Vizualni programski jezici u visokom obrazovanju

Divna Krpan, Saša Mladenović, Goran Zaharija

Prirodoslovno – matematički fakultet, Teslina 12, Split

E-mail: divna.krpan@pmfst.hr, sasa.mladenovic@pmfst.hr, goran.zaharija@pmfst.hr

Sažetak

Složenost programskog jezika za početnike svodi učenje programiranja na učenje sintakse čime početnici ne uspijevaju steći bitne vještine kao što su algoritamski način razmišljanja i rješavanje problema. Poteškoće programera početnika izražene su kod svih uzrasta, no posebno se ističu na fakultetima zbog slabe prolaznosti i odustajanja studenata. U radu opisujemo iskustvo poučavanja studenata kao programera početnika pomoću vizualnih programskih jezika.

Uvod

Učenje i poučavanje programiranja je izazov za nastavnike i učenike, bez obzira na uzrast učenika (radi li se o osnovnoj, srednjoj školi ili fakultetu). Studenti na fakultetima često imaju poteškoće pri polaganju ispita iz predmeta vezanih za programiranje. Višegodišnjim promatranjem i iskustvom rada sa studentima na predmetima početnog programiranja uočavaju se poteškoće usvajanja osnovnih koncepata i algoritama, te rješavanja problema. Često su studenti uvjereni kako su napisali ispravan program, ali ne razumiju zašto njihov program ne radi ono što očekuju [1]. Obzirom da se problemi u početnom programiranju reflektiraju na sve ostale predmete koji od studenata zahtijevaju znanje programiranja, važno je početi rješavati problem na početku. Pristup koji se opisuj u ovom radu pomiče fokus s uobičajenog učenja sintakse na rješavanje problema. Da bi se to ostvarilo, potrebno je odabrati odgovarajuće programsko okruženje, odnosno programski jezik s jednostavnom sintaksom, a u ovom radu su odabrani vizualni programski jezici: Scratch i Byob. U prvom dijelu je dan osvrt na probleme početnog programiranja u visokom obrazovanju, te odabir početnog jezika. Nadalje, analizirani su konkretni problemi na uvodnim predmetima na Prirodoslovno-matematičkom fakultetu (PMF) u Splitu kako bi se otkrili problemi, mogući izvori problema te načini rješavanja.

Problemi početnog programiranja

Predmeti s početnim programiranjem imaju najviše razine odustajanja (eng. dropout rates) [2], [3]. Početnici nailaskom na probleme počinju zaostajati, gube motivaciju i odustaju jer više ne mogu pratiti nastavu. Na predmetima početnog programiranja studenti ne mogu

postati stručnjaci, no može se uočiti da nakon što polože prvi takav predmet i dalje imaju iste probleme [4], odnosno čini se da nisu svladali potrebne vještine. Dok pojedini autori tvrde kako neke osobe jednostavno ne mogu naučiti programirati, ostali smatraju da možda neće svi postati vrsni programeri, ali bi mogli usvojiti osnovna znanja i vještine programiranja [5]. To bi se moglo očekivati od studenata nakon prvog predmeta iz programiranja.

Problemi početnog programiranja su vezani za to koga se poučava, što se poučava i kako [4]. Programiranje se kroz obrazovni sustav uči u svim uzrastima od osnovne škole do fakulteta. Treba se prilagoditi dobi i predznanju osoba koje se poučava jer se pristupi razlikuju. Kod odabira što se poučava, treba odlučiti što se treba usvojiti (znanja i vještine) primjereno karakteristikama učenika. Način poučavanja može se odnositi na odabir paradigme (npr. objektno-orijentirana, proceduralna) i na specifične strategije poučavanja (grupni rad, projekti, rješavanje problema, ...). Uz sve navedeno je bitan odabir programskog jezika i programskog okruženja jer je to dodatan faktor koji utječe na ukupnu uspješnost početnika.

