Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Denis Korinšek

Mobilna igra za učenje programiranja v osnovni šoli

magistrsko delo

študijski program druge stopnje
pedagoško računalništvo in informatika

Mentor: izr. prof. dr. Zoran Bosnić
Somentorica: doc. dr. Irena Nančovska Šerbec

Ljubljana, 2018

Zahvala

Zahvaljujem se mentorju izr. prof. dr. Zoranu Bosniću in mentorici doc. dr. Ireni Nančovski Šerbec za vso strokovno pomoč, nasvete in usmerjanje pri izdelavi magistrskega dela. Zahvalil bi se mami Albini, očetu Robertu in Karin za vse spodbudne besede ter podporo skozi študijska leta. Hvala!

Denis Korinšek, 2018

"The more I learn, the more I realize how much I don't know."

— Albert Einstein

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
GLAT	games for learning algorithmic thinking	igre za učenje alg. razmišljanja
IDE	integrated development environment	integrirano razvojno okolje
SEGAN	serious games network	mreža resnih iger
VCS	version control systems	sistem za kontrolo različic
JVM	java virtual machine	javansko virtualno okolje
XML	extensible markup language	razširljiv označevalni jezik
MVC	model view controler	model pogled kontroler
MVP	model view presenter	model pogled povezovalnik
MVVM	model view viewmodel	model pogled model pogleda

Povzetek

Naslov: Mobilna igra za učenje programiranja v osnovni šoli

Že v zgodnjem otroštvu se začnemo učiti skozi igro. Danes se vse pogosteje učimo s tehnologijami in tudi z računalniškimi igrami. V okviru magistrske naloge smo izdelali mobilno igro, katere cilj je naučiti otroke osnovnih programerskih konceptov in skozi igro razvijati njihove kognitivne spretnosti ter računalniško mišljenje. Izdelali smo metaokolje s katerim se učenci naučijo uporabo konceptov zaporedja ukazov, zank in pogojnih stavkov. S podporo programiranja v več programskih jezikih otrok spozna različne zapise programov. Koncepte lahko zapiše na način gnezdenja blokov, diagrama poteka ali psevdokode v izbranem programskem jeziku. Igra je namenjena učencem v 2. vzgojno-izobraževalnem obdobju. Okolje je predstavljeno s slikovnimi prikazi, animacijami in zvokom, zato je igra primerna tudi za najmlajše otroke. Igro smo analizirali in testirali na vzorcu štirih otrok v njihovem prostem času. Analize so pokazale, da so se otroci med učenjem zabavali in da jim je bil najtežji preskok iz vizualnega programiranja v programsko kodo.

Ključne besede

učenje programiranja za otroke, učenje skozi igro, mobilne igre

Abstract

Title: Mobile game for learning programming in elementary school

From birth, children learn through play. Due to advancements in computer science and technology we are increasingly learning with help of those same technologies, which are very much applicable in primary schools, as well. For our master thesis, we developed a mobile game. The goal is to teach children the basic programming concepts and develop their cognitive abilities through play and computer logic thinking. We developed the meta environment which allows children to learn the concepts of a command sequence, loops and if statements. Featuring support for multiple programming languages, a child is able to learn the differences between them. A child is able to write in chain block, flowchart or pseudocode modes in the programming language of choice. The game is designed for children in the second triennium of primary school. The environment is presented with pictures, animations and sound, making it suitable for even younger children. We analysed and tested the game on four children outside the primary school. Analysis showed that children had fun while learning and that the jump from visual programming to programming with code presented the biggest challenge.

Keywords

teaching programming in an elementary school, learning through games, mobile games

Poglavje 1

Uvod

1.1 Motivacija

Zaradi razvoja računalništva in tehnologije se v šolah vedno več poučuje ob pomoči računalniških iger, ki otroku predstavljajo zanimiv in učinkovit način učenja. Učitelji uporabijo računalniške igre med učnimi urami ali jih priporočijo učencem za utrjevanje. Otroci se zelo radi učijo skozi igre, kjer pridobijo poglobljeno znanje na podlagi sodelovanja, logičnega povezovanja in neposredne interakcije v igri. Papert [1] je ugotovil, da se otroci bolje naučijo stvari, ki jih občutijo, vidijo in zaznajo, zato je vizualno programiranje boljše od tradicionalnega učenja z "ex cathedra" podajanjem računalniških učnih vsebin.

V ospredje torej vedno bolj prihajajo računalniška znanja in tehnologije, zato se ponekod otroci učijo računalniških veščin že v prvem razredu osnovne šole. Učenje otrok osnov programiranja in računalniških konceptov (v razredu) navadno poteka z igrami ali računalniškimi aplikacijami. Igre popestrijo učenje, z uporabo igrifikacije, ki uporablja otrokov način razmišljanja in mehanizme igre pa vplivamo na aktivno učenje in sposobnost reševanja problemov [2]. Veliko obstoječih aplikacij je v tujih jezikih ali ne ustrezajo učiteljevemu načrtu oz. jih učitelji ne morejo uporabiti pri poučevanju. Največkrat so računalniške aplikacije za učenje programiranja namenjene

samostojnemu učenju ali so komercialno usmerjene, zato jih učitelji ne morejo uporabiti med učnimi urami. Večina je omejena na učenje enega programskega jezika ali sintakso tega in ne toliko na razumevanje konceptov programiranja.

V nadaljevanju bomo predstavili predlagano rešitev problema in zakaj je pomembno, da otrok pri učenju programiranja spozna različne zapise programov.

1.2 Pregled rešitve

Glavni prispevek magistrskega dela je implementacija mobilne igre za operacijski sistem Android. Namen igre je učenje konceptov programiranja, kjer učenec z igranjem razume in zna uporabiti zaporedje ukazov, pogojni stavek in zanko. Igra z ustvarjanjem kode v različnih programskih jezikih igralca uči reševanja problemov in logičnega razmišljanja, kjer spoznava, da lahko z različnimi zapisi programa dobi enake rešitve. Različni zapisi programa igralcu razvijejo občutek za uporabo različnih pristopov pri reševanju problemov, razmišljanju, načrtovanju, iskanju in povezovanju rešitev. Igra omogoča prilagajanje vrstnega reda sklopov programiranja z bloki, diagrama poteka in psevdokode. Posledično je primernejša za uporabo v razredu, saj si učenec lahko prilagodi zaporedje sklopov, ki se jih želi najprej naučiti. Ciljna skupina so učenci 2. vzgojno-izobraževalnega obdobja v osnovni šoli, ki bi skozi igro samostojno pridobili predznanje ali utrdili obravnavano učno snov ter tako poglobili znanje.

Podroben opis različnih zapisov programov, ki jih igra podpira, najdemo v poglavju 3.

1.3 Struktura dela

V poglavju 2 bomo naredili pregled sorodnih del in iger ter se seznanili z izobraževalnimi igrami in definicijami učenja skozi igro. V Poglavju 3 pred-

stavimo razvoj mobilne igre in njeno načrtovanje s teoretičnega, didaktičnega in implementacijskega pogleda. Predstavili bomo potek igre in igralnost ter njeno testiranje in analizo delovanja. V poglavju 4 bomo našteli didaktične in implementacijske doprinose, glavne rezultate in predlagali nekaj idej za izboljšavo igre.

Poglavje 2

Pregled področja

V poglavju bomo spoznali, zakaj je učenje z igro že od nekdaj učinkovit način učenja otrok, in pregledali, zakaj so posledično izobraževalne računalniške igre primerne za uporabo v osnovnih šolah. Predstavili bomo podobne igre in sorodna dela, ki se ukvarjajo z reševanjem podobne problematike kot naše magistrsko delo.

2.1 Učenje skozi igro

Igra predstavlja dejavnost, s katero otrok že od rojstva spoznava svet okoli sebe, se uči in razvija. Pripomore k otrokovemu kognitivnemu, socialnemu, emocionalnemu in gibalnemu razvoju, zato ima pomembno vlogo pri poglobljenem učenju, saj s svojimi idejami, občutki in odnosi lažje razume in povezuje pridobljeno znanje [3]. Z odraščanjem otroci zamenjajo svobodne oblike igranja z igrami, ki vsebujejo pravila. Izobraževalna igra običajno potrebuje za uspešno učenje določena pravila.

Učenje je proces pridobivanja in razvijanja znanja, zato so v preteklosti razvili mnoge teorije učenja. Konstruktivizem in konstrukcionizem sta učni teoriji, na katerih temelji naša igra. Teorija konstruktivizma predpostavlja, da pridobivanje znanja ni zgolj prenos znanja in pomnjenje podatkov, temveč konstruiranje, odkrivanje in samostojno razlaganje podatkov, s ka-

terimi učenec ustvarja nova znanja na podlagi predhodnih znanj, izkušenj, stališč, vrednot, osebnostnih lastnosti in okolja [6].Konstrukcionizem je Papert [7] opisal kot učenje, pri katerem se otroci učijo s tem, ko nekaj delajo in so aktivni ob reševanju nekega problema.

Ker obstaja več teorij učenja, so razvili mnogo različnih definicij. Salen in Zimmerman [4] definirata igro kot sistem, kjer igralec sodeluje v umetnem konfliktu, določenem s pravili, posledica česar je kvantitativen rezultat. Torej, kot rezultat izobraževalnih iger pričakujemo pridobivanje znanja, ki ga bomo skušali ovrednotiti kvantitativno. Igra je po definiciji izobraževalna takrat, kadar so v njej zavestno skriti učni cilji [5].

2.2 Izobraževalne računalniške aplikacije

Danes ljudje vsakodnevno uporabljajo računalnike in mobilne naprave v prostem času ali za delo. Otroci so rojeni v digitalnem svetu in že zelo zgodaj uporabljajo mobilne naprave oz. tehnologijo, vendar se ne učijo osnovnih konceptov programiranja in veščin računalništva. Učenje skozi računalniške igre lahko štejemo kot novo obliko poučevanja, ki je predmet mnogih raziskav s področja učenja. Pomembna značilnost uporabe iger kot učnega pripomočka je, da učitelj ve, kaj naj bo naučeno [18].

