O3 – Obrazovni sadržaji za učitelje

O3/A2 – VODIČ ZA UČITELJE O PRISTUPU CODING4GIRLS TEMELJENOM NA OBRAZOVNIM IGRAMA

Podaci o dokumentu

Rezultat: O3/A2 – Vodič za učitelje o pristupu CODING4GIRLS temeljenom na obrazovnim igrama

Intelektualni rezultat: O3 – Obrazovni sadržaji za učitelje

Odgovorni partner: Jugozapadno sveučilište "Neofit Rilski" (Bulgarska)

Grupa autora

Jugozapadno sveučilište "Neofit Rilski", Bulgarska

Daniela Tuparova, Boyana Garkova, Ivanichka Nestorova, Rositsa Georgieva

Sveučilište u Tesaliji, Grčka

Hariklia Tsalapata, Olivier Heidmann, Kostas Katsimentes, Christina, Taka Roxani, Sotiri Evangelou, Nadia Vlachoutsou

Sveučilište u Rijeci, Odjel za informatiku, Hrvatska

Nataša Hoić-Božić, Martina Holenko Dlab, Ivona Franković, Marina Ivašić Kos

EU-Track, Italija

Michela Tramonti, Alden Meirzhanovich Dochshanov, Luigi Tramonti

Virtual Campus, Portugal

Carlos V. Carvalho, Rita Durão.

Sveučilište u Ljubljani, Slovenija

Joze Rugelj, Mateja Bevcic, Spela Cerar, Matej Zapushek

Istanbul Valiligi, Turska

Ahu Şimşek, K. Fatih Mutlu, Abdurrahman Saygın

SADRŽAJ

1. UVOD

Projekt Coding4Girls bavi se otvorenim i inovativnim obrazovanjem koje je usmjereno na vještine programiranja te je na taj način izuzetno važno i traženo za današnje digitalno doba i tehnološko društvo. Isto tako, sposobnost dizajniranja algoritama, kodiranja i programiranja postale su relevantne vještine svakog pojedinca jer su povezane s logičkim zaključivanjem i sposobnostima rješavanja problema. Vještine iz područja informatike su vrlo poželjne jer su dio kompetencija koje poslodavci zahtijevaju u sektorima vođenim inovacijama koji će dovesti do održivog gospodarskog rasta u godinama koje slijede i većeg zapošljavanja te, kao rezultat toga, do socijalne kohezije.

Međutim, trenutno nedostaje stručnjaka iz područja informatike, a tvrtke se suočavaju s poteškoćama u pronalaženju zaposlenika s potrebnim IKT vještinama. Coding4Girls želi premostiti tu prazninu učinkovitim poticanjem razvoja vještina iz područja IKT u višim razredima osnovne škole i prvom razredu srednje škole, što je period kada mnogi učenici gube zanimanje za informatiku. Povrh toga, u ovom je području uočen rodni jaz jer posebice djevojke nisu zainteresirane za bavljenje informatičkom karijerom.

Coding4Girls se bavi inkluzivnim obrazovanjem, obukom i mladima promičući jednake mogućnosti za djevojčice i dječake pri odabiru budućih zanimanja i razvoju karijera u području informatike. Ovaj cilj će se ostvariti utjecanjem na uklanjanje pogrešnih percepcija i stavova učenika, učitelja i roditelja o karijeri iz područja informatike te naglašavanjem prednosti odabira ovih zanimanja i za djevojčice i dječake kroz povezanost sa stvarnim životom i isticanjem činjenice da se takve karijere nagrađuju neovisno o spolu ili porijeklu. Time će se ujedno utjecati na rješavanje problema nedovoljnih postignuća u razvoju računalnih vještina, koje su dio širih kompetencija u znanosti i tehnologiji (STEM).

Projekt pomaže izgraditi poticajno okruženje među učenicima u školama koja će im omogućiti uspjeh u budućoj akademskoj (u tercijarnom ili drugom kontinuiranom obrazovanju iz informatike) ili profesionalnoj (u strukovnim ili profesionalnim aktivnostima vezanim uz informatiku) karijeri. Coding4Girls se također bavi razvojem učiteljskih kompetencija i profila učiteljske profesije osnažujući učitelje da učinkovito grade poželjne informatičke vještine među svojim učenicima. Također omogućuje učiteljima da vode

inicijative koje promiču perspektive rodne ravnopravnosti u akademskom i poslovnom okruženju.

Ovaj vodič za učitelje dio je intelektualnih rezultata razvijenih u okviru projekta. Učitelji će pronaći informacije o: glavnim konceptima na kojima se temelji Coding4Girls metodologija (učenje temeljeno na igrama, obrazovne igre, pristup *design thinking*), položaju djevojaka u IKT i njihovoj percepciji obrazovnih igri, značajkama okruženja za igranje Coding4Girls s dvije komponente – za učitelje i za učenike, pristupima za poučavanje programiranja temeljenim na igrama s primjerima korisnih programskih okruženja. Vodič uključuje i scenarije za učenje programiranja prema Coding4Girls metodologiji na osnovu razvijenog okruženja temeljenog na igrama.

2. OSNOVNI POJMOVI

2.1. Obrazovne igre (eng. *Serious games*)

Igra je širok pojam i teško je ukazati na najvažnije karakteristike koje određenu aktivnost čine igrom. Za Thortona je najvažnija značajka igre interaktivnost. Johnston igru vidi kao aktivnost koja ima pravila koja svi sudionici moraju poštivati, jedan ili nekoliko ciljeva, interakciju i dinamične vizualne elemente (Johnston & de Felix, 1993). Malone (Malone T., 1981.) u svojem radu ističe fantaziju, znatiželju, izazov i kontrolu kao najvažnije komponente svake igre. Najopsežniju analizu daju Garris i suradnici (Garris, Ahlers i Driskell, 2002) tvrdeći da su te komponente natjecanje, izazov, socijalne interakcije, razgovor i fantazija. Prenskyjeva teorija učenja temeljenog na igrama jedna je od najutjecajnijih. Autor tvrdi da se igra sastoji od sljedećih elemenata: pravila, ciljevi i zadaci, rezultati i povratne informacije, konfliktne situacije (natjecanje, izazov i protivljenje), interakcija i maštoviti okvir (Prensky, 2001). Autori knjige "Serious games" definiraju igru kao dobrovoljnu aktivnost (oblik slobode) odvojene od stvarnog života (imaginarni svijet koji može imati ili nema veze sa stvarnim životom), koja upija igračevu punu pažnju i igra se prema utvrđenim pravila kojih se svi igrači moraju pridržavati (Michael & Chen, 2006).

Jedna od najistaknutijih značajki igre je interaktivnost i odnosi se na interakciju s računalom, protivnikom (upravljan računalom ili čovjekom koji igra) ili s drugim timskim igračima. Računalne igre dijele se obzirom na broj igrača koji su istovremeno uključeni u igru na pojedinačne igre, igre za više igrača i masovne igre za više igrača. Prema žanru poznajemo arkadne, akcijske, ratne, igranje uloga, strateške, društvene, sportske, simulacijske i avanturističke igre. Ljestvice popularnosti računalnih igara pokazuju da su najčešće igrane igre masovne igre za više igrača u raznim ulogama (eng. *multiplayer role-playing games*), strateške i akcijske igre.

Kako bi se zaista shvatila snaga igre, prvo je potrebno razmotriti dječju igru. Dječja igra poznata je kao jedna od najvažnijih aktivnosti koja pomaže razviti važne vještine za cijeli život. Kroz igru dijete poboljšava svoje intelektualne sposobnosti otkrivanjem prvih osnovnih pojmova iz stvarnog svijeta i uspostavljanjem održivih veza između njih. Jean Piaget igru vidi kao ugradnju novih materijala u postojeću kognitivnu strukturu i učvršćivanje novo

naučenog ponašanja. Freud naglašava emocionalne blagodati igre tvrdeći da ona smanjuje objektivnu i instinktivnu tjeskobu, pomažući djetetu u rješavanju emocionalnih problema. Socijalni konstruktivisti vjeruju da je igra važna zbog razvoja socijalnih vještina.

Učenje uz igru jedna je od najčešćih aktivnosti u ranoj dobi, ali pozitivni učinci igranja i igara na učenje često se zanemaruju u formalnom obrazovanju. Puno je istraživanja provedeno na polju učenja temeljenog na igrama, pokazujući da predstavljanje sadržaja za učenje u formatu računalne igre ima pozitivne učinke na motivaciju. Igra može pomoći učenicima da posvete pažnju nastavnim sadržajima za učenje, a istodobno je izvor zabave koja im pruža užitak i zadovoljstvo. Ostvarenost ove dvije komponente u procesu učenja rezultira opuštenim i visoko motiviranim učenikom koji je stoga spremniji učiti. Ostale pedagoške prednosti su suradničko učenje (mogućnosti igri s više igrača), eksperimentalno i aktivno učenje, učenje temeljeno na problemima i aktivnosti iz autentičnog okruženja. Korištenje IKT-a danas je uobičajeno u formalnom obrazovanju, što nam daje priliku da razvijemo kvalitetne sadržaje kako bismo potaknuli učenje.

Obrazovne igre obično se odnose na igre koje se koriste za trening, oglašavanje, simulaciju ili edukaciju. Omogućuju učenicima da iskuse situacije koje su u stvarnom svijetu nemoguće iz različitih razloga, poput sigurnosti, troškova ili vremena. Trebale bi imati definirane ishode učenja, a za njih se tvrdi da imaju pozitivne učinke na razvoj novih znanja ili različitih vještina kod igrača. Da bismo osmislili dobru obrazovnu igru, moramo biti sposobni dizajnirati i proizvesti dobar softver. Međutim, obrazovne su igre puno više od softvera jer trebamo biti sposobni dizajnirati i proizvesti također i kvalitetan obrazovni proces. Čini se da je potencijalna vrijednost obrazovnih igara za učenje velika, ali i dalje ostaje otpor upotrebi igara u učionici. Promišljen način da se više učitelja uvjeri u isprobavanju igre je putem pedagogije, povezujući elemente postojećih dizajna igara s prihvaćenim teorijama učenja i poučavanja.

Rieber, Smith i Noah već su 1998. godine izjavili da postoje dvije različite primjene igara u obrazovanju (Rieber, Smith i Noah, 1998): igranje i dizajniranje igara. Igranje igara tradicionalni je pristup u kojem se učenicima pružaju gotove igre. Dizajniranje igara pretpostavlja da je čin izgradnje igre sam po sebi put do učenja, bez obzira na to pokazuje li se igra zanimljiva drugim ljudima. Ideja "učenja dizajniranjem" temelji se na pretpostavci da

je aktivno sudjelovanje u procesu dizajna i razvoja najbolji način da se nešto nauči. Ovaj je pristup danas dobio veću važnost zbog dostupnosti sve više računalnih alata za dizajn igara.

2.2. Učenje uz pomoć igara

Nekoliko je razloga koji odgajateljima skreću pozornost na igre (Gee, 2007). U formalnom obrazovanju doživljavamo pomak s tradicionalnog didaktičkog modela, usredotočenog na podučavanje, na model usmjeren na učenika koji naglašava ulogu aktivnog učenika. Također je promijenjen pogled na ishode učenja, s nižih razina u taksonomiji, poput samo prisjećanja, na više razine, poput pronalaženja i korištenja podataka u novim situacijama (Rugelj, 2016.).

Prensky (Prensky, 2001.), Gee (Gee, 2003.) i Whitton (Whitton, 2009.) definirali su učenje uz pomoć igara kao proces učenja uz upotrebu digitalnih igara. Igre mogu pružiti motivaciju za učenje, povećavajući tako prilike za postizanje željenih ishoda učenja. Učenje se definira kao stjecanje znanja ili vještina iskustvom ili praksom te nema boljeg načina za učenja od igre (Pivec i Kearney, 2007.), (Pivec, Koubek i Dondi, 2004.) Gotovo sve studije o učenju uz pomoć igara pokazuju da su učenici vrlo motivirani kada se sadržaji za učenje prezentiraju u formatu računalne igre i da to pozitivno utječe na ubrzanje procesa učenja (Zapušek i Rugelj, 2013). Učenicima je potrebna motivacija kako bi se usredotočili na ono što trebaju naučiti, ali za kvalitetno učenje to ipak nije dovoljno. Usporedba ishoda učenja učenika koji su učili pomoću računalnih igara i onih koji uče s drugom vrstom materijala za učenje pokazuje da među njima nema značajne razlike. To se obično događa zbog neprikladnog dizajna igre. Igre mogu biti vrlo privlačne učenicima, ali ukoliko samo zabavljaju, a ne podučavaju, uporaba igara u obrazovanju nema puno smisla. Dakle, moramo otkriti koji su elementi koji čine računalnu igru obrazovnom računalnom igrom.

Gross (Gross, 2003) tvrdi da digitalne igre s obrazovnom svrhom moraju imati dobro definirane ishode učenja i moraju promicati razvoj važnih strategija i vještina za povećanje kognitivnih i intelektualnih sposobnosti učenika.

Prema Maloneu (Malone T. W., 1981b) te Garrisu i suradnicima (Garris, Ahlers i Driskell, 2002), elementi koji doprinose obrazovnim vrijednostima digitalnih igara su: senzualni podražaji (vizualni i zvučni prikazi materijala za učenje), fantazija (kontekst prikazan u imaginarnom okruženju), izazov (zahtjevna ili poticajna situacija) i znatiželja (želja za znanjem ili učenjem). Ti se elementi moraju uključiti na objedinjenu platformu kako bi

strukturirali ciljeve i pravila, kontekst smislenog učenja, privlačnu priču, neposredne povratne informacije, visoku razinu interaktivnosti, izazova i natjecanja, slučajnih elemenata iznenađenja i bogatih okruženja za učenje.

Igra se može iskoristiti za učenje jer uključuje mentalnu (a ponekad i fizičku) stimulaciju i razvija praktične vještine - prisiljava igrača da odlučuje, bira, definira prioritete, rješava probleme itd. Neposredna nagrada (i povratna informacija) je glavni motivacijski čimbenik, bilo da se osigurava kroz elemente igre (npr. veća životna snaga, pristup novim razinama, itd.) ili kao neurološki impulsi (npr. sreća, osjećaj postignuća, itd.). Igre mogu biti društveno okruženje te ponekad uključuju velike distribuirane zajednice. Podrazumijevaju i sposobnosti samoučenja (igrači su često dužni tražiti informacije kako bi savladali samu igru), omogućuju prijenos učenja iz drugih stvarnosti i iskustvo koje uključuje angažiranje više osjetila (Baptista i Vaz de Carvalho, 2010.).

Van Eck (van Eck, 2006.) tvrdi da igre i igranje mogu biti učinkovito okruženje za učenje ne zato što su zabavne, već zato što su omogućuju igraču da se u potpunosti uključi, zahtijevaju od igrača da često donosi važne odluke, imaju pametno odabrane ciljeve, prilagođavaju se svakom igraču pojedinačno i uključuju društvenu mrežu.

Ako uzmemo u obzir model učenja temeljenog na igrama koji su radili Garris, Ahlers i Driskell (Garris, Ahlers i Driskell, 2002), glavna karakteristika obrazovne igre je da je nastavni sadržaj skriven unutar karakteristika igre. Učenici se igraju i zabavljaju, zaboravljajući na dio iskustva koji „uči“, iako im se neprestano predstavljaju novi koncepti koje moraju prilagoditi kako bi bili uspješni u igri. Motivacija se potiče dizajnom igara koji propagira ponavljanje ciklusa unutar konteksta igre. Kod igrača će se izazvati poželjna ponašanja temeljena na emocionalnim i kognitivnim reakcijama koje proizlaze iz interakcije s igranjem i od povratnih informacija koje dobiva tijekom igranja.

Gee (Gee, 2003.) tvrdi da se značajke videoigara s visokim potencijalom za učenje mogu podijeliti u dvije kategorije: značajke „ne-igrivosti“ koje se također mogu pojaviti u kontekstu koji nije igra, i značajke „igrivosti“ koje se više odnose na aktivnosti tijekom igranja igara. Unatoč toj razlici, ne treba pretpostaviti da bi značajke „ne-igrivosti“ funkcionirale i za učenje ako bi se odvojile od igre.

Značajke „ne-igrivosti“ s velikim potencijalom za učenje su:

- Empatija prema složenim sustavima – treba promatrati složeni sustav iznutra da bi se razumjelo kako njegove varijable međusobno djeluju.
- Simulacije iskustva i pripreme za akciju – U video igrama igrači vide virtualni svijet na osnovu toga kako omogućuje različite vrste akcija koje trebaju poduzeti da bi postigli cilj pobjede.
- Distribuirana inteligencija stvaranjem pametnog alata – Dobre video igre distribuiraju inteligenciju između osobe iz stvarnog svijeta i virtualnih likova s umjetnom inteligencijom kako bi predstavile makro i mikro prikaz situacije.
- Višefunkcionalni timski rad – Dobre video igre mogu naučiti suradnji i višefunkcionalnom timskom radu što je korisno za institucije poput škola i radnih mjesta. Igrači međusobno komuniciraju ne u smislu njihovih stvarnih karakteristika, već putem njihovih izmišljenih identiteta u igrama. Oni također mogu odlučiti koristiti svoju stvarnu rasu, klasu, kulturu i spol kao strateške resurse, ali nisu prisiljeni na unaprijed određene rasne, rodne, kulturne ili klasne kategorije.
- Situirano značenje – Dijalog i iskustvo su presudni da bi ljudi mogli povezati riječi sa stvarnim radnjama, funkcijama i rješavanjem problema. Budući da su video igre simulacija iskustva, one mogu smjestiti jezik u određeni kontekst.
- Otvorenost: spajanje osobnog i društvenog – U dobrim igrama otvorenog tipa igrači konstruiraju vlastite ciljeve koji se temelje na vlastitim željama, stilovima i porijeklu. Kombinacija osobnih ciljeva i ciljeva u igri je visoko motivirajuća.

Značajke ‘dobre igre’ koja se odnosi na učinkovito učenje su sljedeće:

- Motivacija – Video igre duboko motiviraju igrače i važno je razumjeti izvore ove motivacije ako se žele pružiti temelji za učenje.
- Uloga neuspjeha – Cijena neuspjeha je manje strašna i često se smatra načinom da se nauči temeljni obrazac. Ove značajke neuspjeha u igrama omogućuju igračima da preuzmu rizike koji mogu biti preskupi ili preopasni u stvarnome svijetu.
- Natjecanje i suradnja – Mnogi igrači, uključujući i one mlade, uživaju u natjecanju s drugim igračima u igrama, ali u školi možda ne vide natjecanje kao ugodno i motivirajuće. Konkurenciju u video igrama igrači vide kao društvenu, radi se ne samo o igranju, nego i o pobjedi i porazu.

Dizajn igara se odnosi na:

- Interaktivnost - igrač ne samo da pasivno prima znanje, već ima i kontrolu nad sadržajem.
- Prilagođavanje - temelji se na stilovima učenja i pruža višestruki put do uspjeha.
- Snažni identiteti - dobre igre igračima nude identitete koji potiču duboko uključivanje od strane igrača i koji su jasno povezani s funkcijama, vještinama i ciljevima koje osoba mora ostvariti u virtualnom svijetu-
- Dobro nanizani problemi - u konektivističkim pristupima učenju tvrdi se da je nizanje presudno za učinkovito učenje u složenim domenama.
- Ugodna razina frustracije - prilagođavanje izazova na takav način da većine igrača može igru doživjeti kao izazovnu, ali izvodljivu.
- Ciklus stručnosti - ponovljeni ciklusi omogućuju praksu i testiranje uspješnosti.
- 'Duboko' i 'pošteno' - igra mora biti izazovna, ali postavljena na način koji vodi do uspjeha.

Elementi igranja trebaju u početku biti jednostavni i laki za učenje i postajati složeniji što ih više igrač savlada i napreduje kroz igru.

Dokazano je da računalne igre koje se koriste u obrazovnom okruženju povećavaju motivaciju no visoko motivirani učenici nisu dovoljni za kvalitetno učenje. Potrebni su i dobri sadržaji za učenje, pa će učenici zapravo stjecati nova znanja iz sadržaja za učenje predstavljenih u obliku računalne igre.

Unatoč pozitivnom utjecaju igre na motivaciju, učitelji i dalje oklijevaju koristiti obrazovne igre u nastavi. Dodatan razlog je i taj što igre mogu biti previše vremenski zahtjevne za upotrebu u učionici. Stoga se obrazovne igre često koriste kao aktivnosti za učenje kod kuće ili kao motivacija u uvodnim lekcijama.

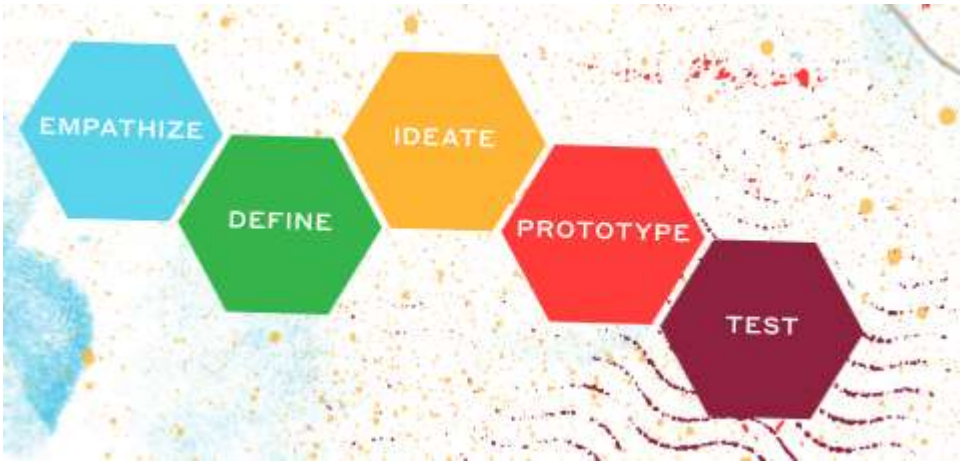
2.3. Metodologija *design thinking*

Ideje o *design thinking* pristupu pojavile su se krajem 1960-ih, ali kao koncept "*design thinking*" se pojavio na Sveučilištu Stanford krajem 1980-ih i početkom 1990-ih. U poslovnom svijetu metoda je postala široko rasprostranjena nakon što ju je promovirao Tim Brown iz IDEO koji je rekao: "*Design thinking* je pristup inovacijama usmjeren na čovjeka koji

koristi alate dizajnera kako bi integrirao potrebe ljudi, tehnološke mogućnosti i zahtjeve poslovnog uspjeha." (IDEO Design thinking, 2008.). Početkom 21. stoljeća *design thinking* pristup je postao izuzetno popularan i IT tvrtke su ga počele primjenjivati pri razvoju svojih proizvoda. Od 2005. na primjer, SAP primjenjuje *design thinking* kao filozofiju za rješavanje problema i kao inovativan pristup usmjeren na krajnjeg korisnika (Innovation Starter, 2015), a kao rezultat toga razvoj novih proizvoda dovodi do željenog rezultata. Tvrtke kao što su P&G, IBM Design, GOOGLE, AMAZON i Cisco integriraju *design thinking* u cjelokupno svoje poslovanje i organizaciju (Naiman, 2019.).

