Usvajanje novih nastavnih sadržaja iz matematike pomoću programskoga jezika Scratch

Vicić, Katarina

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ZA ODGOJNE I OBRAZOVNE ZNANOSTI

Katarina Vicić

USVAJANJE NOVIH NASTAVNIH SADRŽAJA IZ MATEMATIKE POMOĆU PROGRAMSKOG JEZIKA *SCRATCH*

DIPLOMSKI RAD

Osijek, 2018.

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ZA ODGOJNE I OBRAZOVNE ZNANOSTI

Integrirani preddiplomski i diplomski sveučilišni Učiteljski studij

USVAJANJE NOVIH NASTAVNIH SADRŽAJA IZ MATEMATIKE POMOĆU PROGRAMSKOG JEZIKA *SCRATCH*

DIPLOMSKI RAD

Predmet: Programski jezik Logo
Mentor: izv. prof. dr. sc. Ivana Đurđević Babić
Student: Katarina Vicić
Matični broj studenta: 2468
Modul: B modul, informatika

Osijek
srpanj, 2018.

Zahvaljujem se mentorici izv. prof. dr. sc. Ivani Đurđević Babić za svu pomoć i vođenje pri pisanju diplomskoga rada.

Veliko hvala mojoj obitelji i prijateljima koji su me uvijek podržavali i vjerovali u mene.

SAŽETAK

Zbog sve veće prisutnosti informacijsko-komunikacijskih tehnologija i važnosti područja znanosti, tehnologije, inženjerstva i matematike (eng. izraz: *Science, Technology, Engineering and Mathematics*; STEM područje) u ovom radu naglasak je stavljen na računalno programiranje i njegov utjecaj na sposobnost rješavanja problema, apstraktno mišljenje te na razvoj računalno-algoritamskog načina razmišljanja. Navode se prednosti i izazovi poučavanja programiranja kod djece mlađe školske dobi te pristupi kojima se može olakšati uvodno učenje računalnog programiranja. Kao jedan od pristupa, spominje se rad u vizualnim programskim jezicima koji omogućuju učenje računalnog programiranja na djeci zanimljiv način, uz jednostavnost grafičko-korisničkog sučelja i pojednostavljenu sintaksu. Rad također donosi pregled ishoda učenja računalnoga programiranja za niže razrede osnovne škole te rezultate provedenog istraživanja kojemu je cilj bio provjeriti mogu li se u četvrtom razredu osnovne škole usvojiti novi programski sadržaji iz matematike radom u programskom jeziku *Scratch*.

Ključne riječi: računalno programiranje, računalno-algoritamski način razmišljanja, izazovi poučavanja, vizualni programski jezici, matematika

SUMMARY

Due to the increasing presence of ICT and the importance of the field of Science, Technology, Engineering and Mathematics; in this thesis emphasis is on the computer programming and its impact on the ability to solve problems, on abstract thinking and on the development of computer-algorithmic thinking. Thesis lists the advantages and challenges of teaching programming to children in younger school age, and the approaches that can facilitate introductory learning of computer programming. As one of the approaches, there are visual programming languages that are allowing learning of computer programming in an interesting way for children, with the ease of a graphical-user interface and simplified syntax. Thesis describes the learning outcomes of computer programming for elementary school and the results of the conducted research aimed at checking whether the new teaching contents in mathematics could be adopted using a programming language *Scratch* in the fourth grade of elementary school.

Keywords: computer programming, computational thinking, teaching challenges, visual programming languages, mathematics

SADRŽAJ

1. UVOD 1
2. RAČUNALNO PROGRAMIRANJE I PREDNOSTI UČENJA RAČUNALNOG PROGRAMIRANJA 2
2.1. Utjecaj računalnog programiranja na sposobnost rješavanja problema 3
2.2. Apstraktno mišljenje i računalno programiranje 4
2.3. Računalno-algoritamski način razmišljanja 5
3. IZAZOVI POUČAVANJA PROGRAMIRANJA I RAZLIČITI PRISTUPI STJECANJU ZNANJA IZ PROGRAMIRANJA 6
3.1. Računalna igra u učenju programiranja 6
4. UČENJE MATEMATIKE PUTEM PROGRAMIRANJA 8
5. POUČAVANJE UČENIKA MLAĐE ŠKOLSKE DOBI PROGRAMIRANJU 10
5.1. Logo 11
5.2. Scratch 12
5.3. Alice 14
5.4. Kodu 15
6. RAČUNALNO PROGRAMIRANJE U OSNOVNOŠKOLSKOJ NASTAVI U REPUBLICI HRVATSKOJ 16
7. METODOLOGIJA ISTRAŽIVANJA 18
8. REZULTATI I RASPRAVA 20
9. ZAKLJUČAK 28
10. LITERATURA 29

1. UVOD

Pojava i prisutnost informacijsko-komunikacijske tehnologije dovela je do toga da djeca u svoje slobodno vrijeme pretražuju internet i igraju videoigre te takvim neformalnim učenjem usvajaju osnovne koncepte računalno-algoritamskog načina razmišljanja (Wing, 2008). Prema *Nacionalnom okvirnom kurikulumu* (MZOS, 2011), u Republici Hrvatskoj informacijsko-komunikacijska tehnologija odnosi se na usmjeravanje učenika za stjecanje znanja i vještina iz područja informacijsko-komunikacijske tehnologije, kritičko promišljanje o njezinim prednostima i nedostatcima te razvoj digitalne kompetencije kao jedne od osam temeljnih kompetencija svakog pojedinca. U istom dokumentu navedeno je da se informacijsko-komunikacijska tehnologija poučava u okviru izbornog nastavnog predmeta, a učenici mlađe školske dobi uče osnove računalnoga programiranja.

Računalno se programiranje većini učenika može činiti teškim jer iziskuje razvijenu sposobnost apstraktnog mišljenja koja nije karakteristična za fazu konkretnih operacija u kojoj se oni nalaze, a učenjem programiranja kod djece mlađe školske dobi može se utjecati na razvoj te sposobnosti (Bubica i suradnici, 2013; Wing, 2006). Isti autori navode da učenje računalnog programiranja može utjecati i na sposobnost rješavanja problema te računalno-algoritamski način razmišljanja koji uključuje apstrakciju i dekompoziciju tijekom rješavanja određenog problema, a takvim promatranjem i promišljanjem o načinu na koji će riješiti problem učenici shvaćaju mogućnost rješavanja problema na više načina, što mogu usporediti i sa situacijama u svakodnevnom životu.

Budin (2018) tumači da računalno programiranje nije usko vezano samo za područje računalnih znanosti i da se takvi načini razmišljanja trebaju prenositi u druga područja (ponajprije u STEM područja), s ciljem da učenici postanu konkurentniji. Kako bi se olakšalo uvodno učenje programiranja pojednostavljuje se sintaksa, naredbe su grafički blokovi koje djeca mogu povlačiti i ispuštati te tako izraditi program, a programi se mogu odmah i izvoditi (Fessakis i suradnici, 2013). Takva programska okruženja su u vizualnim programskim jezicima koji „osim uklanjanja problema sintakse, omogućuju učenje programiranja u konkretnom okruženju, gdje apstraktni pojmovi poput varijabli, petlji i sl. u vizualnom okruženju pružaju konkretno iskustvo“ (Bubica i suradnici, 2013:4).

Prvi programski jezik koji je bio namijenjen djeci, a služio je za uvodno učenje programiranja jest programski jezik *Logo* u kojemu su djeca, osim programiranja, učila i matematičke koncepte (Papert, 1980). Nakon programskog jezika Logo pojavili su se brojni vizualni programski jezici, kao npr. *Alice*, *Scratch* i *Kodu* koji će biti objašnjeni dalje u tekstu.

2. RAČUNALNO PROGRAMIRANJE I PREDNOSTI UČENJA RAČUNALNOG PROGRAMIRANJA

Računalno programiranje uključuje „rješavanje problema, otklanjanje grešaka, razvijanje logičkog razmišljanja i računalnog razmišljanja, a to podrazumijeva razvoj strategija za rješavanje problema koji se mogu odnositi i na neprogramerska područja“ (Bubica i suradnici, 2013:1). Na temelju pregleda ranijih istraživanja Pea i Kurland (1983) smatraju da postoje četiri kognitivne aktivnosti prisutne u svakom računalnom programiranju: razumijevanje problema, izrada i planiranje njegovog rješavanja, izrada programskog koda te shvaćanje napisanog programa i pronalaženja pogrešaka u kodu. Isti autori navode da uz ove aktivnosti postoji još mnogo kognitivnih aktivnosti uključenih u izradu programskog koda.

Jenkins (2002) tumači da, ovisno o stilu učenja i motivaciji učenika, neki učenici mogu smatrati učenje programiranja jednostavnim, a drugi teškim i dosadnim. Iz tog razloga autor tvrdi da učitelji moraju uložiti velike napore u uvodnom učenju programiranja. Pea i Kurland (1983) ističu pet kognitivnih sposobnosti za koje smatraju da imaju utjecaj na učenje računalnog programiranja, a to su matematička sposobnost, kapacitet memorije, vještine za rješavanje matematičko-logičkih zadataka, vještine uvjetovanog razmišljanja i vještine proceduralnog razmišljanja pa prema tome preispituju jesu li djeca sposobna postići znatan napredak u računalnom programiranju.

Računalno se programiranje može koristiti u raznim područjima i znanostima pa se može reći da je sveprisutno (Wing, 2008). Prema Prensky (2008), u današnjem je svijetu normalno da svaka obrazovana osoba zna programirati. Današnja djeca mlađe školske dobi rođena su u digitalnom dobu i mogu se prilagoditi novim tehnologijama koje jednostavno koriste, stoga postoji potreba za njihovim osposobljavanjem za računalno programiranje (Kalelioğlu, 2015). Isti autor navodi da zbog toga dolazi do potrebe za mijenjanjem programskih sadržaja, posebice iz područja računalne znanosti jer je priroda tog područja dinamična, fleksibilna i inovativna. Svi dionici obrazovnog sustava trebaju dati dovoljno pozornosti računalnoj znanosti kako bi stvorili nove korisnike i stvaratelje na tom području (Kalelioğlu, 2015).

Tay i suradnici (2012) objašnjavaju razliku između učenja *od* informacijsko-komunikacijske tehnologije (eng. izraz: *Information and Communication Technology*; IKT) i učenja *sa* informacijsko-komunikacijskom tehnologijom. Učenje *od* IKT-a autori opisuju kao bihevioristički pristup učenju prilikom čega se mogu razvijati osnovne vještine poput čitanja i slušanja, a računalni sustav smatra se asistentom tijekom takvog procesa učenja. Učenje *sa*

IKT-om opisano je kao konstruktivistički pristup učenju prilikom čega se može olakšati razvoj sposobnosti rješavanja problema i mišljenja višeg reda, kognitivnih i metakognitivnih sposobnosti te kritičkog mišljenja (Tay i suradnici, 2012). Računalno programiranje, kao dio IKT-a, također ima pozitivne učinke na kognitivne i metakognitivne sposobnosti te pomaže i u razvoju apstraktnog mišljenja (Bubica i suradnici, 2013).

2.1. Utjecaj računalnog programiranja na sposobnost rješavanja problema

Pears i suradnici (2007) tvrde da mnogi istraživači smatraju da je računalno programiranje povezano s logičkim načinom razmišljanja i sposobnošću rješavanja problema pa se često povezuje i s matematikom. Isti autori tumače da se sposobnost rješavanja problema može razvijati učenjem programskih jezika te računalno programiranje opisuju kao primjenu te sposobnosti.

