je pogled viden, v neki točki pokliče metodo onDraw(android.graphics.Canvas). (11)

**intersect()**

public boolean intersect(Rect r)

Če navedeni pravokotnik seka ta pravokotnik, metoda vrne true in nastavi ta pravokotnik na to presečišče, sicer vrne false in tega pravokotnika ne spreminja. Preverjanje, ali je kateri koli pravokotnik prazen, ni izvedeno. (12)

## XML

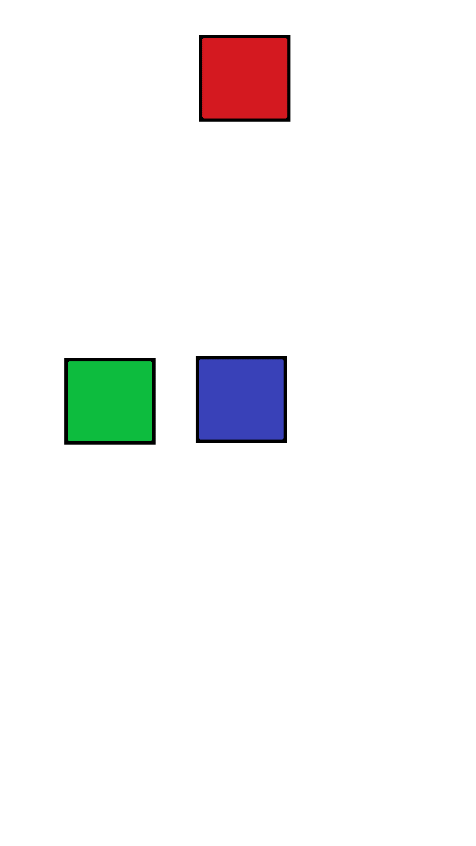
Za razvoj Android aplikacije pomeni znanje programskega jezika Java ali Kotlin , kar tudi uradno podpira Android Studio , ki je Googlovo uradno integrirano razvojno okolje za razvoj v Androidu. Vendar se redko govori o XML in to lahko povzroči zmedo, ko prvič odprete nov projekt in opazite, da sta dejansko dve različni glavni datoteki in dve različni vrsti skriptov. XML[[1]](#footnote-1) je označevalen jezik in se nekoliko razlikuje od programskega jezika. Ker programski jezik kot so C #, C ++, Java in Kotlin omogočajo določanje vedenja, interakcij in pogojev. Označevalni jezik se bolj uporablja za opis podatkov in v tem primeru postavitev. Programski jeziki ustvarjajo dinamične interakcije, medtem ko označevalni jeziki običajno obravnavajo stvari, kot so statični uporabniški vmesniki. Ko začnemo s android projektom, se prikaže hierarhija različnih datotek in map. Zaenkrat se osredotočimo le na dve datoteki in sicer MainActivity.java in activity\_main.xml. V datoteki MainActivity.java z metodo setContentView(R.layout.activity\_main) lahko pokličemo datoteko activity\_main.xml.

Druga alternativna uporaba XML je v Android Manifest ( AndroidManifest.xml ). V njem je veliko podatkov, ki opisujejo aplikacijo, na primer oznaka (ime aplikacije), ikona in navodila, katere dejavnosti je treba najprej naložiti. Pomaga tudi, da se aplikacija pravilno prikaže na začetnem zaslonu, in pravilno funkcionira v trgovini z aplikacijami. (11)

Pomembnejši pogledi oziroma gradniki v datoteki activity\_main.xml so TextView, Button in RelativeLayout. Vsak mora vsebovati svoj ključ (ID), da ga lahko pokličemo iz datoteke MainActivity.java, in po navadi vsebuje še nekatere atribute.

**RelativeLayout**

<RelativeLayout></RelativeLayout>

Relativna postavitev je postavitev, ki razporeja poglede ali gradnike glede na položaj drugih pogledov. Novi pogledi so postavljeni glede na že obstoječe poglede (13). Torej za primer lahko predpostavimo učilnico v kateri so učenci in učitelj. Če učenec A sedi na stolu in učitelj prosi učenca B naj sedi desno od učenca A, bo učenec B vedel kje mora sedeti. Podobno je tudi pri relativni postavitvi le da so tukaj namesto učencev različni pogledi kot so <Button> in <TextView> in namesto navodil določeni atributi. Spodaj je naštetih nekaj atributov.

Slika : Središče pogledov v relativnem pogledu

android: layout\_centerHorizontal = "true"

S tem je pogled postavljen vodoravno v sredino zaslona. Ker nadrejeni pogled pokriva celoten zaslon mobilnega telefona, se pogled postavi na sredino mobilnega zaslona vodoravno. To prikazuje Slika 2 in sicer rdeč kvadrat.

android: layout\_centerVertical = "true"

S tem je pogled postavljen navpično v sredino zaslona. Ker nadrejeni pogled pokriva celoten zaslon mobilnega telefona, se pogled postavi navpično na sredino mobilnega zaslona. To prikazuje Slika 2 in sicer zelen kvadrat.

android: layout\_centerInParent = "true"

Ta atribut bo podrejen pogled postavil v sredino nadrejenega pogleda. Ker pogled pokriva celoten zaslon mobilnega telefona, se podrejeni pogled postavi na sredino mobilnega zaslona tako vodoravno kot navpično. To prikazuje Slika 2 in sicer moder kvadrat.

Recimo, da je v središču en pogled in njegov ključ je podan kot. android:id="@+id/main". Tako lahko druge nove poglede postavimo glede na ta pogled, kot sledi:

android: layout\_toLeftOf = "@ id / main"

Nov pogled se postavi na levo stran pogleda z ključem main. To prikazuje Slika 3 rumen kvadrat.

android: layout\_toRightOf = "@ id / main"