Spletno izobraževanje in mobilno učenje (m-učenje) sta podpodročji e-izobraževanja, ki sta nastali zaradi razvoja tehnologij [8]. Za potrebe m-učenja v osnovnih šolah še ni veliko aplikacij in iger, zato smo z razvojem mobilne igre razširili možnosti uporabe in raznolikost učnih pripomočkov ter s tem učiteljem ponudili dodaten material za kakovostno poučevanje. Otroci vsakodnevno uporabljajo mobilne naprave, kar lahko učitelji izkoristijo kot prednost, saj lahko ustvarijo interaktivne in zanimive učne izkušnje, ki ponujajo avtentične in kontekstno usmerjene učne pogoje [12]. Učenje z uporabo mobilnih naprav je lahko naključno oz. neurejeno in potrebuje vodenje. Ponuja se nam nov način učenja kjer koli in kadar koli tudi zunaj šole. Finska je leta 2016 v kurikulum vključila učenje programiranja v prvem razredu osnovne

šole in s tem postavila nove temelje osnovne izobrazbe [9]. Izpostavili so problem, da na tem področju primanjkuje uradnih informacij in materialov za poučevanje, ki bi jih učitelji lahko uporabili, kot so izobraževalne računalniške igre.

Dobre izobraževalne računalniške igre morajo biti problemsko zasnovane in predstavljene kot zgodbe, ki predstavljajo prilagojen model resničnosti/domišljije, kjer igralec prevzame vlogo in aktivno rešuje nek problem [5]. Uporabo iger, razmišljanje ob igranju in razvoj igralne tehnike, ki jih uporabljamo z namenom povečanja lastnih vložkov ter sodelovanja igralcev pri reševanju problema, imenujemo igrifikacija [10]. Glavni problem v modernem izobraževanju je pomanjkanje sodelovanja in motivacije učencev, prisotnih v izobraževalnem procesu. Učitelji preizkušajo nove tehnike in pristope, pri tem pa je pomembna igrifikacija, ki z uporabo elementov igre in učnega procesa prispeva k povečanju motivacije in sodelovanja v aktivnosti [11].

V zvezi z možno uporabo izobraževalnih računalniških iger v učnem procesu je Evropska unija podprla nekaj projektov. V projektu GLAT, ki se je začel izvajati oktobra 2017 v okviru programa Erasmus+ in bo trajal dve leti, se ukvarjajo s spodbujanjem vključevanja računalniškega in algoritmičnega razmišljanja, veščin reševanja problemov, logike in ustvarjalnosti v dnevno poučevanje različnih predmetov od prvega do četrtega razreda osnovne šole. Program vključuje stokovna usposabljanja učiteljev, ki jim predstavljajo inovativne učne metodologije in uporabo informacijskih tehnologij, s poudarkom na uporabi izobraževalnih strategij učenja z igrami. Program se izvaja na Hrvaškem, zato bodo pripravljene učne e-vsebine v hrvaškem kot tudi v angleškem jeziku, uporabile pa jih bodo lahko tudi druge evropske države [14].

Glavni cilj projekta SEGAN [29] je ustvariti konzorcij za izmenjavo idej in izkušenj, povezanih z izobraževalnimi igrami. Projekt povezuje učitelje, študente, raziskovalce in strokovnjake s področja izobraževalnih iger. V okviru projekta je ustvarjenih veliko novih iger, člankov, pripomočkov ipd., iz tega pa lahko trdimo, da projekt pripomore k reševanju izpostavljenega problema o pomanjkanju materialov.

2.3 Sorodna dela

Predstavili bomo nekaj sorodnih del, kjer so se ukvarjali z raziskovanjem in izdelavo izobraževalnih računalniških iger.

Učenje konceptov programiranja z igro Labirint [13], izdelano v okolju Scratch, so analizirali na delavnici, organizirani na Fakulteti za računalništvo in informatiko, v okviru tedna programiranja leta 2017. V raziskavo so bili vključeni otroci, stari od 8 do 10 let, in nekateri med njimi so že imeli nekaj znanja o programskem jeziku Scratch. Izvedli so kvantitativno raziskavo s pomočjo programa DrScratch, s katerim so zbirali rezultate o napredku otrokovih veščin programiranja, povezanih z računalniškim razmišljanjem. Rezultati prikazujejo, kako se sinhronizacija dogodkov in abstraktno razmišljanje med otroki razlikujeta in da je skozi proces poučevanja z igro večina otrok dosegla visoko razvojno stopnjo znanja. Iz rezultatov je razvidno, da je delavnica pomagala pri razvoju veščin računalniškega mišljenja.

Wilson in Moffat [15] sta naredila raziskavo, kjer so 21 učencev, starih 8 let, pri pouku računalništva 8 tednov učili programiranja s sistemom Scratch. Ugotovila sta, da so se otroci hitro naučili osnov programiranja. Po vsaki uri so ugotavljali, kako se otroci počutijo med delom, pri čemer so ugotovili, da se je večina ob učenju zabavala. Izvedli so 3 različne teste na isti množici učencev, ki so pokazali, da se je uspešnost skozi vsak test izboljšala. Trdijo, da sta za uspešen učni načrt programiranja enako pomembni pozitivno čustveno stanje učenca kot vsebina.

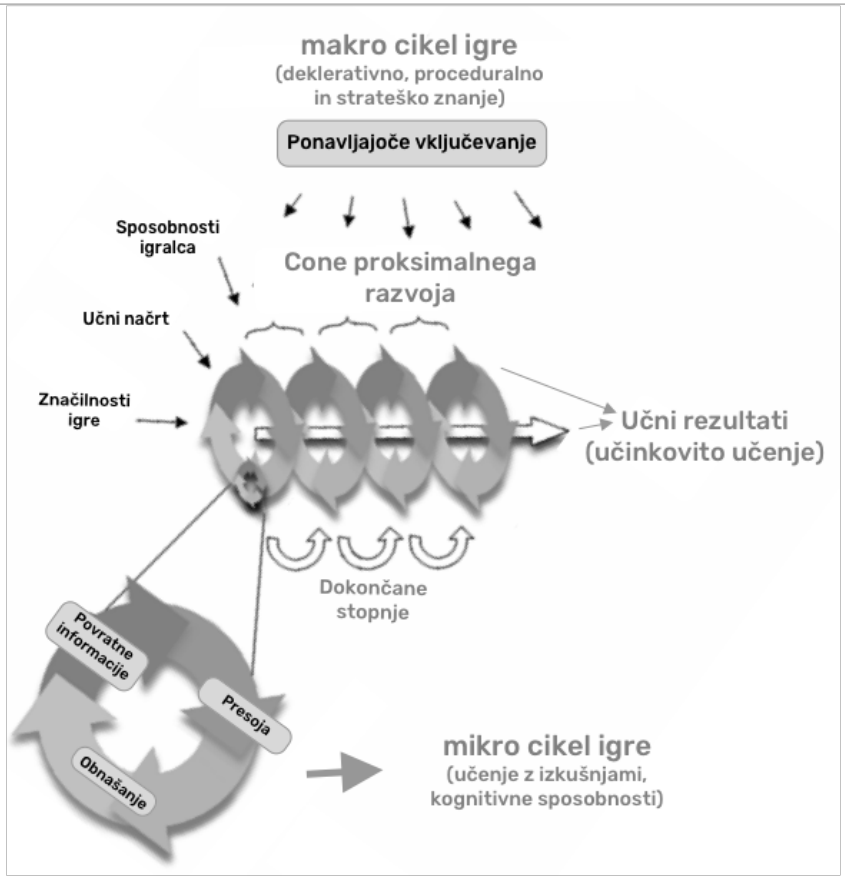
Hinds in sod. [16] so izdelali izobraževalno igro RunJumpCode, katere namen je učenje osnovnih konceptov programiranja s programskim jezikom C#. Cilj igre je izboljšati znanje programiranja z zanimivimi izzivi in ugankami, kjer lahko igralec spreminja nastavitve, kar mu daje ustvarjalno svobodo za dokončanje stopenj. Statistiko igranja so za vsako igro beležili skozi mobilno aplikacijo, ki so jo razvili kot dodatek k igri. Raziskava, pri kateri je sodelovalo 17 učencev srednje šole, je pokazala, da so igro dobro sprejeli, se ob igranju zabavali in izboljšali znanje programiranja.

Brennanova [17] je že leta 2007 raziskovala možnosti vključitve Scratcha

v kurikulum K-12. Dodana vrednost Scratcha je dopuščanje svobode za raziskovanje različnih ustvarjalnih metod. Uporabniku omogočajo, da se povezuje z ostalimi, kot so starši, učitelji ali osebe z znanjem, ki nudijo pomoč v razčlenjevanju problema in razmišljanju pri logičnih izzivih. Avtorica trdi, da je učiteljeva vloga pomembna pri uvajanju Scratcha v razred, zato se ukvarja z razumevanjem, kako učitelji poučujejo, in razvija materiale, ki pomagajo pri doseganju ciljev. Poudarja dva glavna izziva, s katerima se srečujejo učitelji. Prvi opisuje iskanje ravnovesja med pomočjo učencem in dopuščanjem svobode pri ustvarjalnih procesih, drugi pa uporabo socialnega učenja v šolski kulturi, kjer ocenjevanje in ocena predstavljata osnovo za napredek posameznega učenca.