Odabir početnog jezika

Odabir programskog jezika za poučavanje početnika nije jednostavan problem. Početni programski jezik bi morao imati jednostavnu sintaksu, brzu povratnu informaciju i strukturirani način pisanja. Često na odabir programskih jezika na fakultetima utječu zahtjevi tržišta, odnosno ponuda poslova. Prema tome se nastoji poučavati jezike koji su trenutno popularni kako bi studenti nakon završetka studija imali više mogućnosti za zaposlenje. Jezici koji se koriste u industriji moraju biti moćniji, što implicira složenost jezika, te samim tim neprimjerenost za početnike [6]. Programski jezici koji su pisani upravo za učenje i poučavanje programiranja (npr. Pascal) nisu popularni izvan područja obrazovanja, jer se ne mogu dalje primijeniti. Python je jezik kojem raste popularnost u oba svijeta: obrazovanju i industriji. Razvojna programska okruženja za početnike mogu uključivati vizualni prikaz izvršavanja programa (kao npr. u programskom jeziku Logo). Međutim, u ovom istraživanju odabran je vizualni programski jezik Scratch (http://scratch.mit.edu/) koji osim vizualnog prikaza izvršavanja programa ima i grafičke naredbe. Postoji i prijevod na hrvatski jezik što dodatno olakšava rad. Problemi s početnim programiranjem opisani su na konkretnom primjeru sa studentima PMF-a u Split u.

Programeri početnici u visokom obrazovanju

Studenti koji se upisuju na studijske programe na PMF-u Split koji uključuju početne predmete iz programiranja dolaze s nikakvim ili lošim predznanjem. Pretpostavka je da su

neke osnove trebali steći u prethodnom obrazovanju tijekom osnovne i srednje škole. Premda je informatika za osnovne škole definirana Hrvatskim Nacionalnim Obrazovnim Standardom (HNOS), ipak je izborni predmet što znači da će samo dio učenika u nekoj školi upisati informatiku ovisno o opremljenosti škole (broju računala i računalnih učionica). Dodatno, programiranje je samo manji dio koncepata koji se obrađuju u okviru predmeta. Učenici u gimnazijama uče informatiku u jednom ili dva početna razreda, osim prirodoslovno matematičkih gimnazija gdje imaju informatiku sva četiri razreda.

Učenje programiranja omogućuje stjecanje vještina rješavanja problema kod učenika. Problemi koji se rješavaju tijekom učenja programiranja mogu biti iz raznih područja (matematika, fizika, …). Međutim, upravo područje problema kojeg rješavaju može stvarati poteškoće ako početnici ne razumiju problem. Početnici ne razumiju koncepte kao što su varijable, grananja, petlje, rekurzije, ulazne i izlazne naredbe te kako sve te koncepte izraziti pomoću naredbi programskog jezika [7]. Kod studenata u ovom istraživanu uočene su poteškoće pri sastavljanju algoritama. Studenti koji su naučili sintaksu i jednostavne koncepte jezika često ne uspijevaju sastaviti ispravan program iz tih dijelova.

U sljedećem dijelu opisana su dva vizualna programska jezika koji su korišteni u praktičnoj nastavi sa studentima prve godine smjera Informatika na predmetu Informatički projekt 1. Obzirom da se iz prethodne analize može zaključiti da početnici osim sintakse imaju i druge poteškoće kao što su motivacija i rješavanje problema, dan je i kratak osvrt na iskustva ostalih istraživača, te detaljniji opis vlastitog praktičnog iskustva rada sa studentima.

Vizualni programski jezici

Scratch i Byob (Build Your Own Blocks, http://byob.berkeley.edu/) nisu jedini vizualni programski jezici (poznati su npr. Alice, Greenfoot i drugi) no ovdje se posebno ističu ova dva jezika zbog toga što su korišteni u praksi sa studentima. Kao prvi odabrani vizualni programski jezik, jednostavno se nametnuo Scratch odgovorivši na postavljene zahtjeve prilikom traženja odgovarajućeg okruženja za početnike. Byob je dijalekt ili proširenje Scratch-a.