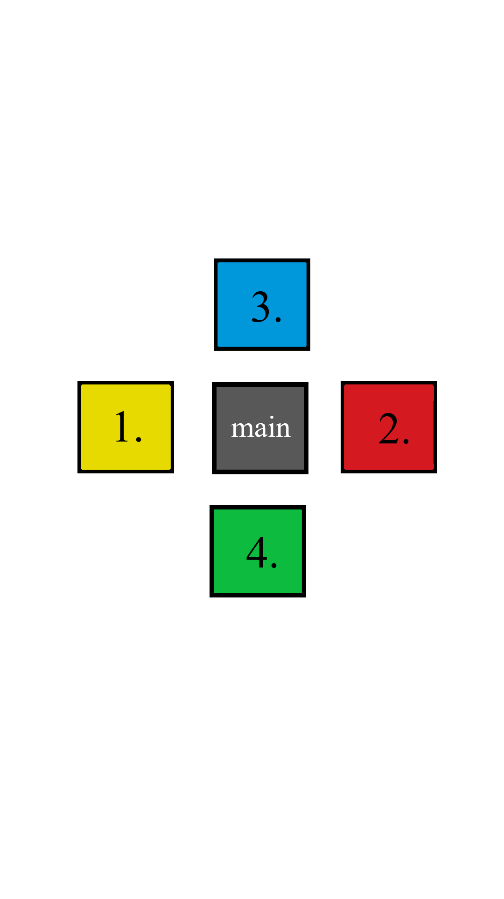
Nov pogled se postavi na desno stran pogleda z ključem main. To prikazuje Slika 3 rdeč kvadrat.

android: layout\_above = "@ id / main"

Nov pogled se postavi na zgornjo stran pogleda z ključem main. To prikazuje Slika 3 moder kvadrat.

android: layout\_below = "@ id / main"

Nov pogled se postavi na spodnjo stran pogleda z ključem main. To prikazuje Slika 3 zelen kvadrat.



Slika : Postavitev pogledov glede na id drugega pogleda

**TextView**

<TextView></TextView>

TextView uporabniku prikaže besedilo in mu po želji omogoča programsko urejanje. TextView je popoln urejevalnik besedil, vendar je osnovni razred nastavljen tako, da ne dovoljuje urejanja, lahko pa ga vseeno urejamo. (14)

Pogled TextView lahko ustvarimo tako, da ga napišemo v postavitvi torej v datoteki XML ali s programskim primerkom v razredu Jave .

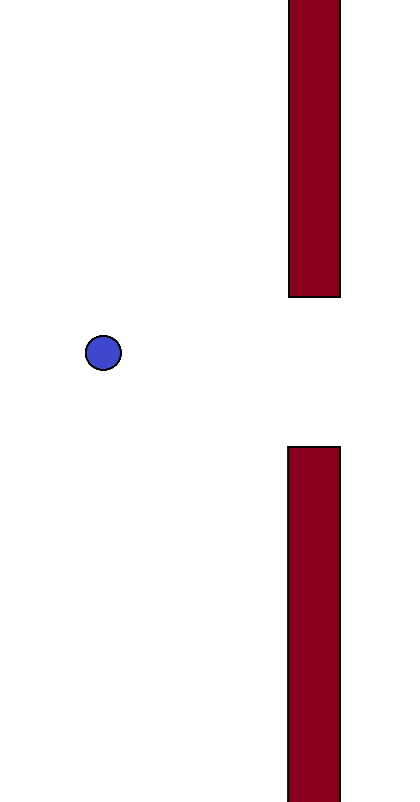
**Button**

<Button></Button>

Button predstavlja gumb. Uporabnik lahko klikne ali pritisne gumbe za izvedbo dejanja. V androidu se uporabljajo različne vrste gumbov, kot so CompoundButton, ToggleButton , RadioButton. Button je podrazred razreda TextView. Na gumbu lahko izvajamo različna dejanja ali dogodke, kot so dogodek klikni, pritisnjeni dogodek, dogodek na dotik itd. (15)

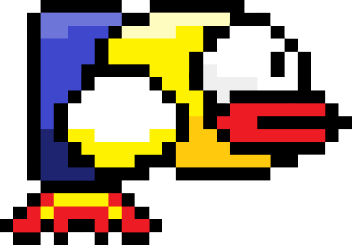
# IZDELAVA IGRE

## NAČRTOVANJE IZDELAVE

Bistvo igre Space Bird je enak igri Flappy Bird. Igra vsebuje ptico in par cevi. Igralec nadzira ptico tako da klika kamor koli na ekran, ta pa se pomakne po ekranu navzgor. Na ptico ves čas deluje privlak proti spodnjemu delu zaslona oziroma nanj deluje gravitacija. Ptici se v bistvu spremenijo le koordinate y, koordinate x pa ostanejo enake. Za videz, da se ptica premika v desno poskrbijo cevi, katera se premikajo v levo. Cevi se generirajo preden pridejo v eter zaslona. Igralec se poskuša izmikati cevem, če pa mu to ne uspe, izgubi igro. Če ptico izgubi iz zaslona na spodnjem ali zgornjem delu enako izgubi. Za vsakem uspešnem izogibu cevi, igralec pridobi točko. Tekom igre jih igralec poskuša nabrati čim več. Na Sliki 4 moder krog predstavlja ptico, dva temno rdeča podolgovata kvadrata pa dve cevi. Zgornja cev se drži zgornjega dela zaslona, spodnja pa spodnjega.

Slika : Idejna skica Space Bird

Sama igra lahko poteka in deluje povsem ne prekinjajoče in pravilno, četudi so v igri uporabljeni le geometrijski liki. Nekoliko je celo zaželeno , da se med programiranjem v grafičnem vmesniku sprva uporabijo liki, saj tako lažje razumemo delovanje igre, še zlasti pri začetnih programerjih. Za izkušnjo igranja igre pa je vendarle boljše poustvariti boljše grafike. Kasneje sem geometrijske like zamenjal s slikami, katere sem sam izdelal in sicer v programu Slikar in 3D-slikar. Samo risanje slik sem se lotil v programu Slikar. Za dokončno določanje širine in višine slike ter odstranitvijo ozadja pa sem slike uspel dokončati v programu 3D-slikar. Slika 8 in Slika 9 so spodnja in zgornja cev, Slika 6 in Slika 5 so dve ptici in obe imata na hrbtu raketni pogon za lažje premikanje po vesolju. Samo ena ptica vsebuje plamen saj se bodo slike ves čas menjavale kot majhna animacija delovanja raketnega pogona. Ozadje v igri je preprosto zvezdno nebo, lahko ga vidite na Sliki 7.



Slika : Ptica

Slika : Ptica s plamenom



Slika : Zgornja cev

Slika : Spodnja cev

Slika : Ozadje igre(Zvezdno nebo)

## PROGRAMIRANJE

### PREDSTAVITEV RAZREDOV PROGRAMA V JAVI

Program je razdeljen na več razredov. Vsak razred opravlja svojo nalogo, nekateri pa se povezujejo oziroma dedujejo. Glavni razred je MainActivity, v katerem program prične svojo pot. V njem se vzpostavijo vse osnovne lastnosti za pravilni zagon in kasnejše delovanje aplikacije. Glavni programski del igre pa poteka v razredu UprizoritevIgre. Razred Konstante vsebuje pomembne konstante v igri, razred OsnovniRazred vsebuje enostavne metode, ki so potrebne za kreiranje, nastavljanje in brisanje različnih lastnosti v igri. Razreda Ptica in Cevi, sta podrazreda razreda OsnovniRazred.