Koncept koji je razvilo Sveučilište Stanford identificira *design thinking* kao „vodeću metodologiju za kreativno rješavanje problema i stvaranje inovacija, a svakodnevno ga koriste tisuće tvrtki i organizacija širom svijeta. Cilj je razviti u adolescenata vještine 21. stoljeća - timski rad, rješavanje problema, kreativnost, empatiju, samopouzdanje, strpljenje, koncentraciju, eksperimentiranje." (Karakehayova, 2019).

Stanfordski model sastoji se od 5 faza (Slika 2.1.).



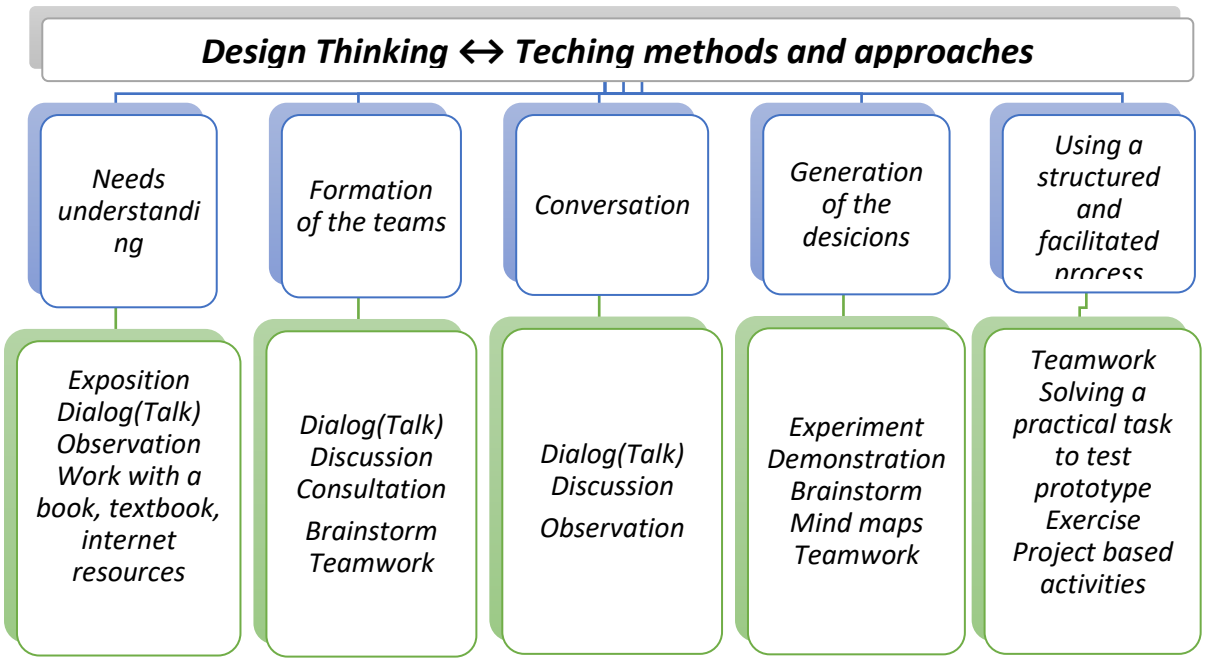
Slika 2.1. Faze Stanfordskog *design thinking* modela (Doorley, Holcomb, Klebahn, Segovia, & Utley, 2018)

Glavne faze i karakteristike *design thinking* mogu se sumirati na način prikazan u Tablici 2.1. (Georgieva & Tuparova, 2020).

Tablica 2.1. Faze i karakteristike design thinking modela. Prema: (Innovation Starter, 2015), (Naiman, 2019), (A project of the K-12 Initiative at Stanford University, 2009)

Prakse	**Načela**	**Faze**		
		Kratki postupak	**Prošireni postupak**	**Stanfordski model**
Razvijanje duboko empatičnog razumijevanja potreba korisnika	Otkriće - Upoznavanje novog izazova. Kako to riješiti?	Promatranje i proučavanje ponašanja i iskustva korisnika, istraživanje i definiranje problema i gledišta kroz empatiju		Empatija
Formiranje heterogenih timova	Tumačenje - što sam naučio i kako to protumačiti?	Generirati mnogo ideja u kratkom vremenskom razdoblju.		Definicija
Razgovori temeljeni na dijalogu	Ideje - vidim priliku, kako je oblikovati u ideju?	Izbor i klasifikacija ideja		Ideje
Donošenje odluka eksperimentiranjem	Eksperimentiranje - Imam ideju kako to izgraditi?	Izrada prototipa - proizvod koji se dobiva brza povratna informacija od korisnika.		Prototip
Korištenje strukturiranog postupka	Evolucija - probao sam nešto novo, kako to razviti?	Testiranje / povratne informacije - kako poboljšati prototip, ako je potrebno		Testiranje
			Testiranje ideje	
			Učenje i poboljšanja na temelju povratnih informacija	

Design thinking koristi se ne samo u poslovanju nego i nevladinom, obrazovnom i zdravstvenom sektoru. Sve se više obrazovnih institucija okreće *design thinking* pristupu. Postupno se povećavaju primjeri dobre prakse za primjenu *design thinking* u obrazovanju (Georgieva i Tuparova, 2020.). U provedbi *design thinking* pristupa učitelj mora uspostaviti vezu između pojedinih faza u modelu *design thinking* i odgovarajućih nastavnih metoda koje se mogu primijeniti. *Design thinking* zahtijeva primjenu širokog spektra nastavnih metoda.



Slika 2.2. Nastavne metode i *design thinking* (Georgieva & Tuparova, 2020)

2.4. Teorije učenja i učenje temeljeno na igrama

U početku je mnogo obrazovnih računalnih igara dizajnirano prema bihevioristočkoj teoriji učenja (Rugelj i Lapina, 2019) i provedeno u obliku programirane nastave. Učenicima je ponuđen poticaj u obliku pitanja ili bilo koje druge vrste zadatka ili problema koji se trebaju riješiti. Učenici odgovaraju odabirom jednog od ponuđenih odgovora. Ako je odgovor točan, igra pruža neku vrstu pozitivnog odgovora u obliku pozitivne reakcije lika ili vesele melodije koja potiče pozitivne emocije. Ovaj slučaj ponavljanja „radnja-reakcija“ nameće vezu između pitanja i točnog odgovora. U slučaju pogrešnog odgovora, pruža se reakcija u obliku negativnih podražaja. Igre i kvizovi oblika „pokaži i odaberi“ imaju ugrađen koncept vježbanja i uvježbavanja i tipični su predstavnici igara temeljenih na biheviorizmu. Prikladni su za učenje osnovnih aritmetičkih operacija ili za potporu pamćenju faktografskih podataka tj. za ostvarivanje ishoda učenja na najnižim razinama Bloomove taksonomije.

Teorija kognitivnog učenja naglašava kognitivnu aktivnost učenika i formiranje odgovarajućih mentalnih modela. Učenici uče temeljne pojmove od učitelja ili resursa za učenje, a zatim koriste logičko zaključivanje kako bi stekli nova znanja. Slagalice i strateške

igre kao okruženja za donošenje odluka primjeri su za kognitivistički pristup. Najnapredniji oblici igara temeljenih na kognitivnoj teoriji temelje se na inteligentnim sustavima podučavanja, gdje se algoritmi strojnog učenja koriste za modeliranje učeničkog i ekspertnog znanja kako bi se dobio personalizirani sadržaj za učenje.

Konstruktivizam kao teorija učenja predstavlja alternativni pogled koji sugerira da učenici sami konstruiraju svoje znanje, kao niz pojedinačno konstruiranih prikaza znanja koji su svi jednako valjani. Učenje je aktivan proces konstruiranja (umjesto stjecanja znanja), izgrađen rekurzivno na znanju koje učenik već ima. U procesu konstrukcije, senzorni podaci kombiniraju se s postojećim znanjem kako bi se stvorili novi mentalni modeli, koji zatim postaju osnova za daljnju izgradnju.

Konstruktivističko učenje naglašava otkrivanje i učenje navodeći kao argumenat da učenici trebaju biti smješteni u okruženje (koje se može modelirati računalnom igrom) u kojem sami grade svoje znanje. Tri temeljna načela definiraju konstruktivistički pogled na učenje:

- svaka osoba oblikuje vlastiti prikaz znanja,
- učenje se događa kada istraživanje učenika otkrije nesklad između njihove trenutne reprezentacije znanja i njihovog iskustva,
- učenje se odvija u socijalnom kontekstu, a interakcija između učenika i njihovih vršnjaka nužan je dio procesa učenja.

Materijali za učenje pružaju poduku koja se sastoji od podupiranja izgradnje znanja, a ne od deklariranja znanja na bihevioristički način. Simulacije računalnih igara preslikavaju različite scenarije iz stvarnog života u formatu računalnih igara. One predstavljaju model apstrahirane stvarnosti u kojem učenik ima određenu ulogu. Zadaća učitelja je pružiti smjernice i povratne informacije kada učenik uči, tj. konstruirati održive mentalne modele.

Igre mogu dovesti do promjena u stavovima, ponašanju i vještinama igrača. Shute i Ke (Shute & Ke, 2012) otkrili su da postoji konvergencija između temeljnih elemenata dobre igre i karakteristika produktivnog učenja. Dizajn igara može nas puno toga naučiti o učenju, a suvremena teorija učenja može nas naučiti o dizajniranju boljih igara (Shute, Rieber i Van Eck, 2012). Marshall McLuhan, kanadski filozof teorije komunikacije, koji je predviđao World Wide Web šezdesetih godina prošlog stoljeća kada je govorio o globalnom selu, izjavio je:

„Svatko tko pravi razliku između igara i učenja ne zna ništa ni o jednom ni o drugom. " (Shute, Rieber i Van Eck, 2012).

Whitton i Moseley (Whitton i Moseley, 2012.) predložili su okvir za dobru praksu u dizajniranju obrazovnih igara iz perspektive aktivnog učenja. Prema smjernicama okvira, okruženje za igru trebalo bi podržati aktivno učenje potičući istraživanje, rješavanje problema i propitivanje, izazvati angažman s eksplicitnim i ostvarivim ciljevima, biti primjereno kontekstu učenja, podržavati i pružati mogućnosti za razmišljanje, pružiti jednake mogućnosti svim učenicima, pružiti stalnu podršku postupnim uvođenjem sve veće složenosti i biti potpomognuti nekim oblikom pomoći ili savjetima.

Nekoliko je studija pokazalo da obrazovne igre mogu podržati učenje motivacijom, angažmanom i zabavom (Hijon-Neira, Velazquez-Iturbide, Pizarro-Romero i Carrico, 2015). Učenje programiranja zahtijeva mnoge kompetencije kao što su logičko razmišljanje, rješavanje problema i sposobnost razumijevanja apstraktnih pojmova. Iz tog razloga, mnogim učenicima je teško naučiti računalno programiranje. Ova činjenica može dovesti do slabe motivacije za izučavanje uvodnih tečajeva programiranja. Kako bi poboljšali motivaciju i poboljšali odnos učenika prema učenju programiranja, učitelji traže poticajne pristupe učenju.

Programiranje se najbolje uči kroz praksu i, ukoliko učenici žele učinkovito učiti, barem će dio ove prakse morati biti usmjeren prema samostalnom učenju, kao i u prema suradnji s vršnjacima. Učiteljeva je ključna uloga nagovoriti učenike na to i time ih motivirati (Feldgen & Clua, 2004).

U uvodnom tečaju računalnog programiranja učenici dizajniraju i razvijaju programe što predstavlja tipičan pristup aktivnom učenju (Rugelj & Lapina, 2019). Ako uvedemo projektni rad u skupinama koji se pokazao vrlo učinkovitim načinom učenja programiranja (Nančovska Šerbec, Kaučič i Rugelj, 2008), zapravo možemo govoriti o trostrukom ili „trijaloškom“ principu učenja (eng. *trialogical learning principle*). Ovaj plan učenja sastoji se od oblika učenja u kojima učenici u sustavnom procesu suradnički razvijaju, mijenjaju ili kreiraju zajednički artefakt - računalni program (Kafai, 1995). Fokus je na interakciji koja se događa stvaranjem konkretnih artefakata, a ne samo između ljudi ("dijaloški pristup") ili unutar nečijeg uma ("monološki" pristup) (Paavola i Hakkarainen, 2005.).

3. DJEVOJKE, IKT I OBRAZOVNE IGRE

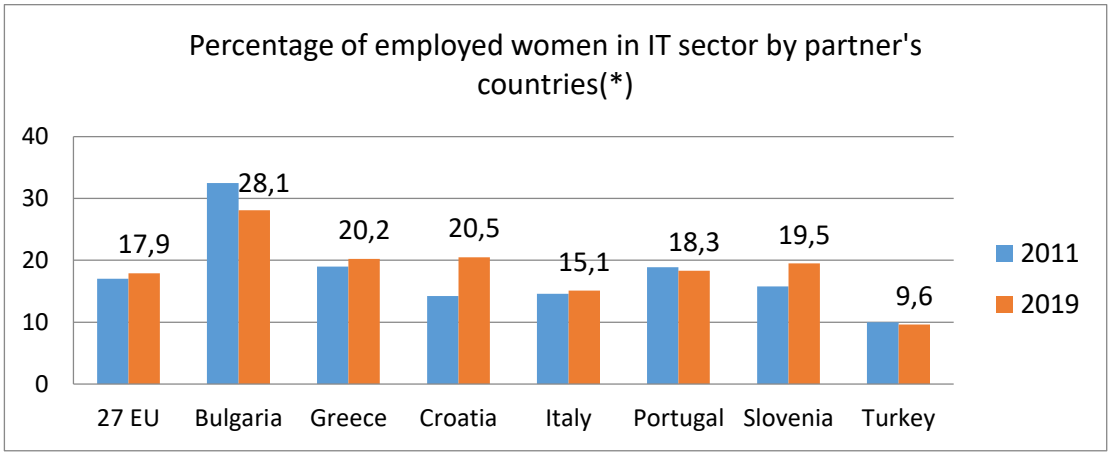
3.1. Položaj žena u IKT sektoru – pregled stanja

Digitalno društvo postavlja sve veće zahtjeve prema kompetencijama suvremenih ljudi. Sve je veća potreba za ljudima koji nisu samo digitalno pismeni, već imaju i vještine za stvaranje i održavanje softverskih aplikacija u raznim poljima ljudskog djelovanja. Mediji neprestano komentiraju nedostatak kvalificiranih stručnjaka u sektoru informacijskih i komunikacijskih tehnologija (IKT) koji pokazuje kontinuiran je rast.

U 2019. oko 7,8 milijuna ljudi radilo je kao IKT stručnjaci širom Europske unije (EU). (Eurostat, 2020) no to predstavlja samo 3,9% svih zaposlenih u EU.

Prema Eurostatu u EU-27 u 2011. godini udio žena u IT sektoru bilo je 17%, a u 2019. oko 17,9%. Prema podacima Eurostata za razdoblje 2007.-2019. u prosjeku za 27 država članica EU, udio žena u IT sektoru smanjio se s 22,5% na 17,9%. U 2019. godini žene čine 17,3% od ukupnog broja osoba zaposlenih u EU s IKT obrazovanjem, u odnosu na 20,2% u 2009. (Eurostat, 2020).

Trenutno stanje zaposlenosti žena u IKT sektoru u zemljama partnerima projekta Coding4girls prikazano je na slici 3.1.



Slika 3.1. Zaposlenost žena u IKT sektoru (u postocima) u 2011. i 2019.

3.2. Stavovi djevojaka prema obrazovnim igrama

Rezultati studije koju su proveli Vermeulen i dr. (Vermeulen, Van Looy, Courtois i De Grove, 2011.) pokazali su da se uočavaju rodne razlike u igranju igara, ali i da prethodna iskustva bitno utječu na rezultate (Rugelj i Lapina, 2019.). Žene igraju igre češće, ali kraće vrijeme od muškaraca. Preferiraju igre koje su apstraktne, kratke i jednostavne za svladavanje poput igara za opuštanje (npr. Tetris) i igara na društvenim mrežama. Muškarci će vjerojatnije igrati „osnovne“ žanrove što se odnosi na igre temeljene na vještinama, koje oduzimaju puno vremena i obično sadrže visokokvalitetnu 3D grafiku, poput pucačina, borbi, akcijskih avantura, sporta, utrka, strategija, igranja uloga i MMO igara. Ostali žanrovi uključuju platformu, avanturu, simulaciju, zabavu, obrazovne, klasične i igre za opuštanje. Studija je otkrila da žene vole slagalice ili puzzle, dok utrke, ritamičke igre, simulacije i virtualne igre igraju i žene i muškarci. Drugo istraživanje otkrilo je da žene favoriziraju zabavne igre (poput glazbe i plesnih igara) i klasične ili „retro“ igre.

Alserri i suradnici (Alserri, Zin i Wook, 2018.) napravili su opsežno istraživanje radova o učinkovitim elementima obrazovne igre, kao što su motivacijski elementi igrača digitalnih igara, učinkoviti obrazovni elementi igre i elementi preferencija od strane ženskih igrača. Rezultate su koristili za stvaranje konceptualnog modela za angažman prema spolu u obrazovnim igrama. Ovaj model koristio se i u projektu Coding4Girls za povećanje motivacije djevojaka za programiranje kroz odgovarajući odabir igara koje će učenici osmisliti i implementirati prilikom učenja temeljenog na dizajnu igara.

Autori su naveli da je studija otkrila kako motivacija za igranjem određene vrste digitalne igre ovisi o spolu. Stereotip da žene ne igraju računalne igre više ne vrijedi. Razlike u preferencijama za digitalne igre prema spolu utvrđene u ovom istraživanju vrlo su slične već predstavljenim razlikama koje su utvrdili Vermeulen i sur. (Vermeulen, Van Looy, Courtois i De Grove, 2011). Žene više vole istraživački i kreativni način igre nego muškarci. Igraju češće, ali kraće vrijeme od muškaraca, a sklonosti žanrovima igara također su različite. Utvrđeno je da muškarci provode više vremena igrajući računalne igre nego žene. Žene također preferiraju puzzle, društvene igre s nagradama koje se nude u igrama, obrazovne igre, žanrove simulacijskih igara, kao i suradničko i istraživačko igranje, virtualni život i svjetove te zabavne igre. Dodatno, žene vole igrati avanturističke igre, ali radije u početku promatraju

druge, prije nego što same započnu igranje. Motivacijski elementi u igrama koji žene preferiraju su izazov, relaksacija, zabava, socijalna interakcija, motivacija, fantazija, natjecanje i uzbuđenje. I muškarci i žene vole trke, simulacije i virtualne igre.

Phan i sur. (Phan, Jardina, Hoyle i Chaparro, 2012.) došli su do vrlo sličnih zaključaka u svojoj studiji o rodnim razlikama između muških i ženskih igrača u smislu upotrebe računalnih igara, preferencija i ponašanja.

Prema studiji Tuparove i suradnika (Tuparova, Tuparov i Veleva, 2019. b), postoje razlike u sklonostima dječaka i djevojčica prema značajkama i elementima igara kao što su: grafički dizajn, zvuk, mogućnost igranja igre na različitim uređajima (računalu, tabletu, pametnom telefonu itd.), mogućnost uključivanja više igrača, mogućnost igranja *online*, mogućnost prelaska na složenije razine igre. Najpoželjnija značajka igara za djevojčice je *mogućnost prelaska na složenije razine igara*. Za dječake je najpoželjnija osobina *logika i zaplet/scenarij igre*. Najmanje zanimljiva značajka za djevojčice je zvuk u igrama, a za dječake nagrade u igri. Što se tiče korištenog uređaja, pametni telefon je najčešće korišten uređaj za igranje kod dječaka i djevojčica. Za djevojčice je drugi uređaj u uporabi prijenosno računalo, a zatim tablet. Za dječake su to stolno računalo i prijenosno računalo. Autori su primijetili da dječaci više od djevojčica vole igrice s pričama, a djevojčice više od dječaka igre pamćenja (eng. *memory*). Ne ovise o spolu preferencije za sljedeće vrste igara: igre s društvenim elementima, igre koje zahtijevaju pažnju/koncentraciju ili brzu reakciju, igre uloga, zagonetke, avanturističke igre. Igra s najmanje interesa za djevojčice i dječake je slagalica.

Uzimajući u obzir specifičnu situaciju u projektu Coding4Girls u kojem se koristi učenje temeljeno na dizajnu igara, cilj je bio pripremiti zadatke koji su zanimljivi i motivirajući za djevojke. Programsko okruženje koje predstavlja programiranje kao sredstvo za stvaranje animiranih filmova (tj. pripovijedanje priča) ili jednostavnih igara može biti prikladno za djevojke jer većina njih može smisliti ideju za priču ili jednostavnu igru koju bi htjele stvoriti (Wolz, Barnes , & Bayliss, 2009). I priča i jednostavna igra su po svojoj prirodi sekvencijalne i malo je vjerojatno da će odmah trebati savladati napredne programske koncepte za njihovu izradu. Dodatno, predstavljaju oblik samoizražavanja i pružaju djevojkama priliku za eksperimentiranje s različitim ulogama. I oni njihovi prijatelji koji ne programiraju mogu lako razumjeti i cijeniti animiranu priču ili igru te im pružiti poticaj kroz pozitivne povratne

informacije. Kelleher i suradnici (Kelleher, Pausch i Kiesler, 2007) su upotrijebili programske okoline *Storytelling Alice* i *Generic Alice* za oba slučaja. Tvrde da je nekoliko studija djece programera otkrilo da ako su djevojke i dječaci sličnog iskustva s računalnim programiranjem, onda su i jednako zainteresirani i učinkoviti u učenju programiranja. No uspješnost u programiranju povezana je s vremenom koje su korisnici utrošili programirajući i njihovim prethodnim iskustvima u programiranju.

4. POUČAVANJE PROGRAMIRANJA KROZ IGRU

Ovo poglavlje će se usredotočiti na predstavljanje i usporedbu različitih pristupa i platformi/okruženja igara koji se mogu koristiti za podučavanje djece vještinama programiranja.

4.1. Pristupi poučavanju programiranja kroz igre

Trenutno postoji čitav niz pristupa za poučavanju početnika u programiranja.

Pristupi se mogu klasificirati kao:

- Učenje programiranja dizajniranjem igara, poznato i kao učenje temeljeno na dizajniranju igara (eng. *game-design based learning*). Predstavlja upotrebu programskih jezika za učenje programiranja dizajniranjem i stvaranjem igara.
- Učenje programiranja u okruženju temeljenom na igrama.
- Učenje programiranja dizajniranjem igara u programskom okruženju temeljenom na igrama (metodologija Coding4Girls pripada ovoj kategoriji).