Scratch

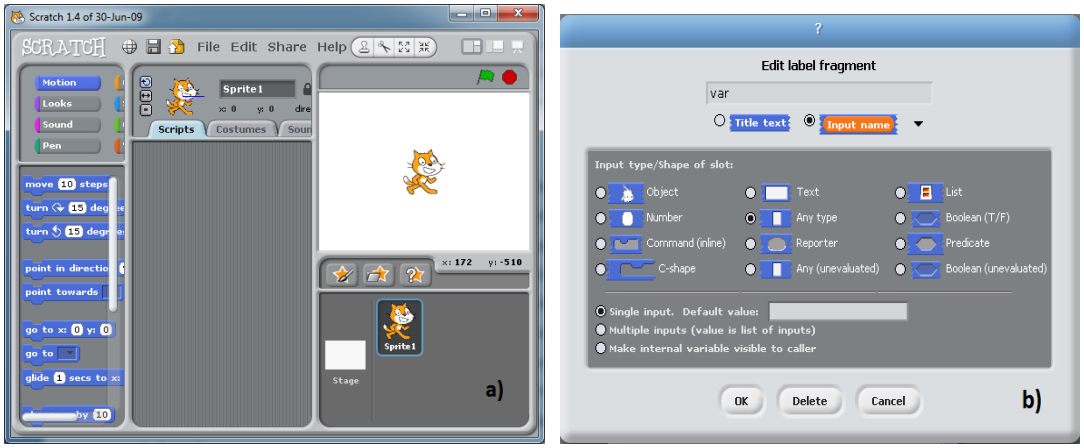
Programski jezik Scratch je nastao na sveučilištu MIT (Massachusetts Institute of Technology) kao projekt grupe „Lifelong Kindergarten Group“. Zasnovan je na idejama programskog jezika Logo, ali umjesto pisanja koda korisnik povlači naredbe koje su u obliku slagalica (inspirirano okruženjima LogoBlocks i EToys) [8] (Slika 1.a)). Scratch omogućuje

stvaranje igara, animiranih priča, te interaktivnih prezentacija. Okruženje za rad sastoji se od pozornice (eng. stage) na kojoj se prikazuju likovi (eng. spites), odnosno na kojoj se izvršavaju programi, područja za izradu skripti i paleta s naredbama. Naredbe su grupirane po paletama ovisno o njihovoj namjeni. Programi se mogu sastojati od jedne ili više skripti za svakog lika, a izrađuju se povlačenjem naredbi iz paleta na područje skripti, te spajanjem s drugim naredbama u obliku slagalice. Nema stroge sintakse već je vizualno jasno mogu li se neke naredbe spojiti ili ne. Naredbe se razlikuju i po oblicima. Scratch ima naredbe grananja, petlje i pokretače događaja (npr. kada je tipka pritisnuta, kad je lik kliknut). Komunikacija među likovima se provodi objavom poruka (eng. broadcast). Likovi mogu objavljivati i primati poruke. Mogu se pisati posebne skripte za svaki pokrenuti događaj ili objavu poruke. Podržane su liste, globalne i lokalne varijable. Obzirom da je u nastavi u to vrijeme bila aktualna Scratch verzija 1.4 (koja je 2010. godine bila najnovija verzija), postojao je problem što nisu podržane funkcije. Naknadno je u Scratch 2 omogućena izrada dodatnih blokova. Naprednija mogućnost Scratch-a je interakcija više različitih projekata (na istom ili više udaljenih računala). Postupak se naziva „mesh“ i omogućuje projektima zajedničko korištenje varijabli i poruka.

Byob

Byob (Build Your Own Blocks) je nadogradnja Scratch-a nastala na Sveučilištu Berkeley (http://snap.berkeley.edu/). Kasnije je preimenovan u “Snap!”. Byob na prvi pogled izgleda isto kao i Scratch, ali je napravljen tako da rješava nedostatke koji su postojali u Scratch 1.4. Omogućuje izradu vlastitih blokova naredbi čime se implementiraju funkcije koje mogu primati parametre. Moguća je izrada i višedimenzionalnih nizova, no to nije baš jednostavno napraviti upotrebom blokova. Dodano je dinamičko stvaranje novih likova (kloniranje) tijekom izvršavanja programa. Klonovi nasljeđuju sve funkcionalnosti originalnog lika. Moguće spremati cijele likove u varijable.

