Učenje i poučavanje programiranja temeljeno na igrama

Monika Mladenović
monika.mladenovic@pmfst.hr

Sažetak — Živimo u digitalnom dobu koje je bitno drugačije od prethodnih naročito kada je riječ o djeci, učenju i poučavanju. Današnja djeca koja se nazivaju „digitalnim urođenicima“ ili“ Net generacijom“ žive u potpuno drugačijem svijetu od prethodnih generacija dok se način poučavanja i učenja nije bitno promijenio. Igralište Net generacije je digitalni, „virtualni svijet“ u koji moramo ući kako bi ih mogli razumjeti i pripremiti za život, što i jest osnovna svrha školovanja. Djeca danas dosta vremena provode za računalom, bilo na društvenim mrežama, igrajući računalne igrice ili surfajući internetom. Informacijska i komunikacijska tehnologija je postala sastavni dio života i teško je zamisliti bilo koje zanimanje budućnosti koje neće podrazumijevati poznavanje i snalaženje s tehnologijom. Zbog toga, osim jezične i matematičke pismenosti u današnje vrijeme neophodno je savladati i informatičku pismenost (korištenje tehnologije), a poželjno je postići informatičku okretnost (razumijevanje tehnologije). Za razumijevanje tehnologije, potrebno je razumjeti jezik računala, odnosno ovladati principom rada računala. Najbolji način za to je učenje programiranja od najranije dobi, ali na način koji je djeci „prirodan“. Činjenicu da djeca dosta svog vremena provode za računalom potrebno je iskoristiti kako bi to vrijeme mogli provesti učeći. Jedan od načina za to je korištenje računalnih igara za učenje programiranja. U radu je predstavljena relevantna literatura koja se bavi navedenom problematikom kao i rezultati istraživanja na istu temu.

Ključne riječi - računalna znanost, poučavanje, programiranje, računalne igre, učenje

I. UVOD

Istraživanja o učenju i poučavanju su doživjela procvat nakon drugog svjetskog rata (industrijsko doba). U novom socijalnom ustroju koji se dogodio nakon rata, obrazovanje i učenje su počeli utjecati na društvo što je prepoznato na svim (društvenim, političkim, sociološkim i dr.) razinama. Za industrijski način proizvodnje bili su potrebni radnici koji znaju pisati i računati dok su sve ostale individualne značajke bile manje važne. Danas živimo u digitalnom dobu za koje su potrebne drugačije sposobnosti. Od vremena informacijskog doba prema digitalnom se, kako u ekonomiji tako i u obrazovanju, događaju promjene koje su međusobno ovisne [1]. Industrija više nije ključni gospodarski element, broj "bijelih ovratnika" nadmašio je broj "plavih ovratnika", obitelj više nije u centru društva, sve je više samohranih roditelja, uvažavaju se razlike među ljudima. Što se tiče učenja i poučavanja, došlo je do promjena u teoriji učenja te u poimanju inteligencije. Do kraja industrijskog doba dominantni biheviorizam zamijenjen je kognitivizmom pa kasnije konstruktivizmom, a tadašnje poimanje inteligencije temeljeno isključivo na matematičkoj i lingvističkoj inteligenciji promijenilo se uvažavanjem koncepta višestruke inteligencije [2] koja se temelji na prepoznavanju devet vrsta inteligencije kod čovjeka. Od sposobnosti zaključivanja i matematičkih sposobnosti, za koje je zadužena lijeva strana mozga, postala je važna kreativnost, sposobnost prilagođavanja, vizualizacije za što je zadužena desna strana mozga [3]. Globalizacijom tržišta i uzletom velikih korporacija došlo je do promjena u društvu koje zahtijevaju promjene u obrazovanju. Daljnjim napretkom tehnologije i virtualizacijom proizvoda i usluga postao je važan "intelektualni kapital" u stvaranju i održavanju vrijednosti u poslu. Učenje i obuka prepoznati su kao važni faktori za zadržavanje kompetitivnosti ljudi i organizacija [4]. Brze promjene rezultirale su bitnim razlikama u očekivanim ishodima učenja i vještinama. Današnje škole se nisu bitno promijenile od industrijskog doba, a kao ishod učenja traže se potpuno druge kompetencije [5]. Glavna značajka sljedeće generacije poslova bit će povećana upotreba tehnologije, rješavanje problema i kompleksne komunikacije [6]. Na primjer, Ranije je programer mogao biti netko tko ima sposobnosti linearnog razmišljanja i izvršavanja rutina, dok se danas od programera očekuje kreativnost (desna strana mozga), a ne isključivo kreiranje algoritama (lijeva strana mozga). Očito je da se nešto mora promijeniti u obrazovanju, jer je sposobnost programiranja u zapadnom svijetu postala samo vještina, a sada se puno više cijene sposobnosti analize i oblikovanja sustava te gledanja cjelokupne slike (eng. big picture) i druge kreativne sposobnosti [7]. Unatoč preporukama da su u učenju što više koristi lijeva i desna strana mozga većina nastavnih programa računalne znanosti je napravljena za „lijevi“ mozak [8].

Današnje doba je digitalno doba koje je bitno drugačije od dosadašnjih. Prijelazi između prethodnih razdoblja tekli su

Rad je predan na ocjenu: 22.04.2014.

Monika Mladenović studentica je poslijediplomskog studija Istraživanje u edukaciji u području prirodnih i tehničkih znanosti, usmjerenja informatika pri Prirodoslovno-matematičkom fakultetu Sveučilišta u Splitu.

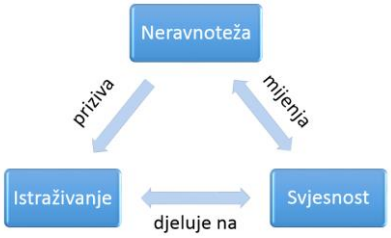
inkrementalno, dok se pri prijelazu na digitalno doba dogodio diskontinuitet. Uzrok ovakvog diskontinuiteta se nalazi u ubrzanim promjenama digitalnih tehnologija od kraja 20. stoljeća. Današnji učenici predstavljaju prvu generaciju koja je odrasla s novim tehnologijama. Oni su cijeli život okruženi računalima, mobitelima, video igrama, digitalnim glazbenim uređajima, video kamerama, i drugim uređajima i igračkama digitalnog doba. „Digitalni urođenici“ preferiraju učenje kroz igru u odnosu na „*ozbiljan* rad“ [9]. Pojavom digitalnog doba pojavljuje se i pojam informatičke ili digitalne pismenosti kao sposobnosti korištenja računala i različitih računalnih alata. Informatička pismenost [10] danas predstavlja minimum pojmova koje bi trebalo poznavati, a većina današnjih poslova zahtjeva ne samo informatičku pismenost već i *informatičku okretnost* [11] koja podrazumijeva ne samo korištenje različitih informatičkih alata nego i sposobnost sustavnog i kreativnog razmišljanja potpomognutog tehnologijom. Računalno razmišljanje je rješavanje problema, dizajniranje sustava, te razumijevanje ljudskog ponašanja koje se temelji na konceptima koji su temelj računalne znanosti [12]. S obzirom na sve navedeno kao i na sveprisutnost računalne znanosti u životu, informatika i računalno razmišljanje su vještine koje bi svi trebali savladati. Računalno razmišljanje bi bila četvrta „analitička sposobnost“, uz čitanje, pisanje i aritmetiku, koja bi za učenje trebala biti dostupna svima [13], a programiranje bi trebala biti vještina koju bi svi trebali usvojiti [14] [15]. Unatoč navedenom, u svijetu postoji trend opadanja interesa za učenjem računalne znanosti [14] [16] dok istodobno raste potražnja za takvim poslovima [17] [18]. Zbog takvog odnosa ponude i potražnje u području računalne znanosti jasno je kako se nešto treba promijeniti u pristupu poučavanju računalne znanosti. Razlog pojave nesrazmjera se možda krije u činjenici što su današnja djeca „digitalni urođenici“, a način poučavanja sadržaja računalne znanosti i programiranja se nije bitno promijenio, odnosno nije prilagođen takvoj populaciji. U ovom radu obrađuje se poučavanje programiranja kroz igru i kroz programiranje igara kao načinima prilagodbe učenju djece koja pripadaju digitalnom dobu.