Računalno se programiranje shvaća kao alat pomoću kojeg razvijamo metakognitivne sposobnosti, a ne samo kao vještina koju je poželjno svladati (Bubica i suradnici, 2013). Metakomponente, kao dio metakognitivnih spoznajnih procesa, odgovorne su za planiranje, kontroliranje i donošenje odluka tijekom rješavanja problema i izvršavanja zadataka, a osim ovih postupaka, uključuju i odlučivanje o prirodi problema, odabiru relevantne strategije za rješavanje problema te praćenje procesa rješavanja problema (Clements i Nastasi, 1999).

Kalelioğlu (2015) objašnjava da računalno programiranje iziskuje i refleksivno mišljenje koje se najbolje očituje u situacijama rješavanja problema, potiče analizu i promišljanje o tome što se događa tijekom rješavanja zadataka te omogućuje učenicima shvaćanje na koji način riješiti problem i koje su prikladne strategije za postizanje cilja. Fessakis i suradnici (2013) u istraživanju koje su proveli s djecom vrtićke dobi došli su do zaključka da djeca tijekom računalnog programiranja podjednako odabiru planiranje i metodu pokušaja i pogrešaka za rješavanje problema.

Učenje računalnog programiranja potiče precizno izražavanje i temeljito razmišljanje, razumijevanje formalnih procedura, funkcija i varijabli kao osnovnih koncepata, izradu plana rješavanja problema, prepoznavanje sličnosti s nekim već riješenim problemom zahvaljujući naučenim strukturama u mozgu, dekompoziciju složenih problema, rješavanje problema metodom pokušaja i pogrešaka te prepoznavanje činjenice da se svaki problem može riješiti na više načina (Budin, 2018).

Clements (1986) je istraživao utjecaj računalnoga programiranja na kognitivne i metakognitivne sposobnosti, kreativnost te uspjeh učenika prvog i trećeg razreda osnovne

škole. Istraživanje je trajalo gotovo šest mjeseci, a prema rezultatima prije i poslije istraživanja autor je zaključio da nema značajne razlike između rezultata učenika prvog i trećeg razreda te nije odbacio hipotezu da i učenici mlađe školske dobi mogu učiti računalno programiranje. Autor je došao do zaključka da računalno programiranje pozitivno utječe na sposobnost rješavanja problema, određivanje prirode problema te proces rješavanja problema, poboljšava razinu kreativnosti i sposobnost opisivanja smjera kretanja. Do sličnog su zaključka došli Bubica i suradnici (2013) koji su u svome radu naveli da je računalno programiranje alat pomoću kojega se razvijaju metakognitivne vještine i apstraktno mišljenje.

Prema Papert (1980), učenici tijekom računalnog programiranja razvijaju i sposobnost za davanje uputa, s obzirom na to da se trebaju pisati točne i sintaktički smislene naredbe kako bi računalo moglo uspješno izvršiti zadatke. Računalno bi programiranje moglo pozitivno utjecati i na divergentno mišljenje, s obzirom na to da učenici tada stvaraju i mijenjaju vlastite projekte te na reflektivno mišljenje i prostornu svijest (Clements i Gullo, 1984).

2.2. Apstraktno mišljenje i računalno programiranje

Djeca mlađe školske dobi u fazi konkretnih operacija uče iz konkretnih iskustava te trebaju razvijati i redovito vježbati sposobnost apstraktnog mišljenja kako im se računalno programiranje ne bi činilo teškim (Bubica i suradnici, 2013). Ako učenik uspješno transformira apstraktne probleme u programski kod kao rješenje na računalu, smatra se da je uspješan u računalnome programiranju (Derus i Ali, 2012). Wing (2008) smatra da je osnova računalnog programiranja apstrakcija, koja je u ovome području složenija nego u matematici i fizici jer se istovremeno radi s najmanje dva sloja apstrakcije.

Aktivnostima koje uključuju apstraktno razmišljanje, a stimulativne su i uzbudljive, može se pomoći djeci u razvoju apstraktnog razmišljanja (Bubica i suradnici, 2013). Wing (2008) tvrdi da se djeci može olakšati računalno programiranje pomoću vizualizacije i činjenice da njihove ideje mogu biti zapisane u algoritmima te tako postaju konkretno iskustvo. Prema Grover i Pea (2013), jedan od načina su vizualno-grafička programska okruženja koja su oslobođena djeci često teške i nerazumljive sintakse pa se učenici mogu lakše koncentrirati na izradu programa i realizaciju vlastitih ideja.

2.3. Računalno-algoritamski način razmišljanja

Računalno-algoritamski način razmišljanja (eng. izraz: *computational thinking*) žarište je istraživanja u području informacijsko-komunikacijske tehnologije i pokušava ga se što više integrirati u kurikulum za osnovnoškolski odgoj i obrazovanje zbog pozitivnog utjecaja na kognitivne sposobnosti (Kazimoglu i suradnici, 2012). Računalno-algoritamski način razmišljanja opisan je kao „misaoni proces uključen u formuliranje problema čija rješenja mogu biti predstavljena kao koraci u računalnom programiranju i algoritmi“ (Aho, 2012:1). Ovakav se način razmišljanja razvija rješavanjem problema koji su povezani sa stvarnim životom, a pri tome se trebaju primijeniti i znanja iz drugih područja, poput prirodoslovlja i matematike (MZOS, 2016b).

Budin (2018) također tumači da je s računalno-algoritamskim načinom razmišljanja povezan i matematički način razmišljanja (eng. izraz: *mathematical thinking*) koji bi se trebao sustavno razvijati. Wing (2006) objašnjava da je razvoj računalno-algoritamskog načina razmišljanja preporučljiv za svakoga te ga je poželjno uključiti u dječje analitičke sposobnosti. Ista autorica navodi da se način rješavanja problema u svakodnevnom životu može poboljšati učenjem kako računalo rješava određeni problem. Svakodnevnim vježbanjem, učenjem sintakse, pronalaženjem i otklanjanjem pogrešaka mogu se stvoriti dobro ustanovljene strukture u mozgu i za rješavanje drugih problema, što i jesu osnovni koncepti računalno-algoritamskog načina razmišljanja (Wing ,2006).

3. IZAZOVI POUČAVANJA PROGRAMIRANJA I RAZLIČITI PRISTUPI STJECANJU ZNANJA IZ PROGRAMIRANJA

Prilikom uključivanja novih tehnologija u nastavne programe najčešće se pojavljuju četiri izazova (Vitolo, 2011): ograničenje trenutnih kurikuluma, sposobnost i mogućnosti koje su dostupne nastavnicima kako bi postali digitalno kompetentni, utvrđivanje nastavnih materijala potrebnih za poboljšanje učenja te prihvaćanje novih tehnologija kao značajnih od strane odgovornih za donošenje odluka. Ako su ove četiri stavke zadovoljene i računalno programiranje postane obavezno u obrazovanju djece mlađe školske dobi, učitelji će morati prilagoditi svoj pristup poučavanju na način da se učenicima olakša njegovo usvajanje (Bubica, 2014).

Na neuspješno učenje programiranja najviše utječe nemogućnost pridržavanja slijeda izvođenja programa, nerazumijevanje sintakse programskih jezika te slaba upotreba strategija za rješavanje problema (Kalelioğlu, 2015). Robins i suradnici (2003) tumače da se učitelji ne trebaju usmjeriti na poučavanje učenika, nego na njihovo samostalno učenje i učinkovitu komunikaciju između učenika i učitelja kako bi se učenici osposobili za cjeloživotno učenje.

Djeca najviše vole učiti uz vizualne prikaze, poput dijagrama, videozapisa, animacija, ali i uz verbalno objašnjavanje koraka te iz samostalnih pokušaja (Zhang i suradnici, 2014). Zaharija i suradnici (2012) objašnjavaju da djeca mogu riješiti samo one probleme koji su povezani sa stvarnim svijetom jer su u fazi konkretnih operacija pa još nisu sposobna za apstraktno, hipotetičko mišljenje. Ovi autori tumače da treba razviti pristup poučavanju računalnog programiranja koji će omogućiti učenicima povezivanje sa stvarnim svijetom i pomoći im u razvijanju logičkog načina razmišljanja i sposobnosti rješavanja problema.

3.1. Računalna igra u učenju programiranja

Leutenegger i Edgington (2007) tvrde da su društvene promjene uzrokovale potrebu za učenjem računalnog programiranja. Programiranje računalnih igara učenicima je zanimljivo i motivirajuće pa programska načela uče u zabavnom okruženju (Kazimoglu i suradnici, 2012). Pri takvom načinu učenja najvećim dijelom se motiviraju početnici u učenju programiranja jer su zainteresirani dok uče, a postoji vizualna komponenta koja im omogućuje uvidjeti pogreške (Leutenegger i Edgington, 2007).

Kazimoglu i suradnici (2012) tumače da je izrada ovakvih igara najbolji pristup poučavanju i uvodnom učenju programiranja, a u tom kontekstu ih nazivaju *ozbiljnim igrama*

(eng. izraz: *serious game*), budući da su razvijene u obrazovnome smislu i usmjerene su na učenje. U svom su istraživanju objasnili da računalne igre imaju pozitivan utjecaj na učenikove ishode učenja, što pokazuje da računalne igre mogu biti učinkovit obrazovni alat za učenje računalnoga programiranja, a takav način učenja zove se *učenje temeljeno na igri* (eng. izraz: *Game-Based Learning*; GBL).

U učenju računalnog programiranja mogu se koristiti i programske igre. Primjer takve igre je *RoboCode* koja je razvijena od strane International Business Machines Corporation (IBM) 2001. godine za učenje programskog jezika *Java* te ostalih programskih jezika na zabavan način (O'Kelly i Gibson, 2006). U ovoj programskoj igri kodira se robot za borbu protiv drugih robota u borbenoj areni, a igrač je programer robota koji nema izravan utjecaj na igru, već programira kako će se robot ponašati i reagirati na događaje u borbenoj areni (Larsen, 2013). Programiranjem računalnih igara učenicima se omogućuje bolje razumijevanje teorije i razvoj većeg samopouzdanja tijekom izrade programskog koda (Hartness, 2004). Često se za poučavanje programiranja koriste posebne računalne programske igre (npr. *Colobot*, *Catacombs*, *Elemental*).

Postoje i vizualni programski jezici, kao što su *Scratch* i *Alice*, no Kazimoglu i suradnici (2012) smatraju da, iako su oni učinkoviti u uvodnom učenju programiranja, ne ulaze u GBL jer im nedostaju određene karakteristike poput pravovremenih povratnih informacija i mehanizma za nagrađivanje.

4. UČENJE MATEMATIKE PUTEM PROGRAMIRANJA

Smatra se da su matematika i računalno programiranje čvrsto povezani zato što su matematičari sudjelovali u početcima razvoja računala i zato što računalni programeri koriste logiku, funkcije i procedure u ostvarivanju svojih ciljeva (Misfeldt i Ejsing-Duun, 2015). Pea i Kurland (1983) navode da neki istraživači smatraju da matematičko znanje nije povezano s računalnim programiranjem, ali činjenica je da oni koji znaju programirati, najčešće znaju i višu razinu matematike.