Pivec [19] izpostavlja dejstvo, da je glavna značilnost izobraževalne igre zabrisana vsebina s karakteristikami igre, kjer je igra motivacijsko naravnana in učenec ponavlja cikle igranja. S ponavljanjem igranja igre kot rezultat interakcije dosežemo, da je učenčevo vedenje usmerjeno na čustvene in kognitivne odzive. Avtorica predstavi tudi model mikro ponavljalnega učenja, povezanega s stopnjami v igri. Slika 2.1 opisuje, kako sposobnosti in izkušnje igralca vplivajo na premagovanje izzivov v igri in doseganje dobrih rezultatov učenja. Model prikazuje vključevanje učnega načrta in karakteristik igre kot pomembnega elementa igre za doseganje učnih rezultatov. Proces pridobivanja znanja in veščin je prikazan kot ponavljajoč cikel, ki spodbuja pridobivanje učnih rezultatov in znanja skozi čas in sposobnosti skozi izkušnje [19].

Frederick in Watson [21] sta izdelala igro za poučevanje, kjer si igralec sam izdela okolje in objekte, ki jih uporablja skozi igro. Premikanje objekta avtomobila in potek igre igralec upravlja z zankami in s pogojnimi pravili. Učenec lahko preveri pravilnost njegove logike z opazovanjem avtomobila na dirkalni stezi, kjer napačno število krogov ali nepričakovano delovanje pomeni napako v programski kodi. Ugotavljajo, da imajo učenci pozitiven odziv na učenje skozi igro in da je učenje popolnih začetnikov programiranja skozi tehniko učenja skozi igro učinkovita za učenje razumevanja osnovnih konceptov.



Slika 2.1: Cikli ponavljanja skozi učenje z igro (povzeto po [20])

Avtorja [21] definirata, da proces učenja konceptov programiranja vključuje štiri faze:

- sprejemanje, ki je lahko v obliki pasivnega sprejemanja predavanj ali branja učnega gradiva;
- vizualiziranje, ki uporablja vizualne pripomočke, ki pomagajo učencu razumeti in graditi miselni model koncepta;
- utrjevanje znanja, kjer sta učencu omogočena ponavljanje in vadba konceptov;
- uporaba znanja in sinteza, kjer so učenci vključeni v izdelavo aplikacij in uporabijo naučeno znanje.

Korteling in sod. [22] so ocenjevali kakovost prenosa znanja, pridobljenega z igranjem igre, kjer so po določenem času igranja poskusni skupini zastavljali izzive iz realnega okolja, dokler niso dosegli vnaprej določene stopnje znanja. Ukvarjali so se s poučevanjem veščin, ki jih potrebujemo v službi za konkretne realne primere. Čas, ki je potreben, da poskusna skupina uspešno doseže stopnjo, primerjajo s časom, ki ga potrebuje kontrolna skupina, ki je bila učena samo z realnimi nalogami. Takšen način ocenjevanja prenosa znanja lahko uporabimo tudi pri izobraževalnih mobilnih igrah.

$$\%T = \frac{T_c - T_e}{T_c} \times 100\%. \tag{2.1}$$

kjer:

T_c = čas potreben, da kontrolna skupina doseže določeno stopnjo znanja
T_e = čas po učenju z igro, ki ga poskusna skupina porabi za doseženo stopnjo znanja
$\%T$ = odstotek prenosa znanja

Iz enačbe (2.1) [22] je razvidno, da kadar je %T nekega izobraževalnega programa 100 %, ni potreben dodaten trening s strani poskusne skupine za dosego enake stopnje znanja.

2.4 Podobne igre

Pregledali in analizirali bomo nekaj aplikacij, ki se ukvarjajo z reševanjem podobnega problema. Predstavili bomo njihove prednosti in slabosti, na podlagi katerih smo zasnovali načrt naše igre.

S programskim jezikom Scratch lahko otroci sami izdelajo interaktivne vsebine, kot so igre, zgodbe, animacije in simulacije. Brennanova in Resnick [28] predstavita glasbeni video Fireflies, simulacijo Countries in igro 10 Levels, ki so jih izdelali učenci osnovnih šol. Takšno okolje, kjer učenci sami ustvarjajo učne vsebine, je vsekakor pomembno, saj s tem dodatno poglobijo znanje in pomagajo sovrstnikom pri učenju. Ugotovila sta, da so vsi otroci z izdelavo omenjenih vsebin razvijali računalniško mišljenje. Skozi leta raziskovanja in preučevanja aktivnosti v Scratchu sta razvila definicijo računalniškega razmišljanja, ki mora vsebovati računalniške koncepte, računalniško prakso in računalniško perspektivo.

Kidlo Coding [23] je mobilna igra, namenjena učenju osnovnih konceptov programiranja. Igralec se z igranjem vlog, kot so gasilec, zobozdravnik, izdelovalec smutijev, s pokanjem balonov in drugimi, uči reševanja problemov, spodbuja spomin in izboljšuje logično razmišljanje. Skozi igro se otroci naučijo zaporedja ukazov, zank, funkcij in uporabe seznamov. Igra ni primerna za učenje v osnovni šoli, saj v brezplačni različici ne ponuja dovolj stopenj, skozi katere bi učenca naučili uporabe omenjenih konceptov. Licenca igre je omejena na enega igralca, kar pomeni, da ne podpira nakupa skupinske licence, ki bi jo lahko uporabili v šoli. Učni materiali, predstavitev konceptov in celota igre so izdelani kakovostno in bi bili z razširitvijo brezplačnih funkcionalnosti igre primerni za poučevanje.

Bit by Bit [24] je mobilna igra, izdelana na več mobilnih platformah, ki

deluje po principu premikanja po puzlih. Igralec za vsako polje, po katerem se premikajo igralni agenti, določi spremembo smeri ali akcijo, ki se zgodi na omenjenem polju. Igro so testirali v osnovni šoli in jo zasnovali na podlagi kurikuluma. Pomanjkljivost igre je, dda njeno igranje nima neposredne povezave s koncepti programiranja, pripomore pa pri predvidevanju dogodkov, reševanju problemov, analitičnem in logičnem razmišljanju. Igra je brezplačna, vendar bi težavnost igranja in upravljanja igre lahko predstavljali težavo za mlajše otroke.

Mobilna igra Coding Planets [25] je igra s tematiko vesolja, kjer igralec z ukazi usmerja robota. Skozi igro otrok spozna osnovne koncepte programiranja, kot so zaporedja ukazov, funkcije in zanke. Igra je brezplačna in primerna za uporabo v šoli, vendar je njena pomanjkljivost, da uporablja prostor 3D. Slabost tega je, ker najmlajši otroci še nimajo dobre prostorske predstave in jim lahko prostorska predstava rotacije in premiki predstavljajo težavo pri reševanju problema.

Igra Algorithm City [26] uči uporabe zaporedja ukazov, funkcij in zank z usmerjanjem lika in zbiranjem zlatnikov. Igra je razdeljena na štiri sklope, kjer je prvi namenjen izobraževanju, preostala pa uporabi naštetih konceptov. Igra je brezplačna, vendar omogoča prilagoditve in menjavo igralnih likov, ki jih lahko dokupimo. Slabost igre je podobna kot pri igri Coding Planets [25], saj uporablja prostor 3D.

Mimo [27] je mobilna aplikacija, ki je po naši oceni primernejša za uporabo, kadar ima otrok že nekaj predznanja programiranja in želi svoje znanje nadgraditi. S testom uporabe smo ugotovili, da je kompleksnost aplikacije primernejša za starejše uporabnike, ki se želijo naučiti programiranja. Uporabnik lahko v aplikaciji popolnoma prilagodi igralno okolje kurikulumu in vaje želenim ciljem ter se nauči uporabe več kot 15 programskih jezikov. V sami aplikaciji lahko izdela igre, aplikacije, spletne strani ipd.

Poglavje 3

Razvoj mobilne igre

V poglavju bomo predstavili izdelavo mobilne igre. Najprej se bomo osredotočili na njene učne cilje in načrtovanje učnih materialov. Predstavili bomo tehnologije, razvojno arhitekturo ter teoretično in slikovno predstavili delovanje igre. Omenili bomo nekaj posebnosti, ki smo jih uporabili pri implementaciji. Predstavili bomo analizo igre in rezultate testiranja.

3.1 Učni cilji

Učni cilji zaključenih izobraževalnih enot opisujejo pomembna in bistvena znanja, ki usmerjajo izobraževanje in jih je učenec dosegel skozi izobraževanje. Preveri se jih lahko na koncu učne ure, modula, semestra, z nalogami, ki preverjajo razumevanje in uporabo pridobljenega znanja [30]. Učne cilje na najvišji stopnji konkretnosti, kjer prehajamo od splošnega do konkretnega znanja, imenujemo operativni učni cilji [31]. Učenec z igranjem razvite mobilne igre v izobraževalnem procesu pridobi različna znanja in veščine, ki jih v naslednjem odstavku predstavimo kot operativne učne cilje naše naloge.

Učenec:

- zna s programsko kodo predstaviti preprosto rešitev;

- zna uporabiti osnovne koncepte programiranja zaporedja ukazov, funkcije in zanke;
- razume programsko kodo, ki reši dan problem;
- zna uporabiti znane programske rešitve za nove problemske situacije;
- zna problem razdeliti na manjše podprobleme;
- zna uporabljati in razume različne zapise programov, kot so blokovsko programiranje, diagram poteka in psevdokoda;
- pozna in razume različne načine reševanja istega problema;
- zna napisati preprost program.

3.2 Načrtovanje

Za ustvarjanje uspešne izobraževalne igre Pivec [32] priporoča uporabo naslednjih elementov pri načrtovanju igre:

- določiti potek učenja;
- izbrati najustreznejši pedagoški pristop za poučevanje;
- problemsko zasnovati nalogo;
- reševanje problema zapakirati v navidezno resničnost;
- dodelati podrobnosti, kjer je poudarek na odpravi morebitnih pomanjkljivosti igre, ki bi lahko zmotile igralca;
- vključiti izobraževalne vsebine in pomoč, če jo igralec potrebuje;
- narediti preslikavo učnih aktivnosti v dejavnosti znotraj igre;
- vključiti preslikavo učnih konceptov v koncepte znotraj igre.