Jedan od primjera je uporaba vizualnih programskih jezika zasnovanih na blokovima koji su posebno prilagođeni korisniku, jednostavni za rukovanje (povlačenjem i ispuštanjem), sprječavaju pogreške u sintaksi i logici (Ouahbi, Kaddari, Darhmaoui, Elachqar i Lahmine, 2015.) i prezentiraju sadržaj na vrlo intuitivan način. Scratch (Resnick i sur., 2009.), https://scratch.mit.edu/, Snap! (Weintrop & Wilensky, 2015) https://snap.berkeley.edu/, ili Alice 2/3 https://www.alice.org/about/ poznati su i uspješni primjeri vizualnih programskih jezika. Tablica 4.1 uspoređuje karakteristike najpoznatijih jezika i okruženja za vizualno programiranje.

Tablica 4.1. Karakteristike nekih vizualnih programskih okruženja temeljenih na blokovima

	PLATFORMA	CILJNA GRUPA	PROGRAMSKI JEZIK (JEZICI)	BESPLATNO/PLAĆA SE
Scratch	*Online* (dostupna je i *offline* verzija)	8-16	Vizualni programski jezik	Besplatan
Snap!	*Online* (dostupna je i *offline* verzija)	12-20	Vizualni programski jezik	Besplatan
Alice	Windows, Mac OS X, Linux	12-20	Vizualni programski jezik	Besplatan
Tynker	*Online*, iOS	7+	Vizualni programski jezik,	Plaća se

			JavaScript, Python	

Računalne igre također se mogu koristiti za učenje programiranja, bilo poticanjem učenika na razvoj i stvaranje vlastitih video igara (učenje dizajniranjem igara) ili omogućavanjem igranja obrazovnih igara čiji ishodi učenja uključuju ishode učenja povezane s programiranjem (učenje temeljeno na igrama). Računalne igre namijenjene obrazovnim aktivnostima mogu stvoriti motivirajuće i zabavno okruženje za učenje jer sadrže aktivnosti koje udovoljavaju obrazovnim standardima, ciljevima učenja, pružaju povratne informacije i mogu postići visoke obrazovne ishode (Georgieva i Tuparova, 2019).

Ideja korištenja igara za podučavanje programiranja dolazi iz činjenice da one angažiraju i motiviraju učenike te im omogućuju da nauče računalno razmišljanje i vještine programiranja u zabavnom i poznatom okruženju, prije nego što te vještine prenesu na učenje programskog jezika (Kazimoglu, Kiernan, Bacon i Mackinnon, 2012.). Ipak prema Kazimogluu i sur. (Kazimoglu, Kiernan, Bacon i Mackinnon, 2012.), obrazovne igre za učenje programiranja i računalnog razmišljanja ne bi trebale biti stvorene samo za zabavu, već bi trebale biti usklađene s kurikulumom, praveći razliku između različitih programskih konstrukcija i potičući dobre prakse u programiranju (npr. putem gamifikacije).

Obrazovne igre također mogu biti korisne za programiranje jer pretvaraju neugodne aktivnosti u zanimljiva iskustva. Shabanah i suradnici (Shabanah, Chen, Wechsler, Carr i Wegman, 2010) stvorili su alat *Algorithm Game Designer* čiji je cilj olakšati stvaranje igara s algoritmima putem sustava za vizualizaciju algoritama. Grivokostopoulou i suradnici (Grivokostopoulou, Perikos i Hatzilygeroudis, 2016.) razvili su igru za podučavanje algoritama umjetne inteligencije, temeljenu na igri *Pacman*, tako da se učenicima kroz vizualizacije algoritama i animacije može pomoći u procesu učenja, demonstrirajući način na koji algoritam djeluje, njegove funkcije te kako donijeti ispravne odluke na temelju određenih parametara.

Razvojem bržih internetskih veza i sve većom dostupnosti računala, započelo je korištenje platformi zasnovanih na mrežnim igrama poput CodeCombat (https://codecombat.com/) i LightBot (https://lightbot.com/). Combéfis i sur. (Combéfis, Beresnevičius i Dagiene, 2016.) pregledali su glavne vrste *online* platformi koje se koriste za poučavanje vještina

programiranja i zaključili su da sljedeći čimbenici mogu obrazovnu igru učiniti više motivirajućom i uspješnijom: 1) povratne informacije i procjena, 2) estetika, 3) aspekti suradnje i više igrača, 4) vođenje kroz igru, 5) nema negativnih posljedica, 6) glazba, 7) izazov srednje razine da se zadrži interes igrača.

Što se tiče ishoda učenja obrazovnih igara za učenje programiranje, prema opsežnom pregledu 49 obrazovnih programskih igara Miljanovića i suradnika (Miljanović i Bradbury, 2018), većina obrazovnih igara za programiranje usredotočeno je na rješavanje problema i temeljne koncepte programiranja, nekoliko igara usredotočeno je na strukture podataka, razvojne metode i dizajn softvera.

Postoje i igre koje su, iako možda ne poučavaju vještine programiranja izravno, sposobne poučavati na neizravan, ali učinkovit način ključne vještine računalnog razmišljanja, poput apstrakcije (npr. Tetris), dekompozicije (npr. Sudoku), algoritamskog razmišljanja (npr. puzzle), evaluacije (npr. igre pamćenja) i generalizacije (npr. igre s uzorcima) (Franković, Hoić-Božić i Načinović-Prskalo, 2018).

U učenju temeljenom na dizajnu igara često imamo kombinaciju upotrebe vizualnih programskih jezika i njihovih okruženja s blokovima s procesom dizajniranja i kodiranja igara. U studiji koja je uspoređivala korištenje okruženja za kodiranje Scratcha i Pascala (proceduralni jezik koji koristi editor u kojem se upisuju naredbe) od strane programera početnika, utvrđeno je da su učenici mnogo motiviraniji za nastavak studija računalnih znanosti ukoliko koriste Scratch (Ouahbi, Kaddari, Darhmaoui, Elachqar i Lahmine, 2015). Studija je također otkrila da je stvaranje igara i priča učenike učinilo kreativnijima i samostalnijima tijekom učenja programiranja. Još jedna studija koju su proveli Weintrop i suradnici (Weintrop i Wilensky, 2015.), uspoređujući upotrebu Snap! i Java programskog jezika od strane srednjoškolaca, ustvrdila je kako je pristup temeljen na blokovima bio lakši za učenje od korištenja Jave, što je u skladu sa stajalištem da je programiranje zasnovano na blokovima (poput Scratch i Snap!) pristupačnije programerima početnicima. Prema nalazima studije, to je bilo zbog činjenice da se blokovi lakše čitaju, daju vizualne upute kako se mogu koristiti te ih je lakše sastavljati.

Integrirani pristup poučavanju programiranja i njegov utjecaj na postignuća učenika starih 14 godina raspravlja se u (Tuparova, Nikolova i Tuparova, 2020). Ovaj pristup kombinira dvije faze:

1. Korištenje postojećih obrazovnih računalnih igara: za stjecanje znanja i vještina, motivaciju za aktivnosti učenja, razvoj algoritamskog razmišljanja, za eksperimentiranje s postojećim obrazovnim računalnim igrama radi istraživanja pravila igre, elemenata sučelja i njihovih svojstava i događaja
2. Dizajn i razvoj obrazovnih računalnih igara, uključujući matematički model igre, dizajn grafičkog korisničkog sučelja (GUI), kodiranje, testiranje i provjeravanje.

Za provedbu pristupa koristilo se programsko okruženje C#. Model je testiran na 126 učenika koji su pohađali informatički smjer, podijeljenih u dvije skupine - eksperimentalnu i kontrolnu skupinu. Postignuća učenika mjerila su se kroz praktični zadatak i test na papiru. Rezultati su pokazali da ne postoji razlika između postignuća dječaka i djevojčica u eksperimentalnoj skupini no autori su ustvrdili da su djevojke u eksperimentalnoj skupini postigle bolje rezultate od djevojčica u kontrolnoj skupini.

4.2. Okruženja temeljena na igrama za poučavanje programiranja

U nastavku će biti predstavljeno nekoliko primjera upotrebe različitih okruženja pomoću kojih se može uključiti igre u podučavanje kodiranja i programiranja.

4.2.1. Učenje dizajniranjem igara

Scratch

Slika 4.1. Primjer programskog kôda u Scratch-u

Scratch (https://scratch.mit.edu/) je programsko okruženje temeljeno na blokovima, odnosno omogućava učenicima da sastavljaju programe spajanjem blokova kôda te da

primaju vizualne povratne informacije (Weintrop & Wilensky, 2015). Na taj se način učenici upoznaju s osnovama programiranja, a ne moraju voditi računa o programskoj sintaksi (Ouahbi, Kaddari, Darhmaoui, Elachqar i Lahmine, 2015).

Scratch se može koristiti kao uvod u programiranje, posebno među mlađim učenicima, ali i kao način za olakšavanje prijelaza u druga programska okruženja. Time se postiže i poticanje interesa za informatiku (Wolz, Maloney i Pulimood, 2008). Prema istraživanju provedenom od strane Meerbaum-Salant i suradnika (2013), većina učenika može postići dobru razinu usvojenosti koncepata računalnih znanosti putem Scratcha. Međutim, poteškoće nastaju pri poučavanju određenih tema koje zahtijevaju veću razinu apstrakcije no to se može riješiti ako ih u procesu prati učitelj.

Web sjedište Scratcha podržava svjetsku mrežu korisnika, a programerima omogućava dijeljenje svojih projekata (Meerbaum-Salant, Armoni i Ben-Ari, 2013). Scratch je dostupan na više od 50 jezika, uključujući bugarski, hrvatski, engleski, grčki, talijanski, portugalski, slovenski i turski. Također ima *offline* editor koji se može preuzeti na računalo i koristiti bez internetske veze.

Snap!

Snap! (https://snap.berkeley.edu/) je također vizualni programski jezik zasnovan na blokovima čiji se dizajn temelji na Scratchu, ali dodaje neke dodatne značajke. Snap! je, slično Scratchu, zasnovan na webu, ali također ima mogućnost pokretanja izvan mreže putem preglednika. Povrh značajki Scratcha, Snap! dodaje klase za liste, procedure, sprajtove, kostime, zvuk i sl. što ga čini prikladnijim za naprednije korisnike u odnosu na Scratch. Snap! dostupan je na preko 40 jezika, uključujući bugarski, hrvatski, engleski, grčki,

talijanski, portugalski, slovenski i turski.

Slika 4.2. Sučelje alata Snap!

Alice

Alice https://www.alice.org/ je programsko okruženje temeljeno na blokovima koje ima za cilj motivirati učenike na programiranje potičući njihovu kreativnost stvaranjem animacija, interaktivnih priča ili jednostavnih 3D igara (Kelleher, Pausch i Kiesler, 2007). Omogućuje stvaranje virtualnih svjetova pomoću programa *Virtual World Editor* gdje učenici mogu dodavati 3D objekte i dodavati funkcije i metode postojećim 3D objektima. Jednom kada se stvori virtualni svijet, učenik može pisati kôd za razvijanje logike igre/priče /animacije (Sykes, 2007). Alice je dobar programski jezik za učenike bez prethodnog iskustva jer im omogućuje da vide kako se animirani programi pokreću dok pišu svoj kôd (Cooper, Dann i Pausch, 2000).

Slika 4.3. Primjer programskog kôda u alatu Alice 3

Štoviše, drugo istraživanje koje se usredotočilo na Storytelling Alice (programsko okruženje temeljeno na Alice 2 s dodatnim kreativnim alatima za pisanje priča) pokazalo je da, iako su i Generic Alice i Storytelling Alice bile podjednako uspješne u podučavanju programskih koncepata za djevojčice, sa Storytelling Alice djevojke su bile motiviranije i više su vremena posvetile programiranju.

Alice 3, najnoviji dio serije, dostupan je na 13 jezika, uključujući engleski, portugalski, grčki, slovenski i bugarski, engleski, španjolski, portugalski i njemački.

Tynker

Tynker https://www.tynker.com/ je obrazovna programska platforma koja se temelji na „povuci i ispusti“ (eng. *drag-and-drop*) vizualnom programskom jeziku. Za razliku od ostalih

predstavljenih programskih okruženja, Tynker je komercijalni proizvod. Jedan aspekt koji dodaje Scratchu sastoji se u tome da se programiranju pristupa kao igri u kojoj korisnici moraju stvoriti program pomoću kojeg će se likovi pomicati, komunicirati i izvoditi različite zadatke. Igračima se predstavljaju problemi koje trebaju riješiti te na taj način djeca počinju prepoznavati obrasce u programiranju (Geist, 2016.). Tynker također nudi ugrađenog virtualnog tutora koji daje detaljne upute, tako da učenik može naučiti kako primijeniti koncepte kodiranja. Učenik također može vizualizirati blokove u JavaScript kodu, omogućavajući im da razumiju blokove i u programskom jeziku temeljenom na tekstu.

Tynker nudi preko 23 tečajeva programiranja, 11 tečajeva za iPad i više od 2000 aktivnosti kodiranja te daje mogućnost odabira individualnih planova i obiteljskih planove (za do 4 člana). Dostupan je samo na engleskom jeziku.

Slika 4.4. Sučelje alata Tynker

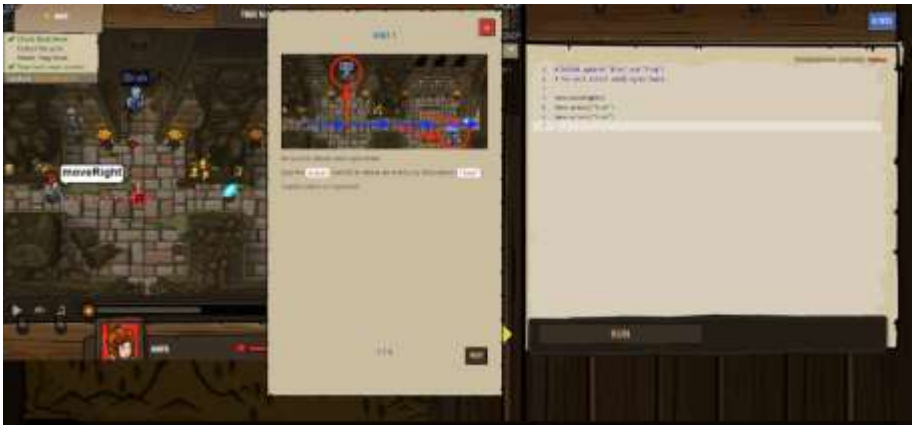
4.2.1. Učenje programiranja u okruženjima temeljenim na igri

Ovo poglavlje predstavlja platforme za igranje koje su namijenjene za razvoj vještina programiranja.

Code Combat

Code Combat https://codecombat.com/ je web igra koja propada žanru avanture. Mogu je koristiti pojedini učenici i cijeli razredi, a nudi i resurse za učitelje tj. za korištenje tijekom nastave. Ima više od 100 razina za igranje koji su dostupni svima te dodatne razine za

pretplatnike. Dostupna je na više od 50 jezika, uključujući bugarski, hrvatski, engleski, grčki, talijanski, portugalski, slovenski i turski.



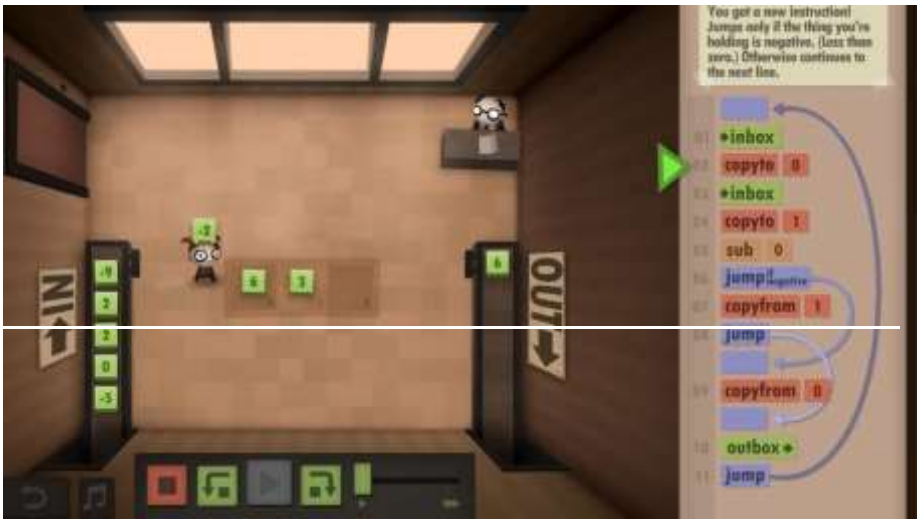
Slika 4.5. Sučelje alata Code Combat

Učenici moraju napisati vlastiti kôd (bilo u JavaScriptu ili Pythonu) kako bi se likovi pomicali, komunicirali i ostvarivali različite zadatke. Nije temeljena na blokovima te zahtijeva od učenika da napišu vlastiti kôd. Pruža pomoć tijekom izrade i postupno uvodi koncepte. Igra je podijeljena u 5 različitih svjetova, uvodeći različite koncepte kako igrač prolazi kroz igru: 1) *Kithgard Dungeon* - sintaksa, metode, parametri, nizovi, petlje i varijable; 2) *Backwoods Forest – if/else*, Boolova logika, relacijski operatori, funkcije, svojstva objekta, upravljanje događajima, upravljanje ulazima; 3) *Sarvensk Desert* - aritmetika, brojači, *while*-petlje, *break, continue, arrays*, usporedba nizova, pronalaženje min/max; 4) *Cloudrip Mountain* - objektni literali, *remote* metoda, invokacija, for-petlje, složene funkcije, crtanje, moduli; 5) *Kelvintaph Glacier* - napredne tehnike.

Prema Miljanoviću i suradnicima (Miljanovic & Bradbury, A Review of Serious Games for Programming, 2018.) obrazovni sadržaj igre uključuje različite konceptualne aspekte dizajna algoritama i rješavanja problema, temeljne programske koncepte (sintaksa i semantika, varijable i primitivni tipovi podataka, izrazi i pridruživanja, ulaz i izlaz, uvjeti i ponavljanja, funkcije i parametri, rekurzija) i temeljne strukture podataka (nizovi i apstraktni tipovi podataka).

Human Resource Machine

Human Resource Machine http://tomorrowcorporation.com/humanresourcemachine je *puzzle* igra zasnovana na vizualnom programskom jeziku. U ovoj igri igrač mora automatizirati zadatak programiranjem radnika u uredu, napredujući kroz sve teže zagonetke (Johnson i dr., 2016). Igrač mora stvoriti slijed poteza povlačenjem i ispuštanjem naredbi, s postupnim uvođenjem novih naredbi za izvršavanje složenijih operacija. Pomoću ovog vizualnog programskog jezika poučavaju se temeljni koncepti programiranja putem



usporedbe algoritama.

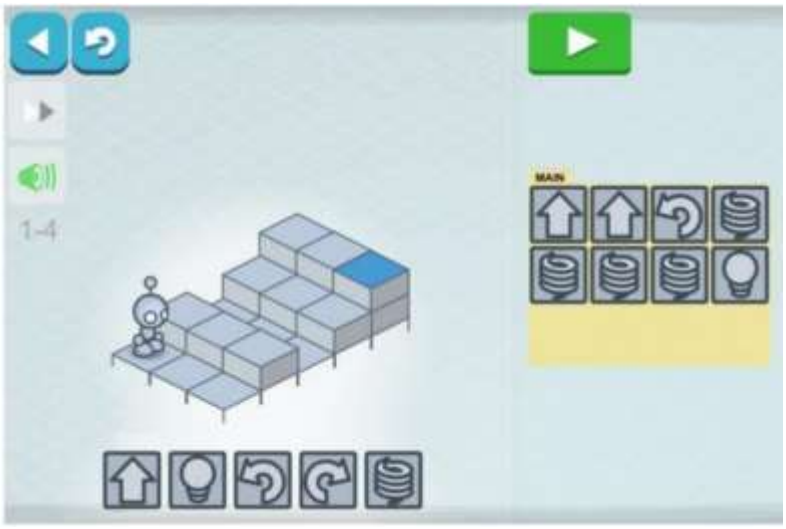
Slika 4.6. Sučelje alata Human Resource Machine

LightBot

LightBot https://lightbot.com/ je *puzzle* igra sa sličnom mehanikom kao Human Resource Machine, koja zahtijeva logičko razmišljanje za napredovanje kroz razine. Igrač mora voditi robota da osvijetli pločice i riješi 40 razina. Naredbe korištene u Lightbotu pojavljuju se kao ikone.

Iako ne postoji eksplicitno učenje programskog jezika, igrači „... razvijaju razumijevanje redoslijeda i implementacije algoritama“ (Miljanović i Bradbury, 2018.) usredotočujući se vještine rješavanja problema. Ipak, kroz igru učenici uče različite temeljne koncepte programiranja, poput uvjeta, ponavljanja, rekurzije i strategija otklanjanja pogrešaka. Kako je prostor za pločice ograničen, učenik mora razmotriti i učinkovitost koda.

U studiji koju su proveli Mathrani i suradnici (Mathrani, Christian i Ponder-Sutton, 2016.) koristio se Lightbot s učenicima srednjoškolskog obrazovanja te se otkrilo da su učenici uživali u igranju igara i da je ovo učinkovit način učenja programskih konstrukcija poput funkcija, procedura, uvjeta i rekurzija. Većina uključenih učenika također se složila kako je ovakav pristup učinkovit način da razumiju koncepte koje bi inače bilo teže shvatiti.



Slika 4.7. Sučelje igre LightBot

May's Journey

May's Journey jedina je igra s ovog popisa koja je izravno usmjerena na djevojke iz srednje škole. Riječ je o 3D puzzle igri u kojoj igrači rješavaju labirint kroz programski jezik

Slika 4.8. Sučelje alata May's Journey

igre, koji je inspiriran Javom.

Igra je osmišljena kako bi privukla djevojke za uključivanje u računalne znanosti poučavanjem osnova programiranja. Programer koristi pseudo-kôd kako bi olakšao prijelaz između vizualnog programiranja i stvarnih programskih jezika. Nastavni sadržaj uključuje osnovne naredbe i logiku niza naredbi, petlje, varijable, if naredbu, komparatore i Boolovu logiku, operacije nad cijelim brojevima i operacije nad nizovima (Jemmali, 2016).

U igri postoje dvije faze, faza istraživanja s mehanikom tipične igre i faza kodiranja. U fazi kodiranja sučelje je podijeljeno tako da igrač može upisati kôd, a uz to dobiva i vizualnu povratnu informaciju o izvođenju programa. Savjeti za kôd također se pružaju tijekom faze istraživanja. U priči je junak djevojka koja živi u svijetu koji se raspada i odvojena je od svog prijatelja te mora popraviti svijet igre i riješiti misterije. Misterij se otkriva postupno, dio po dio, motivirajući igrače da dalje igraju. Kako su pred-tinejdžerice glavna ciljna skupina za igru, glavni lik izgleda poput djevojke iz srednje škole, tako da se igračica može poistovjetiti s njom. Prilikom dizajna također se imalo na umu preferencije djevojaka (pojačana svjetlost, topla shema boja i manje agresivnih ilustracija). Također se pokazalo da je testna skupina bila zadovoljna s pričom igre.