Prvi začetnik ideje da bi se računalnim programiranjem mogla poučavati matematika bio je Seymour Papert izradivši programski jezik *Logo* u kojemu su djeca naredbama pokretala kornjaču na ekranu i tako vježbala geometriju (Papert, 1980). Tvrdio je da djeca mogu učinkovito učiti izrađujući različite geometrijske likove i zanimljive uzorke pomoću traga koji ostavlja kornjača pa je opisao načine pomoću kojih se u programskom jeziku *Logo* može poučavati geometrija.

Kahn i suradnici (2011) u svom su istraživanju učenike učili o pojmu beskonačnosti i kardinalnosti skupa koristeći pri tome animirano programsko okruženje *ToonTalk*. Kao dio projekta *WebLab*s u istraživanju su sudjelovali učenici između devet i dvanaest godina iz šest europskih država. Matematički pojmovi poput parnih i neparnih brojeva, cijelih brojeva i uređenih parova, veličine skupova te racionalnih brojeva prikazani su tijekom šest različitih aktivnosti u kojima se trebao programirati robot koji izvodi zadatke u *ToonTalk*-u. Autori su zaključili da su učenici uspješno rješavali sve aktivnosti, iako se neki od ovih pojmova uče u višim razredima te su sa zanimanjem istraživali i promišljali o pojmu beskonačnosti i kardinalnosti skupa.

Istraživanje provedeno u Danskoj kao dio projekta *Children as Learning Designers in a Digital School* u petom razredu osnovne škole opisali su Misfeldt i Ejsing-Duun (2015). U tom su istraživanju učenici programirali računalne igre te tako učili o koordinatnom sustavu i orijentaciji. Isti autori navode da su učenici uz pomoć učitelja dolazili do zaključaka o tome kako matematičke pojmove zapisati pomoću naredbi pa iz tog razloga tvrde da su u učenju matematike putem programiranja vrlo važni učiteljevi pristupi poučavanju.

Istraživanje o načinu na koji nastaje matematičko razmišljanje dok djeca programiraju u programskom jeziku *Scratch* proveo je Calder (2010) koji je okupio 26 ispitanika šestog razreda osnovne škole. Učenici su nakon dva tjedna rada i upoznavanja sa programskim jezikom *Scratch* dobili zadatak osmisliti matematičku igru vezanu za olakšavanje razumijevanja pojma broja za učenike u prvom razredu. Nakon analize izrađenih

matematičkih igara autor je zaključio da su učenici realizirali svoje matematičke ideje koristeći pri tome geometriju i koordinatni sustav. On smatra da su izradom ovakvih igara učenici razvili prostorni zor, osjećaj za smještanje u koordinatni sustav i sposobnost za mjerenje vremena te da su usvojili pojam i mjerenje kuta. Razvoj matematičkog razmišljanja tijekom programiranja kod učenika je potpomognut samostalnim istraživanjem i kreativnim rješavanjem problema te uporabom logičkog načina razmišljanja (Calder, 2010). Prema Calder (2010), neki od prijedloga zadataka kojima učenici mogu poboljšati matematičko razmišljanje radom u programskom jeziku *Scratch* su izrada programa koji pomiče lik u obliku kvadrata, pravokutnika ili trokuta, izrada tlocrta učionice te izrada programa kojima se vježbaju dosad naučeni pojmovi.

5. POUČAVANJE UČENIKA MLAĐE ŠKOLSKE DOBI PROGRAMIRANJU

Mannila i de Raadt (2006) saželi su kriterije koji programski jezik čine prikladnim za korištenje u uvodnom učenju računalnog programiranja. Oni smatraju da bi programski jezici koji se koriste u uvodnom učenju programiranja trebali imati jednostavnu sintaksu i prirodnu semantiku, mogućnost rješavanja problema u koracima, interaktivno grafičko-korisničko sučelje, otvoren pristup, mogućnost korištenja i izvan obrazovanja, mogućnost proširenja, a njegova korisnost morala bi biti jasna i u budućnosti. Prema ovim kriterijima ustanovili su da su najbolji programski jezici za poučavanje programiranja *Python* i *Eiffel*, no ne umanjuju vrijednost ostalih programskih jezika.

Kako bi programski jezici bili lakši i razumljiviji učenicima, programska okruženja koja se koriste za poučavanje programiranja imaju pojednostavljenu sintaksu, jednostavne naredbe, a naglasak je stavljen na izbjegavanje sintaktičkih pogrešaka (Bubica, 2014). Prema Pears i suradnici (2007), jedan od najutjecajnijih pristupa u poučavanju računalnog programiranja je *objektno orijentirano programiranje* koje se usmjerava na objekte kao dijelove koda i pomoću njih se izražava rješenje problema. Objektno orijentirano programiranje upotrebljava najveći broj obrazovnih instituacija kod poučavanja računalnog programiranja (Pears i suradnici, 2007). Détienne (1997) tumači da se koristi i u obrazovanju i u industriji te da brojni istraživači smatraju da je ovakav pristup najbolji za učenje računalnog programiranja. Objektno orijentirano programiranje započelo je programskim jezikom *SmallTalk*, a njegove najvažnije karakteristike su *objekti* kao prirodne značajke problemske domene i *slučajevi klasa*, gdje klasa određuje ponašanje objekata (Begel, 1997; Robins i suradnici, 2003). Procedure su smještene unutar objekata s kojima su povezane, a objekti imaju svoje atribute i znaju kako izvoditi određene zadatke te postoji mogućnost nasljeđivanja metoda među objektima (Begel, 1997).

U vizualno-grafičkim programskim okruženjima programi se izrađuju na način da se grafički blokovi naredbi za likove na ekran slažu jednostavnom metodom *povuci* i *ispusti* (Grover i Pea, 2013). Budući da djeca istovremeno mogu vidjeti program koji njihov lik izvodi, u njihovom poučavanju računalnog programiranja najviše se koriste vizualni programski jezici jer se uči programirati kroz igru i istraživanje te su učenici zbog toga visoko motivirani (Grover i Pea, 2013). De Raadt i suradnici (2002) u svom su istraživanju htjeli utvrditi koji se programski jezici koriste u uvodnom učenju programiranja na trideset i dva australska sveučilišta. Telefonskim anketiranjem saznali su da su to *Java*, *Visual Basic*, *C++*,

C, *Haskell*, *Eiffel*, *Ada*, *Delphi* i *JBase*. Od ispitanika su prikupili i podatke o kriterijima prilikom izbora prikladnog programskog jezika za uvodno učenje programiranja. Među kriterijima najviše se ističu industrijska važnost, zahtjevi tržišta te zahtjevi studenata, a zatim i pedagoške prednosti programskog jezika, izgled grafičko-korisničkog sučelja, popularnost i cijena. De Raadt i suradnici (2002) su također provjerili koji se pristupi poučavanju računalnog programiranja koriste na sveučilištima te su zaključili da 86,00 % sveučilišta koristi objektno orijentirani pristup u poučavanju računalnog programiranja, a 14,00 % proceduralni pristup.

5.1. Logo

Seymour Papert naziva se ocem programskog jezika *Logo* budući da je bio važan član Artificial Intelligence Laboratory grupe na Massachusetts Institute of Technology (MIT) koja je 1960-ih godina radila na njegovu razvoju (Logo foundation, n.d.). Programski jezik *Logo* jedan je od prvih programskih jezika namijenjenih djeci, a njegovim razvojem računalno je programiranje postalo dostupno svima (Begel, 1997). Od sedamnaest kriterija koje bi programski jezik namijenjen poučavanju trebao zadovoljavati, a koje su naveli Mannila i de Raadt (2006), programski jezik *Logo* zadovoljava devet.

Filozofija poučavanja je konstruktivizam jer ima korijene u Piagetovom radu, stoga se temelji na činjenici da učenici stvaraju svoje znanje kroz interakciju sa stvarima i ljudima oko sebe (Logo foundation, n.d.; Clements i Gullo, 1984, prema Papert, 1980). Korištenjem ovog programskog jezika zastupljen je „piagetovski pristup“, odnosno cilj je učenje bez poučavanja (Logo foundation, n.d.; Clements i Gullo, 1984, prema Papert, 1980).

U programskom jeziku *Logo* glavni lik je kornjača koja se može kretati u svim smjerovima po ekranu i izvršavati naredbe, prilikom čega predstavlja grafički objekt (Logo foundation, n.d.). Začetnici ističu važnost takve interaktivnosti i pojednostavljenja jezika na prirodan i razumljiv način zbog mogućnosti poučavanja djece i neprogramera računalnome programiranju (Logo foundation, n.d.). Begel (1997) tumači da je programski jezik *Logo* omogućio djeci učenje procedura i rekurzija, olakšano uklanjanje pogrešaka u programu jer mogu vidjeti koji točno zadatak kornjača ne izvršava onako kako su htjeli, učenje metodom iskušavanja svih naredbi i razumijevanje načina na koji program funkcionira. Osim što je djeci olakšana sintaksa, tvorci programskog jezika *Logo* imali su za cilj olakšati savladavanje matematičkih koncepata (Papert, 1980). Begel (1997) nadalje smatra da kornjača pomaže djeci u shvaćanju matematike, posebice geometrije, jer djeca okreću kornjaču u različitim smjerovima i pomoću traga koji kornjača ostavlja mogu crtati geometrijske likove. Zbog te

činjenice programski jezik *Logo* vezan je uz pojam *geometrija kornjače* (eng. izraz: *Turtle Geometry*) jer se „kreira okruženje u kojemu nije cilj da djeca nauče formalna pravila, nego da steknu dovoljan uvid u način na koji se oni kreću u prostoru kako bi se omogućio prijenos svijesti o vlastitom kretanju u prostoru u programe koji čine da se i kornjača pokreće“ (Papert, 1980:205).

Pea (1983) smatra da je *Logo* zanimljiv većini istraživača zato što omogućuje učenicima uvod u učenje programiranja i razvoj metakognitivnih sposobnosti kao što su planiranje i rješavanje problema. Programski jezik *Logo* doživio je veliki uspjeh i primjenu u osnovnoj školi većinom zbog obrazovnih knjiga koje su napisali tvorci ovoga programskog jezika, a najvećim doprinosom smatra se knjiga *Mindstorms* (Begel, 1997). Pea (1983) objašnjava da je učenje programiranja u programskom jeziku *Logo* teško ostvariti bez poučavanja te da učitelji moraju uložiti velike napore za razvoj vlastitih vještina kako bi mogli usmjeravati učenike i direktno ih poučavati programiranju, što je suprotno od tvrdnji da učenici mogu učiti samostalno i spontano razvijati svoje vještine. Neke od verzija programskog jezika *Logo* jesu *UCB Logo*, *MSW Logo*, *FMS Logo*, *Terrapin Logo*, *MicroWorlds* te *KinderLogo* u kojima se može raditi s jednom ili više kornjača koje paralelno izvode zadatke (Logo foundation, n.d.).

Smatra se da je programski jezik *Logo* jednostavan za korištenje početnicima u programiranju koji u njemu mogu napredovati i nakon nekog vremena sami izrađivati programe, a ima i dobro razvijenu zajednicu korisnika (Resnick i suradnici, 2009). Vizualno-grafička programska okruženja prikladna su za učenje programiranja početnika u programiranju jer su oslobođena mogućnosti pogrešaka u sintaksi pa se učenici mogu koncentrirati na osmišljavanje i izradu vlastitih programa (Grover i Pea, 2013). Ovakvim pristupom povećava se kreativno programiranje koje je zasnovano na dječjim interesima, kreativnosti i mašti, prilikom čega se mlađim korisnicima omogućuje izrada vlastitih sadržaja, a naglasak je na znanju, vještinama i konceptima koje oni spontano usvajaju dok izrađuju svoje programe (Brennan i suradnici, 2014).