Pri načrtovanju igre smo upoštevali omenjene elemente, kjer smo najprej določili koncepte, ki so predmet poučevanja. Najprej mu predstavimo zaporedja ukazov, kjer bo igralec v prostoru 2D z osnovnimi ukazi, kot so premik igralnega agenta oz. figure gor, dol, levo in desno do končne rešitve problema. Ko igralec osvoji koncept in zna uporabljati zaporedja ukazov, mu predstavimo zanke. Sreča se s problemom, bi bila rešitev z zaporedji ukazov predolga, in zato spozna nov koncept zank, kjer igralec določi, koliko ponovitev zaporedij ukazov, ki jih je definiral, se bo izvedlo. Učenec se nauči uporabe na različnih problemih, kjer znanje povezuje s predhodno pridobljenim. Tretjega koncepta pogojnih stavkov se nauči, ko zna uporabljati zaporedja ukazov in zanke. Namen koncepta je, da lahko igralec določi pogoj, ob katerem bo igralni lik naredil drugačno akcijo kakor sicer. V tej fazi igranja igralec uporablja kombinacije vseh konceptov in išče rešitve za zastavljen problem poti na trenutni stopnji. Razlog za takšen vrstni red izhaja iz načina igranja igre, kjer igralni lik premikamo s preprostimi ukazi, in je pogojni stavek smiseln za uporabo šele, ko se lik ponavljajoče premika, in lahko definiramo spremembo smeri, kadar igralni lik pride na polje, ki ga igralec nastavi kot pogoj.

Navidezna resničnost igralca postavi v vlogo izmišljenega lika z imenom Leo, ki živi v pravljičnem svetu. Otrok se uči različnih zapisov programskih jezikov, ki jih v igri predstavimo na način programiranja:

- vizualno ali blokovsko programiranje,
- sestavljanje diagrama poteka,
- psevdokoda.

Blokovsko ali vizualno programiranje predstavlja način programiranja, ki je podoben programskemu jeziku Scratch. Otrok poljubno sestavlja bloke z metodo povleci in spusti blok za ukaze, zanke in pogojne stavke. Programski jezik omogoča tudi, da različno implementira rešitev. To pomeni, da lahko samo s postavitvijo zaporedja ukazov dosežemo enako premikanje lika, kot če uporabimo koncepta zank in pogojnih stavkov. Z rešitvijo, ki nas s

premikanjem igralnega lika pripelje do polja za konec, uspešno dokončamo stopnjo. Namen in implementacijo stopenj bomo podrobneje predstavili v podpoglavju 3.5.

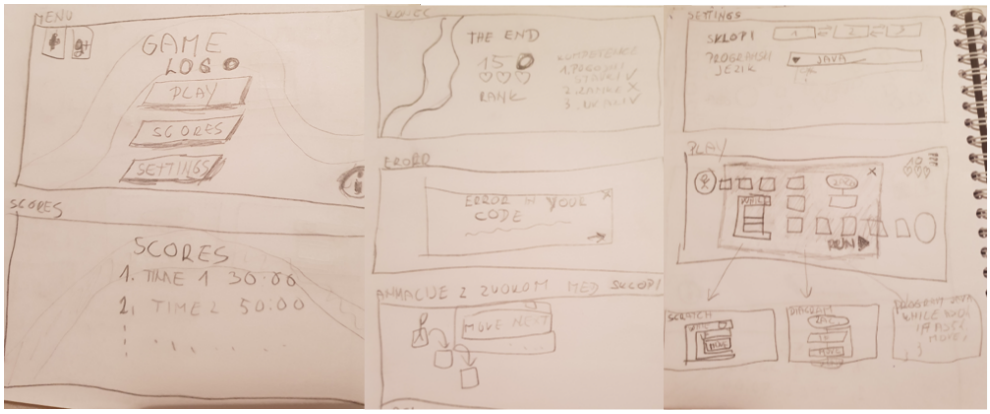
Premikanje lika z diagramom poteka predstavlja način programiranja, kjer z grafično predstavitvijo zaporedja operacij določimo, v kakšnem zaporedju in kako se bomo v igri premikali. Otrok spozna osnovne grafične simbole za predstavitev diagrama poteka, kot so:

- začetni in končni blok, ki označujeta začetek in konec programa;
- procesni blok je namenjen izvajanju operacij (v našem primeru otrok določi, kateri premik se bo na tem mestu zgodil);
- odločitveni blok, ki ga uporabimo, kadar se del programa razveji oz. lahko večkrat ponovi;
- usmerjene povezave med bloki, ki prikazujejo smer izvajanja programa.

Uporaba psevdokode je način programiranja, kjer otrok napiše program v izbranem programskem jeziku. V igri smo podprli programska jezika Kotlin in Python, ki sta si dovolj različna, da igralec razlikuje med različnimi zapisi oz. sintaksami. Podrobnejše delovanje in implementacijo bomo predstavili v podpoglavju 3.7.

Otroke motivira, če vidijo napredek in rezultate, saj lahko tako sami ocenijo napredovanje v znanju. Zato za vsako dokončano stopnjo prejmejo nagrado, ki predstavlja uspešnost odigrane stopnje. Igralec ima pregled nad doseženimi točkami za vsako stopnjo in možnost primerjanja rezultatov predhodno odigranih iger.

Pred začetkom implementacije igre smo načrt in idejo skicirali na papir, kot prikazuje slika 3.1. Tako smo razdelili razvoj na več delov oziroma funkcionalnosti, kjer je vsak del predstavljal svoj implementacijski sklop, saj smo tako lažje ocenili trajanje in kompleksnost pred razvojem projekta.



Slika 3.1: Izdelan načrt pred razvojem igre

3.3 Tehnologije in razvojno okolje

Igro smo razvili za operacijski sistem Android, ki ga je razvilo podjetje Google. Sistem je zasnovan na jedru Linux in spada med odprtokodne projekte, kar pomeni, da je programska koda javno dostopna za preučevanje in spreminjanje [33]. Po podatkih agencije StatCounter Global Stats [34] je Android najbolj razširjen operacijski sistem za mobilne naprave s 77-odstotnim deležem za razliko od konkurenčnega operacijskega sistema iOS, ki dosega 19 %. Zaradi visokega deleža uporabnikov smo se odločili, da aplikacijo razvijemo na omenjeni platformi, ki razvijalcem ponuja dobro podporo in dokumentacijo.

Za razvojno okolje smo uporabili Android Studio [35], ki je tudi uradno podprt IDE za načrtovanje in razvoj mobilnih aplikacij za operacijski sistem Android.

Večina mobilnih aplikacij za Android je razvitih v programskem jeziku Java, ki je bil do maja 2017 uradni jezik za razvoj na platformi. Google je takrat na konferenci predstavil nov uradni programski jezik Kotlin [36]. Pri razvoju smo upoštevali zadnje Googlove smernice in se odločili za razvoj v programskem jeziku Kotlin. Tudi otrokom smo pri programiranju s psevdokodo ponudili možnost izbire programiranja v Kotlinu, saj je vedno bolj

razširjen tudi za uporabno na drugih platformah.

Sistem za nadzor različic (VCS), ki omogoča lažji nadzor in pregled programske kode in smo ga uporabili pri razvoju, je Git. Prednost uporabe sistema je pregled nad starejšimi različicami kode in odkrivanje napak, vendar se bolj izrazi, kadar na projektu sodeluje več razvijalcev. GitHub je platforma, ki za delovanje uporablja Git in smo jo uporabili za javno gostovanje naše programske kode. S tem smo rešitev ponudili drugim in s tem pripomogli pri razvoju iger v programskem jeziku Kotlin, saj lahko razvijalci preučijo, ali lahko uporabijo ideje za reševanje podobnih problemov. Programska koda, razvita v okviru magistrskega dela, je dostopna na povezavi `https://github.com/korinsek/game`.

Gradle [38] je odprtokodni sistem za upravljanje in prevajanje programske kode, ki temelji na Javanskem virtualnem okolju (JVM) in združuje prednosti drugih sistemov v eni rešitvi. Vsak projekt za platformo Android privzeto uporablja sistem Gradle za ustvarjanje aplikacij v obliko, primerno za zagon na operacijskem sistemu. Dodatna funkcionalnost, ki smo jo uporabili pri delu na projektu, je učinkovito orodje za preprosto vključevanje in uporabo knjižnic, ki jih sistem samodejno prenese iz repozitorijev in vključi v projekt [39].

Za izvajanje asinhronih procesov v ozadju in nadzor nad njimi smo uporabili knjižnico RxJava [37], ki uporablja razširitve funkcijskega programiranja Reactive pri razvoju aplikacij. Pri knjižnici RxJava uporabljamo vzorce opazovalca (ang. observer patterns), kjer z operacijami nad tokom podatkov prilagajamo podatke. Uporabili smo jo za izvajanje zanke igre, ki skrbi za obdelavo vhodnih podatkov in interakcij, posodabljanje objektov ter izrisovanje.