II. UČENJE I IGRE

Prijelazom na digitalno doba u odnosu na prethodne promijenio se način učenja i obrade informacija. Kako su prije digitalnog doba od medija bile dostupne samo knjige i pisani materijali, a kasnije i televizija, koristilo se linearno, deduktivno zaključivanje [19]. Zbog velikog utjecaja i uporabe digitalnih tehnologija djeca odrasla u digitalnom dobu imaju sposobnost obrade više informacija istovremeno (eng. paralell processing), sposobnost obavljanja više poslova istovremeno (eng. multitasking), više koriste induktivno zaključivanje, žele brzu i čestu interakciju sa sadržajem i imaju izvanredne vizualne sposobnosti što su sve karakteristike učenja temeljenog na računalnim igrama [19] [20] [21]. Iz navedenog se može zaključiti kako je kod digitalnih urođenika dominantno konstruiranje novih ideja uporabom bilo čega što je pri ruci, što je način razmišljanja u stručnoj literaturi poznat pod nazivom „bricolage“ [19]. „Bricolage“ mislioci rješavaju probleme po principu pokušaja i pogrešaka, istraživanja, eksperimentiranja što je utjecajem interneta i digitalne tehnologije postao dominantan, čak i poželjan način razmišljanja [22]. Iz razmatranih istraživanja proizlazi hipoteza mogućeg budućeg istraživanja kako se upravo u navedenom krije jedan od razloga za smanjenje zanimanja za programiranje koje zahtijeva analitičnost i sustavni pristup rješavanju problema koja nije u skladu s „bricolage“ načinom razmišljanja. Osnovne stilove programiranja odnosno razmišljanja moguće je podijeliti na „tvrdi pristup“ (eng. hard) i „meki pristup“ (eng. soft) koji uključuju nekoliko razlika u pristupu poučavanju programiranja [23]. Tvrdi pristup podrazumijeva logički, hijerarhijski pristup rješavanju problema te pristup „razmisli dvaput kodiraj jednom“ [24], dok meki pristup uključuje fleksibilan, nehijerarhijski pristup rješavanju problema i otvorenost prema eksperimentiranju. Tvrdi pristup, od učenika, zahtijeva visoku razinu apstrakcije dok meki preferira pristup od konkretnog prema apstraktnom. Istraživanja pokazuju da žene preferiraju meki pristup dok muškarci preferiraju tvrdi pristup programiranju [22]. Iz navedenog se može zaključiti da drugačiji, „mekši“ pristup programiranju ne samo da može ohrabriti digitalne urođenike na učenje programiranja već se može ujednačiti i stav prema programiranju u odnosu na spol. Neka istraživanja su pokazala kako se programiranjem igara i pričanjem priča (eng. storytelling) u programskom jeziku Alice u srednjoj školi djevojčice motiviraju za programiranje [25].

Područje istraživanja o utjecaju korištenja računalnih igara za učenje nije dovoljno istraženo kako bi se moglo reći da korištenje igara ima ili nema utjecaj na efekte učenja [26]. Osim istraživanja na tu temu potrebno je odgovoriti na pitanje kako uključiti računalne igre u nastavni proces kako bi se maksimizirao učinak učenja [27]. Iako je posljednjih nekoliko godina uključivanje igara u postupak poučavanja postala aktualna tema o kojoj se puno piše, postoji i bojazan da se lošom primjenom izgubi sve potencijalno dobro. Može se reći kako se nešto takvo i dogodilo, od očekivanja da se uzme najbolje od oba svijeta (igara i učenja) dogodio se obrat koji je rezultirao usvajanjem najgoreg od obaju svjetova [28].

Istraživačko pitanje koje se postavlja je: Koje uvjete igra mora zadovoljavati kako bi bila i zabavna te istodobno tijekom igre omogućavala odvijanje procesa učenja? Prema Piagetovoj teoriji kognitivnog razvoja znanje se asimilira ili akomodira [29]. Proces asimilacije je jednostavniji, a podrazumijeva potpuno novo znanje koje se prihvaća dok je akomodacija puno zahtjevniji proces pri kojem se novo znanje mora uklopiti u postojeće s kojim može biti u konfliktu pri čemu može doći do kognitivne neravnoteže [29] što je važno za kognitivni razvoj. Igrom je lako postići stanje kognitivne neravnoteže koje u pravoj mjeri potiče učenje. Igranje igara potiče ciklus postavljanja hipoteza, testiranja, ispitivanja [27].



Slika 1 Odnos neravnoteže, istraživanja, svjesnosti [23]

Osim kognitivne neravnoteže tijekom učenja se javljaju i kognitivni procesi istraživanje i svjesnost. Neravnoteža, istraživanje i svjesnost su povezani procesi kojima se dolazi do viših razina razmišljanja i rješavanja problema [30].

Prejednostavne igre neće u dovoljnoj mjeri potaknuti sudjelovanje te će brzo postati dosadne. Nasuprot tome preteška igra može kognitivno preopteretiti igrača koji će brzo odustati. Napredovanje u igri mora pratiti ciklus neravnoteže, istraživanja i svjesnosti.

III. IGRE

Igre su vrlo prilagodljivi medij koji se može prilagoditi svakoj tehnologiji od neolitika do visoke tehnologije. Postoji velik broj vrsta igara: ratne igre, slagalice, logičke, strateške, društvene i druge [31].