#### Razred MainActivity

Razred MainActivity se sam generira že s stvaritvijo android projekta. V njem poteka glavni del kode in s tem tudi povezovanje s XML datoteko. Razred vsebuje različne lastnosti, ki so klicana iz datoteke XML. To so lastnosti txt\_tocka in txt\_tocke\_konec tipa TextView, rl\_konec tipa RelativeLayout, privatna lastnost ui tipa UprizoritevIgre, ki sem jo ustvaril sam v datoteki XML, in lastnost gumb\_start tipa Button. Vse lastnosti lahko vidite v Kodi 1.

public static TextView *txt\_tocka*, *txt\_tocke\_konec*;  
public static RelativeLayout *rl\_konec*;  
private UprizoritevIgre ui;  
public static Button *gumb\_start*;

Koda : Lastnosti razreda MainActivity

Začetek igre se zažene v metodi onCreate(), katera vsebuje vse ostale pomembnejše klice razredov in metod. V Kodi 2 lahko takoj po deklariranju metode opazimo del kode, katera odstrani vse prejšnje privzete nastavitve aplikacije. Po poizvedbi te kode zaslon v bistvu postane prazen v katerem lahko lažje brez večjih ovir igro ustvarjamo naprej. S komentarjem sem v kodi tudi označil kje se začne pridobivanje dimenzij posameznega zaslona. Dimenzije pametnega telefona se shranijo v razred Konstante katerega bom kasneje opisal. Funkcija setContentView(R.layout.activity\_main) vzpostavi povezavo s XML dokumentom activity\_main. Vse deklaracije lastnosti razreda spodaj, se deklarirajo s metodo findViewById()in bodo vsebovale vse kar vsebujejo lastnosti iz datoteke XML. Metoda setOnClickListener() vrne true če zazna klik na gumb in igra se prične.

@Override

protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 this.getWindow().setFlags(WindowManager.LayoutParams.*FLAG\_FULLSCREEN*,WindowManager.LayoutParams.*FLAG\_FULLSCREEN*);  
 *//Pridobivanje dimenzije zaslona (sirina in visina)* DisplayMetrics dm = new DisplayMetrics();  
 this.getWindowManager().getDefaultDisplay().getMetrics(dm);  
 Konstante.*ZASLON\_SIRINA* = dm.widthPixels;  
 Konstante.*ZASLON\_VISINA* = dm.heightPixels;  
  
 setContentView(R.layout.*activity\_main*);  
  
 *txt\_tocka* = findViewById(R.id.*txt\_tocka*);  
 *txt\_tocke\_konec* = findViewById(R.id.*txt\_tocke\_konec*);  
 *rl\_konec* = findViewById(R.id.*rl\_konec*);  
 *gumb\_start* = findViewById(R.id.*gumb\_start*);  
 ui = findViewById(R.id.*ui*);  
 *gumb\_start*.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 ui.setStart(true);  
 *txt\_tocka*.setVisibility(View.*VISIBLE*);  
 *gumb\_start*.setVisibility(View.*INVISIBLE*);  
 }  
 });  
 *rl\_konec*.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 *gumb\_start*.setVisibility(View.*VISIBLE*);  
 *rl\_konec*.setVisibility(View.*INVISIBLE*);  
 ui.setStart(false);  
 ui.reset();  
 }  
 });  
}

Koda : Metoda onCreate() v razredu MainActivity

#### Razred Konstante

Razred vsebuje dve najpomembnejši lastnosti v programu, katera vrednost se nastavi že v razredu MainActivity. Lastnost ZASLON\_SIRINA je podatkovnega tipa Integer in je javna kot tudi lastnost ZASLON\_VISINA, ki pa je tudi tipa Integer. Obe lastnosti predstavljajo konstante. Glej Kodo 3.

public class Konstante {  
 public static int *ZASLON\_SIRINA*;  
 public static int *ZASLON\_VISINA*;  
}

Koda : Razred Konstante

#### Razred OsnovniRazred

Razred vsebuje 6 lastnosti. Dve lastnosti sta tipa int in dve tipa float in dva objekta tipa Rect in tipa Bitmap.

Objekt Rect vsebuje štiri celoštevilčne koordinate tipa int za pravokotnik. Pravokotnik predstavljajo koordinate njegovih 4 robov (levo, zgoraj, desno, spodaj). Do teh polj je mogoče neposredno dostopati. Za pridobitev širine in višine pravokotnika uporabite metodi width() in height(). Rect se uporablja za risanje pravokotnikov na zaslon (12). Vsaka točka v koordinatnem sistemu se imenuje piksel. Slikovna pika je sestavljena iz bitov, ki predstavljajo barvo te slikovne pike. Vse, kar je narisano v androidu, je Bitmap. Bitmap primer lahko ustvarimo bodisi z uporabo Bitmap razreda, ki ima metode, ki nam omogočajo manipulacijo s slikovnimi pikami v 2d koordinatnem sistemu, bodisi lahko ustvarimo Bitmap iz slike ali datoteke ali vira z uporabo BitmapFactory razreda (16).

protected float x, y;  
protected int sirina, visina;  
protected Rect rect;  
protected Bitmap bm;

Koda : Lastnosti razreda OsnovniRazred

Konstruktor razreda poskrbi za nastavljanje vrednosti različnih lastnosti razreda (Glej Kodo 5) OsnovniRazred je nadrejen razred razredu Ptica in Cevi. Oba razreda nastavita lastnosti razreda OsnovniRazred v svojem konstruktorju.

public OsnovniRazred(float x, float y, int sirina, int visina) {  
 this.x = x;  
 this.y = y;  
 this.sirina = sirina;  
 this.visina = visina;  
}

Koda : Konstruktor razreda OsnovniRazred

Razred vsebuje enostavne metode za prejem in spremembo vsake lastnosti posebej in skupaj. Vse metode so napisane spodaj v Kodi 6.