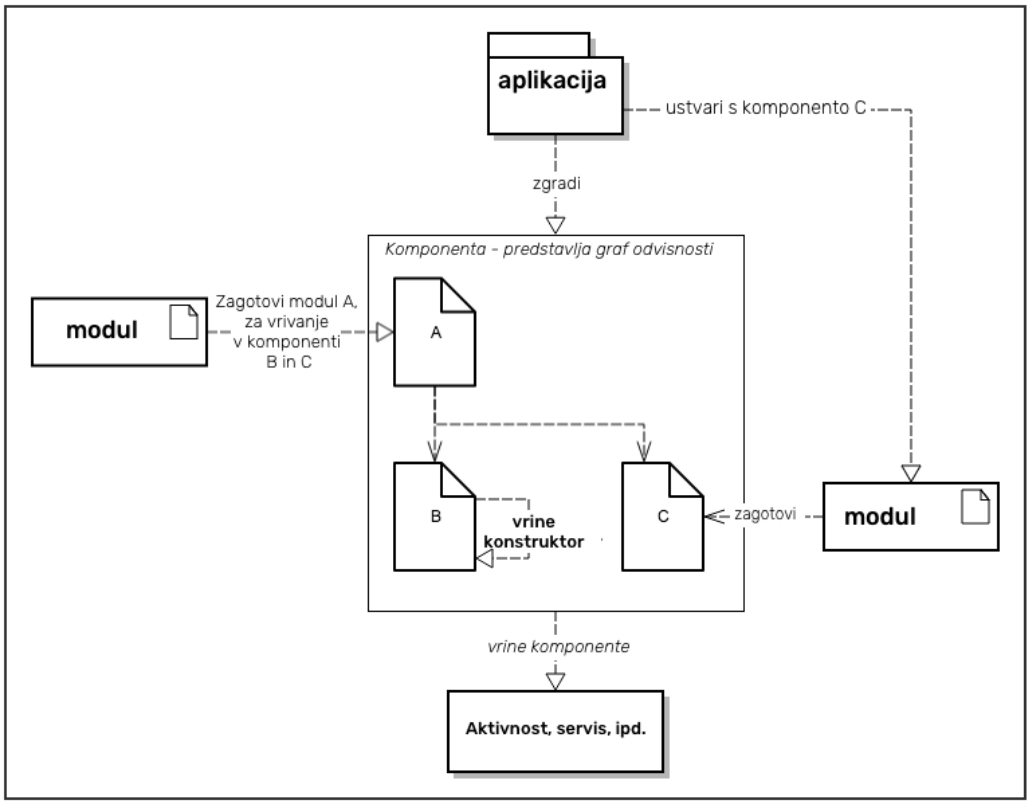
Vrivanje odvisnosti (ang. dependency injection) je tehnika pri razvoju programske opreme, ki izboljša vzdržljivost programske kode, saj je aplikacija razdeljena v več ločenih modulov, ki jih lahko vrinemo v odvisne objekte. Cilj je ločiti programsko kodo na smiselne ločene celote, kjer odvisen objekt ne spreminja vrinjene kode [40]. Uporabili smo ogrodje Dagger za vrivanje

odvisnosti, ki ga vzdržuje podjetje Google. Temelji na označevalnih anotacijah Jave (ang. Java annotation processors) in med prevajanjem opravi analizo ter preveri odvisnosti. Kot prikazuje slika 3.2, obstajajo trije koncepti, ki jih uporablja dagger:

- moduli so mehanizmi za zagotavljanje odvisnosti;
- komponente so povezava med odvisnimi objekti in moduli ter definirajo, kako so zagotovljene odvisnosti s strani modula vrinjene v objekte, ki potrebujejo odvisnost;
- odvisni objekti, ki uporabljajo anotacije, s katerimi zahtevamo odvisen vrinjen objekt [41].

Igro smo zasnovali s podporo tako za mobilne telefone kot tudi tablice, kjer se uporabniški vmesnik prilagaja velikosti zaslona. Tako razširimo število naprav, na katerih lahko igramo mobilno igro. Za postavitev uporabniškega vmesnika smo uporabili Android XML, kjer elementi predstavljajo hierarhijo razredov View in ViewGroup. Razred ViewGroup je neviden element, ki vsebuje strukturo elementov, ki razširjajo razred View. Razširitev razreda ViewGroup vsebujejo elementi LinearLayout, RelativeLayout, Constraint layout ipd. Razred View pa dedujejo elementi Button, TextView, EditText ipd. Predstavitev uporabniškega vmesnika s strukturo XML nam ločuje predstavitveni del aplikacije od kode, ki skrbi skrbi za obnašanje aplikacije [42]. S takšno definicijo je lažje razdeliti predloge za različne velikosti zaslonov.

Ko razvijamo mobilno aplikacijo za Android, je pomembno, da izberemo pravilno različico operacijskega sistema, na katerem bomo še lahko uporabljali aplikacijo. Kadar želimo podpreti stare različice in zajeti čim več mobilnih naprav, običajno izgubimo funkcionalnosti za razvoj aplikacije, saj starejše različice ne podpirajo funkcij, ki so razvite za nove sisteme. Izbrati moramo torej različico, ki podpira, kar potrebujemo za razvoj igre, in da kljub temu zajamemo veliko večino mobilnih naprav. Podprli smo API level 21 in novejše, ki se imenuje Android Lolipop 5.0. Tako smo po podatkih Googla [43] zajeli 87 % vseh mobilnih naprav.

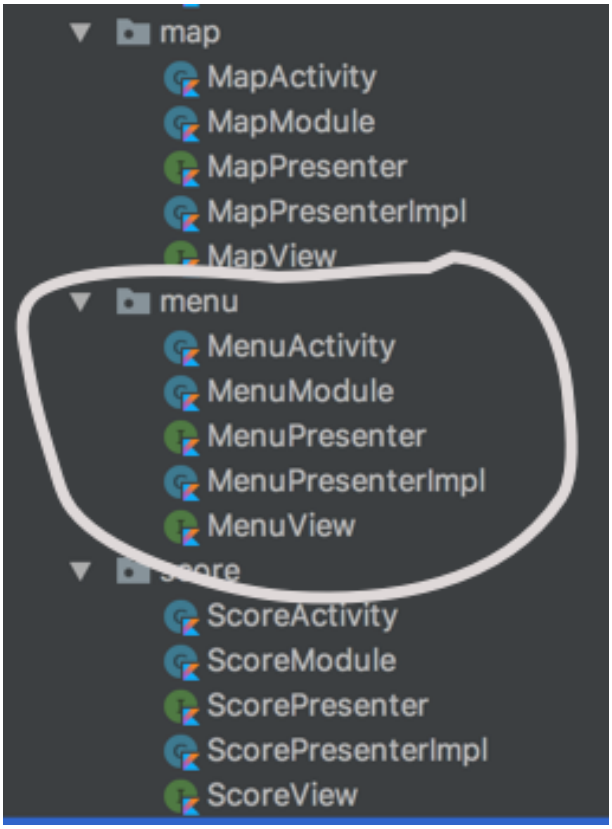


Slika 3.2: Vrivanje odvisnosti z ogrodjem Dagger 2 deluje z grafom odvisnosti, ki ga zgradi na podlagi modulov, v katerih zagotavljamo odvisne komponente. Te lahko vrinemo v konstruktorje drugih komponent, aktivnosti ipd. (povzeto po [41]).

3.4 Razvojna arhitektura

Razvojna arhitektura je način organiziranja kode, ki je pomemben pri projektih, kjer želimo nadzirati razširljivost, preglednost programske kode, programsko testirati in ponuditi končni izdelek z visoko kakovostjo. Najpogosteje uporabljeni arhitekturni vzorci pri razvoju mobilne aplikacije Android so MVC, MVP in MVVM. Arhitekturni vzorec Model View Controller (MVC) je primernejši za spletne aplikacije in ločuje logiko aplikacije na tri dele: model podatkov, vizualizacijo podatkov in logično plast. Model View Presenter (MVP) je različica vzorca MVC, kjer predstavitvena logika skrbi za prikaz in odzive uporabnika, model skrbi za podatke, logični del pa komunicira z obema, zato je pristop primernejši za grajenje uporabniških vmesnikov in uporabo v aplikacijah Android. Model View Viewmodel (MVVM) se od MVC in MVP razlikuje v tem, da obdelavo in pripravo podatkov delamo ločeno od pridobivanja podatkov in poslovne logike. Pri razvoju mobilne igre smo uporabili vzorec MVP, kjer je logika razdeljena na 3 ločene celote, zato smo za komunikacijo med njimi uporabili sistem za vrivanje odvisnosti Dagger.

Slika 3.3 prikazuje hierarhijo naših razredov po arhitekturi MVP. Razred MenuActivity skrbi za predstavitveno plast, torej prikaz podatkov in sprejemanje uporabnikovih interakcij, kjer je MenuView pripadajoča definicija razreda (ang. interface). MenuPresenterImpl in MenuPresenter skrbita za obdelavo podatkov in določata, kaj se bo zgodilo. MenuModule pa je povezovalni razred za Dagger, ki skrbi za vrivanje odvisnosti med ločenimi funkcionalnostmi.