Potrebno je odrediti što je svim igrama zajedničko, definirati pojam igre i uvjete koje mora zadovoljiti kako bi bila uspješna. Ne postoji jedinstvena definicija igre već postoji više interpretacija tog pojma [32]. Jedna od njih je Juulsova definicija koja igru predstavlja u dvije dimenzije. Jedna dimenzija se odnosi na perspektive, a to su: *igra* (svojstva sustava koja definiraju pravila igre), *igrač* (relacija između igre i igrača) i *svijet* (relacija između igre i svijeta) [33]. Kada govorimo o igrama koje bi osim zabave trebale igrati i obrazovnu ulogu ovom modelu možemo dodati i *dimenziju znanja* koja predstavlja relaciju između igrača i obrazovne svrhe igre. Druga dimenzija definira što igra mora zadovoljavati: *pravila*, *mjerljivi ishod*, *vrednovanje ishoda*, *trud igrača* (igra je izazov), *igrač koji je vezan za ishod* (npr. sretan je s pozitivnim ishodom, a nezadovoljan s lošim ishodom), te *prenosive posljedice* (igra se može igrati sa ili bez posljedica u stvarnom svijetu) [33]. Iz svega navedenog igru bi mogli definirati kao formalni sustav temeljen na pravilima s mjerljivim ishodom i varijablama gdje je različit ishod vezan uz različite vrijednosti, a u kojem igrač ulaže napor kako bi utjecao na ishod, te želi ostvariti cilj igre, a posljedice igranja su po izboru prenosive u stvarni svijet. Kako bi igru mogli smatrati pedagoškim alatom osim navedenih značajki igra bi još trebala imati mjerljivi pedagoški ishod.

Tablica1 – Uvjeti koji definiraju igru

	igra	igrač	svijet	znanje
1. pravila	x			
2. mjerljivi ishod	x			x
3. vrednovanje ishoda	x			x
4. trud igrača		x		
5. igrač koji je vezan za ishod		x		
6. prenosive posljedice			x	x

U tablici 1 prikazani su uvjeti koji definiraju igru. Navedeni uvjeti koje igra treba zadovoljiti nisu vezani uz iste razine pa tako uvjeti navedeni pod 1, 2 i 4 opisuju svojstva igre kao formalnog sustava, 3 definira vrijednosti za moguće ishode igre, 4 i 5 opisuju relaciju između igrača i sustava, a 6 definira vezu između igre i ostatka svijeta.

Costikyan definira igre vrlo jednostavno: "bez cilja to je igračka (eng. „*without a goal, it is a 'toy.*'") te tvrdi da je igra skup definiranih pravila dodanih igrački [34]. On je definirao sljedeće elemente igre: cilj, borba (ne nužno natjecateljski oblik što uključuje i suradničke igre), borba sa samim sobom (npr. slagalice, i slični izazovni zadaci koji ne moraju uključivati druge igrače), struktura, smislenost i interaktivna zabava.

Uvjete koje igra treba zadovoljiti kako bi bila zabavna prema LeBlancovoj taksonomiji su: osjet, fantazija, narativnost, izazov, druženje, otkrivanje, izražavanje i podčinjenost [35].

IV. PROGRAMIRANJE

Programiranje je područje koje je izazov za učenje i poučavanje [36]. To je područje koje se smatra teškim za učenje, razumijevanje i savladavanje pogotovo za početnike [37], [38], [39] bez obzira na dob [40], a većina djece u osnovnim školama ima negativno mišljenje o računalnoj znanosti [41]. Programiranje je računalom potpomognuto rješavanje problema, otklanjanje grešaka, razvijanje logičkog i računalnog razmišljanja što podrazumijeva razvoj strategija za rješavanje problema koji se mogu odnositi i na ne-programerska područja. Zbog toga se može reći da programiranje mijenja način razmišljanja [42]. Učenje programiranja podrazumijeva ne samo učenje novih pojmova već i vježbanje vještine primjene naučenog za rješavanje problema u novim situacijama [43]. Iako se područje poučavanja programiranja aktivno istražuje, najbolja metoda za učenje programiranja još nije pronađena [43].

Neka istraživanja pokazuju da faktori kao što su stav, motivacija i pojačani interes za programiranje utječu na uspješnost u učenju programiranja [44]. Bez povezivanja programiranja i koncepata računalne znanosti s učeničkim različitim interesima, klasično poučavanje programiranja ne motivira većinu učenika i obeshrabruje ih u nastavku školovanja prema računalnoj znanosti [14] [45]. „Klasični" alati za programiranje, poput BASIC-a, osim što su vizualno neprivlačni današnjoj djeci, zahtjevaju potpuno sintaktički (i semantički) ispravno napisan program, što je često izvor frustracija i demotivirajući faktor za učenje programiranja. Zbog toga što program mora sintaktički biti potpuno ispravan učenici više pridaju pažnje sintaksi nego semantičkom značenju [46].

Nakon početnog entuzijazma za programiranje koji je postojao 1970-ih i 1980-ih s pojavom osobnih računala, danas takvog entuzijazma gotovo da nema. Danas je jedan jednostavan pametan telefon snažniji od svih sveučilišnih računala 1970-ih. Djeca su uključena u računalno okruženje od malih nogu i očekuju interakciju s računalom pa to područje više nije usko vezano samo uz matematiku i matematičke sposobnosti [13]. Programiranje se uglavnom doživljava kao tehničku aktivnost primjerenu za manji broj ljudi. Što se dogodilo s entuzijazmom? Postoji nekoliko faktora koji su utjecali na pad entuzijazma za programiranje [42] :

- Rani alati za programiranje su bili prekomplicirani za korištenje te dosta djece ne bi uspjelo savladati sintaksu
- Programiranje se obično vezalo uz aktivnosti koje

nisu vezane uz interese djece (uglavnom matematički problemi kao što je traženje prostih brojeva i sl.)
- Programiranje se odvijalo na način da se u slučaju pojave pogrešaka teško samostalno našlo rješenje

U današnje vrijeme postoji cijeli niz novih jezika koji eliminiraju navedene nedostatke. Takvi jezicima će biti detaljnije opisani pod *B.*

A. Programiranje i apstrakcija

Apstrakcija je temelj matematike, znanosti i inženjerstva uopće, te igra kritičnu ulogu u izradi modela za analizu te u izradi inženjerskih rješenja. Prema Piagetovoj teoriji kognitivnog razvoja postoje četiri faze kognitivnog razvoja čovjeka: senzomotorna (0-2 g.), predoperacijska (2.-7. g), konkretnih operacija (7.-11.g) i formalnih operacija (>12 g) [29]. Novija istraživanja pokazuju da samo 34% adolescenata dolazi do ove faze, dok ni kod odraslih taj postotak nije veći [47]. Ovakva razlika upućuje na mogućnost da škole mogu potaknuti razvoj apstraktnog mišljenja kod većeg broj učenika prilagođavanjem kurikuluma učeničkim kognitivnim razinama. Dio odraslih ljudi nikada ne dođe do faze formalnih operacija jer se nisu našli u okruženju koje to zahtjeva od njih, što potiče na propitivanje: Može li se apstrakcija vježbati? [48] Prema Piagetovim istraživanjima došlo se do zaključka da učenici trebaju konkretna iskustva kako bi razumjeli apstrakciju [22]. Jedno od većih poteškoća za učenje programiranja je što su programski jezici umjetni pa zahtijevaju visoku razinu apstrakcije [49]. Programiranje je samo po sebi apstraktno i samim time teško za razumijevanje, ali je u isto vrijeme dobar alat za vježbu i razvoj apstraktnog razmišljanja. Cilj učenja programiranja ne mora biti stvaranje programera već i vježba za poticanje učenja i apstraktnog mišljenja.