public float getX() { return x; }  
  
public void setX(float x) { this.x = x; }  
  
public float getY() { return y;}  
  
public void setY(float y) { this.y = y; }  
  
public int getSirina() { return sirina; }  
  
public void setSirina(int sirina) { this.sirina = sirina; }  
  
public int getVisina() { return visina; }  
  
public void setVisina(int visina) { this.visina = visina; }  
  
public Bitmap getBm() { return bm; }  
  
public void setBm(Bitmap bm) { this.bm = bm; }  
  
public Rect getRect() {  
 return new Rect((int)this.x, (int)this.y, (int)this.x+(int)this.sirina, (int)this.y+(int)this.visina);  
}  
  
public void setRect(Rect rect) { this.rect = rect; }

Koda : Osnovne metode razreda OsnovniRazred

#### Razred Ptica

Razred Ptica obsega vse pomembne metode in lastnosti ptice, od animacij ptice do gravitacije. Spodaj v Kodi 7 je lastnost arrBm, ki je ArrayList tipa Bitmap, stetje, yZamah in idTrenutniBitmap, ki je indeks arrBm trenutnega Bitmap-a, so tipa int, padec pa je tipa float. Vse lastnosti so privatne in se nastavijo v konstruktorju.

private ArrayList<Bitmap> arrBm = new ArrayList<>();  
private int stetje, yZamah, idTrenutniBitmap;private float padec;  
public Ptica() {  
 this.stetje = 0;  
 this.yZamah = 5;  
 this.idTrenutniBitmap = 0;  
 this.padec = 0;  
}

Koda : Lastnosti in konstruktor razreda Ptica

V naslednjem delu kode (Koda 8), metoda draw() na zaslon nariše Bitmap in njegovo premikanje s gravitacijo, kar nas privede do naslednje metode gravitacija(). Omenjena funkcija poskrbi za konstanto padanje ptice s lastnostjo padec ki se vsakih 10 milisekund poveča za 0'6, lastnost y pa spremeni koordinate ptice, da je videz gravitacije viden tudi nam. Metoda setArrBm() prilagodi Bitmap po širini in višini sliko, ki smo jo določili za ptico.

public void draw(Canvas canvas){  
 gravitacija();  
 *//risanje bitmap na canvas* canvas.drawBitmap(this.getBm(), this.x, this.y, null);  
}  
  
private void gravitacija() {  
 this.padec += 0.6;  
 this.y += this.padec;  
}  
  
public ArrayList<Bitmap> getArrBm() {  
 return arrBm;  
}  
  
public void setArrBm(ArrayList<Bitmap> arrBm) {  
 this.arrBm = arrBm;  
 *//we need to scale those bitmaps to the height and width of the bird* for (int i=0; i<arrBm.size(); i++){  
 this.arrBm.set(i, Bitmap.*createScaledBitmap*(this.arrBm.get(i),this.sirina,this.visina, true));  
 }  
}

Koda : Metode draw(), gravitacija() in setArrBm() v razredu Ptica

getBm() je metoda, ki dobi določen Bitmap iz objekta bm, ki smo ga deklarirali v razredu OsnovniRazred. Metoda je v tem primeru namenjena dvema animacijama. Prva animacija neprestano zamenjuje dve sliki ptic vsakič, ko se zaslon osveži, torej vsakih 10 milisekund. Animacija da videz delujočega raketnega pogona, ki ga ima ptica na hrbtu. Koda je označena s komentarjem v Kodi 9.

public Bitmap getBm() {  
 stetje++; if(this.stetje == this.yZamah){ *//OGENJ NA PTICO (menjava dveh slikic)*

for (int i=0; i<arrBm.size(); i++){  
 if(i == arrBm.size()-1){  
 this.idTrenutniBitmap = 0;  
 break;  
 }else if (this.idTrenutniBitmap == i){  
 idTrenutniBitmap = i+1;  
 break;  
 }  
 }  
 stetje = 0;  
 }  
 *//ANIMACIJA PTIČA PRI LETU* if(this.padec < 0){  
 Matrix m = new Matrix();  
 m.postRotate(-25);  
 return Bitmap.*createBitmap*(arrBm.get(idTrenutniBitmap), 0, 0, arrBm.get(idTrenutniBitmap).getWidth(), arrBm.get(idTrenutniBitmap).getHeight(), m, true);  
 }else if (padec >= 0){  
 Matrix m = new Matrix();  
 if(padec < 70)  
 m.postRotate(-25+(padec\*2));  
 else  
 m.postRotate(45);  
 return Bitmap.*createBitmap*(arrBm.get(idTrenutniBitmap), 0, 0, arrBm.get(idTrenutniBitmap).getWidth(), arrBm.get(idTrenutniBitmap).getHeight(), m, true);  
 }  
  
 return this.arrBm.get(idTrenutniBitmap);  
  
 */\*try to get first bitmap of the bitmap array  
 return this.getArrBm().get(0);\*/*}

Koda : Animacije ptice v razredu Ptica

#### Razred Cevi

Lastnost, ki jo razred vsebuje je hitrost, ki pomeni hitrost premikanja cevi iz desne proti levi, proti ptici. V konstruktorju deklariramo lastnosti iz razreda OsnovniRazred in hitrost prilagodimo zaslonu.

public static int *hitrost*;  
public Cevi(float x, float y, int sirina, int visina){  
 super(x, y, sirina, visina);  
 *hitrost* = 10\*Konstante.*ZASLON\_SIRINA*/1080;}

Koda : Lastnosti in konstruktor razreda Cevi

Metoda draw() ima isti lastnost, kot pri razredu Ptica, kjer nariše na zaslon ptico, tukaj pa nariše cev ter ji spreminja x koordinato za videz premikanja ptice v desno. Funkcija nakljucnoY() generira naključno število do števila, ki smo ga dobili z delitvijo višine Bitmap-a in štiri. To število je y koordinata slike cevi. setBm() prilagodi Bitmap sliki cevi. Koda opisanih metod je podana na naslednji strani kot Koda 11.

public void draw(Canvas canvas){  
 this.x -= *hitrost*; canvas.drawBitmap(this.bm, this.x, this.y, null);  
}

public void nakljucnoY(){  
 Random r = new Random();  
 this.y = r.nextInt((0 + this.visina/4) + 1) - this.visina/4;  
}

@Override  
public void setBm(Bitmap bm) {  
 this.bm = Bitmap.*createScaledBitmap*(bm, sirina, visina, true); *//sets the Bitmap's height and width equal to the height and width of the ceve*}

Koda : Metode draw(), nakljucnoY() in setBm() v razredu Cevi

#### Razred UprizoritevIgre

Navedeni razred je najpomembnejši takoj za razredom MainActivity, ker vsebuje vse pomembne povezave do drugi razredov, ter je edini razred na katerega se sklicuje XML datoteka. V njem se ustvarijo cevi in ptica, poteka glavni del igre, nariše ter obnavlja vse na zaslon pametnega telefona, začne in konča igro, zazna, če je bil zaslon telefona dotaknjen in igro začne znova. Lastnost ptica je tip razreda Ptica , arrCev je seznam cevi iz razreda Cevi ter lastnosti tipa int sumCev, ki določi število generiranih cevi, razmaky za razmak med zgornjo in spodnjo cevjo, tocke in še lastnost start tipa boolean, ki določa začetek ali konec igre. Razred vsebuje še lastnosti handler tipa Handler in runnable tipa Runnable.