5.2. Scratch

Lifelong kindergarten grupa je 2003. godine na Massachusetts Institute of Technology (MIT) blisko surađivala s tvrtkom Lego tijekom razvoja njihova robota *Lego Mindstorms* pa se tu javila ideja i o izradi programskog jezika *Scratch* (Resnick i suradnici, 2009). Promatrajući djecu i njihove reakcije tijekom slaganja kockica poželjeli su napraviti

programski jezik koji bi djecu podsjećao na slaganje kockica (Bubica i suradnici, 2013). U programskom jeziku *Scratch* postoje već ugrađeni blokovi naredbi koji se slažu kao kockice i tako tvore program, a oblikovani su tako da se slažu samo na one načine koji imaju sintaktički smisao, zbog čega ne može doći do pogreške u sintaksi (Resnick i suradnici, 2009). Osim toga, grafičko-korisničko sučelje sa *povuci* i *ispusti* interakcijom omogućuje jednostavno upravljanje svim blokovima naredbi. Resnick i suradnici (2009) tumače da programe izvode grafički objekti, odnosno *likovi* (eng. izraz: sprites).

Dijelovi grafičko-korisničkog sučelja u ovom programskom jeziku su paleta blokova, područje skripte i pozornica. U paleti blokova nalaze se sve naredbe koje se mogu koristiti prilikom izrade programa. Naredbe su podijeljene po kategorijama od kojih se neke odnose na kretanje glavnog lika, izgled glavnog lika i pozornice, debljinu i boju traga koje ostavljaju likovi te na naredbe za upravljanje događajima na pozornici. Svaki blok ima svoju boju što korisnicima vizualno olakšava njihovo korištenje. Osim različitim bojama, korisnicima se oblikom olakšava njihova primjena, npr. petlje su oblikovane u slovo C kako bi djeca samostalno došla do zaključka da neki blokovi naredbi trebaju ići unutar nje, a naredbe koje moraju imati argument oblikovane su ovalno. Mjesto gdje se naredbe dovlače i slažu u blokove naredbe naziva se područje skripti, a nalazi se na desnoj strani grafičko-korisničkog sučelja. Blokovi naredbi ne moraju biti uredno posloženi da bi se program izvodio, nego korisnik slaže svoje blokove kako njemu odgovara, pri čemu se očituje sloboda i individualnost svakog pojedinca uzimajući u obzir njegov stil učenja. Područje u kojemu se nalazi lik koji izvodi zadani program naziva se pozornica. Likovi i pozornica mogu se mijenjati tako da ih se izabere iz ugrađene knjižnice, nacrta, učita iz datoteke ili snimi pomoću kamere.

Maloney i suradnici (2010) tvrde da je životnost likova važna odlika programskog jezika *Scratch*. Oni smatraju da se životnost likova očituje u tome što se u bilo kojem trenutku može kliknuti na naredbu ili na samo neki dio grafičkog bloka u području skripte kako bi lik započeo izvoditi program, a to korisnicima omogućuje paralelno testiranje programa, otklanjanje pogrešaka i poboljšanje samoga programa. S obzirom na to da učenici u programskom jeziku *Scratch* mogu eksperimentirati s blokovima prilikom slaganja programa, kombinirati manje jedinice u veće dijelove i otkrivati funkcionalnost samih grafičkih blokova, isti autori smatraju da je prikladan za poticanje učenja.

Programski jezik *Scratch* ima još jednu važnu karakteristiku, a to je postojanje velike zajednice korisnika. Postojanje ovakve zajednice omogućuje razmjenjivanje ideja i gotovih programa s korisnicima diljem svijeta te pružanje podrške i razvijanje međusobne suradnje.

Na službenoj internetskoj stranici korisnici mogu *online* izraditi programe ili preuzeti programski jezik *Scratch* i instalirati na svoje računalo. Gotove programe koje korisnici postave na službenu internetsku stranicu ostali korisnici mogu pokretati, komentirati, preuzimati i mijenjati prema vlastitim potrebama.

5.3. Alice

Programsko okruženje *Alice* olakšava izradu animacija, interaktivnih priča i programiranje jednostavnih 3D igara (Alice, n.d.). Cooper i suradnici (2000) navode da ga je izradio Randy Pausch na Carnegie Mellon University, a glavni mu je cilj bio omogućiti početnicima razvoj zanimljivih virtualnih okruženja i istražiti novi medij interaktivne 3D grafike, zbog čega programiranje u programskom jeziku *Alice* više sliči na izradu filma ili videoigre nego na pravo računalno programiranje. Izrada programa u programskom okruženju *Alice* sastoji od dva dijela: stvaranje scenarija i izrada programskog koda, pri čemu stvaranje scenarija uključuje definiranje glavnih objekata i objekata u pozadini, postavljanje početne scene te planiranje niza naredbi i akcija koje će se odvijati (Boljat, 2014). Prema istom autoru, izrada programskog koda odnosi se na slaganje naredbi jednostavnim principom *povuci* i *ispusti*, a osim naredbi postoje i izrazi za matematičke operacije nad podacima te kontrolne strukture za uvjetno izvođenje i petlje.

Rad u ovom programskom jeziku prilagođen je djeci zbog jednostavnog interaktivnog sučelja, *povuci* i *ispusti* principa za slaganje naredbi u blokove, ugrađenih metoda za pojedini objekt, velikog izbora objekata i pozadina te mogućnosti pokretanja programa u koracima kako bi se testirao i izmijenio (Cooper i suradnici, 2000). Cooper i suradnici (2000) navode da programski jezik *Alice* omogućuje korisnicima kontroliranje pojavnosti i ponašanja objekata jer su akcije animirane pa se samim time može određivati i njihovo trajanje. Boljat (2014) smatra da je programski jezik *Alice* prikladan za objektno orijentirani pristup u poučavanju programiranja jer se koristeći *Alice* temeljni pojmovi (objekti, metode i svojstva) intuitivno usvajaju.

Grafičko-korisničko sučelje programskog okruženja *Alice* sastoji se od prozora u kojem se pokazuje svijet koji se gradi (eng. izraz: *world window*), popisa svih objekata koji postoje u svijetu (eng. izraz: *object tree*), područja za detalje o svojstvima, metodama i funkcijama u svijetu (eng. izraz: *worlds details*), područja za uređivanje (eng. izraz: *editor area*) i područja događaja (eng. izraz: *events area*). *Alice* je programsko okruženje koje se može besplatno preuzeti sa službene internetske stranice, a trenutno je omogućeno

preuzimanje triju verzija ovog programskog okruženja (*Alice 2*, *Alice 3 with Netbeans*, *Alice 3*). Sve tri verzije prikladne su za učenje računalnog programiranja, no prema iskustvu u radu s djecom mlađe školske dobi autori tumače da je *Alice 2* jednostavniji, stoga i prikladniji za početno učenje programiranja (Alice, n.d.).

5.4. Kodu

Programski jezik *Kodu* namijenjen je djeci i koristi se u izradi virtualnih 3D svjetova, a u njemu učenici mogu istraživati i igrati gotove igre, samostalno programirati ponašanje likova i stvarati vlastite igre te ih onda igrati i dijeliti s prijateljima (Sokol i Kralj, 2010). U navedenom programskom jeziku nema upisivanja varijabli, nego se sve odabire pokazivačem ili tipkovnicom, čime se smanjuje mogućnost pogreške prilikom pisanja programskog koda.

Pri dnu grafičko-korisničkog sučelja programskog jezika *Kodu* nalaze se svi alati koji se mogu koristiti tijekom izrade programa: alat za vraćanje na glavni izbornik, alat za pokretanje programa, alat za pomicanje kamere, alat za izbor i programiranje objekata, alat za izradu putanje, tri alata za uređivanje terena i alat za dodavanje vode, alat za brisanje te alat za promjenu postavki. Sokol i Kralj (2010) opisuju da je programiranje igara u *Kodu* vrlo jednostavno jer se vizualni dijelovi slažu u rečenice oblika *Kada* uvjet *Učini* ovo (eng. izraz: *When condition Do this*).

Kodu također promiče objektno orijentirani pristup učenju programiranja jer usvajanje samog programiranja dolazi intuitivno. Koristeći ovaj program korisnici mogu kreirati vlastiti 3D virtualni svijet, a na njegovoj službenoj internetskoj stranici postoje već izrađeni virtualni svjetovi i gotove igre. Ako je korisnik početnik u programiranju, može se poslužiti i uputama za izradu 3D svjetova koje su dostupne na službenoj internetskoj stranici. Kada se igra izradi, može se podijeliti s ostalim korisnicima koji je mogu igrati, komentirati i ocijeniti. Grafičko-korisničko sučelje programskog jezika *Kodu* na engleskom je jeziku, ali svojom jednostavnošću i već ugrađenim izborom naredbi olakšava korištenje pa je stoga prikladan za uvodno učenje programiranja za djecu mlađe školske dobi koja posjeduju osnovno znanje o korištenju računala.

6. RAČUNALNO PROGRAMIRANJE U OSNOVNOŠKOLSKOJ NASTAVI U REPUBLICI HRVATSKOJ

U Republici Hrvatskoj informatika se poučava kao izborni nastavni predmet od prvog do osmog razreda osnovne škole. Prema *Nacionalnom okvirnom kurikulumu* (MZOS, 2011), računalno programiranje nalazi se u temi *Informacijska i komunikacijska tehnologija* te podtemi *Rješavanje problema pomoću računala*. U istom je dokumentu navedeno da su ishodi učenja za osnovnoškolsko poučavanje računalnog programiranja „utvrditi da uporabom prikladnih programskih pomagala mogu i sami stvarati vlastite male programe, upotrebljavati interaktivna programska okruženja za grafičko sklapanje jednostavnih programa čiji se učinak odmah vidi na zaslonu monitora, jednostavnim naredbama za pravocrtno kretanje i okretanje likova na zaslonu monitora crtati jednostavne geometrijske oblike tragovima koje likovi ostavljaju na svojemu putu, prepoznati da nizovi naredbi čine program koji se može pohraniti u datoteku i kasnije opet pokrenuti i preoblikovati“ (MZOS, 2011:165-166).

Prijedlog *Nacionalnog kurikuluma za osnovnoškolski odgoj i obrazovanje* (MZOS, 2016b) usklađen je s temeljnim postavkama Cjelovite kurikularne reforme obrazovanja u Republici Hrvatskoj. U okviru *Tehničkog i informatičkog područja kurikuluma* nalazi se domena *Rješavanje problema i programiranje* koja „obuhvaća razvoj vještina i sposobnosti te usvajanje znanja potrebnih za razvijanje algoritamskoga načina razmišljanja te primjenu u različitim problemskim situacijama“ (MZOS, 2016b:31).