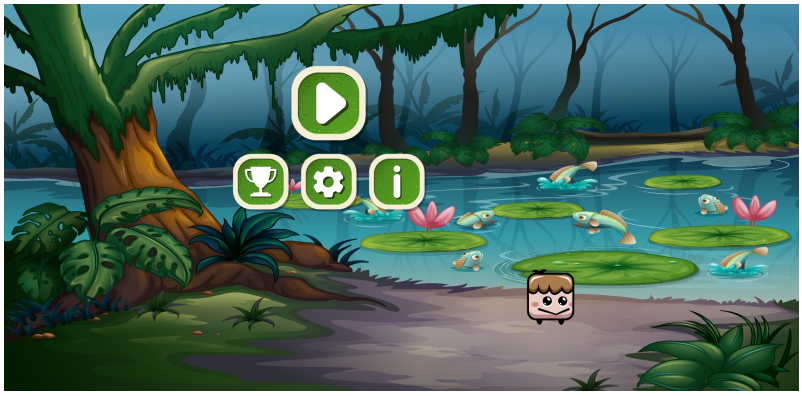
Slika 3.3: Hierarhija razredov po arhitekturi MVP

3.5 Potek igre

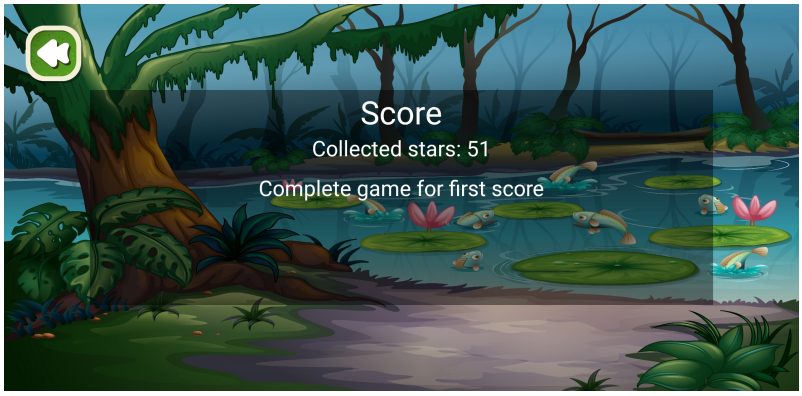
Na začetnem zaslonu, ki ga prikazuje slika 3.4, imamo možnost navigacije po mobilni igri, kjer lahko prehajamo med igranjem igre, stopnjami, rezultati, nastavitvami in informacijami o igri. Med nastavitvami (slika 3.6) imamo možnost izbire programskega jezika, ki bo prikazal igranje s psevdokodo. Ker smo želeli, da lahko igro igrajo tudi otroci iz drugih držav, smo pod nastavitvami dodali možnost izbire jezika. Vrstni red sklopov, ki jih bo učenec spoznal, lahko tudi urejamo, kar pomeni, da lahko igro prilagodimo učnemu načrtu. Stopnjo, ki jo želimo igrati, lahko izberemo na zaslonu, ki ga prikazuje slika 3.7.

S premikanjem levo/desno izbiramo med učnimi sklopi: programiranje z bloki, programiranje z diagramom poteka in programiranje s psevdokodo. Vsak sklop je sestavljen iz več stopenj, ki jih mora igralec uspešno odigrati za dokončano igro. V vsaki stopnji lahko s pravilnim programom zbere do 3 zvezdice, ki predstavljajo uspešnost dokončane stopnje. Zvezdice so v stopnjah postavljene tako, da mora v nekaterih primerih za vse zbrane zvezdice izdelati kompleksnejši program. Vsak koncept je pred začetkom igre predstavljen z animacijami (slika 3.8), ki otroku pomagajo prikazati posledice uporabe različnih konceptov s premiki Lea po zaslonu, s tem pa mu tudi predstavimo način igranja igre.

Programiranje s povleci in spusti blok je sklop (slika 3.9), v katerem poljubno premikamo in sestavljamo bloke ter tako omogočimo zapis rešitve enega problema na več možnih načinov. V drugem sklopu (slika 3.10) učenec spozna diagram poteka, kjer igro upravlja s predstavitvijo programa v obliki grafa. Tretji sklop (slika 3.11) je najtežji in temelji na psevdokodi, kjer uporabnik zapiše program v izbranem programskem jeziku. Metaokolje nam zaradi boljše preglednosti barva kodo in poskrbi za samodejne zamike, ko so ti potrebni. Zapis programa je lahko poljuben, torej več različnih rešitev pripelje do uspešno dokončane stopnje. Za pomoč pri pisanju psevdokode imamo na voljo pomoč (slika 3.12), ki prikazuje koncepte, ukaze in pravilnost sintakse za izbran programski jezik. Ko igralec dokonča igro, lahko

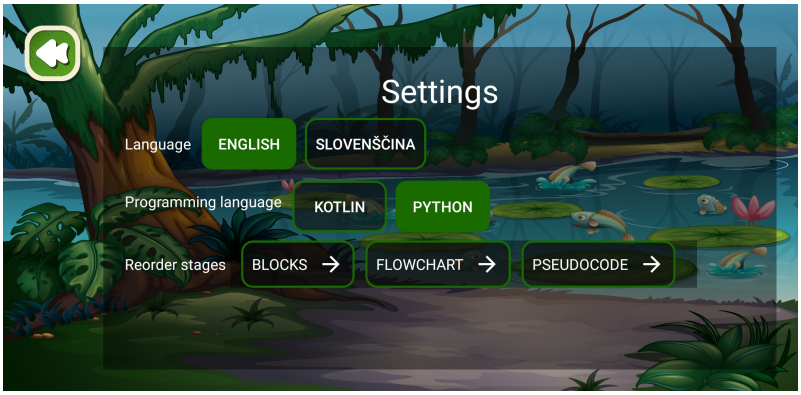


Slika 3.4: Začetni zaslon v obliki menija

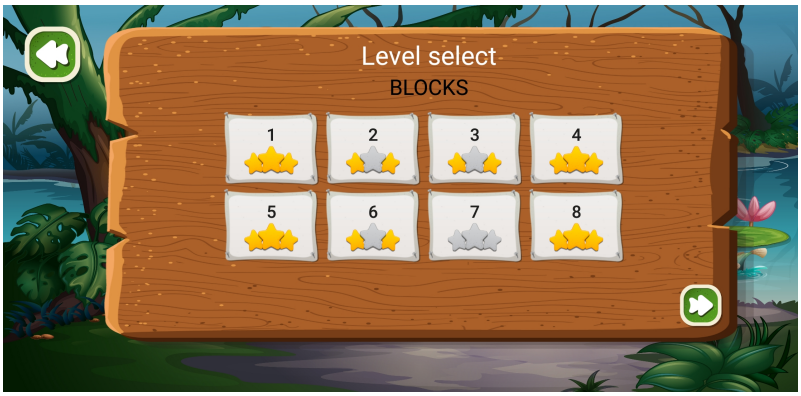


Slika 3.5: Prikaz rezultatov, ko igralec zaključi igro

vidi svoje dosežke na zaslonu za rezultate (slika 3.5), do katerega dostopa z začetnega zaslona. Rezultati so prikazani v obliki lestvice od najboljšega do najslabšega. Igralec lahko tako primerja svoje rezultate in dobi povratno informacijo o napredku.

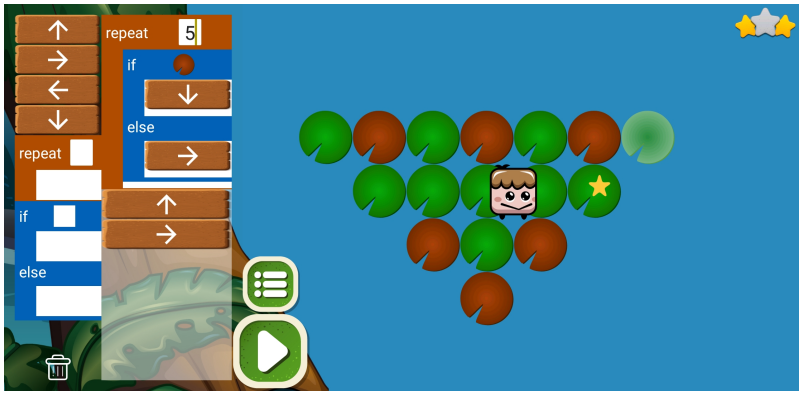


Slika 3.6: Nastavitve jezika vmesnika, programskega jezika in urejanje učnih sklopov

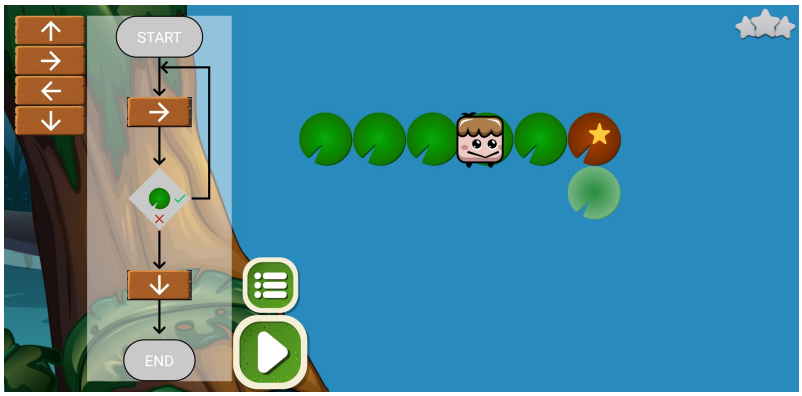


Slika 3.7: Izbira stopnje, sklopa in prikaz rezultatov

Slika 3.8: Predstavitev koncepta v obliki animacij

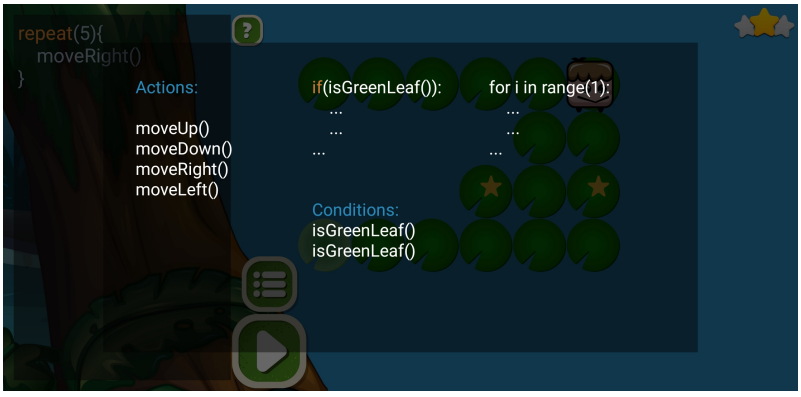


Slika 3.9: Programiranje na način gnezdenja blokov



Slika 3.10: Programiranje z diagramom poteka

Slika 3.11: Primer programa v programskem jeziku Kotlin



Slika 3.12: Sintaktična pomoč v programskem jeziku Python

3.6 Igralnost

Igralnost je izkušnja igralca, ki izraža interakcijo igranja in sodelovanje v igri [4]. V razviti mobilni igri igralec napiše programe, ki igro pripeljejo do konca, in tako sodeluje pri odvijanju igre, saj s svojimi interakcijami nadzira igro. Za uspešno dokončano stopnjo igralec pridobi točke, ki služijo kot motivacija pri igranju in predstavljajo način nagrajevanja, ki daje povratno informacijo o uspešnosti napisanega programa. Igralec lahko na podlagi rezultatov tekmuje s svojimi dosežki ali z drugimi sošolci, ki so lahko bolje odigrali igro, kar otrokom daje še dodatno motivacijo za učenje in da bi bili najboljši.