Iz svega navedenog nameću se tri važna pitanja za računalnu znanost: sveprisutnost računalne znanosti, važnost konteksta, pomak od konkretnog prema apstraktnome. Pitanje pomaka od konkretnog prema apstraktnome je za računalnu znanost ključna jer je sama disciplina apstraktna. Konkretna iskustva su najučinkovitiji način učenja kada se pojavljuju u kontekstu relevantne konceptualne strukture. [50].

B. Programski jezici za početno učenje programiranja

Ljudima je prirodno razmišljati od konkretnog prema apstraktnom, pa je upravo po tom principu Papert sa suradnicima razvio programski jezik LOGO koji je prilagođen djeci, jer naredbe koje se zadaju računalu vide kroz konkretne, vizualne rezultate što povećava i afektivnu domenu [51]. Papert, je tvrdio da programski jezici trebaju imati „nizak pod“ (eng. *Low floor*) (početak treba biti lak), „visoki strop“ (eng. *High ceiling*) (s vremenom treba postojati mogućnost izraditi sve kompleksnije projekte), i „široke zidove“ (eng. *Wide walls*) (treba podržati različite tipove projekata primjerene osobama različitih interesa i stilova učenja). Zadovoljavanje trojke *niski pod/visoki strop/ široki zidovi* nije lako [42]. LOGO se smatra prvim jezikom prilagođenim za početnike koji je potakao razvoj drugih takvih jezika koji se nazivaju mini-jezicima. Značajke mini-jezika su da su mali i jednostavni kako bi olakšali prve korake u programiranju [52]. U većini mini-jezika učenici uče programirati kroz kontrolu nekog fizičkog ili virtualnog lika. Fizički lik predstavljen je fizičkim uređajem, koji je predstavljen kornjačom u početku LOGO-a, ili robotom danas. Virtualni lik predstavljen je računalnim modelom lika u mikrosvijetu kojim se upravlja vrlo jednostavnom sintaksom. Mini-jezici u novije vrijeme nastali po uzoru na LOGO, su vizualni programski jezici poput Scratch, Alice, Greenfoot, GameMaker, Kodu, NXTG (LegoMindstorms) i drugi. Takvi jezici pružaju mogućnost stvaranja priča, kreiranja video igrica, izrada scenarija i slično. Vizualni programski jezici korisniku pružaju iskustvo programiranja od konkretnog prema apstraktnom u određenom kontekstu. Inače teški, kompleksni koncepti programiranja se kroz ovakve jezike doživljavaju preko konkretnih događaja, pa se lakše prenose u apstrakciju, na primjer apstraktni pojmovi poput varijabli, petlji i slično u vizualnom okruženju pružaju konkretno iskustvo kroz vizualizaciju pojedinih koncepata. Navedeni programski jezici zadovoljavaju i potrebe digitalnih urođenika za interakcijom s računalom, vizualno su privlačni te omogućuju programiranje u kontekstu. Kontekst programiranja se na ovaj način pomiče od aktivnosti koje nisu privlačne digitalnim urođenicima i stavlja se u okruženje koje omogućuje razvijanje ne samo matematičkih sposobnosti, već potiče kreativnost na drugim poljima, čime se uzimaju u obzir i druge vrste inteligencije, a ne isključivo matematičko-logička. Ovakav način učenja programiranja otvara put učenja programiranja učenicima koji pripadaju digitalnom dobu.

C. Učenje programiranja kroz programiranje igara

Prethodno navedeni jezici su dobri za početno učenje programiranja, ali ne mogu se nazvati igrama jer nemaju cilj već se mogu nazvati okruženjima u kojima se može igrati i izrađivati igrice. Kako bi se povezala navedena dva svijeta, programiranja i igara, nameće se ideja o učenju programiranja kroz programiranje igara. Jezici koji su navedeni u prethodnom poglavlju su pogodni i za programiranje igara. Na taj način učenicima je zabavnije programirati, a prebacivanjem težišta sa sintakse na semantičku strukturu programa učenici zapravo nisu svjesni da vježbaju rješavanje problema, otklanjaju pogreške u programima, izrađuju scenarije, već su uvjereni kako izrađuju računalne igre. Poticanje učenika na učenje jednog dok oni misle da rade nešto sasvim drugo, Pauch je nazvao prijevara glave (eng. „head fake“) [53].

Većina istraživanja u ovom području se, između ostalog zbog nedostatka primjerenih programa u školama, provodi u klubovima ili ljetnim kampovima.

Brojna istraživanja su pokazala da se programiranjem igara, neovisno o spolu, povećava motivacija učenika te da se na taj način mogu usvojiti određeni koncepti programiranja kroz različita okruženja: korištenjem programskih jezika Alice [54] [55] [56], Scratch [57] [58] [59] i Kodu [60]. Osim toga uočeno je kako korištenje ovakvih jezika potiče informatičku okretnost [56], te da učenici kroz rad u Scratchu kao prvom programskom jeziku u školi mogu usvojiti koncepte programiranja [61].

U Finskoj je provedeno longitudinalno istraživanje na uzrastu 12-16 godina. Kroz tri godine pratili su se učenici koji

su pohađali jednotjedne ljetne tečajeve programiranja igara. Istraživanje je pokazalo da je većina učenika nastavila s programiranjem te da se njihovo zanimanje za računalnu znanost povećalo, osim toga su s programiranjem igara djevojčice bile podjednako zadovoljne kao i dječaci [62].

Nad studentskom populacijom provedeno je istraživanje u sklopu kolegija programiranja 3D igara i to za studente koji studiraju umjetnost, jer se u izradi igara isprepliću umjetnost i programiranje, odnosno lijeva i desna strana mozga. Rezultat kolegija su deseci igara različitih žanrova rađeni kroz projekte u timovima koji su brojili od 3 do 6 studenata. Igre su izrađene pomoću više alata poput Macromedia Director, Flash, Java, Virtools i Quest3D. Studentske igre bile su jako dobre, a često i bolje od igara koje su izradili studenti računalne znanosti što se objasnilo činjenicom kako su studenti studija umjetnosti posjedovali bolje vještine potrebne za izradu sadržaja same igre [8].

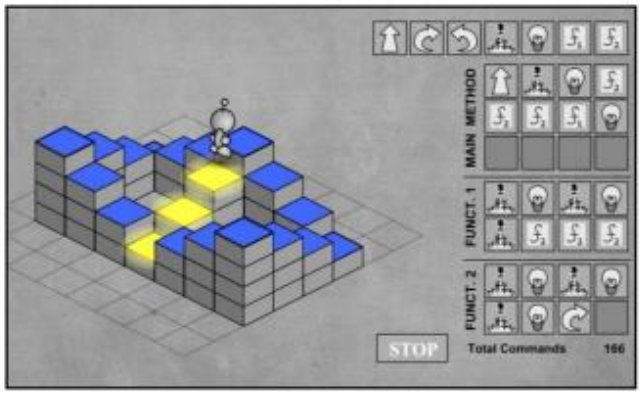
Istraživanje u kojem se primijenio početni pristup učenja programiranja programirajući igre (eng. *game first approach*) je pokazalo kako se takvim pristupom poboljšalo znanje osnovnih programerskih koncepata, povećao ostanak na studiju računalne znanosti te kako je povećano ukupno zadovoljstvo studenata [63].