U sklopu Cjelovite kurikularne reforme obrazovanja doneseni su i prijedlozi predmetnih kurikuluma pa se tako u prijedlogu *Nacionalnog kurikuluma nastavnog predmeta Informatika* (MZOS, 2016a) računalno programiranje nalazi u domeni *Računalno razmišljanje i programiranje*. Neki od ishoda učenja za prva četiri razreda osnovne škole su rješavanje jednostavnog logičkog zadatka, praćenje i prikazivanje slijeda koraka potrebnih za rješavanje nekog jednostavnog zadatka, analiziranje niza uputa koje izvode jednostavan zadatak, stvaranje programa korištenjem vizualnog okruženja u kojem se koristi slijed, ponavljanje, odluka i ulazne vrijednosti te rješavanje složenijih logičkih zadataka s uporabom računala ili bez uporabe računala (MZOS, 2016a).

U *Nacionalnom kurikulumu nastavnog predmeta Informatika* (MZOS, 2016a) također se objašnjava važnost poznavanja programiranja, algoritama i struktura podataka, budući da računalno programiranje razvija sposobnost rješavanja problema, modeliranje, apstrahiranje, preciznost i sustavnost, čime se učenici osposobljavaju za inovativnost, stvaralaštvo i poduzetnost u budućem profesionalnom životu. Učenici trebaju samostalno izrađivati vlastite projekte i realizirati svoje ideje, zbog čega se računalno programiranje smatra sve važnijom sastavnicom u obrazovanju djece u suvremenom svijetu (MZOS, 2011).

7. METODOLOGIJA ISTRAŽIVANJA

Cilj ovog istraživanja bio je provjeriti mogu li se neki programski sadržaji iz matematike u četvrtom razredu osnovne škole usvojiti pomoću programskog jezika *Scratch* te provjeriti koliko su učenici zadovoljni radom u programskom jeziku *Scratch*.

U istraživanju su sudjelovali učenici četvrtog razreda Osnovne škole „Vladimir Nazor“ u Komletincima (N = 12). Distribucija učenika prema spolu u trenutku provedbe istraživanja prikazana je u tablici 1. iz koje je vidljivo da je u istraživanju sudjelovao jednak broj učenika (N = 6) i učenica (N = 6).

Tablica 1. Ispitanici prema spolu.

SPOL	UČENICI	
	N	%
Muški	6	50,00
Ženski	6	50,00

Ovo istraživanje provedeno je tijekom veljače 2018. godine. Prije početka istraživanja dobivena je suglasnost roditelja, ravnateljice i učiteljice za sudjelovanje učenika u istraživanju. Učenici su bili upoznati sa svrhom istraživanja te su mogli odustati od sudjelovanja u istraživanju u bilo kojem trenutku.

Istraživanje je bilo podijeljeno u dva dijela. Prvi dio istraživanja obuhvaćao je upoznavanje s radom u programskom jeziku *Scratch*, što je napravljeno tijekom redovite nastave u sklopu jednog školskog sata. Na početku sata učenici su bili upoznati s osnovnim pojmovima vezanim uz računalno programiranje te su im bila prikazana tri primjera gotovih programa izrađenih u programskom jeziku *Scratch*. Nakon što su učenici samostalno pokrenuli programski jezik *Scratch* na svojim računalima, analizirali su dijelove grafičko-korisničkog sučelja ovog programskog jezika. Učenici su izvršili par osnovnih naredbi, samostalno istraživali koje se sve naredbe mogu koristiti, a onda i samostalno napisali svoj prvi program. Svaki učenik je na papiru crtao skicu rješenja i zapisivao dijelove programa te istovremeno slagao naredbe u programskom jeziku. Nakon toga učenici su odgovorili na tri pitanja koja su služila za utvrđivanje njihove razine zadovoljstva radom u ovom programskom jeziku, njihove procjene korisnosti ovog programa u njihovom obrazovanju te spremnost učenika da ponovno programiranju u njemu.

U drugom dijelu istraživanja na satu matematike učenici su pokušali usvojiti nove programske sadržaje pomoću programskog jezika *Scratch*. Ovaj dio istraživanja proveden je tijekom obrade nastavne teme *Pravokutnik i kvadrat*. Obrazovna postignuća nastavne teme

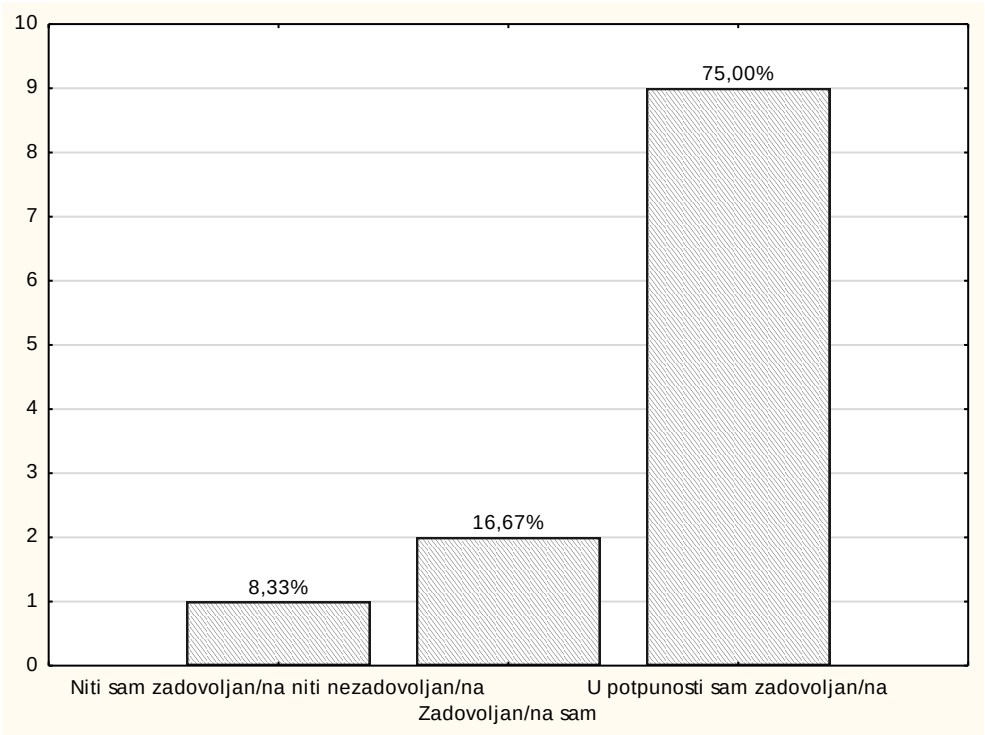
jesu prepoznati i razlikovati pravokutnik i kvadrat, označivati stranice i vrhove pravokutnika i kvadrata te crtati pravokutnik i kvadrat (MZOS, 2006). Svaki je učenik opet istovremeno crtao skicu rješenja i zapisivao dijelove programa te pisao program u programskom jeziku *Scratch*.

Kako bi se utvrdila razina usvojenosti znanja o obilježjima pravokutnika i kvadrata, učenici su riješili test koji je sadržavao pet pitanja. Nakon toga od učenika se tražilo da procijene u kojim bi se nastavnim predmetima programski jezik *Scratch* mogao koristiti s obzirom na usvajanje znanja iz tih predmeta te da ponovno procijene njegovu korisnost u obrazovanju i svoje zadovoljstvo radom u ovom programskom jeziku.

8. REZULTATI I RASPRAVA

Na kraju prvoga i drugog dijela istraživanja svim je učenicima postavljeno pitanje smatraju li da je programski jezik *Scratch* koristan u njihovom obrazovanju. Na temelju zaključnih odgovora svi učenici (N = 12) smatraju da je programski jezik *Scratch* koristan u njihovom obrazovanju.

Nakon prvog dijela istraživanja od učenika se tražilo da na skali od 1 (u potpunosti sam nezadovoljan/na) do 5 (u potpunosti sam zadovoljan/na) iskažu svoje zadovoljstvo radom u programskom jeziku *Scratch*. Na slici 1. prikazani su rezultati iz kojih je vidljivo da jedan učenik (8,33 %) nije niti zadovoljan niti nezadovoljan radom u programskom jeziku *Scratch*, dva učenika (16,66 %) su zadovoljna, a devet učenika (75,00 %) je u potpunosti zadovoljno radom u programskom jeziku *Scratch*.

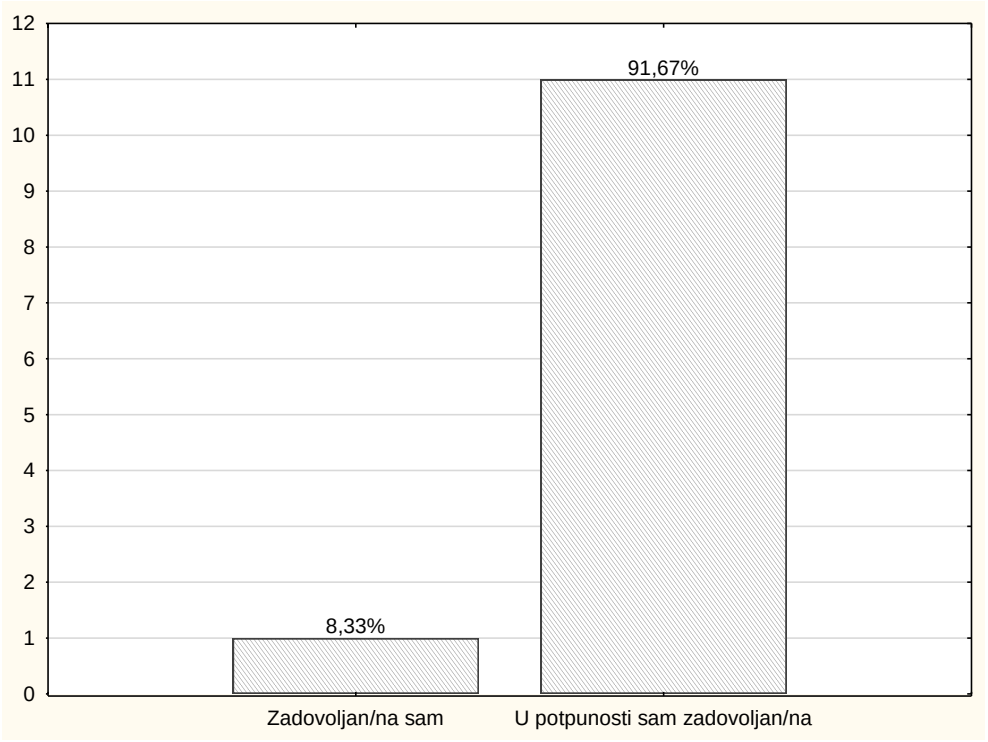


Slika 1. Distribucija učenika prema samoprocijenjenom zadovoljstvu radom u programskom jeziku *Scratch* nakon prvog dijela istraživanja.

Vidljivo je da je većina učenika zadovoljna svojim radom u programskom jeziku *Scratch* iako su prvi put radili u ovom programskom jeziku. Osim toga, učenike se pitalo žele

li opet programirati u programskom jeziku *Scratch*. Samo jedan učenik (8,33 %) je odgovorio da ne želi ponovno programirati, dok su svi ostali odgovorili pozitivno (91,67 %).

Slika 2. prikazuje distribuciju učenika prema samoprocijenjenom zadovoljstvu radom u programskom jeziku *Scratch* nakon drugog dijela istraživanja. Rezultati pokazuju da je 91,67 % učenika (N = 11) u potpunosti zadovoljno radom u programskom jeziku *Scratch*, a 8,33 % učenika (N = 1) je zadovoljno radom u ovom programskom jeziku. Usporedbom rezultata o zadovoljstvu nakon prvog i drugog dijela istraživanja vidljivo je da je zadovoljstvo veće nakon drugog dijela istraživanja, što može biti povezano s činjenicom da su učenici stekli više iskustva te bolje ovladali korištenjem osnovnih naredbi u ovom programskom jeziku.