V ozadju igra glasba, ki motivira igralca pri reševanju problemov. Izbrali smo glasbo, ki ni moteča in se ujema s slogom igre.

3.7 Implementacija

Predstavili bomo implementirane rešitve in algoritme, ki smo jih uporabili pri razvoju naše igre. Zanko igre, ki skrbi za sprejemanje vhodnih podatkov, posodabljanje objektov in izris na zaslon, nadziramo s funkcijskim programiranjem v knjižnici RxJava. Tipična rešitev omenjenega problema se rešuje z neskončno zanko, ki teče v ločeni niti, kjer preverjamo čas na podlagi konstante, ki določa, koliko sličic na sekundo se bo izrisalo. Način nadzora s knjižnico RxJava je učinkovitejši, saj imamo boljši pregled nad izvajanjem in upravljanjem zanke.

Izdelali smo prevajalnik programske kode, ki služi za prevedbo napisanega programskega jezika v programski jezik Kotlin. Pri programiranju z bloki, diagramom poteka in psevdokodo najprej bloke, gradnike ali kodo preberemo iz uporabniškega vmesnika. Za reševanje problema, kjer je lahko gnezdenih poljubno mnogo ukazov in konceptov, smo uporabili rekurzijo, ki se sprehodi po vseh elementih in jih shrani v obliko, primernejšo za izvajanje ukazov. Ukazi, pripravljeni za izvajanje, so shranjeni v seznamu, kjer lahko objekt vsebuje druge podobjekte. Zanka je zasnovana tako, da vsebuje neko zapo-

redje ukazov, torej ima lahko objekt zanka poljubno mnogo podobjektov, ki se bodo izvajali ob klicu zanke. Vsak objekt znotraj zanke ima lahko tudi poljubno mnogo drugih objektov, ki so lahko nove zanke, pogojni stavki ali ukazi. Tako izdelamo hierarhijo ukazov, ki se bodo izvajali.

V prvi fazi smo napisan program spremenili v obliko, razumljivo prevajalniku, v drugi pa smo poskrbeli za dodajanje ukazov na seznam izvajanja, ki predstavlja zaporedje ukazov v vrstnem redu, kot se bodo izvajale operacije. Ukaze izvajamo rekurzivno, tako da za vse ukaze, ki jih prejme funkcija, preverimo tip in na podlagi njegovih značilnosti dodamo ukaze ter nove objekte v rekurzivni klic. Ker pri igranju uporabljamo premikanje lika, poskrbimo, da se ukaz, ki je na vrsti za izvajanje, vzame iz vrste in poskrbi za prikaz animacij. Na vsakem koraku preverimo, ali je po premiku igralni lik še na polju in ali premik pomeni konec stopnje. Po uspešnem premiku iz vrste vzamemo nov ukaz in postopek ponavljamo, dokler igra ni končana ali je postopek napačen.

Stopnje smo izdelali v obliki seznamov 2D, kjer vsako polje sestavlja poligon stopnje. Takšen način predstavitve stopnje (slika 3.13) omogoča lažjo nadgradnjo in dodajanje novih stopenj. Število 1 nam predstavlja zeleno igralno polje, številka 2 rdeče in črka F polje za končanje stopnje. Črka S za številko pomeni, da je na tem tipu polja zvezdica, ki ob obisku pristavlja pridobljeno točko.

```
private val level6 : List<List<String>> = listOf(
        listOf("1", "1", "1", "1", "2S", "0", "0"),
        listOf("2", "0", "1", "0", "2", "0", "0"),
        listOf("2", "0", "1", "1", "2", "0", "0"),
        listOf("2", "0", "0", "0", "2", "0", "0"),
        listOf("2", "1", "1", "1S", "1", "1S", "F"))

private val level7 : List<List<String>> = listOf(
        listOf("0", "0", "0", "0", "0", "0", "0"),
        listOf("1", "2", "1", "2", "1", "2", "F"),
        listOf("0", "1S", "1", "1S", "1", "1S", "0"),
        listOf("0", "0", "2", "1", "2", "0", "0"),
        listOf("0", "0", "0", "2", "0", "0", "0"),
        listOf("0", "0", "0", "0", "0", "0", "0"))

private val level8 : List<List<String>> = listOf(
        listOf("0", "0", "0", "0", "0", "0", "1"),
        listOf("0", "0", "0", "0", "0", "1", "2"),
        listOf("0", "0", "0", "0", "1S", "2", "1"),
        listOf("0", "0", "0", "1", "2S", "1", "0"),
        listOf("0", "0", "1S", "2", "1", "1", "F"))
```

Slika 3.13: Zapis stopenj s seznamom 2D, ki omogoča lažje prilagajanje težavnosti stopenj, saj predstavlja vsak element seznama igralno polje, ki ga določimo s številko in črko.

3.8 Analiza in testiranje

Igro smo testirali na 6 različno zmogljivih mobilnih napravah: Samsung Galaxy S9, Samsung Galaxy A6, Huawei Y7, HTC Desire 12+, Sony Xperia XZ2 in Lenovo Tab 4. Igra deluje nemoteno na vseh mobilnih napravah. Uporabniški vmestnik se prilagaja glede na velikost zaslona in se glede na ločljivost malenkost razlikuje, vendar je na vseh napravah pregleden, zato lahko sklepamo, da igra deluje na večini naprav z operacijskim sistemom Android, ne glede na velikost zaslona.

Igro so preizkusili štirje otroci, stari med 12 in 14 let, v svojem prostem času. Najprej smo jim predstavili teoretično osnovo konceptov programiranja in jih motivirali z igro. Vsi otroci so uživali ob igranju in se strinjali, da bi bila razvita igra primerna tudi za poučevanje njihovih vrstnikov. Eden je poudaril, da so nekatere stopnje pretežke. Ker je zapis stopenj v obliki, ki jo lahko preprosto spreminjamo, nadgradnja ne predstavlja problema. Pri programiranju s psevdokodo so potrebovali dodatno razlago, saj je bil za popolnega začetnika preskok iz vizualnega programiranja v programsko kodo prevelik in bi bilo treba razviti funkcionalnost za predstavitev dodatnih informacij, ki povežejo znanje vizualnega programiranja z ukazi v psevdokodi. Ugotovimo lahko, da je treba, preden iz tega kar koli sklepamo, dodatno testiranje in odziv s strani učiteljev.

Poglavje 4

Sklepne ugotovitve

Cilj magistrskega dela je bil razvoj mobilne aplikacije za učenje osnovnih konceptov programiranja in logičnega razmišljanja. Izdelali smo metaokolje, s katerim se bodo učenci naučili uporabe konceptov v različnih programskih jezikih. Razvili smo igro za operacijski sistem Android s podporo večine mobilnih naprav, telefonov in tudi tablic. Pri razvoju smo uporabili najnovejše koncepte in smernice za razvoj mobilnih aplikacij Android ter programsko kodo javno objavili na platformi GitHub. Ker je igra namenjena otrokom v 2. triletju osnovne šole, so omenjeni koncepti primerni za njihovo raven abstrakcije in razvojne stopnje razmišljanja, kjer smo okolje predstavili z veliko slikami in malo besedila. Ker je igra namenjena učenju v osnovni šoli, je pomembno, da vključuje učno snov, s čimer izpolnjujemo zastavljene učne cilje. Upoštevali smo, da ima igra pravo razmerje igralnosti in učenja konceptov. Takšno igro lahko učitelj uporabi pri načrtovanju učne ure, čeprav se nekateri tega izogibajo zaradi predsodka, da se otroci skozi igro preveč igrajo in da se ničesar ne naučijo.

Poudarili bomo didaktične in implementacijske prispevke mobilne igre, ki smo jo predstavili v prejšnjih poglavjih. Predstavljamo tudi nekaj možnosti za nadaljnje delo, ki bi igro izboljšale.

4.1 Didaktični doprinos

Izdelali smo izobraževalno mobilno igro, ki je primerna za poučevanje otrok v osnovni šoli. Ker pri poučevanju računalništva primanjkuje učnih pripomočkov za učenje osnovnih konceptov programiranja, smo z razvojem k temu pripomogli. Mobilna igra je primerna tudi za igranje doma, kar pomeni, da jo lahko učitelji uporabijo za domačo nalogo in tako ponudijo zanimivejši pristop, kar lahko učitelj preveri na podlagi zbranih točk. Prednost je v tem, da za razliko od drugih iger, ki učijo koncepte programiranja, otroci spoznajo tudi različne zapise programov, kjer lahko rešitve in probleme povezujejo med seboj. Igra je uporaben didaktičen pripomoček ob prehodu iz blokovskega programiranja na tekstovno.

4.2 Implementacijski doprinosi

Mobilna igra je razvita za operacijski sistem Android, kjer smo uporabili programski jezik Kotlin. Ker je ta še v razvoju in veliko razvijalcev še vedno uporablja programski jezik Java, smo ponudili igro z arhitekturo MVP, ki smo jo naložili na GitHub in jo lahko razvijalci prenesejo, analizirajo ter uporabijo rešitve pri razvoju novih inovativnih iger.