Handler je vrsta sporočil. Poslano sporočilo sčasoma obdela, tako da pokliče svojo metodo zagon in mu sporočilo posreduje. Ker se bodo ti klici izvajali vedno po vrstnem redu sporočil, prejetih v isti niti, vam omogoča serializacijo dogodkov (8). Runnable vmesnik je primarna predloga za kateri koli predmet, ki naj bi ga izvedela nit. Določa eno samo metodo run(), ki naj bi vsebovala kodo, ki jo izvede nit (9).

Konstruktor deklarira tocke, start, ustvari ptico in cevi in preko Runnable vmesnika vsebuje metodo run(). Prej navedena metoda run() vsebuje metodo invalidate() katere naloga je obnavljanje metode draw(), v kateri je bistvo celotne igre.

private Ptica ptica;  
private Handler handler;  
private Runnable runnable;  
private ArrayList<Cevi> arrCev;  
private int sumCev, razmaky;  
private int tocke;  
private boolean start;  
public UprizoritevIgre(Context context, @Nullable AttributeSet attrs) {  
 super(context, attrs);  
 tocke = 0;  
 start = false;  
 ustvariPtico();  
 ustvariCev();  
  
 handler = new Handler();  
 runnable = new Runnable() {  
 @Override  
 public void run() {  
 invalidate(); *//unvalidate() to update method draw* }  
 };  
}

Koda : Lastnosti in konstruktor razreda UprizoritevIgre

Funkcija ustvariCev() postavi generiranje cevi na število 6 in razmak med spodnjo in zgornjo cevjo na zmnožek razmerja zaslona s 400. Metoda inicializira seznam cevi in prične z zanko, ki polovico cevi nariše zgoraj, drugo polovico pa spodaj. Zgornjim cevem doda Sliko 9: Zgornja cev, spodnjim pa Sliko 8: Spodnja cev. Ko cev preide skozi zaslon se v seznamu cevi zbriše, na začetku pa se ustvari nova cev. Metoda je zapisana v Kodi 13.

private void ustvariCev() {  
 sumCev = 6;  
 razmaky = 400\*Konstante.*ZASLON\_VISINA*/1920;  
 arrCev = new ArrayList<>(); *//ArrayList cevi* for (int i=0; i<sumCev; i++){  
 if (i<sumCev/2){  
 this.arrCev.add(new Cevi(Konstante.*ZASLON\_SIRINA* + i\*((Konstante.*ZASLON\_SIRINA* + 200\*Konstante.*ZASLON\_SIRINA*/1080)/(sumCev/2)),  
 0, 200\*Konstante.*ZASLON\_SIRINA*/1080, Konstante.*ZASLON\_VISINA*/2));  
 this.arrCev.get(this.arrCev.size()-1).setBm(BitmapFactory.*decodeResource*(this.getResources(), R.drawable.*cev2*));  
 this.arrCev.get(this.arrCev.size()-1).nakljucnoY();  
 }else{  
 this.arrCev.add(new Cevi(this.arrCev.get(i - sumCev/2).getX(), this.arrCev.get(i - sumCev/2).getY() + this.arrCev.get(i - sumCev/2).getVisina() + this.razmaky,  
 200\*Konstante.*ZASLON\_SIRINA*/1080, Konstante.*ZASLON\_VISINA*/2));  
 this.arrCev.get(this.arrCev.size()-1).setBm(BitmapFactory.*decodeResource*(this.getResources(), R.drawable.*cev1*));  
 }  
 }  
}

Koda : Metoda ustvariCev() v razredu UprizoritevIgre

Objekt ptica se inicializira v metodi ustvariPtico() in se ji dodeli vrednost širine in višine glede na razmerje zaslona mobilnega telefona ter koordinate njene prvotne lege pred začetkom igre. Inicializira se nov seznam Bitmap v katerega se shranijo obe sliki ptic.

private void ustvariPtico() {  
 ptica = new Ptica();  
 ptica.setSirina(100\*Konstante.*ZASLON\_SIRINA*/1080);  
 ptica.setVisina(100\*Konstante.*ZASLON\_VISINA*/1920);  
 ptica.setX(100\*Konstante.*ZASLON\_VISINA*/1080);  
 ptica.setY(Konstante.*ZASLON\_VISINA*/2-ptica.getVisina()/2);  
 ArrayList<Bitmap> arrBm = new ArrayList<>();  
 arrBm.add(BitmapFactory.*decodeResource*(this.getResources(), R.drawable.*ptica1*));  
 arrBm.add(BitmapFactory.*decodeResource*(this.getResources(), R.drawable.*ptica21*));  
 ptica.setArrBm(arrBm);  
}

Koda : Metoda ustvariPtico() v razredu UprizoritevIgre

Začetni if stavek v metodi draw() je poglaviten saj določi ali se bo igra izvajala ali ne. Predpostavimo, da je vrednost start-a true in da igra poteka normalno. Najprej bo program na zaslon narisal ptico s vsemi lastnostmi iz razreda Ptica. Nadaljuje se z zanko, ki se bo ponavljala tolikokrat kolikor je vrednost lastnosti sumCev. Naslednji if stavek s pomočjo metode intersect() preveri ali se ptica in spodnja ali spodnja cev prekrivajo oziroma, če ležijo na istih koordinatah. Če je odgovor na ta pogojni stavek pritrdilni, se hitrost cevi, ki se premikajo v levo nastavi na 0, ter se izpiše doseženo število točk, ter dve besedili: ''KONEC IGRE'' in ''Klikni za ponovno igro…''. Enak rezultat dobimo, če ptica preseže y koordinate v pozitivno in negativno vrednost. Naslednji pogojni stavek preveri, če se je ptica varno izognila cevem in če vrne vrednost true, se vrednost točk poveča za eno in ta vrednost se izpiše na zaslon. Zadnji if stavek ugotovi, če je cev prečkala zaslon in če je stavek pravilen ustvari na desni strani nov par cevi s naključno začetno višino. Zadnji stavek v ne ustavljeni igri nariše novo cev na določenem indeksu na seznamu. Če je vrednost prvega if stavka false se ptica generira na točno določenih koordinatah vendar gravitacija na njo še vedno deluje, tako da metoda setPadec() to prepreči in ustvari učinek samostojnega letenja ptice. Stavek handler.postDelayed(runnable, 10) povzroči posodabljanje na vsakih 10 milisekund vmesnika runnable in posledično tudi metode run(). Metoda draw() je zapisana v Kodi 15.