D. Igre za poučavanje

Iako su istraživanja pokazala da se vještine poput rješavanja problema povećavaju i čak prenose tijekom igranja, teško je takve vještine prenijeti izvan igre [64]. Curtis i Lawson su ustanovili tek umjereni prijenos vještine rješavanja problema [65]. Prenesene vještine su, čini se, ograničene na nisku razinu prijenosa [66]. Takva istraživanja pokazuju da nedostaje posredovani transfer [67], odnosno, igra i (ili) pedagoški pristup u kojem bi se vještine i znanja stečene u igri mogle prenijeti u stvarni svijet, odnosno na koncepte programiranja koje su bile cilj učenja. Postoje dvije kategorije posredovanog transfera. Prva kategorija je „premoštenje“ (eng. „bridging“) u kojem nastavnik pomaže učenicima premostiti put od konteksta u kojem je koncept naučen do novog potencijalnog konteksta. Druga kategorija je „prigrljivanje“ (eng. „hugging“) u kojoj nastavnik kreira situaciju učenja sličnu situaciji u kojoj se transfer očekuje [67]. Od igara se ponekad previše očekuje, očekuje se igra u kojoj će učenik igrajući dobiti sva znanja koja se očekuju. Na učitelju je omogućiti prijenos znanja iz igre u željene ciljeve učenja, dakle učitelj je taj koji mora pomoći učeniku premostiti put od igre prema željenom cilju učenja. Primjer za navedeno je istraživanje provedeno igrom *Crystal Island* koje je izvršeno u srednjoj školi *Ocoee* na Floridi gdje su učenici kroz igru učili koncepte iz mikrobiologije na način da su zapisivali svoja otkrića i hipoteze kroz igru [68]. Prestankom igranja ne prestaje učenje, upravo tada je najvažnije vrijeme kada govorimo o korištenju igre u učenju jer upravo tada treba povezati koncepte naučene u igri i staviti ih u željeni kontekst gdje ključnu ulogu ima učitelj [69]. Provedena meta-analiza na 17 studija pokazala je nužnost podrške učitelja kako bi se koncepti naučeni u igri prenijeli u druge kontekste [70]. Iz navedenog se zaključuje da pedagoška primjena igara u učenju i poučavanju neće zamijeniti učitelje i učionice, ali može zamijeniti neke udžbenike ili laboratorije, te kako igre neće postati nastavna metoda već ih treba koristiti kao nastavno sredstvo.

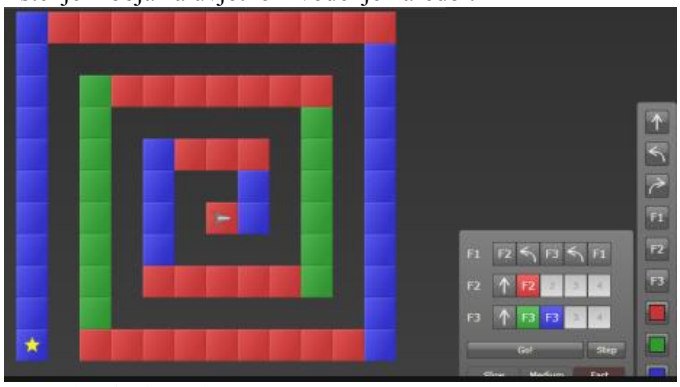
Područje izrade igara u svrhu poučavanja programiranja i računalne znanosti pomoću igara je relativno novo pa nema dovoljno empirijskih dokaza o učenju pomoći igara u tom području što ukazuje na područje otvoreno za istraživanje i proučavanje. Neke od igara koje se koriste za poučavanje programiranja su LightBot [71] i RoboZZle [72]. Navedene igre spadaju u skupinu puzzle igara u kojima se slažu *povuci i pusti* (eng. d*rag and drop*) naredbe kako bi se pokrenuo robot i izvršio zadani cilj. Za rješavanje zadanog problema igrač mora ponuditi semantički točno rješenje, a u isto vrijeme ne postoji problem sa sintaksom. Napretkom kroz igru igrače se potiče na učinkovito korištenje naredbi i na rješavanje sve kompleksnijih problema.

Cilj igre LightBot je robotom kojim igrač upravlja posjetiti sva plava polja i osvijetliti ih. Igra počinje s jednostavnim mapama i naredbama (naprijed, lijevo, desno, skoči, uključi lampicu) da bi napretkom kroz igru dobivao sve složenije zadatke što podrazumijeva ograničenost broja naredbi, uvođenje funkcija i rekurzije.



Slika 2. LightBot okruženje

Slično je i kod igre RoboZZle, gdje ima manje naredbi (naprijed, lijevo, desno), ali je podržana mogućnost grananja korištenjem boja za uvjetno izvođenje naredbi.



Slika 3. Robozzle okruženje

Navedene igre su besplatne i dostupne za izvođenje na više platformi, uključujući i mobilna računala. Iz ovih igara je nastao cijeli niz novih igara kojima se ispitivao utjecaj na učenje osnovnih koncepata programiranja.

Jedna od takvih igara je i *Conveyor* pomoću koje je provedeno istraživanje o usvajanju osnovnih koncepata programiranja igrajući igru. U istraživanju su sudjelovali studenti računalne znanosti i različitih inženjerskih smjerova s prethodnim programerskim iskustvom, ali i studenti drugih smjerova bez programerskog iskustva. Studenti su igrali igru 45 minuta i imali 9 razina koje su mogli proći. Svaka razina je bila teža i upoznavala je studente s novim konceptima. Pokazalo se da je na 3. i 4. razini blago opalo zanimanje za igru, vjerojatno jer su učenici savladali osnovne koncepte pa im je igra postala manje zanimljiva, ali na 5. razini su uvedena grananja što je otežalo igru, povećalo zainteresiranost, ali i broj pokušaja u kojem se igra uspješno završila. Kvalitativna analiza je pokazala da je učenicima igra bila zabavna, ali nisu bili zadovoljni grafičkim i audio performansama. Od učenika koji su studirali inženjerstvo 63% njih je za 10% povećalo svoju prosječnu ocjenu studija [73].

Još jedan sličan sustav je VISPROCON pomoću kojeg se vršilo ispitivanje o utjecaju igara na usvajanje programerskih koncepata, gdje se uočilo kako igra motivira učenike ne samo na uspješno završavanje igre već i na vježbanje kako bi dobili što bolja rješenja. U istraživanju su sudjelovali učenici različitih uzrasta pa se može zaključiti kako se s ovakvim pristupom može početi dosta rano [74].

U školskoj godini 2010./2011. je provedeno pilot istraživanje u pet viših škola u SAD-u, a u svakoj školi se poučavalo različitim pristupom. U jednoj od škola se istraživao pristup učenju osnovnih principa računalne znanosti (eng. *CS principles*) u kojem je korišten i LightBot kao alat za učenje osnovnih koncepata programiranja. Kao pedagoški pristup u poučavanju se koristio oportunistički pristup s ciljem pojačanja interakcije čovjek-čovjek (eng. *human-to-human interaction*). Način koji se pokazalo uspješnim je aktivno učenje kroz predavanja i korištenje LightBota, a uporaba numeričkih primjera nije dala dobre rezultate, što se pokazalo kao najvećom pogreškom, vjerojatno zbog „matofobije" većine učenika, pa su se puno učinkovitijim pokazali nenumerički primjeri za iste koncepte poučavanja [75]. Li i Watson su ponudili tri pristupa učenju programiranja:

- Pristup temeljen na izradi igara
- Pristup temeljen na igranju
- Pristup temeljen na vizualizaciji

U tablici 2 je vidljiva podjela različitih alata za učenje programiranja prema takvom pristupu [76].