No Bug's Snack Bar

No Bug's Snack Bar je obrazovna istraživačka igra koja se fokusira na rješavanje problema i koristi pristup temeljen na povlačenju i ispuštanju. Ishodi učenja se odnose na upravljanje varijablama, niz akcija, uvjete i iteracije te strategije otklanjanja pogrešaka (Vahldick, 2017.). U igri glavni lik radi u zalogajnici i mora razriješiti različite misije pomoću programiranja. Inovacija integrirana u ovoj igri je alat za učitelje koji omogućuje da prate napredak učenika. Postoji i pristup gamifikacije, pri čemu učenik osvaja bodove tijekom napredovanja kroz igre i pronalaženja načina da poboljša svoja rješenja za predstavljene probleme. Igra uključuje pomoćnika sa savjetima i administrativni sustav za učitelja koji mu omogućuje da vidi učenike kojima treba pomoć s ciljem da im pošalje personalizirane savjete. Prisutnost učitelja je prema tome, neophodna tijekom igre.

Slika 4.9. Sučelje alata No Bug's Snack Bar

Robot ON!

Robot ON! je još jedna istraživačka igra tipa puzzle namijenjena studentima preddiplomskog studija koji uče C ++. U ovoj igri igrač je znanstvenik koji mora aktivirati *Mech Suit* kroz niz zadataka. Nakon što igrač završi razinu, aktivira se novi sustav robota. Učitelji mogu stvoriti vlastitu razinu koristeći druge programske jezike. Igrač ima na raspolaganju različite alate koji su označeni bojom, boja na programskom kôdu odgovara alatima. Igrač upoznaje jedan po jedan alat, uz pomoć odgovarajućih tutorijala (Miljanović i Bradbury, 2016).

Slika 4.10. Igra Robot ON!

Umjesto da se učenik usredotoči na pisanje kôda, ova mu igra omogućuje da se posveti razumijevanju programiranja s ciljem da nauči vještine ispravljanja pogreški i razumije kôd koji su napisali drugi. Kako napreduje kroz igru, igrač dobiva sljedeće alate: 1) *activator* (za učenje programskog tijeka), 2) *commenter/un-commenter* (za učenje ponašanja koda), 3) *namer* (za učenje upotrebe varijabli) i 4) *checker* (za učenje protoka podataka).

Educational Pacman Game

Educational Pacman Game istraživačka je obrazovna igra namijenjena poučavanju algoritama pretraživanja, omogućujući studentima da putem grafičkog prikaza vide kako se ponašaju različiti algoritmi pretraživanja (Grivokostopoulou, Perikos i Hatzilygeroudis, 2016).

Slika 4.11. Igra Educational Pacman

Igra ima dva načina rada:

1) Obrazovni način: učenik može pročitati tekstualni opis, grafički dijagram toka i pseudokod algoritma po svom izboru. Igra također omogućava studentu da nauči algoritam pomoću vizualizacija na različitim labirintima *Pacman Maze*.

2) Način za igranje: učenik mora riješiti razine labirinta u različitim uvjetima i uz vremensko ograničenje. Te su razine napravljene tako da učenik mora primijeniti određeni algoritam pretraživanja za pomicanje *Pacmana* u labirintu (od studenta se traži da, iz slučajno odabranog položaja u labirintu, premjesti lik na određeni element).

CMX

CMX je *online* igra za više igrača u različitim ulogama (eng. *Massive Multiplayer Online Role Playing Game - MMORPG*) za učenje programiranja. U igri postoje dva tima – razbijači kôda i hakeri (eng. *crackers and hackers*) koji se međusobno natječu u pronalaženju lozinki u globalnoj tvornici otrovnog otpada. *Senseis* im pomažu u učenju programskog jezika C. Igra uključuje tri razine *Senseis* do kojih se može doći otključavanjem lozinke na prethodnoj razini. Čini se da su studenti kroz igru povećali razinu razumijevanja C jezika - mogli su ne samo odgovarati na teorijska pitanja, već i pisati programe u C (Malliarakis, Satratzemi i Xinogalos, 2014).

Slika 4.12. Igra CMX

Tablica 4.2. Usporedba okruženja za učenje programiranja

IGRA	PLATFORMA	CILJNA GRUPA	JEZIK (JEZICI)	TEME	TIP IGRE
Code Combat	*online*	9+	JavaScript, Python	Sintaksa, metode, parametri, nizovi, petlje i varijable, uvjeti, logika, relacijski operatori, funkcije, svojstva objekta, događaji, ulazne vrijednosti, aritmetika, brojači, while-petlje, prekid, nastavak, nizovi, usporedba niza, pronalaženje najmanjeg i najvećeg elementa, objektni literali, pozivanje, for petlje, složene funkcije, crtanje, modulo	Komercijalna igra Besplatna osnovna verzija (*Freemium*)
Human Resource Machine	Windows, Mac OS X, Linux, iOS, Android	9+	Vizualni programiski jezik	Ulaz i izlaz, uvjeti i ponavljanja, usporedba algoritma	Komercijalna igra (plaća se) Puzzle
Lightbot	iOS, Android, Windows, Mac OS X	9+	Vizualni programiski jezik	Uvjeti i ponavljanja, rekurzije, strategije otklanjanja pogrešaka	Komercijalna igra (plaća se) Puzzle
May's Journey	Windows	12-18	Prilagođeni programski jezik (nadahnut objektno orijentiranim jezicima poput Jave)	Osnovne upute i logika, petlje, varijable, uvjeti, uspoređivanje, operacije na cijelim brojevima, operacije na nizovima	Istraživačka igra Puzzle
No Bug's Snack Bar	*online*	18+	Vizualni programiski jezik	Manipulacija varijablama, slijed radnji, uvjeti i ponavljanja, strategije otklanjanja pogrešaka	Istraživačka igra
Robot ON!	Windows	18+	C++	Sintaksa i semantika, varijabilni i primitivni tipovi podataka, izrazi i dodjele, uvjeti i ponavljanja, razumijevanje programa	Besplatna Istraživačka igra
Educational 'Pacman'	Windows	18+	Igra za učenje AI pretraživanja Algoritam	Algoritmi (osnovni tekstualni opis algoritama i odgovarajući grafički dijagram toka zajedno s pseudokodom)	Istraživačka igra
CMX	Windows	18+	C	Teorija o nizovima, sintaksa, unos vrijednosti varijabli u niz, izlaz, izračun zbroja elemenata niza, izračun maksimalne vrijednosti u nizu	Istraživačka igra MMORPG

5. CODING4GIRLS OKRUŽENJE ZA IGRANJE ZA UČITELJE I UČENIKE

5.1. Coding4Girls platforma i *design thinking* pristup

Rezultat projekta *O2 - Promicanje razvoja programskih vještina kod djevojčica kroz obrazovne igre* imao je kao cilj izgradnju programskih vještina među mladima, uglavnom djevojčicama, u osnovnim i srednjim školama pomoću softvera razvijenog u okviru projekta te je primjer za izvrsnu kombinaciju IKT i obrazovne igre.

Ovaj je softver dizajniran za prevladavanje postojećeg jaza između muškog i ženskog sudjelovanja u obrazovanju i karijeri iz područja informatike, uvođenjem metodoloških intervencija učenja koje informatiku čine privlačnom za sve. Softver i aktivnosti učenja koje će se provoditi razvijene su s ciljem poticanja djevojčica na sudjelovanja u računalnim znanostima i informatici. Dizajnirani su prema *design thinking* pristupu kao kreativnom procesu koji pomaže učenicima u zajedničkom dizajniranju konkretnih rješenja zajedno sa svojim vršnjacima.

Ovaj postupak dizajna vrlo je učinkovit zahvaljujući svom strukturiranom okviru za prepoznavanje izazova, prikupljanje informacija, generiranje potencijalnih rješenja, pročišćavanje ideja i testiranje.

Proces je rekurzivne prirode i zahtijeva interakciju. Svaka faza u procesu zahtijeva ponovno pregledavanje i pozivanje tijekom procesa učenja kako bi se potaknulo eksperimentiranje, izvedivost rješenja i promišljanje (Luka, 2014).

Design thinking aktivnosti omogućuju da se u velikoj mjeri ostvari suradničko iskustvo pri učenju, u učionici i izvan nje. Učenic su izravno uključeni u prikupljanje informacija, generiranje znanja, komunikaciju i prezentaciju.

U projektu Coding4Girls, osmišljeni pristup *design thinking* koristi glavne prednosti učenja temeljenog na igrama kako bi potaknuo i motivirao učenike u njihovom procesu učenja. Koristi neke važne elemente igre (npr. bodove i izazove) i scenarije koji određenu aktivnost čine zabavnijom na način da povećavaju stupanj uključenosti, motivacije i ostvarivost rezultata.

Korištenje izazova pomaže učenicima da vide cjelokupnu sliku prije dizajniranja detaljnog rješenja te potiče kod njih poduzetničko razmišljanje o digitalnim tehnologijama i načinu na koji se one mogu koristiti za rješavanje problema iz stvarnog svijeta.

Softver, razvijen u okviru projekta, sastoji se od dva različita no međusobno povezana dijela koju čine platforma za učitelje i okruženje igre za učenike.

5.2 Platforma za učitelje

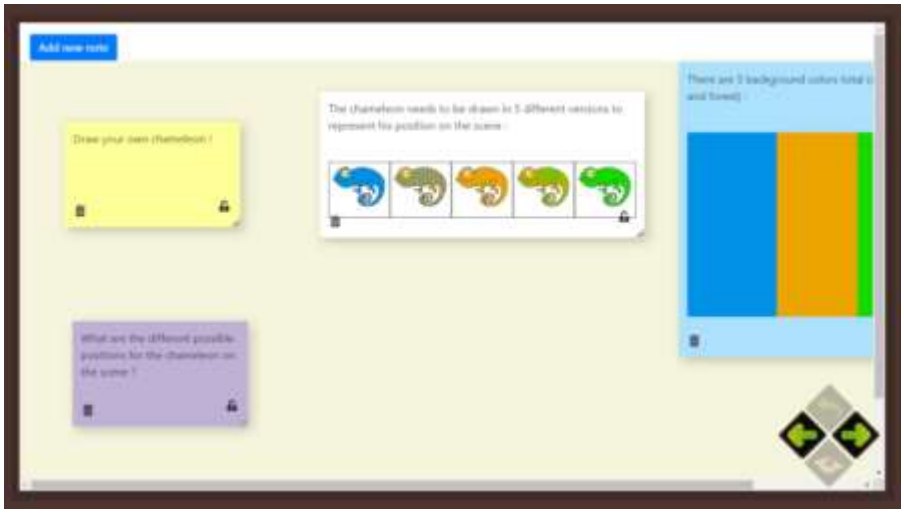
Platforma za učitelje (eng. *The Teachers' Platform*) je web platforma na kojoj učitelji mogu dizajnirati i pripremiti određene tečajeve za razvijanje vještina kodiranja svojih učenika pomoću programskog okruženja Snap!.

Unutar platforme svaki učitelj ima na raspolaganju i privatno i javno područje. U prvom mogu pripremiti svoje tečajeve na temelju potreba svojih učenika. Drugi je repozitorij svih kreiranih tečajeva drugih učitelja. Cilj je podijeliti, koristiti se ili nadahnuti aktivnostima učenja koje su već pripremili drugi kolege (slika 5. 1). Naravno, svaki učitelj može prilagoditi sve javne tečajeve prema specifičnosti svojih predavanja.

Slika 5.1. Platforma za učitelje i neki dostupni tečajevi kodiranja

Pripremljeni tečajevi složeni su kao prostor za grupiranje tematski povezanih aktivnosti, a sve je povezano s odabranim općenitim problemom. Problem je predstavljen učenicima na samom početku tečaja ili predmeta kako bi svi zajedno mogli promišljati o njemu i zajednički razraditi okvirna rješenja. Ovo je vrlo važna faza procesa učenja u kojoj učenici mogu generirati nove ideje i biti kreativno udubljeni u proces rješavanja problema.

Ploču za razmjenu ideja mogu pripremiti učitelji postavljanjem ključnih pitanja ili primjedbi koje će se koristiti tijekom rasprave učenika. Softver nudi mogućnost pripreme „*post-it* papirića“ na ploči korištenjem teksta, slika ili video zapisa kao što je prikazano na slici 5.2.

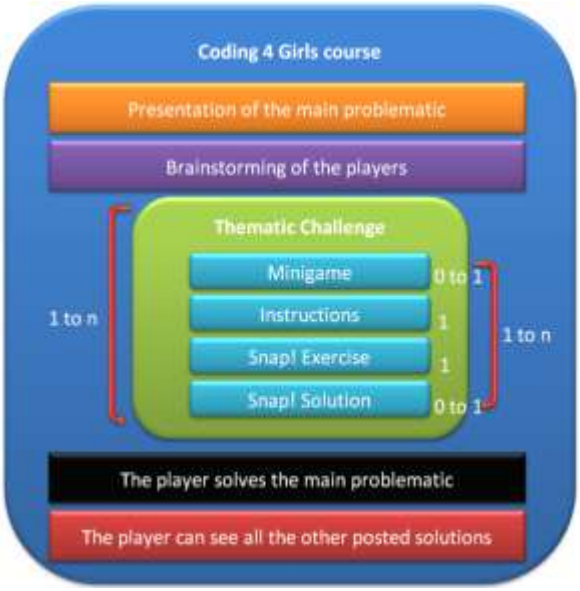


Slika 5.2. Ploča za razmjenu ideja u sučelju za učenike

Svi „*post-it* papirići“ imaju ikonu lokota koja se može zaključati ili otvoriti kada se žele uređivati. Samo učitelji imaju mogućnost zaključavanja i otključavanja, učenici to ne mogu činiti.

Nakon faze promišljanja, učenicima se postepeno, korak po korak, zadaju specifične aktivnosti (predstavljene uzastopnim redoslijedom) namijenjene predstavljanju koncepata potrebnih za rješavanje početnog, općeg problema. Ove aktivnosti programiranja predstavit će se pomoću okruženja Snap! kroz djelomično riješene zadatke kako bi potakli učenike u rješavanju problema i konačnom prezentiranju gotovog rješenja.

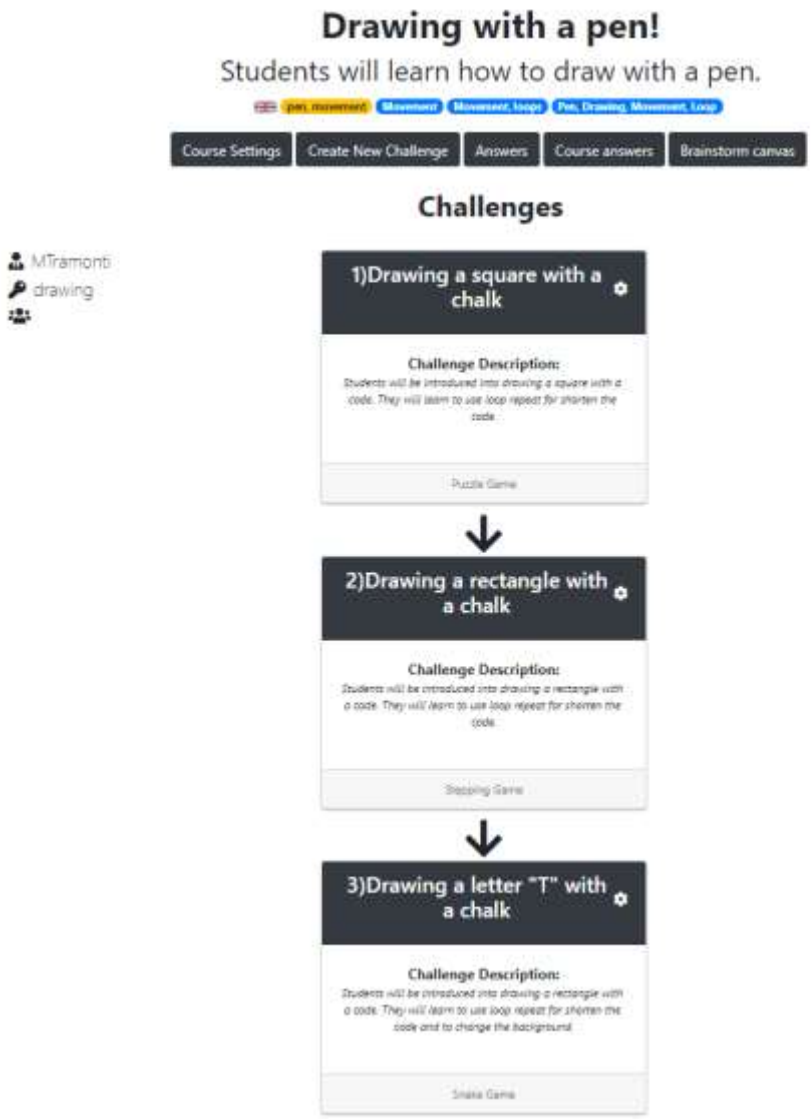
Na slici 5.3 prikazana je struktura C4G tečajeva objavljenih na platformi za učitelje.



Slika 5.3. Struktura tečaja C4G

Svaka grupirana aktivnost u tečaju naziva se izazovom (eng. *challenge*) kojeg učitelj prezentira učenicima u određenom redoslijedu (slika 5.4). Na primjer, ako učitelj želi kreirati tečaj o osnovnom znanju programiranja, prva aktivnost mogla bi se odnositi na koncept Boolovih operatora, druga na strukture za uvjete i zadnja na petlje. To će omogućiti učiteljima da pripreme za svoje učenike prilagođene korake učenja na platformi za učitelje koji će zatim učenike postupno voditi do konačnog cilja učenja u okruženje igre za učenike.

Naravno, učenici se trebaju igrati i riješiti sve izazove redom koji su odredili učitelji, prije ostvarivanja konačnog rješenja.

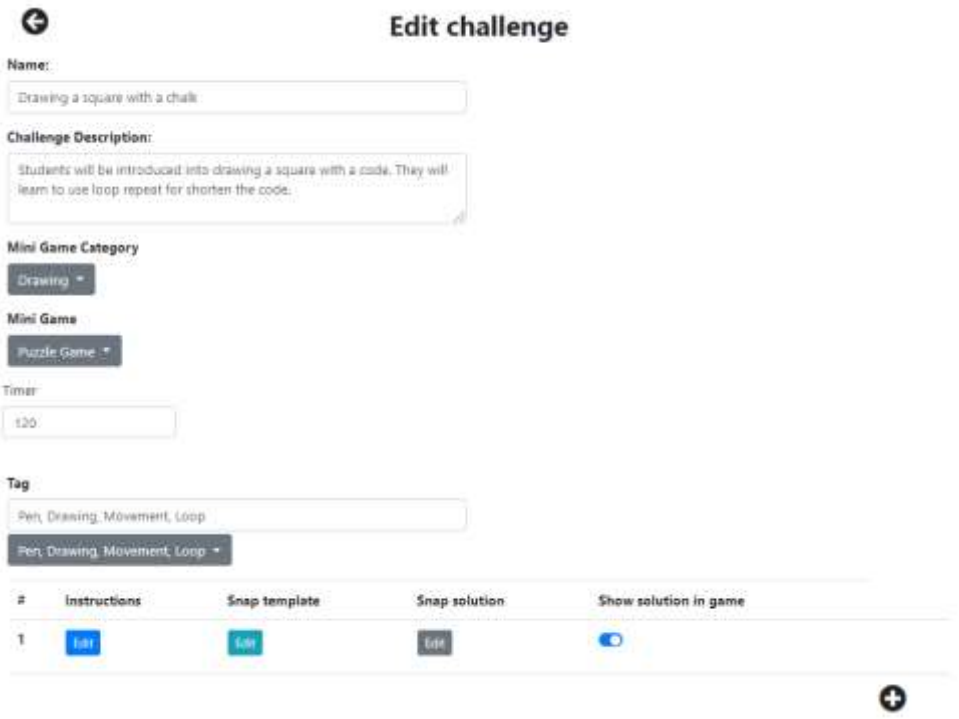


Slika 5.4. Primjer izazova u tečaju u sučelju za učitelje

Svaki izazov može biti povezan s mini-igricom koju je razvio projektni tim. Cilj je pomoći učenicima da bolje razumiju što bi mogao biti konačni rezultat određenog svojstva kodiranja koje su pročavači.

Na primjer, ako se izazov bavi proučavanjem svojstva petlje "*loop*", *Match-3* je jedna od razvijenih mini-igara koje bi mogle biti povezane s tim izazovom.

Stoga bi svaki izazov mogao imati, osim Snap! canvasa za rješavanje zadatka, mini-igru koju će učenik trebati igrati slijedeći stranicu s uputama koje je definirao učitelj tijekom faze pripreme na platformi za učitelje (slika 5.5).



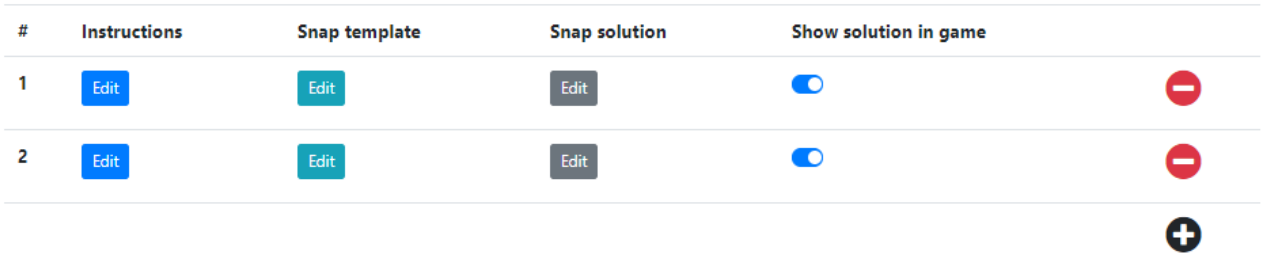
Slika 5.5. Pogled iznutra na izazov u sulječu za učitelje

Sve mini-igre koje se učenicima nude (trenutno ih je 11) u pojednostavljenom obliku odgovaraju stvarnim programskim konceptima na kojima se temelji dizajn C4G tečajeva i scenarija. Međutim, nije obavezno uključiti mini-igru u izazov, to ovisi o izboru učitelja.

Svaki se izazov sastoji od tri dijela:

1. Upute, u kojima učitelj može dati neke naznake ili objašnjenja o zadatku koji će izvršiti učenici.
2. Snap! predložak, gdje će učenici pokušati riješiti dodijeljeni zadatak pomoću Snap! programiranja pomoću blokova u okruženje igre za učenike.
3. Snap! rješenje, pri čemu učitelji mogu odlučiti pokazati ili ne konačno rješenje zadatka.