public void draw(Canvas canvas){  
 super.draw(canvas);  
 if (start){  
 ptica.draw(canvas);  
 for (int i=0; i<sumCev; i++){  
 if (ptica.getRect().intersect(arrCev.get(i).getRect()) || ptica.getY()-ptica.getVisina()<0 || ptica.getY()>Konstante.*ZASLON\_VISINA*){  
 Cevi.*hitrost* = 0;  
 MainActivity.*txt\_tocke\_konec*.setText(MainActivity.*txt\_tocka*.getText());  
 MainActivity.*txt\_tocka*.setVisibility(*INVISIBLE*);  
 MainActivity.*rl\_konec*.setVisibility(*VISIBLE*);  
 }  
 if(this.ptica.getX()+this.ptica.getSirina() > arrCev.get(i).getX()+arrCev.get(i).getSirina()/2  
 && this.ptica.getX()+this.ptica.getSirina() <= arrCev.get(i).getX()+arrCev.get(i).getSirina()/2+  
 Cevi.*hitrost* && i<sumCev/2){  
 tocke++;  
 MainActivity.*txt\_tocka*.setText(""+tocke);  
 }  
 if(this.arrCev.get(i).getX() < -arrCev.get(i).getSirina()){  
 this.arrCev.get(i).setX(Konstante.*ZASLON\_SIRINA*);  
 if(i < sumCev/2){  
 arrCev.get(i).nakljucnoY();  
 }else{  
 arrCev.get(i).setY(this.arrCev.get(i - sumCev/2).getY() + this.arrCev.get(i - sumCev/2).getVisina() + this.razmaky);  
 }  
 }  
 this.arrCev.get(i).draw(canvas);  
 }  
 }else{  
 if (ptica.getY()>Konstante.*ZASLON\_VISINA*/2){  
 ptica.setPadec(-15\*Konstante.*ZASLON\_VISINA*/1920);  
 }  
 ptica.draw(canvas);  
 }  
 handler.postDelayed(runnable, 10); *//update every 10 miliseconds*}

Koda : Metoda draw() v razredu UprizoritevIgre

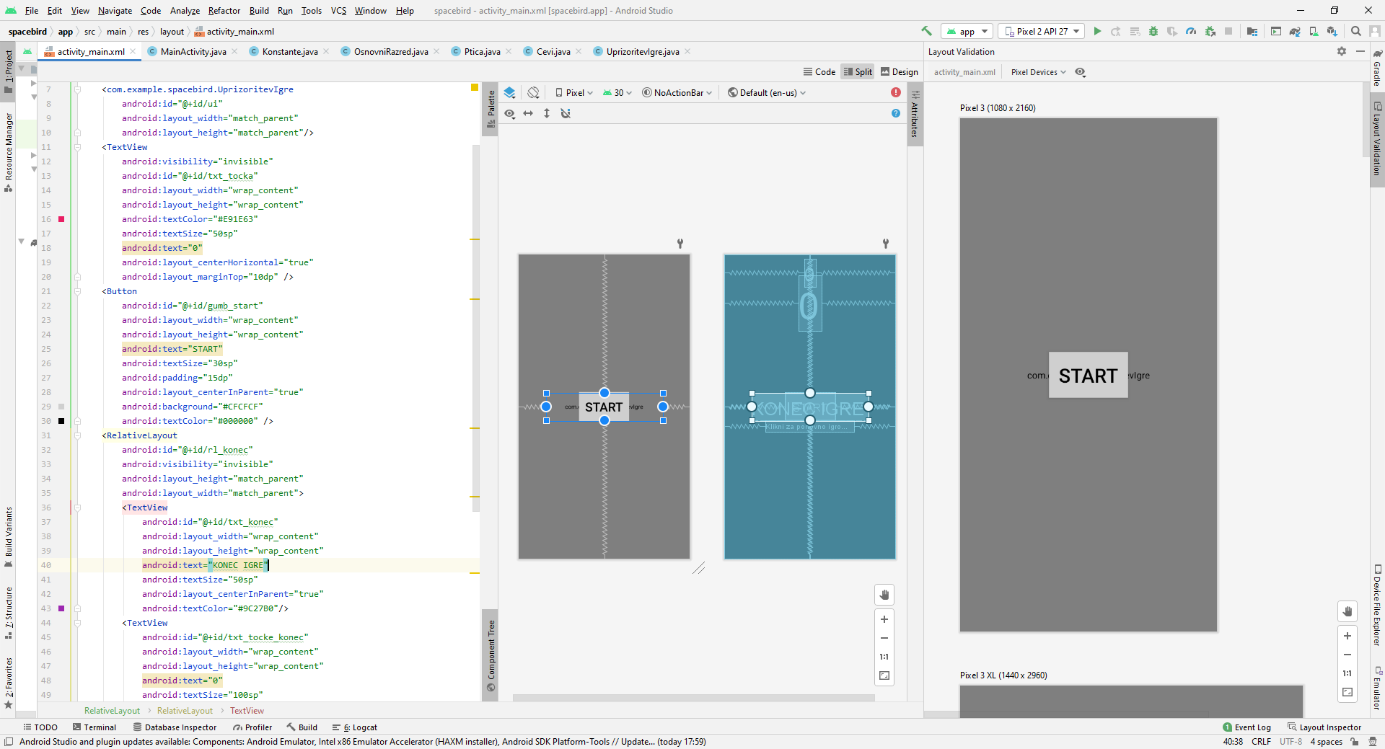
Dana funkcija onTouchEvent(MotionEvent event) je privzeta funkcija, ki zazna dotik prsta na zaslon ter posledično izvede določen stavek ali ukaz. V izseku kode v Kodi 16 pa s dotikom prstom na zaslon povzročimo letenje ptice. V metodi isStart() ugotovimo kakšna je vrednost spremenljivke start, v metodi setStart() pa jo nastavimo. V zadnji metodi reset() se število doseženih točk na vrhu zaslona ponastavi na nič in tudi vrednost lastnosti tocke se nastavi na nič, ustvarita pa se ponovno ptica in cev.