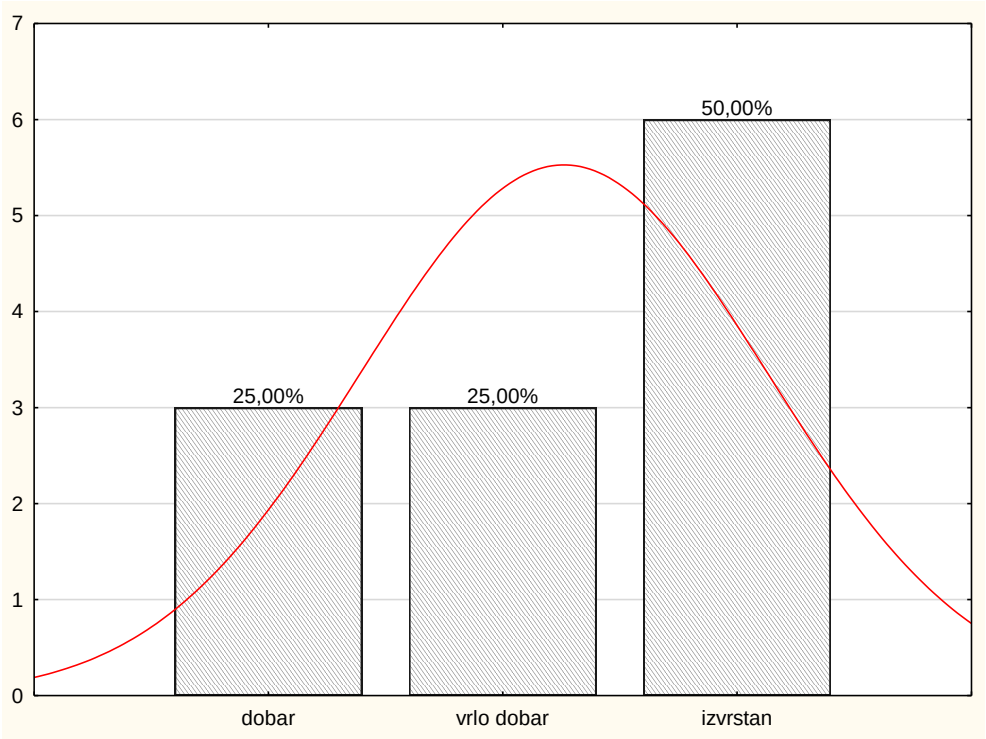
Slika 2. Distribucija učenika prema samoprocijenjenom zadovoljstvu radom u programskom jeziku *Scratch* nakon drugog dijela istraživanja.

Svaki učenik ocijenjen je iz crtanja i označivanja vrhova i stranica likova na papiru te je vrednovan na način prikazan u tablici 2.

Tablica 2. Način vrednovanja učenika iz crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru.

Postotak	0-50 %	51-62 %	63-75 %	76-87 %	88-100 %
Ocjena	Nedovoljan (1)	Dovoljan (2)	Dobar (3)	Vrlo dobar (4)	Izvrstan (5)

Konačne ocjene prikazane su na slici 3. Vidljivo je da je isti broj učenika dobio ocjenu dobar (25,00 %) i vrlo dobar (25,00 %), dok je šest učenika dobilo ocjenu izvrstan (50,00 %).



Slika 3. Prikaz dobivenih ocjena iz crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru.

Mann-Whitney U testom provjereno je postoji li statistički značajna razlika u dobivenim ocjenama iz crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru između dječaka i djevojčica, a razina značajnosti bila je 0,05. Ovim testom pokazalo se

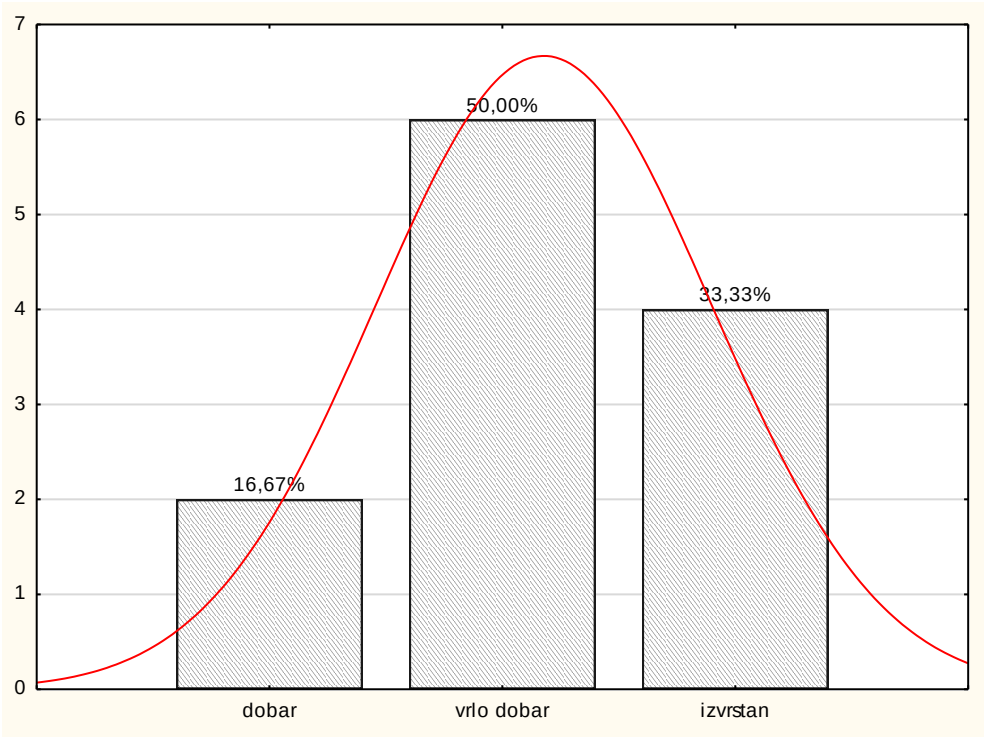
da ne postoji statistički značajna razlika ($U = 12$; $p = 0{,}39$) u ocjenama djevojčica i dječaka iz iz crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru.

Učenici su bili ocijenjeni i za izradu programa prilikom čega se gledao pravilan programski kod za crtanje i označivanje vrhova i stranica pravokutnika i kvadrata u programskom jeziku *Scratch*. Njihovi uradci vrednovali su se na način prikazan u tablici 3.

Tablica 3. Način vrednovanja učenika iz programiranja u programskom jeziku *Scratch* te označivanja vrhova i stranica kvadrata i pravokutnika.

Postotak	0-50 %	51-62 %	63-75 %	76-87 %	88-100 %
Ocjena	Nedovoljan (1)	Dovoljan (2)	Dobar (3)	Vrlo dobar (4)	Izvrstan (5)

Na slici 4. prikazana je distribucija učenika prema ocjenama iz crtanja i označivanja kvadrata i pravokutnika u programskom jeziku *Scratch*. Vidljivo je da je najviše učenika dobilo ocjenu vrlo dobar (50,00 %). Ocjenu dobar dobilo je dvoje učenika (16,67 %), a ocjenu izvrstan četvero učenika (33,33 %).



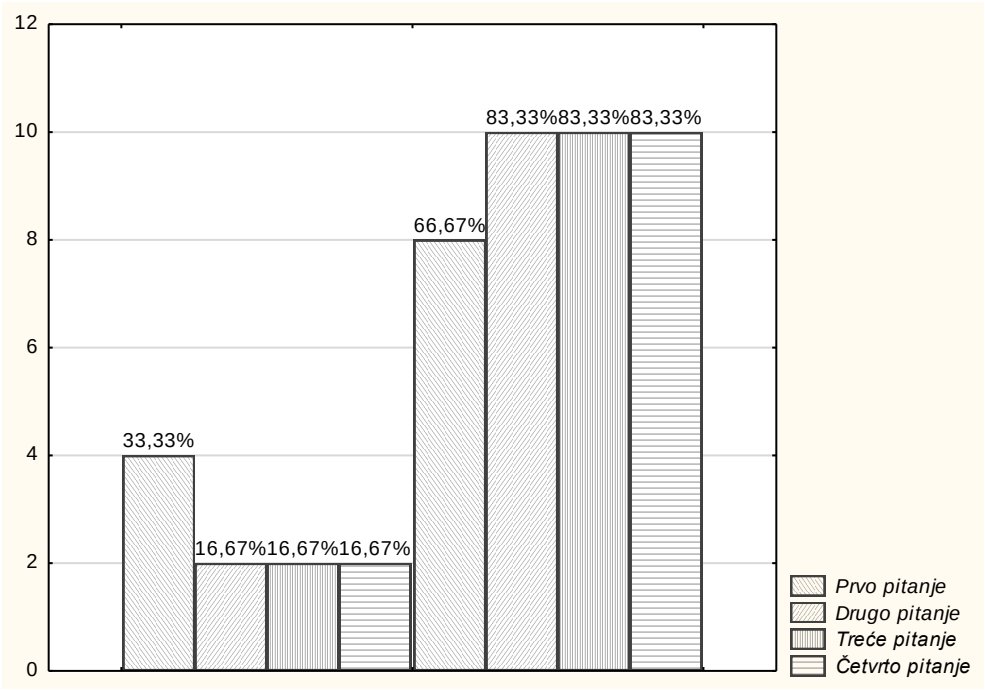
Slika 4. Prikaz dobivenih ocjena iz programiranja u programskom jeziku *Scratch* te označivanja vrhova i stranica kvadrata i pravokutnika.

Mann-Whitney U testom također se htjelo provjeriti postoji li statistički značajna razlika u razini usvojenosti znanja između dječaka i djevojčica iz programiranja u programskom jeziku *Scratch*. Na razini značajnosti 0,05 pokazalo se da ne postoji statistički značajna razlika ($U = 14$; $p = 0,56$) u ocjenama između djevojčica i dječaka.

Prema ocjenama koje su učenici dobili može se reći da je prilikom crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru polovica učenika dobila ocjenu izvrstan (N = 6), a za crtanje i označivanje vrhova i stranica kvadrata i pravokutnika u programskom jeziku *Scratch* polovica učenika dobila je ocjenu vrlo dobar (N = 6). Budući da je šest učenika dobilo ocjenu vrlo dobar, odnosno točno su napisali program koji crta kvadrat i pravokutnik, ali su djelomično točno označili vrhove i stranice kvadrata i pravokutnika, može se zaključiti da su učenici ovladali korištenjem osnovnih naredbi u programskom jeziku *Scratch* jer su učinkovito koristili računalno programiranje u usvajanju matematičkih sadržaja.