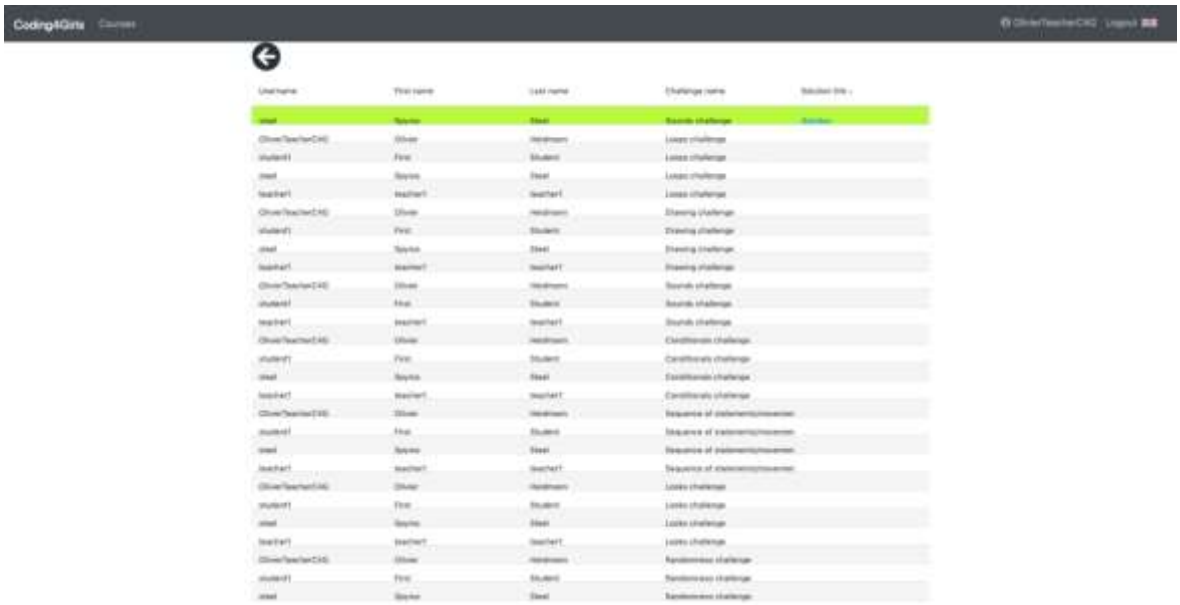
Ako je izazov složen u smislu znanja o programiranju, može se strukturirati u više zadataka (slika 5.6). Na taj će način učitelji moći korak po korak voditi svoje učenike kroz različite faze kako bi riješili složenu aktivnost raščlanjivanjem na manje dijelove.



Slika 5.6. Podjela složenog izazova na manje zadatke

Dodatno, učitelji mogu pregledati sve Snap! odgovore svojih učenika nakon što ih predaju za sve izazove pomoću prikaza u tablici na platformi za učitelje. Tablica sadrži popis korisnika i njegove detalje za svaki izazov koji postoji u tečaju. Svaki redak prikazuje korisničko ime, ime i prezime učenika, naziv izazova i poveznicu na rješenje.

Ukoliko nema poslanog rješenja, stupac "Poveznica rješenja" bit će prazan. U suprotnom, redak će biti posebno označen i u zadnjem stupcu pojavit će se riječ "Rješenje", a klikom na njega prikazat će se rješenje učenika (slika 5.7).



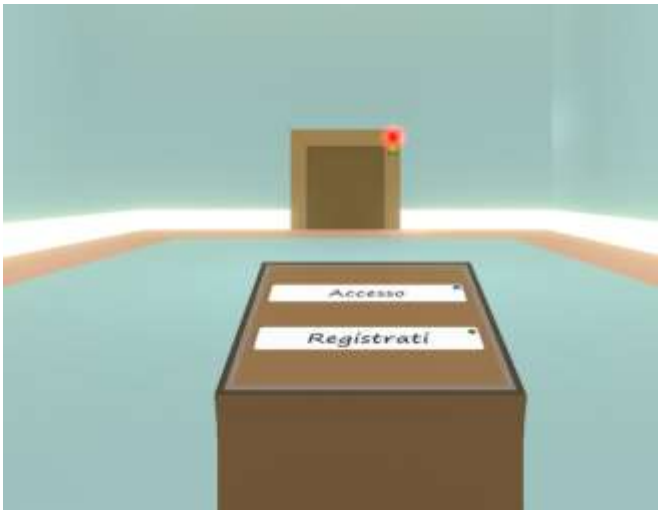
Slika 5.7. Odgovori učenika s rješenjem

Na taj način učitelji uvijek mogu nadgledati i pratiti napredak i rezultate svojih učenika.

5.3. Okruženje igre za učenike

Učenici će kreiranim izazovima pristupiti kroz okruženje igre za učenike (eng. *Student Game Environment*) dostupno kao Unity 3D video igra koju se može preuzeti i instalirati na računalo. Učenici pomoću igre mogu na zabavan, zanimljiv i razigran način otkriti i dovršiti tečajeve koje su pripremili njihovi učitelji.

U okviru tečajeva, koristeći elemente *design thinking* pristupa, učenicima se predstavljaju projekti i alate koji omogućuju njihovo rješavanje postupnim pristupom, korak po korak. Kao što je prethodno spomenuto, učitelji pripremaju i objavljuju tečajeve na platformi za učitelje, a učenici im mogu pristupiti kroz okruženje igre.



Slika 5.8. Prva soba za pristup učeničkom okruženju igre

Učenici se mogu pridružiti tečaju unošenjem ispravnog koda na terminalu u uloznoj sobi. Kôd je jedinstven i trebali bi ga dostaviti učitelji koji su već pripremili tečaj na platformi za učitelje. Nakon prijave u tečaj, njegov je naziv vidljiv na vrhu portala, što ukazuje da je to trenutno odabrani tečaj (slika 5.9).

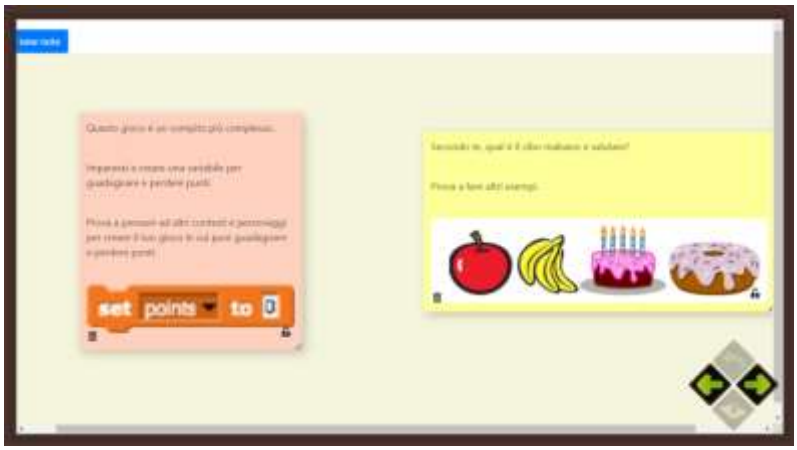
Slika 5.9 Druga soba s dva terminala: jedan za pridruživanje tečaju po prvi puta i jedan za odabir među upisanim tečajevima

Učenici se potiču da dizajniraju i programiraju igre koje se bave određenim potrebama ili problemima iz stvarnog života (ovisno o izboru učitelja). Koristi se i pristup *„low entry high ceiling"* koji učenicima omogućuje da započnu s lakim problemima te kasnije nastave sa zahtjevnijim zadacima. Učitelji osmišljavaju i predstavljaju svojim učenicima polu-gotove (eng. *half-baked*) scenarije u kojima je rješenje djelomično spremno, potičući ih da izvrše zadatak. Stoga učitelji mogu pripremiti male i upravljive module pomoću softvera Coding4Girls (C4G) koji je prikladan za potrebe i znanje njihovih učenika.

Slijedeći pristup *design thinking*, projektni tim je pripremio nekoliko tečajeva i scenarija učenja osmišljenih da potaknu učenike u rješavanju određenog problema iz programiranja. Trenutno je dostupan 21 scenarij učenja koji su bili prikupljen u izvještaju „Kolekcija scenariji za učitelje temeljenih na učenju programiranja pomoću igara", te ponovno prilagođen pristupu *design thinking* slijedeći strukturu softvera C4G.

Scenariji imaju različite razine, od osnovne do napredne (za sposobnije učenike), a dostupni su na engleskom, slovenskom, talijanskom, hrvatskom, bugarskom i grčkom jeziku.

Učenici zadatke mogu riješiti pojedinačno i u međusobnoj suradnji putem grupne rasprave u učionici ili u okruženju igre za učenike. Softver nudi prostor za zajedničko promišljanje u kojem učenici mogu raspravljati i dijeliti svoje ideje postavljanjem i upravljanjem multimedijskim *post-it*-ima, kao što je prikazano na slici 5.11. Svi učenici uključeni u tečaj mogu napisati svoj doprinos koji je vidljiv svima ostalim upisanima.



Slika 5.11. Ploča za razmjenu ideja u scenariju učenja „Kupnja hrane za piknik“

Sljedeća slika prikazuje panel s izazovima u okruženju igre za učenike demo tečaja koji je kreirao učitelj u platformi za učitelje. Brojevi od 0 do 13 predstavlja izazove. Na ovom panelu, izazov broj 0 predstavlja opće upute koje se trebaju pročitati na početku igre, a otvaranje će odvesti učenike do općenitih uputa o problemima, odgovarajući Snap! predložak popraćen pločom za rasprave. Ostali izazovi, od broja 1 do 13, predstavljaju zadatke koje su učitelji pripremili na svojoj platformi.

Slika 5.12. Ploča s izazovima za učenike

Nakon što se odabere jedan od izazova, detalji o njemu će se pojaviti na dnu zaslona: naziv na lijevoj strani, mini-igra povezana s izazovom u sredini i gumb za pokretanje izazova desno. Slika 5.12 prikazuje naziv izazova koji odgovara temi koja se proučava, u ovom slučaju "Uvjeti". Uz temu izazova, sustav prikazuje 3D mini igru povezanu s njom, na pr. "Pronađi svoj put".

Učitelji mogu odlučiti koju će mini-igru njihovi učenici igrati te po želji odabrati jednu od postojećih mini-igara, s nazivima: *Match3, Dice game, Inventory fame, Find your path game, Stepping game, Sound game, Snake game, Puzzle game, Pattern matching game.* Umjesto igre mogu odabrati kviz s pitanjima višestrukog izbora.

Slika 5.13 Primjer dostupne mini igre - *Dice Game*

Svaki izazov u okruženju igre za učenike strukturiran je na sljedeći način: uvodna mini-igra koja ilustrira koncept iz programiranja, stranica s uputama za zadatak koji treba riješiti u Snapu!, dva Snap! okruženja - jedno se koristi za rješavanje zadatka, a drugo za prikaz rješenja aktivnosti.

Stranica s uputama je u HTML-u (slika 5.14.), obično obogaćena slikama ili videozapisima kako bi se prikazao kontekst i specifični cilj zadatka učenja koji treba ispuniti u Snap!-u.

Slika 5.14 Primjer stranice s uputama izazova za učenike

Nakon ove stranice s uputama slijedi Snap! okruženje za programiranje (slika 14.) s predloškom koji je pripremio učitelj i koje sadrži aktivnost za programiranje ili problem koji će učenici izvršiti ili riješiti.

Slika 3.15. Primjer predloška projeta u Snap!-u

Drugo Snap! okruženje služit će za prikaz rješenja predloženog izazova. Međutim, samo učitelj može dozvoliti hoće li se prikazati okruženje s rješenjem ili neće, što će ovisiti o modelu nastavnog procesa koji je odabran.

5.16. Primjer Snap! okruženja s rješenjem

Budući da svaki tečaj uključuje više izazova, opisani koraci će se ponavljati onoliko puta koliki je broj ukupnih izazova koje su utvrdili učitelji. To omogućuje razlaganje složene aktivnosti programiranja u jednostavne elementarne korake gdje odgovor i/ili rješenje za prethodnu aktivnost postaje predložak u kojem treba izvršiti sljedeću aktivnost.

Na kraju, tečaj se sastoji od određenog broja izazova koji prezentiraju učenje programiranja kao proces koji uključuje kontinuiranu podršku za učenike.

Nakon što učenici završe sve izazove tečaja, vraćaju se na početno postavljeni problem iz programiranja i od njih će se tražiti da sintetiziraju nova znanja koja su upravo stekli. Na samom kraju tečaja, igrači mogu vidjeti i sva rješenja problema koja su predložili drugi učenici.

Opisani C4G pristup i alati omogućuju učenicima da razvijaju, poboljšavaju i ojačavaju vještine programiranja kroz personalizirane aktivnosti poučavanja, kombinirajući zabavu sa zadacima učenja.

6. PRIMJERI LEKCIJA (SCENARIJI UČENJA)

6.1. Scenariji učenja

U okviru projekta Coding4Girls razvijena su 22 scenarija učenja. Oni opisuju hibridne aktivnosti učenja koje primjenjuju CODING4GIRLS *design thinking* pristup i učenje temeljeno na igrama. Dostupni su na engleskom jeziku, kao i na nacionalnim jezicima projektnih partnera - bugarskom, hrvatskom, grčkom, talijanskom, portugalskom, slovenskom i turskom. Svi su dostupni na web sjedištu projekta (https://www.coding4girls.eu/results_03.php).

Pripremljeni scenariji učenja sažeto predstavljaju informacije koje će pomoći učiteljima da integriraju predložene obrazovne igre i *design thinking* metodologiju učenja u svoju nastavnu praksu. Slijede CODING4GIRLS pristup temeljen na dizajnu igara i uključuju informacije za svaku aktivnost učenja koja će se služiti za razvoj vještina programiranja kod djevojčica i dječaka. Dostupne su sljedeće informacije:

- opći obrazovni cilj za aktivnost učenja
- pojmovi obuhvaćeni aktivnošću učenja
- specifični odgojno-obrazovni ciljevi
- očekivani ishode učenja
- opis koraka potrebnih za realizaciju CODING4GIRLS pristupa učenju temeljenog na dizajnu igara
- metode ocjenjivanja za vrednovanje usvojenog znanja
- pitanja za poticanje diskusije među učenicima prilikom suradnje u razredu.

Učitelji mogu koristiti scenarije i igre u predloženom slijedu ili ih mogu slobodno birati prema svojim željama i potrebama. Također, učitelj mora voditi računa o prilagodbi nekih scenarija prema znanju učenika, primjerice o nekim pojmovima naučenim iz matematike kao što su koordinatni sustav, koordinate, negativni brojevi, razlomci i sl.

Scenariji učenja pokrivaju i generičku funkcionalnost predložene obrazovne igre, uključujući procese interakcije s korisnikom i generiranje povratnih informacija, kao i opise svih aktivnosti učenja koje će se implementirati u predloženu igru.

Pripremljeni scenariji učenja slijede od osnovnih s jednim konceptom programiranja do naprednijih s više koncepata programiranja. Sljedeća tablica predstavlja predloženi redoslijed aktivnosti.

Tablica 6.1. Popis scenarija učenja razvijenih tijekom projekta Coding4Girls

OSNOVNI SCENARIJI UČENJA		
1.	**Uvod u sučelje alata Snap!** Upoznavanje sa sučeljem alata Snap! za vizualno programiranje	UL
2.	**Vrijeme je za oživljavanje vašeg objekta** Pronalaženje blokova koda i njihovo spajanje, pomicanje objekta, omogućavanje da objekt nešto kaže	UL
3.	**Kretanje po pozornici** Izrada smislenog slijeda blokova	UL
4.	**Mijenjanje kostima i okretanje**	UL
5.	**Zvukovi s farme** Dodavanje, uvoženje, snimanje i reproduciranje zvuka	UL
6.	**Kameleonov ljetni odmor** Upoznavanje s događajima, očitavanje boje, logičke vrijednosti, provjeravanje i reagiranje na dva različita stanja igre	UL
7.	**Pomaganje princu i princezi u pronalasku njihovih životinja** Korištenje uvjeta, crtanje	UL
8.	**Crtanje s kredom** Korištenje petlji, okretanje, promjena pozadine	UL
9.	**Skupljanje otpadaka i čišćenje parka** Uvod u varijable, dupliciranje objekata, blokovi koda	UL
10.	**Hranjenje mačaka** Korištenje varijabli (unutar i izvan petlji), petlje, spajanje stringova, slučajni brojevi, operatori, ulazne vrijednosti	UL
11.	**Pogađanje broja mačaka u skloništu** Korištenje slučajnih vrijednosti, unosa varijabli, uvjeta, operatora usporedbe, brojača	UL
NAPREDNI SCENARIJI UČENJA		
12.	**Hvatanje zdrave hrane** Korištenje varijabli, uvjeta, petlji, kretanje u smjeru, slučajne vrijednosti	UL
13.	**Crtanje**	UNIRI

14.	**Pričanje priča**	SWU
15.	**Uhvati miša** Petlje, uvjeti, varijable	UL
16.	**Kupnja hrane za piknik** Varijable, uvjeti, operatori	UL
17.	**Operacije**	SWU
18.	**Recikliranje**	SWU
19.1	**Sviranje klavira 1**	SWU
19.2	**Sviranje klavira 2**	UNIRI
20.	**Test**	SWU
21.	**Pojednostavljena igra PACMAN** Korištenje kretanja objekta temeljeno na događajima, očitavanje boje, logičke vrijednosti, provjeravanje i reagiranje na dva različita stanja igre	UL

Svih 22 scenarija za učenje su uključeni u Prilog 1. U Prilogu 2 dani su kodovi za sve igre uključene u dio s javno dostupnim igarama u okruženju Coding4Girls.

6.2. Primjeri korištenja okruženja za igre razvijenog u okviru projekta Coding4Girls

Okruženje igre za učenike je obrazovna video igra zasnovana na Unityju i dostupna na Windows i Mac OS X platformama. Nakon što započne igrati igru, korisniku se prvo na 4 sekunde prikaže uvodna scena s europskom izjavomo odricanju odgovornosti i logotipovi Erasmus+ i projekta.

Igrač nakon toga dolazi u veliku praznu sobu koja se zove predvorje, kontrolira nevidljivog avatar i promatra svijet igre iz perspektive prvog lica. Konzola je postavljena u sredini prostorije, što omogućuje korisniku da se prijavi na C4G poslužitelj pomoću svojih podataka ili stvaranjem novog računa. Nakon što se prijavi, vrata se otvaraju i igrač može nastaviti u drugu sobu gdje se može pretplatiti na novi tečaj ili pristupiti tečaju koji je već aktiviran. Nakon odabira željenog tečaja, korisnik će proći kroz lebdeći portal kako bi nastavio s igrom i otkrio točan sadržaj odabranog tečaja.

Tečaj se formira kao kolekcija povezanih izazova kodiranja pomoću Snap!, svaki je dopunjen mini-igricom. Idealno bi bilo da svaki izazov ilustrira jedan aspekt ili korak lekcije koju treba naučiti na tečaju.

Prema zadanim postavkama, C4G je definirao neke osnovne koncepte kodiranja koji su ilustrirani mini-igrom: petlje, uvjete, varijable, naredbe, paralelizam, operatore, događaje. Osim toga, učitelji također mogu odlučiti zamijeniti mini-igru kvizom s višestrukim izborom pitanja ili ne uključiti ništa od toga, ostavljajući samo Snap! vježbe u izazovima.

Slika 6.1. Kategorije dostupnih mini-igara

Svaki izazov uključuje zadatke kodiranja koje treba riješiti u Snapu! i može se povezati s mini-igrom prikazanom na ploči s izazovima (slika 6.2). Integracija mini-igre u specifični izazov ilustrira koncept programiranja u vezi zadatka koji su osmislili učitelji no to nije obavezno. To znači da učitelj može odlučiti uključiti ili ne mini-igru za svoje učenike u izazovu i koju mini igru igrati, odabirući ih s popisa postojećih mini igara.

Treba razmotriti potrebu povezivanja mini-igre s izazovom, te je uključiti ukoliko ona donosi dodanu vrijednost u smislu razumijevanja, provjere koncepta, vizualizacije mehanike i boljeg uključivanja i motiviranja učenika na njihovom putu učenja.

Slika 6.2. Panel s izazovima

Trenutno postoji 11 različitih mini-igara, u rasponu od *Match3* igre do kviza s pitanjima višestrukog izbora.

Slika 6.3 Mini-igre dostupne na platformi učitelja

Kad korisnik uđe u izazov povezan s mini igrom, bit će teleportiran na mjesto na kojem se mini-igra odvija. Postoji mnogo različitih lokacija za mini-igre , od pješčanih plaža do snježnih planina.

Slika 6.4 Primjer mjesta povezanog s jednom mini igrom

Među dostupnim mini-igrama su: *Snake game, Match3 game, Dice game, Inventory game, Find your path game, Stepping game, Sound game, Puzzle game, Pattern matching game* i kviz s pitanjima višestrukog izbora.

Svaka mini-igra ima svoj zadatak koji treba savladati, a raspon dostupnih mini-igara pokriva različite dobne skupine učenika i/ili teme iz programiranja koje se predaju na tečajevima.

Jedna od najjednostavnijih mini-igara je igra *Snake* (zmija), postavljena je na obali rijeke i igrač treba slijediti drveni znak strelice do crvenog kruga da započne aktivnost. (Slika 6.5.)

Slika 6.5. Primjer mjesta povezanog s igrom *Snake*

Igra (slika 6.6.) se odvija u vodi. Zmijom se upravlja pomoću 4 tipke sa strelicama na tipkovnici i pokušava preživjeti što je duže moguće. Zmija se povećava jedući jarko zelene točkice, ali može umrijeti ako uhvati svoj rep, udari o rub zaslona ili pojede crvenu točkicu. Svaka pojedena zelena točkica pridodaje po jedan bod rezultatu igrača.

Slika 6.6. Igra *Snake*

Ova jednostavna mini igra dobro ilustrira nekoliko programskih koncepata kao što su petlje, pokreti, mijenjanje grafike i slično.

Drugi primjer je igra *Match3*. Odvija se u snježnom, brdovitom području kojim igrači mogu slobodno lutati.

Slika 6.7. Primjer mjesta povezanog s igrom *Match3*

Mini-igra *Match3* temelji se na tipičnoj aktivnosti *Match3* gdje morate povući i ispustiti susjedne kvadrate kako biste nestali ukoliko formiraju skupinu od tri ili više pločica iste vrste.

.

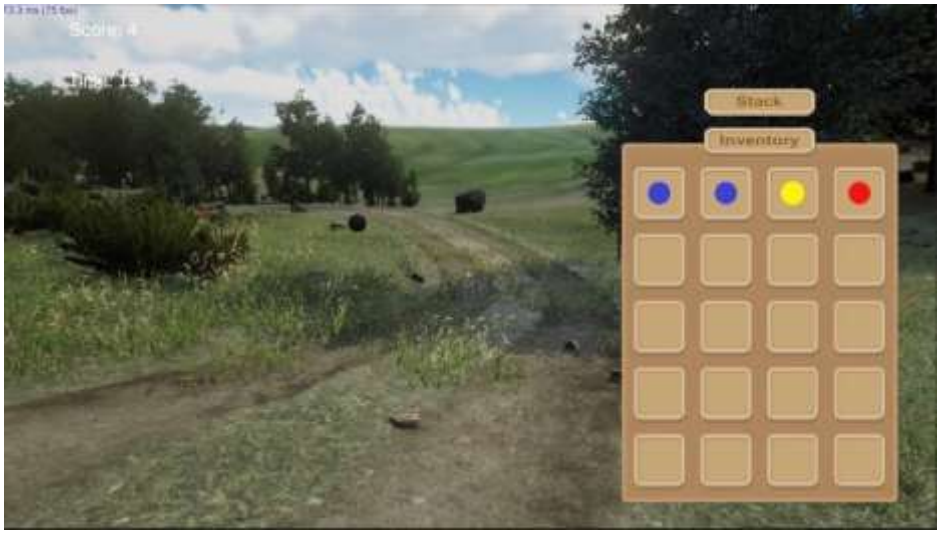
Slika 6.8. Mini igra *Match3*

Dice game (igra kockica) odvija se u mračnoj šumi. Interaktivni dio mini-igre nalazi se u napuštenoj kabini, na stolici preko puta vrata. To je drvena ploča na kojoj su 2 kocke.