Bitna prednost je izrada vlastitih blokova. Byob omogućuje izradu tri vrste bloka: *command* (ne vraća vrijednosti), *reporter* (vraća vrijednosti), *predicate* (vraća logičke vrijednosti „*true*“ ili „*false*“). Moguće je i odabrati vrstu parametra koja se šalje u blok (Slika 1b).



Slika 1: a) Programski jezik Scratch b) Odabir vrste parametara (Byob)

Byob instalacija nema prijevod na hrvatski jezik. Moguće je „uvoziti“ skripte iz postojećih projekata čime se uvoze i dodatni blokovi koji su napisani u tim projektima. Prema tome, može se definirati i koristiti biblioteka vlastitih blokova. Scratch 2 podržava izradu novih blokova, ali ti blokovi ipak imaju ograničene mogućnosti u odnosu na Byob. Moguće je definirati samo blokove koji mogu primati parametre (odgovara *command* bloku u Byob-u). Byob omogućuje prevođenje projekta u izvršnu datoteku koja se može pokrenuti na drugom računalu kao neovisna aplikacija.

Rješavanje problema u Scratch-u

Kordaki u radu [9] definira 11 različitih aktivnosti koje se ostvaruju tijekom učenja programiranja u Scratch-u. Aktivnosti se globalno mogu svrstati u tri kategorije: (i) slobodno istraživanje (eksperimentiranje sa Scratch alatima i stvaranje vlastitih programa), (ii) rješavanje problema i (iii) rad na gotovim Scratch projektima (izmjene potpunih ili nepotpunih projekata). Brown et al. u radu [10] zaključuju kako je u njihovom istraživanju Scratch pozitivno utjecao na razvoj vještina rješavanja problema kod učenika. Kalelioglu & Gulbahar [11] su analizirali više provedenih istraživanja u kojima se koristio Scratch, a koja su ispitivala različite aspekte primjene Scratch-a na učenicima dobi od 8-18 godina. Bavili su se ispitivanjem da li Scratch djeluje na kreativno razmišljanje, logičko razmišljanje, rješavanje problema, razvoj igara, suradnju, koliko utječe na napredak učenika pri usvajanju programerskih vještina i koncepata, te koliko je Scratch poticajan. Motivirani pozitivnim rezultatima, autori provode vlastito istraživanje u kojem ispituju utjecaj na vještine rješavanja problema na 49 učenika osnovne škole. Zbog kratkog vremena istraživanja nisu dobili značajne razlike u rezultatima, no ipak smatraju da je postojao blagi pozitivan utjecaj, te da su učenici dobili više samouvjerenosti kod programiranja.

Primjena vizualnih programskih jezika

Studenti prve godine PMF-a Split imaju više kolegija koji se bave programiranjem. U prvom semestru nekoliko studijskih programa sluša predmet *Programiranje 1* (P1) (smjerovi: matematika, matematika i informatika, informatika, informatika i tehnika, fizika i informatika, te inženjerska fizika). Studenti smjera Informatika imaju dodatan predmet *Informatički projekt 1* (IP1). Tijekom drugog semestra, svi smjerovi također slušaju predmet *Programiranje 2* (P2). P1 je prvi susret studenata s programiranjem. Od ak. god. 2010/11 za smjer Informatika uvodi se predmet IP1. Početna ideja izrade naprednijih projekata je odbačena, obzirom da bi slabiji studenti bili u nepovoljnoj situaciji jer već imaju problema s osnovama programiranja. Trebalo je odabrati okruženje koje će im omogućiti stjecanje znanja i vještina programiranja, ali bez nametanja dodatne sintakse. Scratch se pokazao kao jezik i okruženje koje odgovara na postavljene zahtjeve. Dodatno se očekivalo poticanje motivacije u toj prvoj godini rada sa Scratch-om. Studenti su imali dva sata vježbi tijekom 15 tjedana prvog semestra. Obzirom da su paralelno slušali i P1, nastojalo se pratiti osnovne koncepte koji su se usvajali na tom predmetu. Svaki tjedan tijekom vježbi u uvodnom dijelu nastavnik ukratko upoznaje studente s osnovnim konceptom ili načinom rješavanja određenog problema koji je u fokusu za te vježbe, a studenti samostalno rješavaju postavljeni zadatak. Nakon svakih vježbi su imali proširenje zadatka za izradu kod kuće. Sva rješenja s vježbi i domaćih zadaća su se ocjenjivala, te su na kraju studenti morali osmisliti i izraditi završni projekt kojeg su branili na usmenom dijelu ispita.