@Override  
public boolean onTouchEvent(MotionEvent event) {  
 if(event.getAction() == MotionEvent.*ACTION\_DOWN*){  
 ptica.setPadec(-15);  
 }  
 return true;  
}  
public boolean isStart() { return start; }  
public void setStart(boolean start) { this.start = start; }  
public void reset() {  
 MainActivity.*txt\_tocka*.setText("0");  
 tocke = 0;  
 ustvariPtico();  
 ustvariCev();  
}

Koda : Metode onTouchEvent(), isStart(), setStart() in reset() v razredu UprizoritevIgre

### PROGRAMIRANJE V XML

XML je označevalni jezik, podobno kot HTML, ki se uporablja za opisovanje podatkov. XML oznake niso vnaprej določene. Določiti moramo svoje oznake. XML kot sam je dobro berljiv tako človeku kot stroju. Prav tako je prilagodljiv in enostaven za razvoj. V Androidu uporabljamo XML za oblikovanje postavitev in ker je dokaj enostaven jezik ne otežuje postavitve in urejanja. Spodaj na sliki je je prikazan Arduino Studio v pogledu za urejanje XML datoteke, kateri omogoča več vrst urejanja, preko kode ali pa direktno preko grafičnega vmesnika.



Slika : Android Studio v pogledu urejanja XML datoteke

Za postavitev v XML sem se odločil za RelativeLayout zaradi lažje postavitve besedila in gumba na zaslon. Androidov RelativeLayout omogoča določitev, kako bodo podrejeni pogledi postavljeni med seboj. Položaj vsakega pogleda se lahko določi glede na elemente sorojence ali glede na nadrejenega starševskega pogleda. RelativeLayout vsebuje vse ostale poglede in lastnosti pogledov. Vsebuje tri svoje lastnosti in sicer layout\_width="match\_parent" in layout\_height="match\_parent", ki postavita velikost pogleda na velikost zaslona. Lastnost background="@drawable/ozadje" pa nastavi za ozadje sliko iz mape drawable. V pogledu <com.example.spacebird.UprizoritevIgre> poteka igra. Pogled sem ustvaril sam in vsebuje svoj id. Pogled je enako velik kot RelativeLayout. Prvi TextView pogled se imenuje txt\_tocka in je namenjena prikazovanju točk med igro. Pogled Button se imenuje gumb\_start in je namenjen začetku igre. Znotraj prejšnjega RelativeLayout pogleda sem naredil še enega v namen pri izgubi igre. Vsebuje tri tekstovne poglede in sicer txt\_konec, ki napiše besedilo ''KONEC IGRE'', tocke\_txt\_konec prikaže doseženo število točk. Zadnje besedilo, ki vsebuje niz ''Klikni za ponovno igro…'', nima svojega id naslova ampak je kar postavljen pod txt\_konec s ukazom layout\_below=''@id/txt\_konec''. Vsak pogled pa še vsebuje nekaj grafičnih atributov za lepšo in bolj urejeno postavitev. Spodaj v kodi so navedeni vsi pogledi s vsemi svojimi lastnostmi.

<RelativeLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="@drawable/ozadje"  
 xmlns:android=<http://schemas.android.com/apk/res/android>>

<com.example.spacebird.UprizoritevIgre  
 android:id="@+id/ui"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"

/>  
 <TextView  
 android:visibility="invisible"  
 android:id="@+id/txt\_tocka"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:textColor="#E91E63"  
 android:textSize="50sp"  
 android:text="0"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="10dp"

/>  
 <Button  
 android:id="@+id/gumb\_start"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="START"  
 android:textSize="30sp"  
 android:padding="15dp"  
 android:layout\_centerInParent="true"  
 android:background="#CFCFCF"  
 android:textColor="#000000"

/>  
 <RelativeLayout  
 android:id="@+id/rl\_konec"  
 android:visibility="invisible"  
 android:layout\_height="match\_parent"  
 android:layout\_width="match\_parent">

<TextView  
 android:id="@+id/txt\_konec"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="KONEC IGRE"  
 android:textSize="50sp"  
 android:layout\_centerInParent="true"  
 android:textColor="#9C27B0"

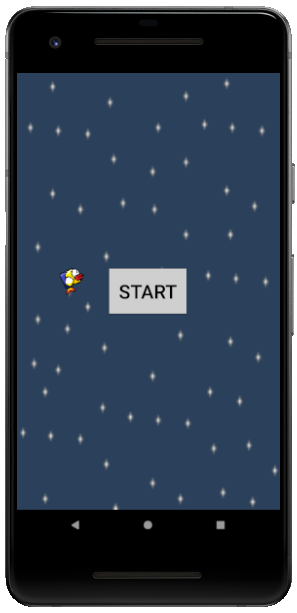
/>

<TextView  
 android:id="@+id/txt\_tocke\_konec"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="0"  
 android:textSize="100sp"  
 android:textColor="#673AB7"  
 android:layout\_marginTop="50dp"  
 android:layout\_centerHorizontal="true"  
 />  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Klikni za ponovno igro..."  
 android:textSize="20sp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_below="@id/txt\_konec"  
 android:textColor="#ffffff"  
 />  
 </RelativeLayout>  
</RelativeLayout>

Koda : Datoteka activity\_main.xml

## TESTIRANJE

Ob zagonu igre se odpre aplikacija kot prikazuje Slika 12. Ptica leti samodejno do pritiska gumba ob njeni desni strani na sredini zaslona. Ob pritisku na gumb, igralec lahko začne igrati. Slika 11prikazuje med igranjem igre in Slika 13 izgubo igre. Samo igro sem med razvojem testiral v Androidovi virtualni napravi, ki sem jo nastavil v Android Studiu. Za virtualno napravo sem uporabljal pametni telefon Pixel 2 API 27. Ko sem igro končal in mi je na virtualnem telefonu delovala pravilno sem si igro naložil na moj osebni telefon, kjer deluje brezhibno. Prosil sem še sošolce sorodnike in znance za namestitev te igre. Nikomur ni igra delovala nepravilno.