Tablica 2: Postojeći sustavi temeljeni na igrama za učenje programiranja prema [76]

		Izrada igara				Vizualizacija					Igre								
		2006: RoboCode	2010: Alice 2	2010: Greenfoot	2010: Scratch	2003: Jeroo	2006: objectKarel++	2007: C-Sheep	2008: Bomberman	2008: Turtlet	2007: Catacombs	2007: Saving Sera	2009: EleMental	2009: Resource Craft	2009: Wu's Castle	2010: Prog & Play	2010: Robozzle	2011: Lightbot	2011: PlayLOGO 3D
Stil programiranja	proceduralno							x	x	x				x		višestruko	x	x	x
	funkcionalno													x					
	temeljeno na objektima		x		x														
	objektno-orjentirano	x		x		x	x				x		x	x	x				
Programski jezici	uobičajeni jezici	x		x				x	x	x	x	x	x	x	x	x			
	posebno osmišljeni jezici		x		x	x	x										x	x	x
Naučeni koncepti	uvjeti	Java jezik	x	Java jezik	x	x	x	x	C jezik	x	x		x	x	x	x			
	for petlja				x	x	x	x		x	x	x		x	x	x			
	while petlja		x		x	x	x	x		x	x	x		x	x	x			
	varijabla		x		x	x		x		x			x	x	x	x			
	parametar		x		x	x	x	x		x			x			x			x
	funkcija/metoda		x		x	x	x	x		x			x			x	x	x	x
	tip podatka definiran od strane korisnika																		
	rekurzija							x					x				x	x	
	nizovi/polja		x			x		x						x	x				
Prikaz koda	tekst	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x
	slike																x	x	
Konstrukcija koda	pisanje koda	x		x		x	x	x	x	x	x	x	x	x	x	x			x
	sastavljanje grafičkih objekata		x		x												x	x	
	biranje/ ispunjavanje forma						x			x									
Metafora mikrosvijeta	općenito		x	x	x														
	specifično	x				x	x	x	x	x	x	x	x	x	x	x	x	x	x

V. MODEL IGRE ZA UČENJE PROGRAMIRANJA

Prema navedenoj teoretskoj podlozi o igrama, učenju i programiranju, model igre koja služi za poučavanje može se opisati kroz tri dimenzije:

- pravila igre: zadovoljavanje tehničkih pravila igre (prema Juulsovoj definiciji),
- zabava: mora zadovoljavati uvjete zabavnosti igre (prema LeBlancovoj taksonomiji),
- edukativnost: opisuje obrazovni aspekt igre.

Prve dvije dimenzije su opisane u III, a potrebno je dodatno definirati edukativnu dimenziju igre. Edukativna dimenzija odnosi se na obrazovni aspekt igre. Igra kod igrača treba izazivati optimalnu razinu težinskog ciklusa: neravnoteža, istraživanje, svjesnost, u području poučavanja prema opisanom u II. Kako je navedena razina različita za svaku osobu, igra mora sadržavati elemente inteligencije u području prepoznavanja značajki igrača. Igra oblikovana prema značajkama igrača pruža različite razine težine ciklusa kroz promjene slijeda kako bi kod igrača zadržala efekt izazova te ne bi postala preteška niti prelagana.

Uloga učenja pomoću igara može se također podijeliti u dvije dimenzije:

- učenik kao igrač: niža razina u kojoj učenik uči određene koncepte kroz interakciju s igrom,
- učenik kao dizajner igre: viša razina u kojoj učenik mora razumjeti igru iz perspektive igrača i dizajnera.

Predloženi model će biti detaljno razrađen u budućem radu.

VI. ZAKLJUČAK

Jasno je kako se vrijeme promijenilo te da promjene koje su došle utječu na sve vidove života: socijalni, ekonomski, politički, obrazovni i dr. Iako se promjene događaju se u svim područjima života, područje koje se najsporije mijenja je područje obrazovanja koje se nije značajno promijenilo još od industrijskog doba. Pitanje koje se postavlja je: Što napraviti kako bi obrazovni sustav što bolje pratio promjene u društvu i tehnologiji? Današnje doba je digitalno doba i već sada je teško zamisliti život bez tehnologije. Može se reći da je moderno, zapadno društvo postalo ovisno o tehnologiji. Ipak, ona nije na pravi način integrirana u školski sustav. Način učenja i poučavanja nije se bitno promijenio, a životom uz tehnologiju promijenio se način razmišljanja pa tako generacija digitalnih urođenika uči na drugačiji način u usporedbi s ne tako davnim prethodnim generacijama. Osim što se ne uvažava drugačiji način učenja i poučavanja u školi, informatika nije ušla u školu na zadovoljavajući način. Koji je to zadovoljavajući način? Djeca se u školama pripremaju za samostalni život čiji su sastavni dio računala bez kojih je teško već danas zamisliti bilo koje zanimanje, a pogotovo neko zanimanje u budućnosti. Snalaženje u tehnološkom okruženju podrazumijeva razumijevanje tehnologije i računala kako bi se djeca bolje mogla uklopili u taj svijet. Već sad bi minimum usvojenog znanja o računalima bila znanja koja se izjednačavaju s pojmom informatičke pismenosti, informatička okretnost te osnove računalnog razmišljanja koja se smatra četvrtom analitičkom sposobnošću i čiju dostupnost treba osigurati djeci kroz školovanje. Računalno razmišljanje se najbolje uči kroz učenje programiranja. Unatoč početnom entuzijazmu za programiranjem 1980-ih, posljednje desetljeće obilježeno je velikim padom zanimanja za programiranjem i općenito učenjem računalne znanosti, a povećava se potražnja za takvim radnim mjestima. Kako se dogodio uočeni disbalans i kako ublažiti i poboljšati taj omjer pitanja su od ključne važnosti za moderno društvo koje za cilj ima osposobiti kroz školovanje pojedince za samostalan i produktivan život. Kada se pogleda način poučavanja programiranja u samim počecima pojave programiranja i usporedi sa stanjem danas, može se vidjeti da nije došlo do velike promjene u pristupu poučavanju, a u isto vrijeme smo kao korisnike obrazovnog procesa dobili generaciju digitalnih urođenika. Kod poučavanja programiranja se naglasak i dalje stavlja na algoritme, na linearno razmišljanje, rješavanje matematičkih problema što je današnjoj djeci potpuno strano. Djeca odrasla u digitalnom dobu imaju drugačiji način razmišljanja od prethodnika, oni su pretežno „bricolage“ mislioci. U školama i na fakultetima kao početni programski jezici još uvijek dominiraju „klasični“ proceduralni jezici koji su današnjoj djeci potpuno strani i izlaze van njihovog prirodnog tehnologijskog okruženja. Programiranje je samo po sebi jako teško za učenje i poučavanje, pa kad se tome doda i težak programski jezik u kojem program mora biti semantički i sintaktički dobro napravljen, kao rezultat se dobije opadanje općeg zanimanja za programiranje. U ovom radu prikazan je pregled istraženosti područja učenja i poučavanja programiranja temeljenog na igrama. Većina prikazanih istraživanja ne daje jasan odgovor na pitanje: Kakav utjecaj igre imaju na učenje programiranja? Kao rezultat pregleda istraženosti predložen je model igre za učenje i programiranje (opisan u V) koji uvodi i pedagoški aspekt igre koji kao okosnicu koristi razinu težinskog ciklusa: neravnoteže, istraživanje, svjesnost. Predloženi model igre sadržava elemente inteligencije u smislu prilagodbe navedenog težinskog ciklusa pojedincu. Pored navedenog, modela uvode se i različite uloge: igrača i dizajnera koje nisu jasno razdvojene, već se po potrebi izmjenjuju s ciljem ostvarivanja što boljeg usvajanja znanja programiranja. Model i uloge zajedno čine okvir za učenje i poučavanje programiranja temeljen na igrama. Kako bi se ispitala učinkovitost postavljenog okvira potrebno je provesti istraživanje kojim će se kroz konkretnu primjenu na odabranim pojmovima iz područja programiranja vrednovati navedeni okvir.