Neki od koncepata koji su se usvajali i na P1 su: naredbe grananja, petlje, logički uvjeti, varijable i nizovi. Međutim, već prilikom pripreme nastavnih materijala uočava se kako nije moguće doslovno pratiti redoslijed usvajanja koncepata na P1 niti rješavati iste zadatke. Npr. da bi se lik ptice mogao kretati po pozornici, već je u prvom primjeru bilo potrebno upotrijebiti petlju za rješavanje zadatka (petlja se na P1 počinje koristiti u praktičnim zadacima tek nakon par tjedana). Kako bi ptica dok se kreće istovremeno mahala krilima, bilo je potrebno složiti dvije skripte koje su se izvršavale paralelno. Svaki lik ima jednu ili više skripti, svoje atribute (veličina, boja, koordinate, ime, lokalne varijable, …). Studenti su morali programirati i interakcije likova s korisnikom i s drugim likovima (odnosno reagiranje na događaje i poruke). Prema tome, zapravo se skup koncepata koje usvajaju dosta brzo proširio. Promatranjem rada i intervjuima sa studentima uočeno je da slabiji studenti uspijevaju na svakim vježbama ispuniti minimum zahtjeva. Studenti koji dolaze nespremni na vježbe se ipak uspijevaju snaći iako im za to treba više vremena pa naprave manje. Međutim,

bilo je i negativnih reakcija. Bolji studenti s vremenom postaju sve više frustrirani jer svoje ideje ne mogu izraziti zbog nedostataka okruženja, te postaje nezgodno domišljati se kako napraviti ono što je će Scratch prihvatiti. Dodatno, kako rade sve složenije skripte tako područje za izradu postaje premalo i nepregledno. Smatrali su da bi bilo puno zgodnije tipkati i isticali su da im jako nedostaje „for“ petlja. Bili su problematični zadaci koje su inače radili često na P1, tipičan primjer je sortiranje niza ili razni matematički zadaci koje je bilo teže sastaviti od blokova nego natipkati u Python-u.

Obzirom na navedene nedostatke, u ak. god. 2011/12 odabran je Byob. Već je istaknuto kako Byob nema prijevod na hrvatski jezik pa su studenti radili prvi dio semestra u Scratch-u, do trenutka kad se javi potreba za dodatnim blokovima. Više nije bilo primjedbi na nedostatak pojedinih naredbi jer se sve što im treba moglo napisati ili iskoristiti iz prethodnih programa, ali se i dalje zadržao problem nepreglednih skripti, no kako je prijelaz na Byob bio jednostavan, u idućoj godini koristi se Byob u potpunosti uz dodatne koncepte kloniranja i višekorisničkih igara.

Studenti pred kraj semestra ocjenjuju provedbu nastave studentskim anketama. Tijekom ak. god. 2010/11 anketirano je 20 studenata. Provedba nastave je ocijenjena prosječnom ocjenom 4.3. Za ak. god. anketirano je ukupno 34 studenta uz prosječnu ocjenu 4.2, a za ak. god. 2012/13 anketirano je 26 studenata (prosječna ocjena 4.1). Ocjena studenata je subjektivan dojam, no moguće je da reflektira njihovo zadovoljstvo.