Nakon crtanja i označivanja vrhova i stranica pravokutnika i kvadrata na papiru i izrade programa u programskom jeziku *Scratch* učenici su riješili test za provjeru usvojenosti znanja o pravokutniku i kvadratu, a na slici 5. prikazani su rezultati. U prvom pitanju učenici su morali dopisati jesu li kvadrat i pravokutnik geometrijski likovi ili geometrijska tijela. Na ovo pitanje osam učenika (66,67 %) odgovorilo je točno. Četvero učenika (33,33 %) napisalo

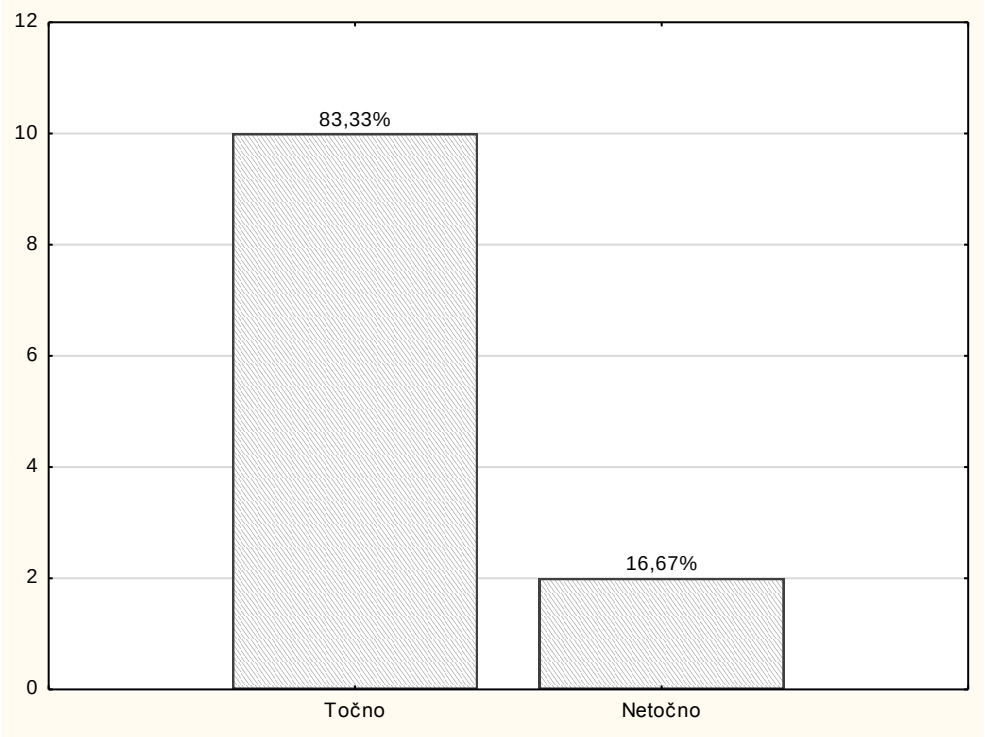
je da su kvadrat i pravokutnik geometrijska tijela što je netočan odgovor. U drugom pitanju učenici su morali dopisati koliko kvadrat ima vrhova, kutova i stranica te jesu li mu stranice jednake ili različite duljine. Deset učenika (83,33 %) dalo je točan odgovor, dok to nisu uspjela dva učenika (16,67 %). Treće pitanje u kojemu su učenici trebali dopisati koje su stranice pravokutnika jednake duljine i četvrto pitanje u kojemu su učenici trebali dopisati koje su vrste unutarnji kutovi u kvadratu i pravokutniku, također je uspješno riješilo deset učenika (83,33 %), a po dva učenika (16,67 %) netočno su odgovorila na postavljena pitanja.



Slika 5. Rezultati testa provjere znanja o obilježjima kvadrata i pravokutnika.

S obzirom na to da je više od 60,00 % učenika uspješno odgovorilo na sve zadatke u testu, može se reći da su učenici uspješno usvojili matematičke sadržaje pomoću programskog jezika *Scratch.* Najviše učenika netočno je odgovorilo na pitanje jesu li kvadrat i pravokutnik geometrijska tijela ili geometrijski likovi (33,33 %). U trećem pitanju dva učenika (16,67 %) nisu točno napisala broj vrhova, kutova i stranica u kvadratu, ali je jedan od njih napisao da kvadrat ima sve stranice jednake duljine unatoč netočnom prvom dijelu zadatka. U četvrtom su pitanju učenici trebali napisati da su kod pravokutnika i kvadrata svi unutarnji kutovi pravi, a od dvoje učenika (16,67 %) koji su netočno odgovorili, jedan je zaključio da svi unutarnji kutovi kod kvadrata i pravokutnika imaju jednaku mjeru.

Osim zadataka u kojima su učenici morali dopisati određene dijelove rečenice, u zadnjem zadatku tražilo se da označe vrhove i stranice nacrtanome kvadratu i pravokutniku. Na slici 6. prikazani su rezultati zadatka za označivanje vrhova i stranica kvadrata i pravokutnika. Može se zaključiti da je zadatak točno riješilo deset učenika (83,33 %), a od dvoje učenika (16,67 %) koji nisu sve označili uspješno, jedan učenik označio je samo vrhove kvadrata i pravokutnika, a drugi je netočno označio stranice pravokutnika.



Slika 6. Distribucija učenika prema zadatku za označivanje vrhova i stranica kvadrata i pravokutnika.

Ukupni rezultati testa provjere znanja o kvadratu i pravokutniku pokazuju da je od pet pitanja u testu na prvo pitanje točno odgovorilo osam učenika (66,67 %), a na ostala četiri pitanja točno je odgovorilo deset učenika (83,33 %). Budući da su učenici prvi put usvajali matematičke sadržaje pomoću programskog jezika *Scratch* ohrabrujući je rezultat da je 66,67 % učenika usvojilo matematičke pojmove putem računalnog programiranja.

Učenici su naveli da bi se programski jezik *Scratch* mogao koristiti za usvajanje znanja iz Hrvatskog jezika, Matematike, Likovne i Glazbene kulture, Engleskog jezika te Prirode i društva (vidi tablicu 4.). Može se pretpostaviti da su učenici prepoznali da slaganjem naredbi u smislene rečenice vježbaju sintaksu i semantiku pa zbog toga smatraju da se programski jezik *Scratch* može primijeniti u nastavnom predmetu Hrvatski jezik. Zbog mogućnosti usvajanja i vježbanja matematičkih pojmova pomoću programiranja, kao što su učenici usvajali obilježja pravokutnika i kvadrata, može se pretpostaviti da su uvidjeli

primjenu ovog programskog jezika i u nastavnom predmetu Matematika. Moglo bi se reći da mišljenje učenika o mogućnosti primjene programskog jezika *Scratch* u nastavnom predmetu Likovna kultura proizlazi iz činjenice da ih upotreba različitih grafičkih objekata, boja te crtanja vlastitih likova, pozadina i raznih drugih oblika u programskom jeziku *Scratch* podsjeća na neka nastavna područja u Likovnoj kulturi.

Tablica 4. Primjena programskog jezika *Scratch* u nastavnim predmetima prema mišljenju učenika.

NASTAVNI PREDMETI	UČENICI	
	N	%
Hrvatski jezik	11	92
Matematika	9	75
Likovna kultura	7	58
Glazbena kultura	4	33
Engleski jezik	4	33
Priroda i društvo	4	33

9. ZAKLJUČAK

Računalno programiranje smatra se alatom kojim se mijenja način razmišljanja (Resnick i suradnici, 2009). Poučavanje računalnog programiranja kod djece mlađe školske dobi ima svoje prednosti, ali i izazove. Kao prednost učenja programiranja navodi se razvoj sposobnosti rješavanja problema, logičkog načina razmišljanja, računalno-algoritamskog načina razmišljanja te poticaj za razvoj apstraktnog mišljenja (Bubica i suradnici, 2013). Izazovi poučavanja računalnog programiranja kod djece mlađe školske dobi očituju se u tome da bi učitelji trebali biti kompetentni za kvalitetno izvođenje nastave računalnog programiranja pomoću kvalitetnih i dostupnih materijala (Vitolo, 2011). Uz prikladne pristupe poučavanju i upotrebom programskih jezika za poučavanje djece mlađe školske dobi matematički koncepti mogu se poučavati u okviru računalnog programiranja (Papert, 1980).

Osnovni cilj ovog istraživanja bio je provjeriti mogu li učenici četvrtog razreda osnovne škole pomoću programskog jezika *Scratch* usvojiti matematičke pojmove kvadrat i pravokutnik, odnosno njihova obilježja, crtanje i označivanje njihovih vrhova i stranica. Analizom i ocjenjivanjem učeničkih programa u programskom jeziku i radova na papiru, učeničkim odgovorima na pitanja te ocjenjivanjem testa za provjeru usvojenog znanja o obilježjima kvadrata i pravokutnika prikazani su rezultati o njihovoj uspješnosti.

Dobiveni rezultati iz crtanja i označivanja vrhova i stranica kvadrata i pravokutnika na papiru pokazuju da je 50,00 % učenika usvojilo matematičke pojmove pravokutnik i kvadrat za ocjenu izvrstan, a 25,00 % učenika to je napravilo za ocjenu dobar i vrlo dobar. Rezultati analize računalnih programa u programskom jeziku *Scratch* pokazuju da je 50,00 % učenika dobilo ocjenu vrlo dobar, 33,33 % učenika dobilo je ocjenu izvrstan, a 16,67 % učenika dobilo je ocjenu dobar. Učenje ovih matematičkih pojmova uz pomoć programskog jezika *Scratch* bilo je uspješno jer nijedan učenik nije dobio ocjenu manju od ocjene dobar (3) za izradu programa u programskom jeziku *Scratch*. Testom se napravila dodatna provjera razine usvojenosti znanja iz obilježja te označivanja vrhova i stranica kvadrata i pravokutnika, a rezultati pokazuju da je na svako od pet pitanja točno odgovorilo barem 66,67 % učenika, odnosno barem osam učenika. Provjereno je i koliko su učenici zadovoljni radom u programskom jeziku *Scratch*, a na kraju istraživanja 91,67 % učenika u potpunosti je zadovoljno radom u ovom programu. Učenici su najveću primjenu programskog jezika *Scratch* uvidjeli u nastavnim predmetima Hrvatski jezik, Matematika i Likovna kultura. Programski jezik *Scratch* poticao je učenike na rad i učenje u novom okruženju te je razvio kod učenika pozitivan stav prema računalnom programiranju.