I igrač i sama igra putem umjetne inteligencije bi trebali redom baciti kockice, a onaj s najvećim zbrojem dvije kocke pobjeđuje u tom krugu.

Slika 6.9. Mini igra *Dice game*

Igra *Inventory game* odvija se na poljima, pokraj male šume. Okolo plutaju šarene kugle i igrač ih mora sakupiti, prema uputama koje mu je učitelj dao na drvenom znaku.



Slika 6.10. Mini igra *Inventory game*

Igra *Stepping game* odvija se na jednoj plaži na otoku, u zalasku sunca. Igrač mora odgovoriti na pitanje napisano na velikoj stijeni u središtu plaže tako što će stati na slovo s ispisom točnog odgovora.

Kad igrač stane na jedan od kamena sa slovom koje tvori odgovor, slovo će biti označeno zelenom bojom i pojavit će se na drugoj velikoj stijeni pored one na kojoj je napisano pitanje. Ako korisnik stane na pogrešan kamen, bit će označen crvenom bojom i morat će ponovno započeti s ispisom odgovora.

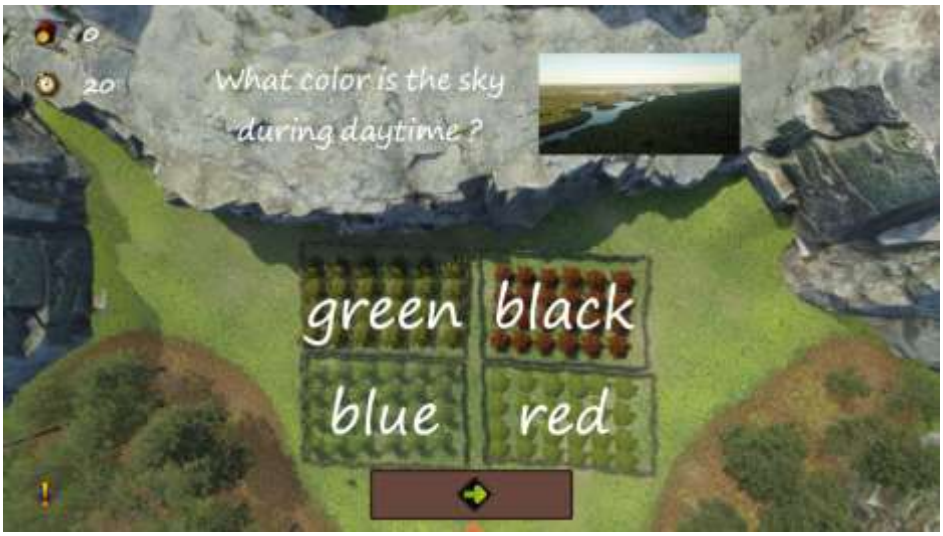
Slika 6.11. Mini igra *Stepping game*

Kviz s pitanjima višestrukog izbora postavljen je u polju ispod litice, a igrači mogu slobodno lutati prije nego što započnu ovu mini-igru.

Pitanja se pojavljuju na vrhu litice, a svako polje sadrži četiri moguća odgovora, koja mogu biti i sa slikama.

Ikona medalje predstavlja trenutni rezultat učenika na temelju točnih odgovora i ostvarenog vremena (uključen je sat koji odbrojava sekunde do završetka ovog zadatka).

Gumb sa strelicom na dnu omogućuje da se prijeđe na sljedeće pitanje.



Slika 6.12. Kviz s pitanjima višestrukog izbora

Mini-igra *Puzzle* se odvija u stjenovitom području karte. Cilj je pronaći raštrkane ploče na tom području i riješiti zagonetke stvaranjem crte koja započinje od bijelog kruga ploče koja ide do crvenog kvadrata.

Slika 6.13. Ploča s mini igrom *Puzzle*

Igra *Pattern matching game* (podudaranja uzoraka) odvija se na obalama rijeke. Kutije s brojevima ispisanima na boku počet će plutati rijekom prema moru. Cilj igre je da ih igrač pokupi kako bi dovršio određenu matematičku operaciju.

Drugim riječima, igrači trebaju koristiti brojeve i dostupne operacije kako bi došli do definiranog broja zapisanog na plavom stolu.

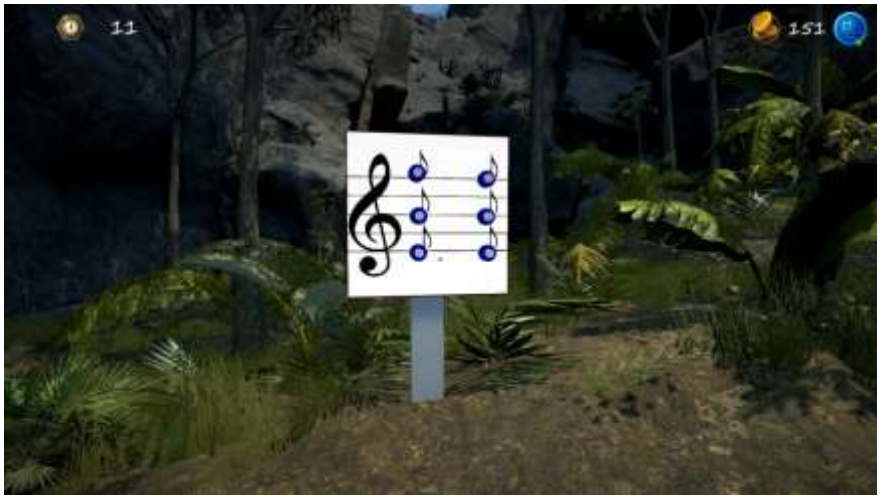
Brojevi se generiraju nasumično, ali na način koji osigurava da uvijek postoji moguća kombinacija za postizanje ciljnog broja.

Dostupne operacije ovise o učitelju i mogu biti sljedeće:

- osnovne operacije (+, -, *, /)
- napredne operacije (potencije, drugi korijen, modulo)
- Trigonometrija (cos, sin, tan)
- Boolean (AND, OR, XOR).

Slika 6.14. Mini igra *Pattern matching*

Još jedan primjer mini-igre je *sound game* koja zahtijeva puno pažnje igrača kako bi uspio u svom zadatku. Odvija se u tropskoj šumi i igrač treba pronaći pet razbacanih ploča i za svaku od njih riješiti zagonetku na temelju zvukova koji se mogu čuti u džungli.

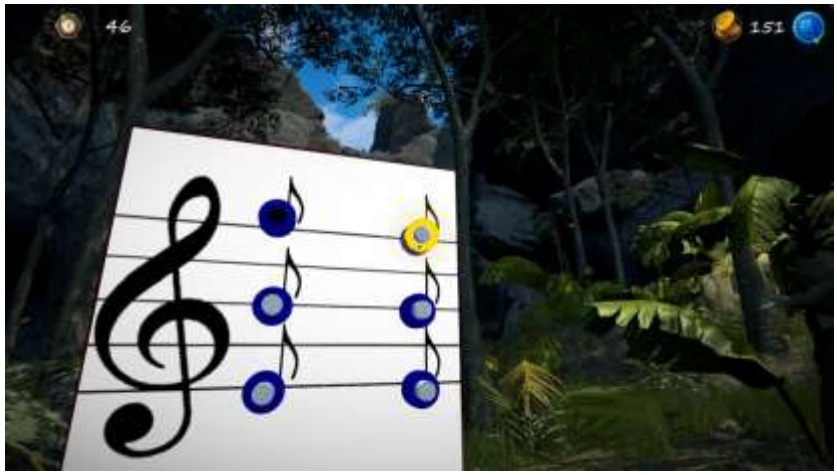


Slika 6.15. Ploča s igrom *Sound game*

Svaki stupac ploče predstavlja jedan zvuk, a svaki redak ton zvuka. Gumbi na vrhu služe za visoke tonove, a gumbi na dnu za niske. Kad se korisnik nalazi u blizini ploče, reproducirat će se kombinacija 3 moguća zvuka (jedan poziv ptice s niskim tonom, jedan poziv ptice sa srednjim tonom i jedan poziv ptice s visokim tonom). Kombinacija ptičjih poziva ponavlja se u petlji, a svako ponavljanje odvojeno je od ostalih tišinom u trajanju od 3 sekunde.

Igrač će morati obratiti veliku pozornost na redoslijed kojim se izvode pozivi ptica i u skladu s tim reproducirati ga na ploči.

Na primjer, na prvoj ploči (slika 6.16.) korisnik može prvo čuti poziv s niskim tonom, a zatim poziv s visokim tonom. Točan je odgovor stoga pritisnuti donji gumb na lijevoj strani ploče i zatim gornji gumb na desnoj strani.



Slika 6.16. Odgovor učenika prikazan na ploči u igri *Sound game*

Da bi odabrao određeni gumb, korisnik jednostavno mora usmjeriti pokazivač miša na njega i gumb će biti označen žutom bojom. Ako je pritisnuta tipka ispravna, tada će izgledati zeleno, u suprotnom će izgledati crno.

Coding4Girls okruženje igre za učenike mogu koristiti svi učenici, uglavnom u dobi od 10 do 15 godina, a cilj mu je potaknuti ravnopravno sudjelovanje u aktivnostima kodiranja unutar i izvan učionice. Prikazuje programske koncepte kroz 3D virtualno okruženje koje učenici mogu istraživati, mini-igre koje demonstriraju koncepte, okruženje za razmjenu ideja među učenicima u razredu i zajedničko traženje rješenja, pregled i međusobnu usporedbu rješenja drugih učenika i usporedbu vlastitih rješenja s onim koje je predstavio učitelj.

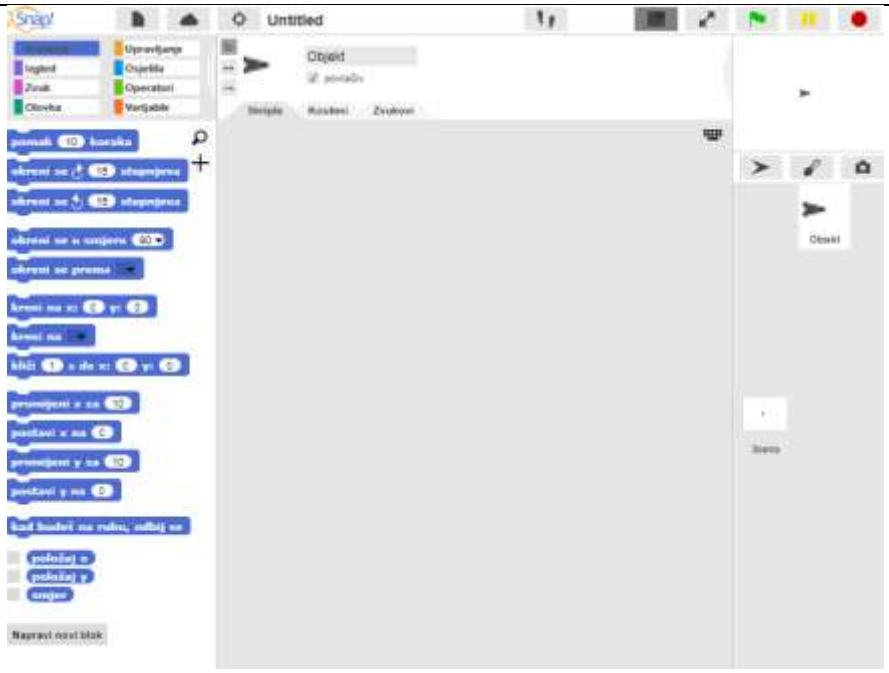
Budući razvoj i poboljšanje C4G softvera uključiti će proširivanje putem dodavanja novih mini-igara koje će ilustrirati dodatne koncepte programiranja, što će biti moguće učiniti s obzirom na modularnu prirodu obrazovne igre C4G i način na koji je dizajnirana u Unityju.

PRILOG 1. SCENARIJI UČENJA

OSNOVNI SCENARIJI UČENJA

Scenarij učenja 1 – Uvod u sučelje alata Snap!

Naziv scenarija	Uvod u sučelje alata Snap!
Potrebno predznanje iz programiranja	/
Ishodi učenja	Koncepti programiranja: • objekti i pozadine u vizualnom programskom okruženju Snap! Ishodi učenja: • Učenik će moći dodati novi objekt • Učenik će moći dodati kostime i urediti ih • Učenik će moći centrirati objekt, kako bi rotacija radila na ispravan način • Učenik će moći dodati različite pozadine na pozornicu i uređivati ih
Cilj, zadaci i kratki opis aktivnosti	Učenik dodaje nove objekte, dodaje kostime objektima, uređuje kostime te ih briše. Učenik stvara nove pozadine na pozornici, uređuje ju, te neželjene briše. **Cilj:** Do kraja sata učenici će nacrtati svoj omiljeni objekt i okruženje u kojem živi, stvarno ili imaginarno, kako bi ga mogli koristiti u igrici. Kako bi aktivnost bila motivirajuća za sve učenike, crtež objekta mora biti prikladan za ciljanu skupinu.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Demonstracija Samostalni rad
Oblici poučavanja	Frontalni rad Individualni rad
Razrada aktivnosti	Do kraja sata učenici će nacrtati svoj omiljeni objekt i okruženje u kojem živi, stvarno ili imaginarno, kako bi ga mogli koristiti u igrici. [Korak 1] Učenicima je potrebno navesti web stranicu na kojoj mogu pronaći program Snap! (https://snap.berkeley.edu/). Pokazuju se različiti dijelovi sučelja: odjeljak s blokovima, odjeljak u kojem mogu sastavljati skripte, mijenjati kostime, dodavati zvukove, pozornicu sa objektima, listu objekata.



[KORAK 2]
Možete kreirati novi objekt klikom na jedan od sljedeća tri gumba:

Pokušat ćete nacrtati novi objekt, stoga ako kliknete na kist, otvorit će vam se prozor u kojem možete nacrtati svoj objekt na sličan način kao i u Paint-u.

Zadatak za učenike: Nacrtajte svoj prvi objekt. Imate 10 minuta. Nakon što nacrtate navedeni objekt, trebali biste se uvjeriti da je centar rotacije objekta onakav kakav želite da bude. Kako biste to

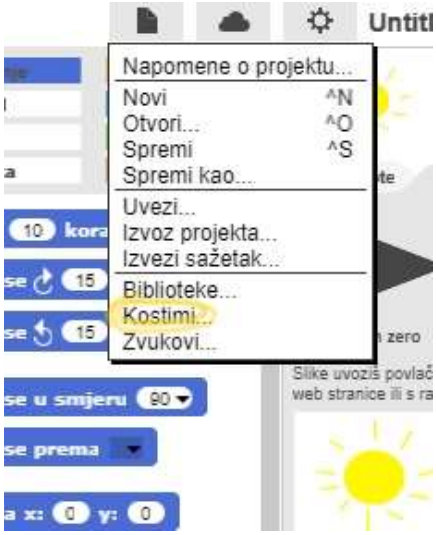
učinili koristite  .
Zadatak za učenike: Centrirajte svoj objekt.

[KORAK 3]

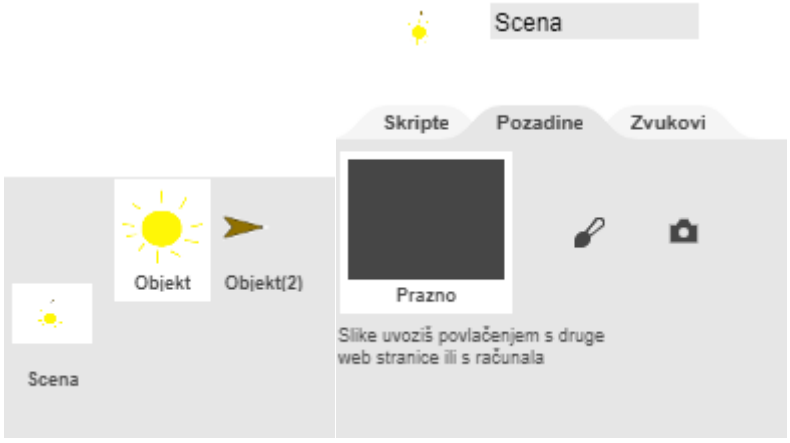
Kako biste uredili svoj objekt, odaberite karticu Kostimi koja je vidljiva samo kada je objekt označen. Desnom tipkom miša kliknite na željeni kostim i odaberite uredi. Također, u istom izborniku možete duplicirati kostim ili ga izbrisati.

[KORAK 4]

Da biste umetnuli već postojeći kostim, kliknite na ikonu na kojoj je nacrtan komadić papira i odaberite Kostimi…

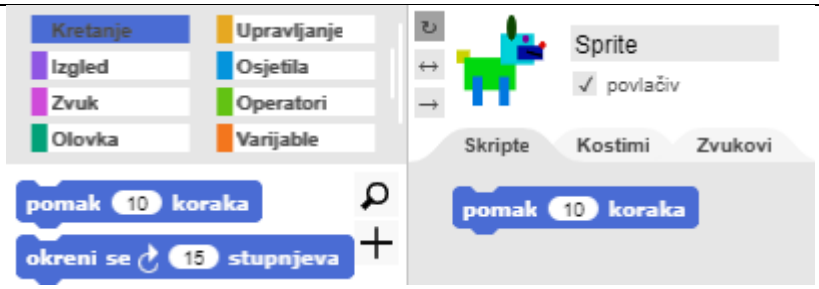


Ova opcija će se prikazati tek kada je vaš objekt označen na pozornici (sceni).

	Zadaci za učenike: odaberite kostim i dodajte ga objektu [KORAK 5] Sada kad imate svoj lik morate dodati neku pozadinu na scenu. Kako bi to napravili najprije trebamo kliknuti na Scena, a ne na objekt. Za dodavanje nove pozadine odabiremo karticu Kostimi:  Zadatak za učenike: Nacrtajte željenu pozadinu. Zadatak za učenike: Pretražite postojeće pozadine i dodajte jednu tako da ju uvezete, tako da imate dvije. Zadatak za učenike: Pronađite način na koji ćete urediti svoju pozadinu. Pronađite način brisanja pozadine tako da ostane samo jedna. Refleksija i evaluacija: Jesu li učenici uspijeli nacrtati svoj objekt i okruženje u kojem živi? Jesu li imali problema? Kako su ih riješili?
Alati i materijali za učitelje	Alat Snap!: https://snap.berkeley.edu/
Alati i materijali za učenike	/

Scenarij učenja 2 – Vrijeme je za oživljavanje vašeg objekta

Naziv scenarija	Vrijeme je za oživljavanje vašeg objekta
Potrebno predznanje iz programiranja	/
Ishodi učenja	Koncepti programiranja: • slijed • objekt (sprite) Ishodi učenja: • učenik će moći pronaći odgovarajuće programske blokove i povezati ih u niz • učenik će moći pomicati objekt • učenik će moći omogućiti da objekt nešto kaže
Cilj, zadaci i kratki opis aktivnosti	Učenik otkriva gdje su pohranjeni programski blokovi i kako pronaći odgovarajuće programske blokove, koje su kategorije blokova i kako se blokovi povezuju u niz.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Demonstracija Samostalni rad
Oblici poučavanja	Frontalni rad Individualni rad
Razrada aktivnosti	Učenicima se prezentira problem: Omogućit ćete da se vaš lik pomakne i nešto kaže tijekom ovog sata. Učenicima se može pokazati primjer programa koji će programirati na ovom satu. [Korak 1] Najprije pogledajmo gdje se nalaze programski blokovi koji su Vam dostupni za korištenje. Gdje se oni nalaze? Na lijevoj strani možete pronaći različite kategorije blokova: Kretanje, Izgled, Zvuk, Olovka, Upravljanje, Osjetila, Operatori i Varijable. Najprije ćemo koristiti  blokove. Zadatak za učenike: Najprije pronađite blok, a zatim dvaput kliknite na njega. Što se dogodilo? [Korak 2] Da biste započeli programirati, trebate povući i ispustiti  blokove u kartici Skripte.

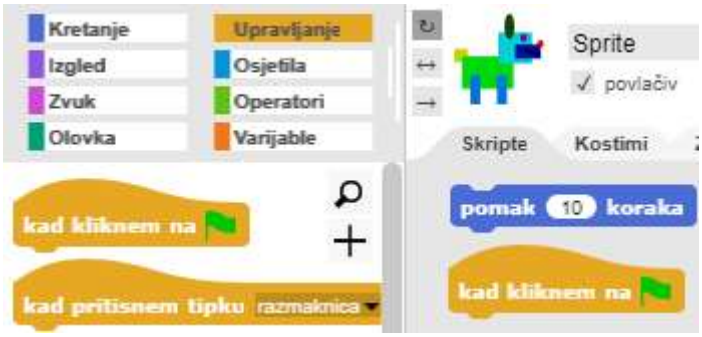


Dvostrukim klikom na blok u kartici Skripte omogućujete izvršavanje koda.

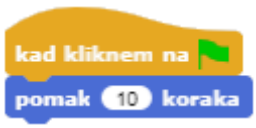
[Korak 3]
Programi u Snap!-u uobičajeno se pokreću klikom na zelenu zastavicu.

Zadatak za učenike: Klikom na različite kategorije blokova pokušajte pronaći blok koji omogućuje pokretanje programa ako se klikne zelena zastavica.

Rješenje:

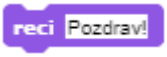
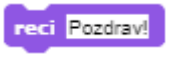


Ako želite da program radi u ispravnom slijedu koraka, blokovi moraju biti povezani kao kod slagalica (*puzzle*).
Na primjer ovako:

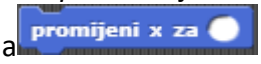


Sada će se, svaki put kada kliknete na zelenu zastavicu, objekt pomaknuti za 10 koraka, samo s drugog položaja na slici.

[Korak 4]
Ako se u bloku nalazi bijeli prostor, to znači da možete promijeniti brojeve ili slova koji su tamo upisani.
Zadatak za učenike: Omogućite da se vaš lik pomiče za 30 koraka odjednom, a umjesto samo 10.