Zaključak

Učenje programiranja je zahtjevno, te je predmet brojnih rasprava. Početni predmeti iz programiranja imaju slabu prolaznost i na našem fakultetu. Jedan od razloga je prethodno obrazovanje studenta u kojem svi nemaju priliku steći predznanje. Problemi s početnim programiranjem prisutni su u svim razinama obrazovanja, no posebno je problematično studentima koji se kao odrasle osobe prvi put susreću s programiranjem, a moraju ga usvojiti u relativno kratkom vremenu.

Učenje programiranja se često svodi na učenje sintakse, te se zanemaruje algoritamsko rješavanje problema što je često problematično. Studenti imaju poteškoće i pri praćenju izvršavanja gotovih programa. Ponekad predmeti početnog programiranja postaju prejednostavni za bolje studente, a slabiji studenti ipak nisu u stanju pratiti. Neki studenti priznaju kako kod kuće gotovo uopće ne vježbaju.

Odabir programskog jezika je bitan faktor pri učenju programiranja. U ovom radu su odabrani vizualni jezici na dodatnom predmetu kod jedne grupe studenata početnika iako su

Scratch i Byob u prvom redu namijenjen djeci. Uklanjanjem složenosti sintakse očekivalo se da će studenti kroz zabavno okruženje lakše usvojiti potrebne vještine i rješavati složenije probleme. Reakcije studenata su uglavnom pozitivne, ostvarena je angažiranost slabijih studenata od početka do kraja vježbi, te su uspijevali programirati složene interakcije među likovima, paralelno izvršavanje više skripti, te sinkroniziranje putem poruka. Bolji studenti nisu bili zadovoljni ograničenjima Scratch-a jer su se već dosta dobro snalazili u Python-u, no radi se o jako malom broju studenata. Prelaskom na Byob bilo je manje primjedbi, ali zaključak je da bi boljim studentima trebalo osmisliti dodatne izazove.

Popis literature

1. Garner, S., P. Haden, and A. Robins. *My program is correct but it doesn't run: a preliminary investigation of novice programmers' problems*. in *Proceedings of the 7th Australasian conference on Computing education*. 2005: Australian Computer Society, Inc.
2. Robins, A., J. Rountree, and N. Rountree, *Learning and teaching programming: A review and discussion.* Computer Science Education, 2003. **13**(2): p. 137-172.
3. McCracken, M., et al., *A multi-national, multi-institutional study of assessment of programming skills of first-year CS students.* ACM SIGCSE Bulletin, 2001. **33**(4): p. 125-180.
4. Pedroni, M., *Teaching introductory programming with the inverted curriculum approach*, in *Department of Computer Science, ETH*. 2003, ETH, Eidgenössische Technische Hochschule Zürich: Zurich. p. 87.
5. Caspersen, M.E., *Educating novices in the skills of programming*, in *Department of Computer Science*. 2007, University of Aarhus: Aarhus. p. 323.
6. Grandell, L., et al. *Why complicate things?: introducing programming in high school using Python*. in *Proceedings of the 8th Australian Conference on Computing Education*. 2006.
7. Eckerdal, A., *Novice programming students' learning of concepts and practise*, in *Faculty of Science and Technology*. 2009, Uppsala University. p. 194.
8. Maloney, J.H., et al. *Programming by choice: urban youth learning programming with scratch*. in *ACM SIGCSE Bulletin*. 2008: ACM.
9. Kordaki, M., *Diverse categories of programming learning activities could be performed within Scratch.* Procedia-Social and Behavioral Sciences, 2012. **46**: p. 1162-1166.
10. Brown, Q., et al., *Computer Aided Instruction as a Vehicle for Problem Solving: Scratch Programming Environment in the Middle Years Classroom.* Proceedings of the ASEE National Conference, 2008: p. 22-25.
11. Kalelioglu, F. and Y. Gulbahar, *The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective.* Informatics in Education, 2014. **13**(1): p. 33-50.