Slika : Posnetek zaslona med igranjem igre

Slika : Posnetek zaslona pred igranjem igre

Slika : Posnetek zaslona ob izgubi igre

# ZAKLJUČEK

Pri pisanju te seminarske naloge sem se bolje seznanil z delovanjem časovnika v Javi in kako ga izkoristiti. Bolje sem se tudi seznanil z dogodkom dotika (angl. onTouchEvent), ki je zelo uporaben pri izdelavi različnih mobilnih aplikacijah. Spoznal sem se s z novim integriranim programskim okoljem Android Studio, ter se seznanil s razvijanjem aplikacij za operacijski sistem Android. Spoznal sem, da lahko, ko programiraš, narediš program, tak kakršnega si si zamislil, le če se zavzameš za to.

Igro bi lahko še izboljšal tako, da bi dodal težavnost igre, ki bi se aktivirala po točno določenem številu točk, dodal bi lahko premikajoče se zvezde v ozadju. Za določeno število doseženih točk, bi igralec dobil medaljo, ter najboljše število točk bi se shranilo v aplikacijo. Dodajanje zvokov in pesmi v ozadju bi igralno izkušnjo precej izboljšalo. Naj enostavnejša nadgradnja igre bi bila izboljšava slike za ptico, cevi in ozadja.

# ZAHVALA

Zahvaljujem se mentorju, ki mi je odgovor na vsa moja vprašanja in mi svetoval pri izdelavi seminarske naloge.

Zahvaljujem se svoji družini za vse spodbude in pomoč in za iskrena mnenja pri testiranju, še posebej pa se zahvaljujem svojemu bratu Janu, ki mi je pomagala pri izvedbi te seminarske naloge.

Zahvaljujem se svojim sošolcem, ki so mi svetovali in me spodbujali.

Zahvaljujem se še vsem ostalim, ki so mi kakorkoli pomagali pri izdelavi igre in seminarske naloge.

David Gole

# VIRI IN LITERATURA

1. **Jin, Dal Yong.** *Mobile Gaming in Asia: Politics, Culture and Emerging Technologies.* Burnaby : Springer, 2016. str. 6-7. 978-94-024-0824-9.

2. **Ozmec, Janez.** *Izdelava igre za pametni telefon.* Telekomunikacije, Univerza v Mariboru. Maribor : s.n., 2014. str. 1, Diplomsko delo.

3. **Wikipedia.** Android Studio. [Elektronski] 6. april 2021. [Navedeno: 2. april 2021.] https://en.wikipedia.org/wiki/Android\_Studio.

4. **Android Developers.** Meet android studio. [Elektronski] 24. februar 2021. [Navedeno: 2. april 2021.] https://developer.android.com/studio/intro.

5. **Mesojedec, Uroš in Fabjan, Borut.** *Java2 temelji programiranja.* Ljubljana : Pasadena, 2004. 961-6361-30-9.

6. **Davison, Andrew.** *Killer Game Programming in Java.* Sebastopol : O'REILLY, 2005. str. 1. 978-0-596-00730-0.

7. **Android Developers.** Activity. *developers.* [Elektronski] 17. marec 2021. [Navedeno: 4. april 2021.] https://developer.android.com/reference/android/app/Activity.

8. —. Handler. *developers.* [Elektronski] 24. februar 2021. [Navedeno: 5. april 2021.] https://developer.android.com/reference/android/os/Handler.

9. —. Runnable. [Elektronski] 27. december 2019. [Navedeno: 4. april 2021.] https://developer.android.com/reference/java/lang/Runnable.

10. **Morgun, Ivan.** How to work with Bitmap in Android. *proft.me.* [Elektronski] 2. avgust 2018. [Navedeno: 5. april 2021.] https://en.proft.me/2017/08/2/how-work-bitmap-android/#.

11. **Android Developers.** View. [Elektronski] 17. marec 2021. [Navedeno: 4. april 2021.] https://developer.android.com/reference/android/view/View.

12. —. Rect. [Elektronski] 24. februar 2021. [Navedeno: 4. april 2021.] https://developer.android.com/reference/android/graphics/Rect.

13. —. Relative Layout. *developers.* [Elektronski] 18. november 2021. [Navedeno: 6. april 2021.] https://developer.android.com/guide/topics/ui/layout/relative.

14. **Abhishek.** TextView With Example In Android Studio. *AbhiAndroid.* [Elektronski] [Navedeno: 6. april 2021.] https://abhiandroid.com/ui/textview.

15. —. Button Tutorial With Examples In Android Studio. *AbhiAndroid.* [Elektronski] [Navedeno: 6. april 2021.] https://abhiandroid.com/ui/button.

# PRILOGE

# STVARNO KAZALO

3D-slikar 11

Android 2, 1, 2, 3, 4, 6, 7, 26, 27, 29

Android Studio 2, 3, 4, 6, 27

AndroidManifest 3, 7

animacija 11, 17

Bitmap 5, 6, 14, 15, 16, 17, 18, 20, 29

boolean 5, 6, 18, 19, 22

Button 4, 7, 8, 9, 12, 13, 23, 24, 29

cev 1, 11, 18, 19, 21, 22

class 14

findViewById 4, 13

Flappy Bird 2, 11

float 14, 15, 16, 18

Handler 5, 18, 19, 29

Igra 2, 11

int 4, 6, 14, 15, 16, 17, 18, 19, 21

**intersect** 6, 21

**invalidate** 6, 19

Java 2, 3, 4, 5, 6, 29

Konstante 1, 12, 13, 14, 18, 19, 20, 21

koordinate 11, 14, 16, 20, 21

Kotlin 2, 6

MainActivity 1, 7, 12, 13, 14, 18, 21, 22

**manifests** 3

Metoda 2, 1, 4, 5, 13, 14, 16, 17, 18, 19, 20, 21, 22

Ozadje 1, 11

pametni telefon 2, 26, 29

private 13, 16, 19, 20

protected 13, 14

Ptica 1, 2, 12, 15, 16, 17, 18, 19, 20, 21, 26

public 4, 5, 6, 12, 13, 14, 15, 16, 17, 18, 19, 21, 22

Razred 2, 1, 5, 12, 14, 15, 17, 18

Rect 6, 14, 15, 29

RelativeLayout 7, 12, 23, 24, 25

return 15, 16, 17, 22

run 5, 19, 21

Runnable 5, 18, 19, 29

setOnClickListener 4, 13

Slikar 2, 11

Space Bird 2, 1, 2, 11

**SPACE BIRD** 1

static 6, 12, 14, 18

TextView 7, 8, 9, 12, 23, 24, 29

void 4, 5, 6, 13, 15, 16, 18, 19, 20, 21, 22

XML 1, 3, 4, 6, 7, 9, 12, 13, 18, 22, 23

1. XML je kratica za Extensible Markup Language [↑](#footnote-ref-1)