LITERATURA

[1] C. M. Reigeluth, »Instructional Theory and Technology for the New Paradigm of Education,« *RED. Revista de Educación a Distancia,* svez. 32, pp. 1-18, 2012.

[2] T. Armstrong, Multiple intelligences in the classroom, Alexandria, Va: Association for Supervision and Curriculum Development, 2000.

[3] D. H. Pink, »Revenge of the Right Brain,« *PUBLIC MANAGEMENT -LAWRENCE THEN WASHINGTON*, svez. 89, br. 6, pp. 10-13, 2007.

[4] M. Fox, »Learning design and e-learning,« *An Epic White Paper*, 2003.

[5] M. Prensky, Teaching digital natives: partnering for real learning, Thousand Oaks, Calif: Corwin, 2010.

[6] F. Levy i R. J. Murnane, The New Division of Labor: How Computers Are Creating the Next Job Market, Princeton University Press, 2012.

[7] D. H. Pink, A Whole New Mind: Moving from the Information Age to the Conceptual Age, Riverhead Hardcover, 2005.

[8] M.-H. Tsai, C.-H. Huang i J.-Y. Zeng, »Game Programming Courses for Non Programmers,« u *Proceedings of the 2006 international conference on Game research and development*, Perth, 2006.

[9] M. Prensky, »Digital natives, digital immigrants.,« *On the Horizon*, pp. 1-6, 2001.

[10] Office of Technology Assessment, Computerized Manufacturing Automation: Employment, Education and the Workplace, Washington, D. C.: U.S. Congress, 1984.

[11] National Research Council Committee on Information Technology Literacy, Being fluent with information technology, Washington: National Academy Press, 1999.

[12] J. Wing, »Computational thinking,« *Communications of the ACM*, svez. 49, br. 3, p. 33–35, 2006.

[13] J. Wing, »Computational thinking and thinking about computing,« *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, svez. 366, pp. 3717-3725, 2008.

[14] S. Uludag, M. Karakus i S. W. Turner, »Implementing IT0/CS0 with Scratch, App Inventor for Android, and Lego Mindstorms,« u *SIGITE '11: Proceedings of the 2011 conference on Information technology education* , West Point, NY, USA, 2011..

[15] A. Perlis, »The computer in the university,« u *In M. Greenberger, Ed., Computers and the World of the Future*, Cambridge, MA, USA, MIT Press, 1962..

[16] P. J. Denning i A. McGettrick, »Recentering computer science,« *Communications of the ACM, 48*, pp. 15-19, November 2005..

[17] C. Litecky, B. Prabhakar i K. Arnett, »The IT/IS Job Market: A Longitudinal Perspective,« u *SIGMIS CPR '06: Proceedings of the 2006 ACM SIGMIS CPR conference on computer personnel research: Forty four years of computer personnel research: achievements, challenges & the future* , 2006..

[18] U. Gupta i L. Houtz, »High School Students' Perceptions of Information Technology Skills and Careers,« *Journal of Industrial Technology*, svez. 16, br. 4, pp. 2-8, 2000.

[19] J. S. Brown, »GROWING UP DIGITAL How the Web Changes Work, Education, and the Ways People Learn,« *US Distance Learning Association Journal*, svez. 16, br. 2, 2002.

[20] M. Prensky, »Digital Game-based Learning,« *Computers in Entertainment*, svez. 1, br. 1, pp. 21-21, 2003.

[21] D. Oblinger i J. Oblinger, »Is It Age or IT: First Steps Toward Understanding the Net Generation,« *CSLA Journal*, svez. 29, br. 2, pp. 8-16, 2006.

[22] S. Turkle i S. Papert, »Epistemological Pluralism and the Revaluation of the Concrete,« *Journal of Mathematical Behavior*, svez. 11, br. 1, pp. 3-33, 1992.

[23] Y. Sedelmaier i D. Landes, »Practicing Soft Skills in Software Engineering: A Project-Based Didactical Approach,« u *Overcoming Challenges in Software Engineering Education*, IGI Global, 2014, pp. 161-179.

[24] Bailey i M. W., »IRONCODE: think-twice, code-once programming,« u *Proceedings of the 36th SIGCSE technical symposium on Computer science education*, St. Louis, Missouri, 2005.

[25] C. Kelleher, R. Pausch i S. Kiesler, »Storytelling Alice Motivates Middle School Girls to Learn,« u *CHI 2007 Proceedings - Programming By & With End-Users*, San Jose, CA, USA, 2007..

[26] P. Wouters, v. d. S. Erik D. i v. O. Herre, »Current Practices in Serious Game Research: A Review from a Learning Outcomes Perspective,« u *Games-Based Learning Advancements for Multi-Sensory Human Computer Interfaces: Techniques and Effective Practices*, IGI Global, 2009, pp. 232-250.

[27] R. Van Eck, »Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless,« *EDUCAUSE Review*, svez. 41, br. 2, p. 16–30, 2006.

[28] S. Papert, »Does Easy Do It? Children, Games, and Learning,« *Game Developer*, svez. 5, br. 6, 1998.

[29] J. Piaget, The origins of intelligence in children, New York: International Universities Press, 1952.

[30] T. Yuen i M. Liu, »How interactive multimedia authoring transforms object-oriented thinking,« u *Proceedings of the 41st ACM technical symposium on Computer science education*, Milwaukee, Wisconsin, 2010.

[31] J. Kirriemuir i A. McFarlane, »Literature Review in Games and Learning,« Futurelab, Bristol, 2004.

[32] J. Smed i H. Hakonen, »Towards a Definition of a Computer Game,« Turku Centre for Computer Science, Turku, 2003.

[33] J. Juul, »The Game, the Player, the World: Looking for a Heart of,« u *Level Up: Digital Games Research Conference Proceedings*, Utrecht, 2003.