	[Korak 5] Omogućite da Vaš lik nešto kaže. Gdje ćete potražiti blok s naredbom „reci“? Isprobajte koja je razlika između  i  te objasnite razliku susjedu u klupi. [Korak 6] Pronašli ste obje „reci“ naredbe u kategoriji Izgled. Glavna razlika je u tome da s  Vi ne kažete programu da pričeka __ sekundi prije nastavka provođenja koda ili da bi ga trebao prestati izgovarati u bilo kojem trenutku. [Korak 7] Uzmite svoj lik s prethodnog sata. Povlačenjem na pozornicu, pomaknite ga na lijevu stranu pozornice i napišite program koji omogućuje liku  s njegove na lijevoj strani na desnu stranu pozornice. Nakon svakog pomaka lik bi trebao nešto reći. Napravite više od samo jednog pomaka. Isprobajte. Završi li lik na točno istom mjestu svaki put kada se program pokrene? Možete li pronaći blok koji bi osigurao da Vaš lik uvijek krene s iste pozicije i da ne pobjegne s pozornice? Savjet za nastavnika: Ako lik pobjegne s pozornice, možete ga vratiti na pozornicu klikom na njega desnom tipkom miša i odabirom opcije „Pokaži“. Blok koji tražite je  . Da biste odredili koje vrijednosti x i y su poželjne, možete pomaknuti svoj lik na mjesto na kojem želite da bude i klikom na položaj x i položaj y (na dnu kategorije blokova Kretanje) će se pokazati trenutne vrijednosti x i y. Vi ih samo morate upisati u bijela polja u bloku. Razmišljanje i evaluacija: Koliko puta Vaš lik mora ponoviti pokrete i izgovoriti nešto kako bi izvršio zadatak? Je li taj broj isti za sve u razredu? Zašto je to tako?
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home
Alati i materijali za učenike	Ako učenik nije nacrtao svoj vlastiti objekt i pozadinu, može upotrijebiti: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dog_goes_home_tmp

Scenarij učenja 3 – Kretanje po pozornici

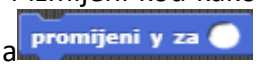
Naziv scenarija	Kretanje po pozornici
Potrebno predznanje iz programiranja	Dodavanje novih objekata i pozadine Mijenjanje kostima Povezivanje blokova
Ishodi učenja	Koncepti programiranja: ● Smisleno povezivati blokove Ishodi učenja: ● Učenik će moći postavljati objekt na pozornici ● Učenik će moći mijenjati *x* i *y* pozicije objekta ● Učenik će moći koristiti blok *ponavljaj* ● Učenik će koristiti naredbu *pomak _koraka* za definiranje kretanja objekta u smjeru u kojem je objekt okrenut
Cilj, zadaci i kratki opis aktivnosti	**Opis aktivnosti:** Učenik će izvoditi kretanje objekta po pozornici u smjeru *x* i *y*, napraviti jednostavan program za izvršenje dobivenog zadatka, te mijenjati smjer objekta i uočiti utjecaj istog na blok *pomak _ koraka.* **Zadaci:** Izraditi program u kojem će se objekt kretati u smjeru *x*, te izraditi program u kojem će se objekt kretati u smjeru *y.* **Cilj:** Učenik će razlikovati kretanja u smjeru *x* od kretanja u smjeru *y* te će koristiti petlju *ponavljaj*
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Demonstracija Individualni rad
Oblici poučavanja	Frontalni rad Individualni rad
Razrada aktivnosti	Učenicima se prezentira problem: Različitim životinjama je potrebno pomoći u ostvarenju njihovih ciljeva. Kako bi se to ostvarilo, treba ih uputiti kako se gibati po pozornici. [Korak 1] Otvori program *Uhvati loptu* i izmijeni kod kako bi pas uhvatio loptu. Koristi se naredbama  i  kako bi se pas pomicao prema lopti.

Moguće rješenje problema:

Kako bismo pokrenuli ovaj blok naredbi, potrebno je kliknuti na zastavicu označenu zelenom bojom . Kao što se može primijetiti, vrijednost *x* se mijenja, što daje za rezultat da se objekt pas kreće u smjeru *x*.

Kada bismo postavili početnu vrijednost *x* na 0, tada bi se objekt pas nalazio na sredini pozornice. Ukoliko je vrijednost *x* postavljena na vrijednost koja je manja od 0 (kao u ovom primjeru), onda se objekt pas nalazi u lijevoj polovici pozornice. Ukoliko je vrijednost *x* postavljena na vrijednost koja je veća od 0, onda se objekt pas nalazi u desnoj polovici pozornice. Uočite da je u tom slučaju pas bliži lopti.

Savjet:
Ukoliko je se radi o radu sa starijim učenicima kojima su unaprijed poznati brojevi u decimalnom zapisu, vrijeme čekanja se može postaviti na kraći period, npr. 0.1. Ukoliko su učenici upoznati s koordinatnim sustavom, objašnjenje istog se može izbjeći.

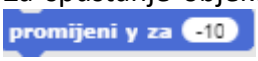
[Korak 2]
Otvori *Pomozi majmunu da se popne na drvo* i izmijeni kod kako bi majmun uhvatio banane. Koristi se blokovima 

i 

kako bi se majmun popeo na palmu.

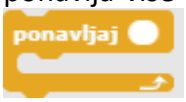
Moguće rješenje problema:

Kako bismo pokrenuli ovaj blok naredbi, potrebno je kliknuti na zastavicu označenu zelenom bojom  . Kao što se može primijetiti, vrijednost *y* se mijenja, što daje za rezultat da se objekt pas kreće u smjeru *y*.

Kada bismo postavili početnu vrijednost *y* na 0, tada bi se objekt pas nalazio na sredini pozornice. Ukoliko je vrijednost *y* postavljena na vrijednost koja je veća od 0, onda se objekt pas nalazi u gornjoj polovici pozornice. Kako bismo objekt smjestili ispod sredine pozornice, možemo zamisliti da se objekt nalazi na površini mora (*y=0*) , te da je potrebno zaroniti u more pri čemu se spuštamo u dubinu i time odmičemo od površine mora. Nakon spuštanja u dubinu mora, glavno pitanje je koliko metara smo ispod površine mora, drugim riječima: koliko koraka smo ispod sredine pozornice. Za spuštanje objekta majmuna sa palme potrebno je koristiti blok 

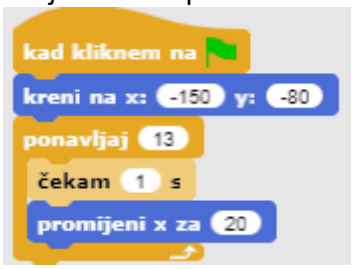
.

Savjet:
Ukoliko je rad sa starijim učenicima kojima su poznati brojevi u decimalnom zapisu, vrijeme čekanja se može postaviti na kraći period, npr. 0.1. Ukoliko su učenici upoznati s koordinatnim sustavom, objašnjenje istog se može izbjeći.

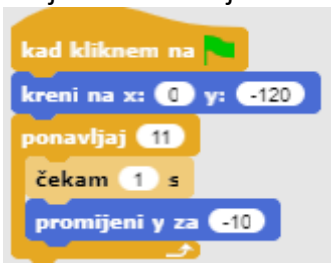
[Korak 3]
U oba zadatka koristila se kombinacija dvaju blokova. Koliko puta ste **ponavljali isti kod**?
Postoji kraći način zapisa istog koda tako da se kompjuteru kaže da isti kod ponovi određeni broj puta. Radi se o petlji *ponavljaj__*. Petlja *ponavljaj___* se može koristiti ako se jedna radnja ili niz radnji ponavlja više puta. U oba zadatka izmijeni kod koristeći se petljom 

. Dio koda koji želiš ponavljati je potrebno smjestiti unutar bloka *ponavljaj__,* i potrebno je zapisati broj ponavljanja u prazni dio bloka.

Linije koda za psa:

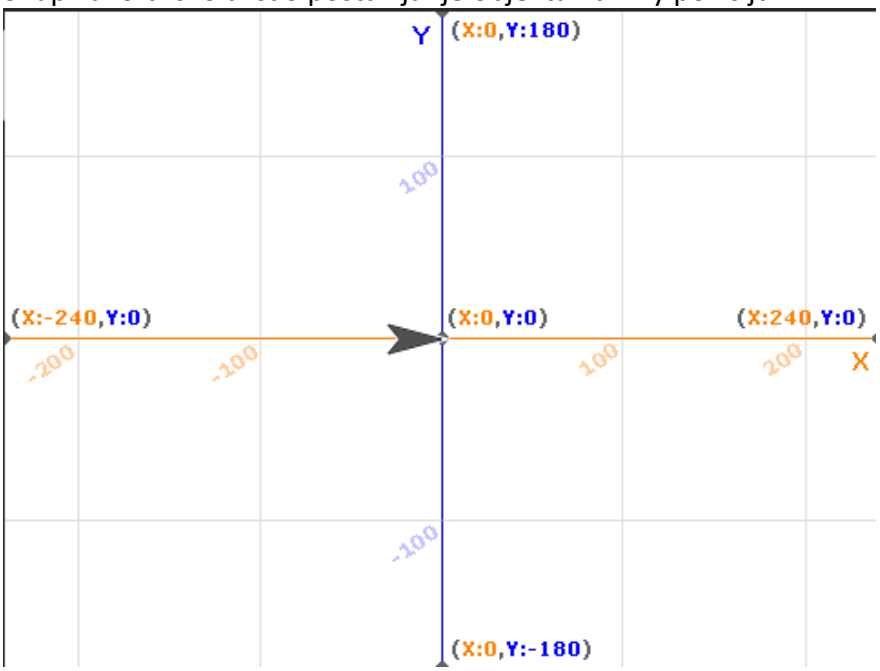


Linije koda za majmuna:

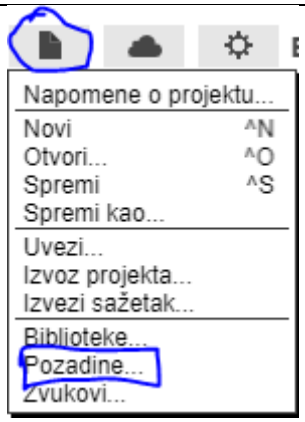


Zadatak: Pokušaj napraviti da pas trči k lopti i nazad.
Zadatak: Pokušaj napraviti da se majmun penje po stablu i zatim spušta.

Što ti se najviše dopalo? Koristi XY pozadinsku mrežu u programu Snap kako bi si olakšao postavljanje objekta na x i y poziciju:



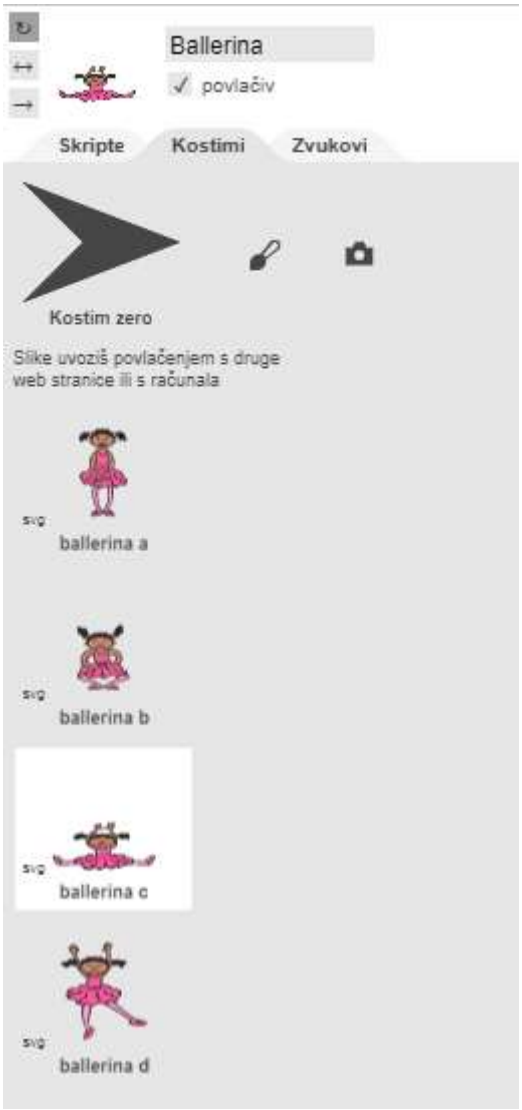
Za postavljanje XY pozadinske mreže, prvo se smjestite na pozornicu (ne na lika), kliknite na ikonu lista u lijevom gornjem kutu, odaberite *pozadine* i zatim izaberite prikladnu pozadinu pod nazivom XY grid.

	
Alati i materijali za učitelje	Moguće rješenje za *Uhvati loptu:* https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_x Moguće rješenje za *Pomozi majmunu popeti se na drvo:* https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_moving_y
Alati i materijali za učenike	*Uhvati loptu:* https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Catch_the_ball Pomozi majmunu popeti se na drvo: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Help_monkey_climb_the_tree

Scenarij učenja 4 – Mijenjanje kostima i okretanje

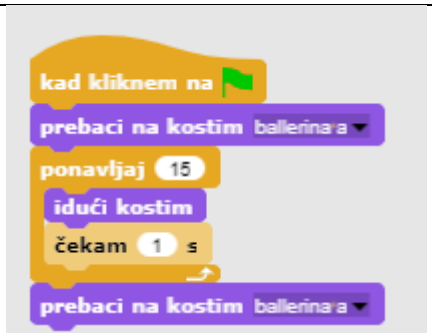
Naziv scenarija	Mijenjanje kostima i okretanje
Potrebno predznanje iz programiranja	Pomicanje
Ishodi učenja	Koncepti programiranja: ● Izrada smislenog slijeda blokova Ishodi učenja: ● Učenik će moći mijenjati kostim objekta kako bi napravio animaciju ● Učenik će moći mijenjati rotaciju likova
Cilj, zadaci i kratki opis aktivnosti	**OPIS AKTIVNOSTI:** Učenik uči kako promijeniti kostur objekta te kako napraviti animaciju. Također uči kako se između njih može mijenjati različite vrste rotacije objekta. **ZADACI:** Stvoriti program koji mijenja kostim objekta te u svakom programu postaviti odgovarajuću vrstu rotacije za svaki objekt. **CILJEVI:** Moći promijeniti kostur objekta i postaviti prikladnu vrstu rotacije objekta.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Game Based Learning – igra Razgovor Demonstracija Rješavanje problema Individualni zadatak
Oblici poučavanja	Frontalni rad Rad u paru Individualni rad Grupni rad (svi učenici)
Razrada aktivnosti	Učenici će naučiti kako napraviti animaciju objekta tako da izgleda kao hodanje, plesanje i slično. [KORAK 1] Otvorite novi prazan projekt, kliknite na ikonu koja izgleda kao bijeli komad papira i odaberite *Kostimi*... Kliknite na balerinu „A“ i zatim kliknite Uvoz (ili „Import“). Učinite isto s balerinom „B“, balerinom „C“ i balerinom „D“. Na kartici *Kostimi* vašeg objekta sada imate 4 kostima balerine. Možete preimenovati objekt u Balerinu tako da promijenite tekst iznad.

Kartica kostima:



Vratite se na karticu Skripta i pokušajte stvoriti kod koji će početi kad se klikne na zelenu zastavu i 15 puta se mijenja pojava balerine svake sekunde. Morat ćete koristiti blok 

.
Pazite da balerina započne i završi ples s obje noge na podu. Početni i krajnji položaj nisu dio njenog plesa.
RJEŠENJE:

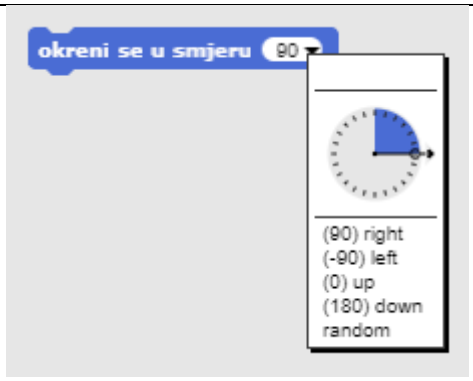


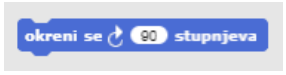
[KORAK 2]
Naša balerina ne želi cijelo vrijeme biti na istoj poziciji, pa radi male pokrete svaki put kad promijeni kostim. Dodajte ovaj pokret njenom plesu.
MOGUĆE RJEŠENJE:

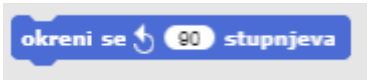
[KORAK 3]
Otvorite novi prazan projekt i uvezite “Avery walking”. Dodajte prigodnu pozadinu za Avery da krene dalje. Animirajte Avery tako da hoda s lijeve strane pozornice na desnu stranu pozornice. Pokušajte shvatiti kako animirati Avery tako da njezini koraci izgledaju povezani kao u stvarnom životu.
MOGUĆE RJEŠENJE:

[KORAK 4]
Do sada ste uvijek pisali program u kojem se objekt kretao samo u jednom smjeru. U ovom koraku morat ćete okrenuti miš kako bi došli do sira. Da bi ga okrenuli, možete odabrati sljedeće:

a) naredbu u kojem smjeru mora pogledati



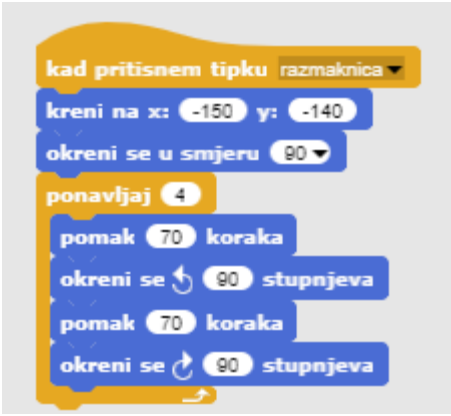
b) ili možete definirati da se okreće za određeni kut u smjeru kazaljke na satu 

ili u smjeru suprotnom od kazaljke na satu 

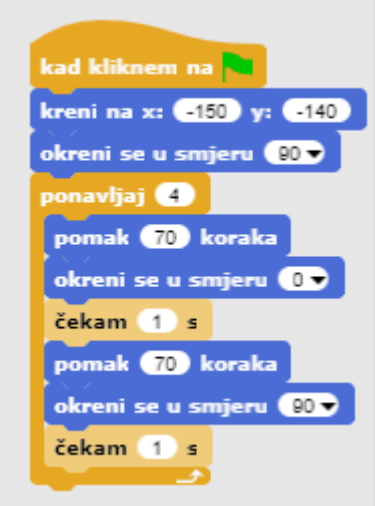
Puni krug ima 360 stupnjeva, pa ako se želite okrenuti suprotno od mjesta gdje trenutno stojite, okrenite se za 180 stupnjeva. Ako želite skrenuti lijevo, skrenite za 90 stupnjeva kazaljke na satu. Ako želite skrenuti s desne strane, skrenite 90 stupnjeva u smjeru kazaljke na satu.

Otvorite http://bit.do/fkEZc. Napišite naredbe koje miš mora slijediti kako bi došao do sira ako mora hodati samo po zelenoj površini. Usmjerite miša u smjeru u kojem se kreće i pomaknite __ koraka. Da bi vidjeli kako se miš kreće, upotrijebite naredbu "pričekajte 1 sekundu između redova".

RJEŠENJE:



Sada pokušajte napisati program s okretanjem za 90 stupnjeva.
RJEŠENJE:



[KORAK 5]
Kao što ste vidjeli, miš se okrenuo u različitim smjerovima kako bi došao do sira. Ponekad ne želite da se vaš objekt okrene naopako već želite da se samo okrene na lijevo ili desno tako da ne hoda po svojoj glavi. Da biste bili sigurni da se vaš objekt okreće onako kako želite, morate kliknuti na odgovarajuću ikonu lijevo od vašeg objekta:

Kružna strelica znači da se vaš objekt može okretati u bilo kojem smjeru (poput vašeg miša).

Strelica <-> znači da će se vaš objekt okrenuti samo ulijevo ili udesno (ovo biste koristili za psa da ne hoda po glavi).
Zadnja -> strelica znači da će objekt uvijek izgledati onako kako jest (ovo možete koristiti za majmune).
Pokušajte napraviti svoje programe za psa i majmuna po uzoru na ovaj, tako da hodaju do predmeta i okrenu se natrag. Pripazite na primjenu pravilnog rotacijskog stila.

Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_dancing *Avery hodanje*: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Avery_walking *Pronađi sir - rješenje:*

	https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese_solution
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_Find_cheese

Scenarij učenja 5 – Zvukovi s farme

Naziv scenarija	Zvukovi s farme
Potrebno predznanje iz programiranja	Postavljanje pozadine Dodavanje novog objekta Omogućiti da objekt nešto kaže
Ishodi učenja	Koncepti programiranja: ● Dodavanje zvuka iz Snap!-ove medijske biblioteke, ● Uključivanje zvuka iz drugih medija, ● Snimanje novog zvuka, ● Sviranje zvuka na pritisak tipke. Ishodi učenja: ● Učenik će moći dodati zvuk iz Snap!-ove medijske biblioteke i pustiti ga da svira pritiskom određene tipke na tipkovnici, ● Učenik će moći dodati zvuk sa računala i pustiti ga da svira pritiskom određene tipke na tipkovnici, ● Učenik će moći snimiti novi zvuk i pustiti ga da svira pritiskom određene tipke na tipkovnici.
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Programirati jednostavnu igru u kojoj igrač uči zvukove životinja pritiskanjem određenih tipki na tipkovnici. **Zadatak:** U prvom koraku učenik treba odabrati pozadinu scene. Zatim, učenik treba programirati farmericu koja govori upute: 1) Ako želiš čuti psa, pritisni tipku „D“!; 2) Ako želiš čuti svinju, pritisni tipku „P“!; 3) Ako želiš čuti kravu, pritisni tipku „C“!; 4) Ako želiš čuti ovcu, pritisni tipku „S“!; 5) Ako želiš čuti konja, pritisni tipku „H“!. Nakon toga, učenik treba programirati sviranje zvukova. **Cilj**: Učenici će se upoznati s načinom kako dodati novi zvuk i kako ga koristiti. Također će naučiti kako koristiti blok *Zvuk* („*odsviraj zvuk [naziv zvuka]*“) i blok *Upravljanje* („*kad pritisnem tipku [tipka*]“).
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	**Na primjer:** Game Based Learning – igra Aktivno učenje
Oblici poučavanja	Frontalni rad Individualni rad

Razrada aktivnosti	Motiviramo učenike igranjem igre (oni ne vide kod). Cilj je napraviti igru kao što je ova koju učenici igraju.  [Korak 1] Prvi korak je utvrđivanje pozadine igre. Pozadina treba sadržavati različite životinje. Postoje tri opcije: 1. učenici sami crtaju pozadinu; 2. učenici traže besplatnu sliku na mreži; 3. učitelj pripremi pozadinu (zbog uštede vremena). Učenici već znaju kako dodati pozadinu pa to rade samostalno.  [Korak 2] Drugi korak je dodati farmericu. Kao u prvom koraku, tri su opcije: 1. učenici sami crtaju farmericu; 2. učenici traže besplatnu sliku farmerice na mreži; 3. učitelj pripremi sliku farmerice učenicima (ako želimo uštedjeti vrijeme). Učenici već znaju kako dodati novi objekt pa to rade samostalno.