10. LITERATURA

1. Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal*, *55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074
2. Alice (n.d.). Alice – Tell Stories. Build Games. Learn to Program. Pribavljeno 7. svibnja 2018. s http://www.alice.org/
3. Begel, A. B. (1997). *Bongo--a kids' programming environment for creating video games on the Web* (Doktorska disertacija, Massachusetts Institute of Technology). Pribavljeno 3. ožujka 2018. s https://dspace.mit.edu/bitstream/handle/1721.1/43517/38198678-MIT.pdf?sequence=2
4. Boljat, I. (2014). Novices Programmers Teaching by Alice-Impact on Motivation and Choice of Career. *Život i škola: časopis za teoriju i praksu odgoja i obrazovanja*. Pribavljeno 2. ožujka 2018. s http://bib.irb.hr/datoteka/702050.Alice_-_motivacija_i_izbor_karijere_-_Zivot_i_skola.pdf
5. Brennan, K., Balch, C. i Chung, M. (2014). Creative computing. *Harvard Graduate School of Education*. Pribavljeno 2. ožujka 2018. s http://scratched.gse.harvard.edu/guide/files/CreativeComputing20140806.pdf
6. Bubica, N. (2014). Strategije poučavanja i faktori koji utječu na unapređenje znanja programera početnika. Pribavljeno 18. travnja 2018. s http://www.pmfst.unist.hr/wp-content/uploads/2014/06/Istraziva--ki-seminar1-Bubica.pdf
7. Bubica, N., Mladenović, M. i Boljat, I. (2013, siječanj). Programming as a tool for the development of abstract thinking. U *"Dohvati znanje"-15. CARNetova korisnička konferencija-CUC 2013-*. Pribavljeno 1. ožujka 2018. s https://www.researchgate.net/publication/275317798_Programiranje_kao_alat_za_razvoj_apstraktnog_misljenja
8. Budin, L. (2018). Računalno razmišljanje i programiranje u višim razredima osnovne škole u sklopu novog kurikuluma predmeta informatika [Webinar]. Na *III. webinar edukacije učitelja u sklopu kurikularne reforme Ministarstva znanosti i obrazovanja*. Pribavljeno 6. travnja 2018. s https://mzo.hr/sites/default/files/dokumenti/2018/OBRAZOVANJE/Nacionalni-kurikulumi/Edukacija/prezentacija_webinara_racunalno_razmisljanje_i_programiranje_u_visim_razredima_osnovne_skole.pdf

9. Calder, N. (2010). Using Scratch: An Integrated Problem-Solving Approach to Mathematical Thinking. *Australian Primary Mathematics Classroom*, *15*(4), 9-14. Pribavljeno 1. ožujka 2018. s https://files.eric.ed.gov/fulltext/EJ906680.pdf
10. Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. *Journal of Educational Psychology*, *78*(4), 309. http://dx.doi.org/10.1037/0022-0663.78.4.309
11. Clements, D. H. i Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational psychology*, *76*(6), 1051. http://dx.doi.org/10.1037/0022-0663.76.6.1051
12. Clements, D. H. i Nastasi, B. K. (1999). Metacognition, learning, and educational computer environments. *Information Technology in Childhood Education Annual*, *1999*(1), 3-36. https://doi.org/10.2304/ciec.2002.3.2.2
13. Cooper, S., Dann, W. i Pausch, R. (2000, travanj). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges*, *15*(5), 107-116. Consortium for Computing Sciences in Colleges. Pribavljeno 2. ožujka 2018. s http://cse.unl.edu/~scooper/alice/ccscne00.PDF
14. de Raadt, M., Watson, R. i Toleman, M. (2002). Language trends in introductory programming courses. U *Proceedings of the 2002 Informing Science+ Information Technology Education Joint Conference (InSITE 2002)*, 229-337. Informing Science Institute. Pribavljeno 13. ožujka 2018. shttps://eprints.usq.edu.au/5195/1/de_Raadt_Watson_Toleman_Insite_2002_PV.pdf
15. Derus, S. R. M. i Ali, A. Z. M. (2012). Difficulties in learning programming: Views of students. U *1st International Conference on Current Issues in Education (ICCIE 2012)*, 74-79. Pribavljeno 3. travnja 2018. s https://www.researchgate.net/publication/267338258_Difficulties_in_learning_programming_Views_of_students
16. Détienne, F. (1997). Assessing the cognitive consequences of the object-oriented approach: A survey of empirical research on object-oriented design by individuals and teams. *Interacting with computers*, *9*(1), 47-72. https://doi.org/10.1016/S0953-5438(97)00006-4
17. Fessakis, G., Gouli, E. i Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

18. Grover, S. i Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. https://doi.org/10.3102/0013189X12463051
19. Hartness, K. (2004). Robocode: using games to teach artificial intelligence. *Journal of Computing Sciences in Colleges*, *19*(4), 287-291. Pribavljeno 2. ožujka 2018. s http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.2090&rep=rep1&type=pdf
20. Jenkins, T. (2002, kolovoz). On the difficulty of learning to program. U *Proceedings of the 3rd Annual Conference of the LTSN Centre for Information and Computer Sciences*, *4*(2002), 53-58. Pribavljeno 2. ožujka 2018. s http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.596.9994&rep=rep1&type=pdf
21. Kahn, K., Sendova, E., Sacristán, A. I. i Noss, R. (2011). Young students exploring cardinality by constructing infinite processes. Technology, Knowledge and Learning, *16*(1), 3-34. https://doi.org/10.1007/s10758-011-9175-0
22. Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior*, *52*, 200-210. https://doi.org/10.1016/j.chb.2015.05.047
23. Kazimoglu, C., Kiernan, M., Bacon, L. i Mackinnon, L. (2012). A serious game for developing computational thinking and learning introductory computer programming. *Procedia-Social and Behavioral Sciences*, *47*, 1991-1999. https://doi.org/10.1016/j.sbspro.2012.06.938
24. Larsen, F. N. (2013, veljača). *ReadMe for Robocode*. Pribavljeno 19. travnja 2018. s http://robocode.sourceforge.net/docs/ReadMe.html.
25. Leutenegger, S. i Edgington, J. (2007, ožujak). A games first approach to teaching introductory programming. U *ACM SIGCSE Bulletin*, *39*(1), 115-118. ACM. https://doi.org/10.1145/1227310.1227352
26. Logo foundation (n.d.). What is Logo?. Pribavljeno 24. travnja 2018. s http://el.media.mit.edu/logo-foundation/what_is_logo/index.html
27. Maloney, J., Resnick, M., Rusk, N., Silverman, B. i Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 16. Pribavljeno 27. siječnja 2018. s https://s3.amazonaws.com/academia.edu.documents/6721160/scratchlangandenvironment.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1524837084&Signature=N3YLj8PXNcTmejsARMfdufTdMzA%3D&response-content-disposition=inline%3B%20filename%3DThe_Scratch_Programming_Language_and_Env.pdf

28. Mannila, L. i de Raadt, M. (2006, veljača). An objective comparison of languages for teaching introductory programming. U *Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling 2006*, 32-37. ACM. https://doi.org/10.1145/1315803.1315811
29. Misfeldt, M. i Ejsing-Duun, S. (2015, veljača). Learning mathematics through programming: An instrumental approach to potentials and pitfalls. U *CERME 9-Ninth Congress of the European Society for Research in Mathematics Education*, 2524-2530. Pribavljeno 2. ožujka 2018. s https://hal.archives-ouvertes.fr/hal-01289367/document
30. [MZOS] Ministarstvo znanosti, obrazovanja i sporta RH (2016a), *Nacionalni kurikulum nastavnoga predmeta Informatika – prijedlog*, Zagreb: Ministarstvo znanosti, obrazovanja i sporta RH. Pribavljeno 17. veljače 2018. s http://www.kurikulum.hr/wp-content/uploads/2016/03/Informatika.pdf
31. [MZOS] Ministarstvo znanosti, obrazovanja i sporta RH (2016b), *Nacionalni kurikulum za osnovnoškolski odgoj i obrazovanje – prijedlog*, Zagreb: Ministarstvo znanosti, obrazovanja i sporta RH. Pribavljeno 17. veljače 2018. s http://www.kurikulum.hr/wp-content/uploads/2016/03/NKOO-1.pdf
32. [MZOS] Ministarstvo znanosti, obrazovanja i sporta RH (2011), *Nacionalni okvirni kurikulum za predškolski odgoj i obrazovanje te opće obvezno i srednjoškolsko obrazovanje*, Zagreb: Ministarstvo znanosti, obrazovanja i sporta RH. Pribavljeno 25. veljače 2018. s http://mzos.hr/datoteke/Nacionalni_okvirni_kurikulum.pdf
33. [MZOS] Ministarstvo znanosti, obrazovanja i sporta RH (2006), *Nastavni plan i program za osnovnu školu*, Zagreb: Ministarstvo znanosti, obrazovanja i sporta RH. Pribavljeno 25. veljače 2018. s http://www.azoo.hr/images/AZOO/Ravnatelji/RM/Nastavni_plan_i_program_za_osnovnu_skolu_-_MZOS_2006_.pdf
34. O'Kelly, J. i Gibson, J. P. (2006, lipanj). RoboCode & problem-based learning: a non-prescriptive approach to teaching programming. U *ACM SIGCSE Bulletin*, *38*(3), 217-221. ACM. Pribavljeno 19. travnja 2018. s http://eprints.teachingandlearning.ie/3449/1/OKelly%20and%20Gibson%202006.pdf
35. Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc. Pribavljeno 6. travnja 2018. s http://worrydream.com/refs/Papert%20-%20Mindstorms%201st%20ed.pdf
36. Pea, R. D. (1983). *Logo Programming and Problem Solving*. (Technical Report No. 12). Pribavljeno 2. ožujka 2018. s https://files.eric.ed.gov/fulltext/ED319371.pdf

37. Pea, R. D. i Kurland, D. M. (1983). *On the Cognitive Prerequisites of Learning Computer Programming*. (Technical Report No. 18). Pribavljeno 1. travnja 2018. s https://eric.ed.gov/?id=ED249931
38. Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., ... Paterson, J. (2007, prosinac). A survey of literature on the teaching of introductory programming. U *ACM SIGCSE Bulletin*, *39*(4), 204-223. ACM. Pribavljeno 2. ožujka 2018. s http://www.seas.upenn.edu/~eas285/Readings/Pears_SurveyTeachingIntroProgramming.pdf
39. Prensky, M. (2008). Programming is the new literacy. *Edutopia magazine*. Pribavljeno 2. ožujka 2018. s http://classtap.pbworks.com/f/Prensky+-+Programming:+The+New+Literacy.pdf
40. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... i Kafai, Y. (2009). Scratch: programming for all. *Communications of the ACM*, *52*(11), 60-67. Pribavljeno 1. ožujka 2018. s https://web.media.mit.edu/~mres/papers/Scratch-CACM-final.pdf
41. Robins, A., Rountree, J. i Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer science education*, *13*(2), 137-172. https://doi.org/10.1076/csed.13.2.137.14200
42. Sokol, G. i Kralj, L. (2010, listopad). Kodu – programiranje igara. *"Novi izazovi"-12. CARNetova korisnička konferencija-CUC 2010*. Pribavljeno 3. svibnja 2018. s https://cuc.carnet.hr/2010/images/a1_76c9d.pdf?dm_document_id=139&dm_dnl=1
43. Tay, L., Lim, S., Lim, C. i Koh, J. (2012). Pedagogical approaches for ICT integration into primary school English and mathematics: A Singapore case study. *Australasian Journal of Educational Technology*, *28*(4), 740-754. https://doi.org/10.14742/ajet.838
44. Vitolo, T. M. (2011). Teaching emerging technology: challenges, outcomes, and options. *Journal of Computing Sciences in Colleges*, *26*(3), 75-82. Pribavljeno 5. ožujka 2018. s https://www.researchgate.net/profile/Debra_Major/publication/234803601_A_CS0_course_using_Scratch/links/55ee340d08ae0af8ee19f652/A-CS0-course-using-Scratch.pdf#page=84
45. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35. https://doi.org/10.1145/1118178.1118215

46. Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725. https://dx.doi.org/10.1098%2Frsta.2008.0118
47. Zaharija, G., Mladenović, S. i Boljat, I. (2013). Introducing basic programming concepts to elementary school children. *Procedia-social and behavioral sciences*, *106*, 1576. https://doi.org/10.1016/j.sbspro.2013.12.178
48. Zhang, J. X., Liu, L., de Pablos, P. O. i She, J. (2014). The auxiliary role of information technology in teaching: Enhancing programming course using Alice. *International Journal of Engineering Education*, *30*(3), 560-565. Pribavljeno 19. travnja 2018. s http://or.nsfc.gov.cn/bitstream/00001903-5/356833/1/1000013384665.pdf