4.3 Predlogi izboljšav

Vsekakor je treba pred nadgradnjo igro še dodatno testirati na večjem vzorcu otrok in jo preizkusiti za poučevanje v razredu. Na podlagi rezultatov bomo lahko ugotovili, kje so večje pomanjkljivosti igre iz didaktičnega kot tudi implementacijskega dela.

Nadgradnja in dodelava zgodbe bi igro med seboj še bolj povezala in jo naredila zanimivejšo. Tako bi povečali igralnost in motivacijo med igranjem. Nova grafična podoba, izdelana s strani grafičnega razvijalca, in zgodba bi naredili boljšo uporabniško izkušnjo. Z več stopnjami in z boljšim sistemom za nagrajevanje bi omogočili dodatno utrjevanje in uporabo znanja.

Razširitev aplikacije za druge mobilne operacijske sisteme bi sicer pomenila popolnoma novo implementacijo za nov operacijski sistem, vendar bi s tem pokrili večji delež uporabnikov, ki bi lahko igrali igro.

Igro bi lahko razširli z dodajanjem novih učnih sklopov, ki bi se nanašali na razumevanje dodatnih računalniških konceptov. S tem bi igralec skozi igro spoznal večino uvodnih konceptov programiranja. Z dodatnimi pristopi programiranja bi otrok odkrival nove načine, medsebojno povezoval in dopolnjeval veščine logičnega razmišljanja.

Funkcionalnost, kjer bi otrok videl opisno oceno, bi bila vsekakor dobrodošla za učenca in tudi učitelja. S tem bi dobil podrobno povratno informacijo, na podlagi katere bi vedel, katere učne sklope in katere koncepte mora še poglobiti.

Z izdelavo magistrskega dela smo pridobili nova znanja s področja izobraževanja računalništva, predstavitve konceptov in izdelave izobraževalnih iger. Med razvojem smo spoznali nova orodja, algoritme, knjižnice in razširili znanje na področju izdelave mobilnih aplikacij Android. Pridobljena znanja bodo prispevala pri nadaljnjem razvoju novih izobraževalnih materialov.

Literatura

[1] S. Papert, "Mindstorms: Children, Computers, and Powerful Ideas", *Basic Books, Inc.*, New York, USA, 1980.

[2] B. Olga, "Gamification – how games can level up our everyday life?", *VU University*, Amsterdam, 2011.

[3] L. Marjanovič Umek, M. Zupančič "Psihologija otroške igre: od rojstva do vstopa v šolo", *Znanstvenoraziskovalni inštitut Filozofske fakultete*, Ljubljana, 2001.

[4] K. Salen, K. S. Tekinbaş, E. Zimmerman "Rules of Play: Game Design Fundamentals", *MIT Press*, 2004.

[5] J. Rugelj, "Izobraževalne računalniške igre", *Pedagoška fakulteta, Univerza v Ljubljani*, Ljubljana, 2015.

[6] U. Marn, "Konstruktivizem v šoli kot podlaga učenja nenasilnih vzorcev vedenja", Socialna pedagogika, Ljubljana, letnik 10, številka 3, str. 365-386, 2006.

[7] S. Papert, "The Children's Machine: Rethinking School In The Age Of The Computer", *Basic Books, Inc.*, New York, 1993.

[8] J. Knežević, "M-učenje", *Odsjek za informa- cijske znanosti, Filozofski fakultet, Sveučilište u zagrebu*, Zagreb, 2011.

[9] T. Hiltunen, "Learning and Teaching Programming Skills in Finnish Primary Schools: The Potential of Games", *University of Oulu*, Oulu, 2016.

[10] K. M. Kapp, "The Gamification of Learning and Instruction: Game-based Methods and Strategies for Training and Education", John Wiley & Sons, 2012.

[11] G. Kiryakova, N. Angelova, L. Yordanova, "Gamification in education", v zborniku: 9th International Balkan Education and Science Conference, 2014.

[12] H. Meishar-Tal, M. Ronen, "Experiencing a Mobile Game and Its Impact on Teachers' Attitudes towards Mobile Learning", International Association for Development of the Information Society, Holon Institute of Technology H.I.T, Israel, 2016.

[13] Ž. Ternik, T. Koron, A. Koron, I. Nenčovska Šerbec, "Learning Programming Concepts through Maze Game in Scratch", European Conference on Games Based Learning, Academic Conferences International Limited, str. 661-670, 2017.

[14] N. Hoić-Božič, M. Holenko Dlab, L. Načinović Prskalo, J. Rugelj, I. Nančovska Šerbec, "GLAT: Games for Learning Algorithmic Thinking", *University of Rijeka, Department of Informatics, Croatia and University of Ljubljana, Faculty of Education, Slovenia*, 2018.

[15] A. Wilson, D. C. Moffat, "Evaluating Scratch to introduce younger schoolchildren to programming", Proceedings of the 22nd Annual Psychology of Programming Interest Group, Universidad Carlos III de Madrid, Leganés, Spain, 2010.

[16] M. Hinds, N. Baghaei, P. Ragon, J. Lambert, T. Dajakaruna, T. Houghton, S. Dacey, J. Casey, "Designing a Novel Educational Game for

Teaching C# Programming", *Bruce M. McLaren (Ed.)*, 9th International Conference on Computer Supported Education, Porto, Portugal, 2017, str. 81-86.

[17] K. Brennan, "Learning Computing through Creating and Connecting", v zborniku: Computer, *Harvard University*, letnik 46, št. 9, str. 52-59, 2013.

[18] B. S. Seker, G. G. Sahin, "Sample game applications in social studies teaching", Procedia-Social and Behavioral Sciences, 46, str. 1679-1683, 2012.

[19] M. Pivec, P. Kearney, "Games for Learning and Learning from Games", *Informatica*, Slovenia, 21, str. 419-423, 2007.

[20] P. Kearney, M. Pivec, "Recursive Loops of Game-Based Learning: a Conceptual model", v EdMedia: World Conference on Educational Media and Technology, Association for the Advancement of Computing in Education (AACE), str. 2546-2553, junij 2007.

[21] W. B. Frederick, C. Watson, "Game-Based Concept Visualization for Learning Programming", v zborniku: Third International ACM Workshop on Multimedia Technologies for Distance Learning, *ACM*, str. 37-42, december 2011.

[22] J.E. Korteling, A. S. Helsdingen, R. R. Sluimer, M. L. van Emmerik, B. Kappe, "Transfer of Gaming: Transfer of training in serious gaming", *TNO report*, Soesterberg, 2011.

[23] Kidlo Coding Game, IDZ Digital Private Limited, Dostopno na: `https://play.google.com/store/apps/details?id=com.internetdesignzone.kidlolandcoding`, (pridobljeno: 6. 8. 2018).

[24] Bit by Bit - Programming Game, Rikai Games, Dostopno na: `http://rikaigames.com/bitbybit/`,(pridobljeno: 6. 8. 2018).

[25] Coding Planets, Min Thura Zaw, Dostopno na: https://play.google.com/store/apps/details?id=com.material.design.codingplanet, (pridobljeno: 6. 8. 2018).

[26] Algorithm City, Mustern, Dostopno na: https://play.google.com/store/apps/details?id=air.MusterenGames.ElHarezmiCoding, (pridobljeno: 7. 8. 2018).

[27] Mimo: Learn to Code, Mimohello GmbH, Dostopno na: https://getmimo.com/, (pridobljeno: 7. 8. 2018).

[28] K. Brennan, M. Resnick, "New frameworks for studying and assessing the development of computational thinking", v zborniku: The 2012 annual meeting of the American Educational Research Association, Vancouver, Canada, letnik 1, str. 25, 2012.

[29] SEGAN, Serious Games Network, Dostopno na: http://seriousgamesnet.eu/, , (pridobljeno: 13. 8. 2018).

[30] A Basic Guide to Learning Objectives, Learning Excellence & Innovation Department, Dostopno na: http://www.northernc.on.ca/leid/docs/guidetolearningobjectives.pdf, (pridobljeno: 13. 8. 2018).

[31] F. Strmčnik, "Didaktika: osrednje teoretične teme", Ljubljana, *Znanstveni inštitut Filozofske fakultete*, 2001.

[32] M. Pivec, O. Dziabenko, I. Schinnerl-Beikircher, "Aspects of Game Based Learning", In 3rd International Conference on Knowledge Management, Graz, Austria, str. 216-225, 2003.

[33] A. Chryssa, "Android Programming Cookbook", *Exelixis Media P.C.*, 2016.

[34] Mobile Operating System Market Share Worldwide, 2018. Dostopno na: http://gs.statcounter.com/os-market-share/mobile/worldwide, (pridobljeno: 14. 8. 2018)

[35] Android Studio. Dostopno na: `https://developer.android.com/studio/intro/`, (pridobljeno: 14. 8. 2018)

[36] M. Shafirov, Kotlin on Android. Dostopno na: `https://blog.jetbrains.com/kotlin/2017/05/kotlin-on-android-now-official/`, (pridobljeno: 14. 8. 2018)

[37] RxJava. Dostopno na: `https://github.com/ReactiveX/RxJava`, (pridobljeno: 16. 8. 2018)

[38] Gradle. Dostopno na: `https://gradle.org/`, (pridobljeno: 16. 8. 2018)

[39] B. Muschko, "Gradle in action", *Manning Publications*, 2014.

[40] D. Baharestani, "Mastering Ninject for Dependency Injection", *Packt Publishing Ltd.*, 2013.

[41] Dependency Injection with Dagger 2. Dostopno na: `https://github.com/codepath/android_guides/wiki/Dependency-Injection-with-Dagger-2`, (pridobljeno: 17. 8. 2018)

[42] Layouts. Android developers. Dostopno na: `https://developer.android.com/guide/topics/ui/declaring-layout`, (pridobljeno: 18. 8. 2018)

[43] Platform versions. Dostopno na: `https://developer.android.com/about/dashboards/`, (pridobljeno: 14. 8. 2018)