[34] G. Costikyan, »I Have No Words & I Must Design: Toward a Critical Vocabulary for Games,« u *Computer Games and Digital Cultures Conference Proceedings*, 2002.

[35] R. Hunicke, M. LeBlanc i R. Zubek, »MDA: A formal approach to game design and game research,« u *Proceedings of the AAAI Workshop on Challenges in Game AI*, San Jose, 2004.

[36] I. Milne i G. Rowe, »Difficulties in Learning and Teaching Programming—Views of Students and Tutors,« *Education and Information Technologies*, svez. 7, br. 1, pp. 55 - 66, 2002.

[37] A. Gomes i A. J. Mendes, »Learning to Program – Difficulties and Solutions,« u *International conference on Engineering Education*, Coimbra, Portugal, 2007.

[38] L. E. Winslow, »Programming pedagogy - a psychological overview,« *SIGCSE Bulletin*, svez. 28, br. 3, pp. 17-22, 1996.

[39] A. Robins, J. Rountree i N. Rountree, »Learning and Teaching Programming: A Review and Discussion,« *Computer Science Education*, svez. 13, br. 2, pp. 137 - 172, 2003.

[40] M. Guzdial, »Programming environments for novices,« u *Computer Science Education Research*, 2004, pp. 127-155.

[41] B. Violino, »Time to Reboot,« *Communications of the ACM - A Direct Path to Dependable Software*, svez. 52, br. 4, p. 19, 2009.

[42] M. Resnick, J. Maloney, M. Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman i Y. Kafai, »Scratch: Programming for all,« *Communications of the acm*, svez. 52, br. 11, pp. 60-57, 2009.

[43] M. Hassinen i H. Mäyrä, »Learning programming by programming: a case study,« u *Baltic Sea '06*, Koli, Finland, 2006.

[44] N. F. A. Zainal, S. Shahrani, N. F. M. Yatim, R. A. Rahman, M. Rahmat i R. Latih, »Students' perception and motivation towards programming,« u *UKM Teaching and Learning Congress*, 2011..

[45] A. Forte i M. Guzdial, »Motivation and nonmajors in computer science: identifying discrete audiences for introductory courses,« *Education, IEEE Transactions on, 48(2)*, pp. 248-253, May 2005..

[46] C. M. Lewis, »How programming environment shapes perception, learning and goals: logo vs. scratch,« New York, 2010.

[47] E. Fusco, »Matching curriculum to studet's cognitive levels,« *Educational Leadership*, p. 47, 1981.

[48] J. Kramer, »Is abstraction the key to computing?,« *Communications of the ACM*, svez. 50, br. 4, 2004.

[49] R. Moser, »A fantasy adventure game as a learning environment: why learning to program is so difficult and what can be done about it,« u *ITiCSE '97*, Uppsala, Sweden, 1997.

[50] C. Dann, »Alice 3: Concrete to Abstract,« svez. 52, br. 8, 2009.

[51] S. Papert, Mindstorms: children, computers, and powerful ideas, New York: ACM, 1980.

[52] P. Brusilovsky, E. Calabrese, J. Hvorecky, A. Kouchnirenko i P. Miller, *Education and Information Technologies*, svez. 2, br. 1, pp. 65-83, 1997.

[53] R. Pausch i J. Zaslow, The Last Lecture, New York: Hyperion, 2008.

[54] L. Werner, S. Campe i J. Denner, »Children learning computer science concepts via Alice game-programming,« u *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, New York, 2012.

[55] L. L. Werner, S. Campe i J. Denner, »Middle school girls + games programming = information technology fluency,« u *Proceedings of the 6th conference on Information technology education*, Newark, 2005.

[56] D. B. R. Werner, »Can Middle-Schoolers use Storytelling Alice to Make Games? Results of a Pilot Study,« Orlando, 2009.

[57] P. K. R. R. Maloney, »Programming by choice: urban youth learning programming with scratch,« Portland, 2008.

[58] L. Malan, »Scratch for budding computer scientists,« New York, 2007.

[59] A. Wilson, T. Hainey i T. Connolly, »Evaluation of Computer Games Developed by Primary School Children to Gauge Understanding of Programming Concepts,« *Interational Journal of Games-based Learning*, svez. 3, br. 1, pp. 93-109, 2013.

[60] A. Fowler i B. Cusack, »Kodu game lab: improving the motivation for learning programming concepts,« u *Proceedings of the 6th International Conference on Foundations of Digital Games*, Bordeaux, 2011.

[61] A. B.-A. Meerbaum-Salant, »Learning computer science concepts with scratch,« Aarhus, Denmark, 2010.

[62] A.-J. Lakanen, V. Isomöttönen i V. Lappalainen, »Life two years after a game programming course: longitudinal viewpoints on K-12 outreach,« u *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 2012.

[63] S. Leutenegger i J. Edgington, »A games first approach to teaching introductory programming,« u *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, Covington, Kentucky, 2007.

[64] S. Egenfeldt-Nielsen, »Overview of research on the educational use of video games,« *Digital kompetanse*, svez. 1, br. 3, pp. 184-213, 2006.

[65] D. D. Curtis i M. J. Lawson, »Computer Adventure Games as Problem-Solving Environments,« *International Education Journal*, svez. 3, br. 4, 2002.

[66] S. Egenfeldt-Nielsen, »Third generation educational use of computer games,« *Journal of Educational Multimedia and Hypermedia*, svez. 16, br. 3, pp. 263-281, 2007.

[67] W. D. a. D. C. a. D. S. a. D. C. a. S. Cooper, »Mediated transfer: Alice 3 to Java,« *43rd ACM technical symposium on Computer Science Education*, 2012.

[68] K. Ash, »Digital Gaming Goes Academic,« *Education Week*, svez. 30, br. 25, pp. 24-28, 2011.

[69] K. L. McClarty, A. Orr, P. M. Frey, R. P. Dolan, V. Vassileva i A. McVay, »A Literature Review of Gaming in Education,« *Gaming In Education*, 2012.

[70] F. Ke, »A qualitative meta-analysis of computer games as learning tools,« *Handbook of research on effective electronic gaming in education*, svez. 1, pp. 1-32, 2009.

[71] LightBot, »http://light-bot.com/,« LightBot. [Mrežno].

[72] »http://www.robozzle.com/,« roboZZle. [Mrežno].

[73] B. R. C. Marques, S. P. Levitt i K. J. Nixon, »Software visualisation through video games,« u *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, 2012.

[74] K. Bromwich, M. Masoodian i B. Rogers, »Crossing the game threshold: A system for teaching basic programming constructs,« u *Proceedings of the 13th International Conference of the NZ Chapter of the ACM's Special Interest Group on Human-Computer Interaction*, Dunedin, 2012.

[75] L. Snyder, T. Barnes, D. Garcia, J. Paul i B. Simon, »The first five computer science principles pilots: summary and comparisons,« *ACM Inroads*, svez. 3, br. 2, pp. 54-57, 2012.

[76] F. W. Li i C. Watson, »Game-based concept visualization for learning programming,« u *Proceedings of the third international ACM workshop on Multimedia technologies for distance learning*, 2011.

[77] C. D. Werner, »Children learning computer science concepts via Alice game-programming,« New York, 2012.