[Korak 3]

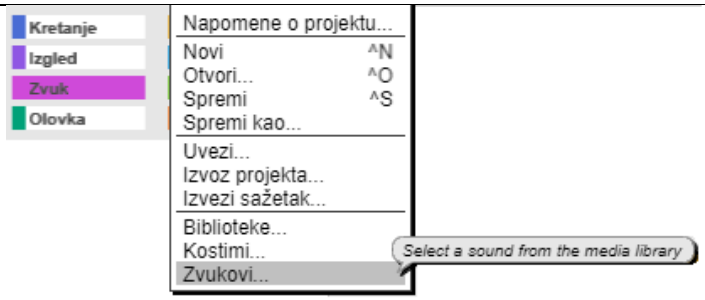
Nadalje, učenici trebaju programirati upute za igrača. Upute daje farmerica. Učenici to čine koristeći *Izgled/reci[znak*] i *čekam[n]* blok. Učenici već znaju kako to napraviti pa to izvršavaju samostalno.

U nastavku pokažemo učenicima kako dodati zvuk u igru. Imamo tri opcije:

1. Dodavanje zvuka iz Snap!-ove medijske biblioteke;
2. Dodavanje zvuka s našeg računala povlačenjem u Snap!;
3. Snimanjem novog zvuka u Snap!-u.

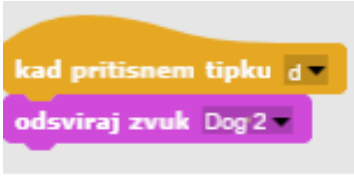
Pokažemo učenicima sve tri opcije frontalnim načinom rada. Kada ih sve demonstriramo, sljedeće zadatke učenici počinju programirati samostalno (uz pomoć učitelja).

[Korak 4]
Učenici trebaju programirati zvuk psa. Kada igrač pritisne tipku “D”, pas treba lajati. Prvo, učenici dodaju zvuk iz Snap! medijske biblioteke u pozadinsku karticu *Zvukovi*.



Zatim, biraju zvuk psa (Dog 1 ili Dog 2).

Učenici trebaju programirati zvuk psa koji će svirati kada je pritisnuta tipka "D". To rade koristeći *Upravljanje/kada pritisnem tipku [tipka]* blok i *Zvuk/odsviraj zvuk [ime_zvuka]* blok.



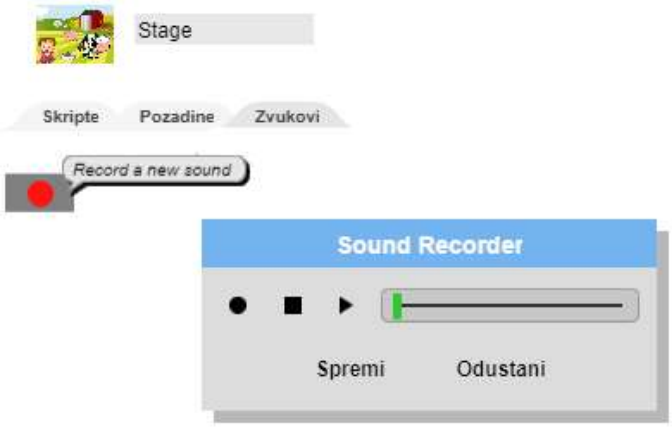
[Korak 5]
Učenici trebaju programirati zvuk životinja. Prvo, trebaju dodati zvukove sa svojih računala. To čine povlačenjem zvukova u pozadinsku karticu *Zvukovi*.

Jednom kad su dodani zvukovi, možemo ih preimenovati desnim klikom na pojedini zvuk. U našem slučaju zvukovi se zovu krava, prasac, konj, ovca i pas.

Nadalje, učenici trebaju dodati zvukove u pozadinsku karticu *Skripte*. To čine koristeći *Upravljanje/kad pritisnem tipku [tipka]* i *Zvukovi/odsviraj zvuk [naziv_zvuka]* blok.

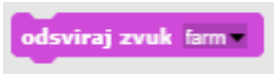
[Korak 6]
Idući korak je programirati farmeričin pozdrav dobrodošlice. Kada igrač započinje igru farmericu treba reći: “Dobro došli na moju formu”. Prvo, učenici trebaju snimiti farmeričin pozdrav. To će

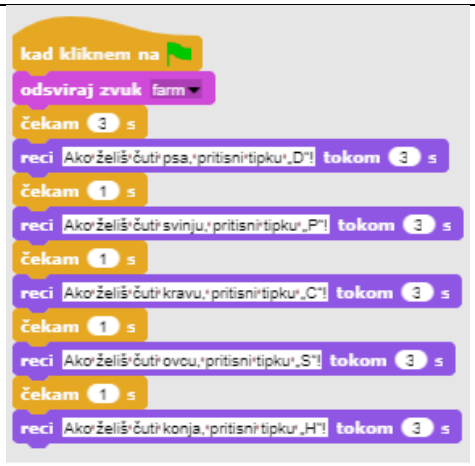
učiniti pomoću snimača zvuka (crveni gumb) koji se nalazi u kartici *Zvukovi*. Kada snime zvuk, trebaju ga spremiti (gumb Spremi).



Jednom kad je zvuk spremljen, možemo ga preimenovati desnim klikom na njega.

Sada učenici trebaju dodati zvuk u skriptu farmerice. To čine koristeći *Zvuk/odsviraj zvuk [naziv_zvuka]* blok.





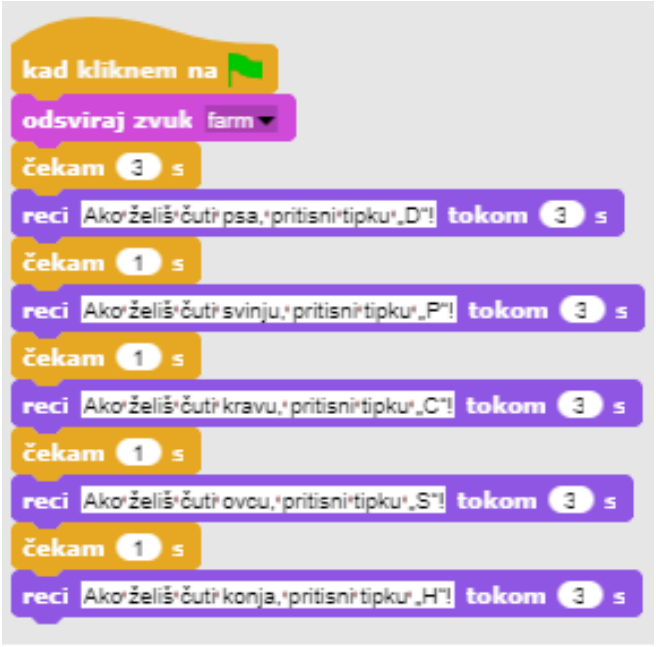
Refleksija i evaluacija

S učenicima se ponavlja:

- Kako su dodali zvukove u njihov kod.
- Koje su blokove koristili da bi dodali zvuk u kod.
- Koje upravljačke blokove su koristili u svom kodu.
- Zašto i kako su koristili blokove za zvuk i upravljačke blokove.

[Cijeli kod]

Farmerica

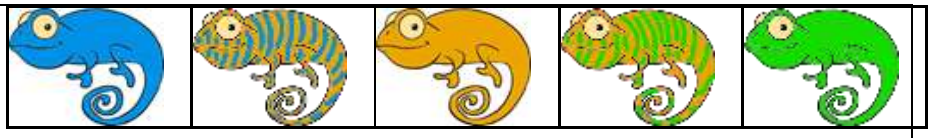


	Pozadina  **[Dodatni zadatak]** Učenici mogu doraditi igru tako što će dodati nove objekte (farmera, kokoš, traktor,…) i zvukove.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Farm Web-stranica za besplatno preuzimanje slika: https://pixabay.com/ Web-stranica za besplatno preuzimanje zvukova: https://www.zapsplat.com/ Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen u alatu Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Sounds%20of%20the%20farm_0 Web-stranica za besplatno preuzimanje slika: https://pixabay.com/ Web-stranica za besplatno preuzimanje zvukove: https://www.zapsplat.com/

Scenarij učenja 6 – Kameleonov ljetni odmor

Naziv scenarija	Kameleonov ljetni odmor
Potrebno predznanje iz programiranja	Dodavanje i uređivanje pozadine i lika Omogućiti da lik govori Korištenje naredbe *Ako ... onda ...* Kretanje objekta pomoću određenih tipki (npr. strelica)
Ishodi učenja	Koncepti programiranja: • kretanje objekta na temelju događaja • očitavanje jedne ili više boja • čitanje logičke vrijednosti u logičkim izrazima • definiranje, razlikovanje, dinamičko provjeravanje i reagiranje na različita stanja igre Ishodi učenja: • učenik će moći omogućiti kretanje objekta korištenjem tipki sa strelicama, koristeći događaje i uzima u obzir ograničenja, • učenik će moći koristiti blok osjetila u boji za dobivanje logičke vrijednosti pri očitavanju jedne ili više boja, • učenik će moći uočiti da se stanje objekta može mijenjati u ovisnosti boja koje dodiruje, • učenik će moći razlikovati pet različitih stanja i zna kako ih izraziti logičkim izrazima, • učenik će moći uočiti da se položaj objekta dinamički mijenja i koristi zauvijek petlju za ponovnu provjeru trenutnog stanja, • učenik će moći koristiti naredbe *Ako ... onda ...* za različite odgovore na temelju trenutnog položaja objekta.
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Programirajte jednostavnu igru u kojoj će objekt promijeniti svoj kostim na temelju boje pozadine. **Zadaci:** Učenici moraju programirati kameleona da promijeni svoj izgled (kostim) i reći gdje se on nalazi u pet različitih situacija: 1) kad pliva u moru, mora promijeniti boju u plavu i reći „Plivam u moru ", 2) kad se nalazi između mora i plaže, koža mu postaje pola plava, pola pješčane boje i kaže:" Nalazim se između mora i plaže ", 3) na plaži poprima pješčanu boju i kaže "Opuštam se na plaži", 4) između plaže i šume, poprima pola zeleno, pola pješčanu boju i kaže "Nalazim se između plaže i šume", 5) u šumi, koža mu postaje zelena i on kaže: „Hladim se u hladu drveća“.

	Cilj: Učenici će biti upoznati s blokom naredbi osjeta boja i kako ga koristiti u logičkim izrazima da bi razlikovali dinamično promjenjiva stanja igre i dali prave odgovore.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Suradničko učenje Rješavanje problema
Oblici poučavanja	Frontalni rad Rad u paru Individualni rad Grupni rad (svi učenici)
Razrada aktivnosti	Učenicima se prezentira problem: Kameleon je otišao na ljetni odmor. Voli se kupati u moru, uživati u opuštanju na plaži, a kad je previše vruće, voli ići u sklonište obližnjih stabala kako bi se rashladio. Budući da se radi o kameleonu, on treba mijenja boju u skladu s trenutnom pozadinom. [Korak 1] Učenici bi trebali urediti pozadinu scene tako da je podijeljena na tri jednaka dijela, a svaki predstavlja drugo mjesto: plava boja je za more, pješčana boja za plažu i zelena za šumu. Učenici mogu dodati druge stavke kako bi pozadinu učinili što realnijom poput valova, školjki, dvoraca s pijeskom, suncobrana, drveća itd., ali moraju biti oprezni da dodani predmeti nisu veći od samog glavnog lika, jer u tom slučaj kameleon neće dodirnuti nijednu od tri boje, a Snap-ova senzorna značajka neće moći prepoznati na kojem dijelu scene je lik.  [Korak 2] Učenici trebaju nacrtati kameleona i obojiti mu kožu u pet različitih kombinacija koje predstavljaju njegov položaj na sceni:



[Korak 3]
Prvo učenici trebaju napraviti da se njihov kameleon okreće u četiri smjera pomoću tipki. Mogu odabrati vlastitu kombinaciju tipki (npr. tipke sa strelicama ili tipke W, A, S i D). U ovom trenutku pretpostavljamo da znaju kako to učiniti. Potrebno je upozoriti učenike da ne zaborave da se taj lik može maknuti sa scene ako ne koristimo odgovarajući blok pri programiranju pokreta (kameleon se treba odbiti ako je na rubu bloka).
Da bi kretanje kameleona bilo malo realnije, želimo da skrene ulijevo ili udesno u vodoravnom smjeru u kojem se nalazimo (koristeći naredbu u bloku smjera).

[Korak 4]
Upoznajemo učenike s konceptom osjeta boje koju lik dodiruje. Pomoću naredbe "*Dodiruje?*" možemo dobiti informacije u obliku logičnih vrijednosti - istina ili laž ako dotakne jednu ili čak više boja u isto vrijeme. Budući da iz ovog bloka dobivamo logičku vrijednost, možemo je upotrijebiti u uvjetu *Ako ... onda ...*- naredbe, gdje se odlučuje hoće li se izvršavati naredbe navedene u njegovom tijelu ili ne.
Zatim razgovaramo s učenicima koji su različiti položaji kameleona na sceni i kako ih možemo izraziti blokom naredbi „*Dodiruje?*“.
Brzo saznajemo da ih ima pet:
1. Kameleon je u potpunosti na plavom dijelu -> *Dodiruje [plava]?*
2. Nalazi se između plavog i pješčanog dijela -> *Dodiruje [plava]?* i *Dodiruje [pijesak]?*
3. On je u cijelosti na pješčanom dijelu -> *Dodiruje [pijesak]?*
4. Nalazi se između pješčanog i zelenog dijela -> *Dodiruje [pijesak]?* I *Dodiruje [zeleno]?*
5. On je u cijelosti na zelenom dijelu -> *Dodiruje [zeleno]?*
Kad kameleon dodirne određenu boju / boje, moramo promijeniti njegov izgled, te moramo napraviti da kaže gdje se nalazi. Izgled kameleona možemo promijeniti zamjenom njegovih kostima. To se postiže blokom *Izgled / prebaci na kostim [opcija]* gdje odabiremo koji od mogućih kostima želimo prikazati. Da bi kameleon govorio koristimo blok *Izgled / reci [tekst].*

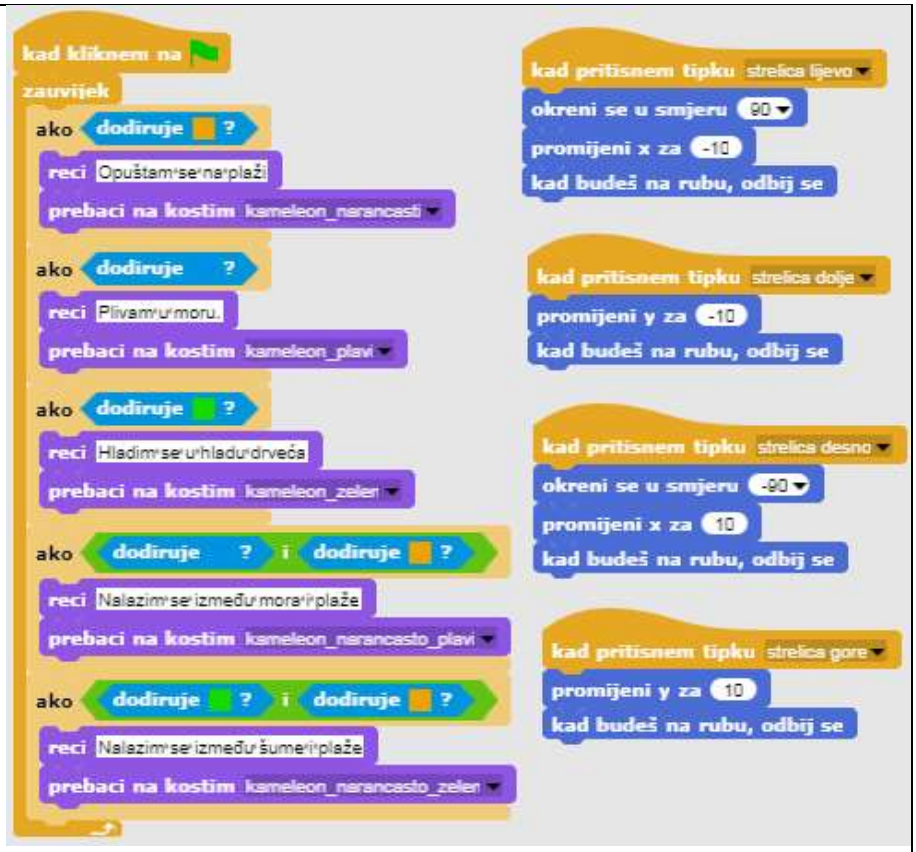
Prvo se pobrinemo za jednostavnije situacije kada je kameleon u potpunosti na istoj boji u sceni:

Dalje oblikujemo logički izraz upotrebom logičkog operatera *I*, jer želimo provjeriti dodiruje li kameleon dvije boje istovremeno:

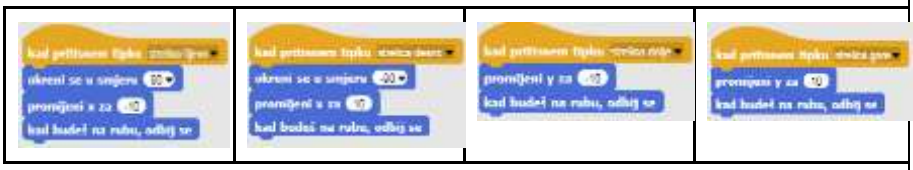
Ako kombiniramo gore navedene uvjetne rečenice i stavimo ih pod *„Kad kliknem na Zelena zastava“* naredbu, primijetit ćemo da će se ovi uvjeti provjeriti točno jednom. Tada učenicima pomažemo uočiti da zbog toga što pomičemo glavnog lika po sceni, kameleonova pozicija će se mijenjati tijekom cijele igre. Zbog toga moramo stalno provjeravati te uvjete, ne samo jednom, nego doslovno cijelo vrijeme!

[Korak 5]
Za situacije kada moramo izvršavati određene naredbe tijekom cjelokupnog trajanja izvršavanja programa koristimo *zauvijek* petlju. Sve napisano ispod tijela *zauvijek* petlje izvršit će se iznova i iznova. S učenicima razgovaramo o tome da je u našem slučaju to upravo ono što želimo / trebamo kako bismo stvorili ovu igru.

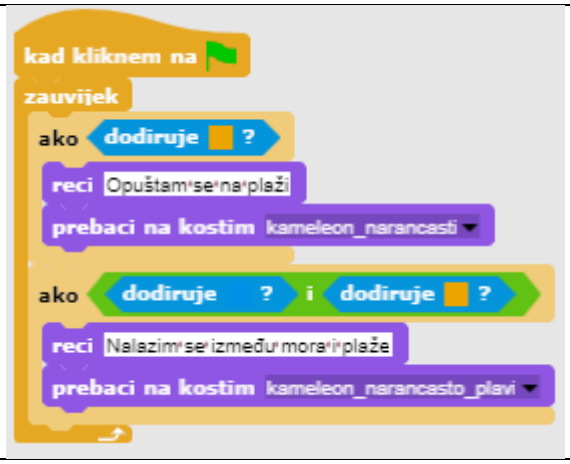
[Konačni kod]



[Prilagođavanje koda učenicima]
Da bismo pojednostavili ovu aktivnost, prethodno možemo pripremiti dio koda u datoteci predloška i uputiti učenike da ga dovrše.
Učenici koji su slijedili predloženi put učenja već su naučili kako pomicati objekt s tipkama. Dakle, možemo uključiti kôd kretanja u datoteku predloška. Učenici mogu promijeniti postavke tipki s tipki sa strelicama u neki vlastit raspored (npr. W, A, S, D).

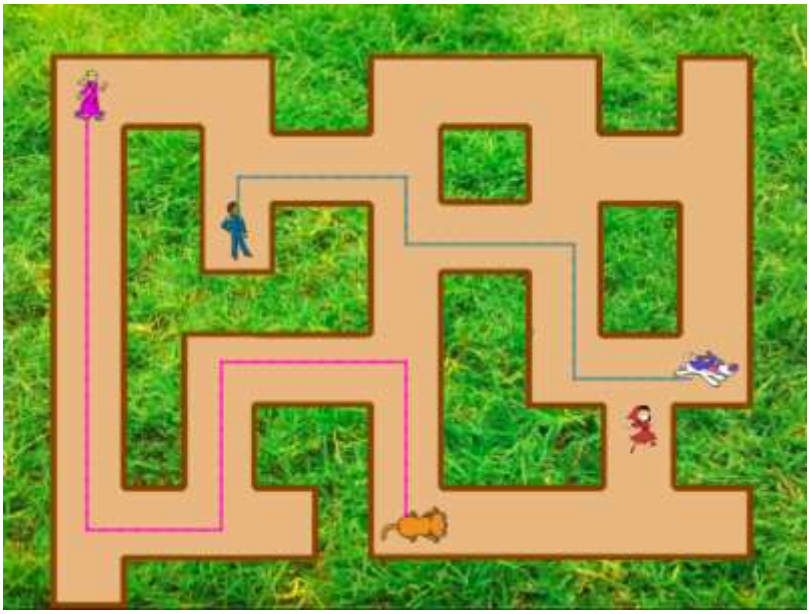


Da bismo pomogli učenicima da razumiju koncept petlje *zauvijek* i kako je koristiti za otkrivanje boje pozadine, možemo uključiti kôd za otkrivanje dvije situacije: 1) objekt je u potpunosti u jednoj boji, 2) objekt dodiruje dvije boje istovremeno. Potrebno je uputili učenike da popunjavaju kod za svaki od dva moguća slučaja.
Predloženi predložak koda:

	
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon Lajović, S. (2011). Ogrebotina. Naučite kako programirati i postati računalna mačka. Ljubljana: Pasadena. Worderman, C. (2017). Računalo programiranje za djecu. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen u alatu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_template Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=zapusek&project=chameleon_half_baked

Scenarij učenja 7 – Pomaganje princu i princezi u pronalasku njihovih životinja

Naziv scenarija	Pomaganje princu i princezi u pronalasku njihovih životinja
Potrebno predznanje iz programiranja	Dodavanje teksta objektu Kretanje objekta pomoću strelica koristeći događaje Korištenje uvjeta za *objekt dodiruje* za stanje objekta Korištenje događaja
Ishodi učenja	Koncepti programiranja: ● Uvjeti ako *objekt dodiruje* određenu boju ● Kretanje po koordinatama ● *Olovku digni*, *olovku pritisni* ● Boja olovke Ishodi učenja: ● Učenik će moći koristiti naredbu *ako* za utvrđivanje stanja objekta i za vraćanje objekta natrag u slučaju da dotakne određenu boju ● Učenik će moći postaviti objektu početne x i y koordinate ● Učenik će moći koristiti olovku digni i olovku pritisni za crtanje linije/puta ● Učenik će moći mijenjati boju olovke ovisno o paru kojeg spaja ● Učenik će biti sposoban uvidjeti da na početku mora očistiti sve prethodno nacrtane putove
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Djevojčica mora pomoći princezi pronaći njezinu mačku i princu pronaći njegovog psa. To radi na način da otiđe do princeze i pokaže joj, crtanjem linije, put do njezine mačke; na sličan način djevojčica pokazuje princu put do njegovog psa. Na tom putu djevojčica mora izbjegavati susret životinja kako se putovi do njih ne bi preklapali. **Zadatak:** U prvom koraku učenici moraju odabrati prikladnu pozadinsku sliku (labirint). Dodaju u labirint pet objekata – vlastiti objekt (djevojčicu), princezu, princa, mačku i psa. Nadalje, programiraju kretanje djevojčice pomoću tipki (koristeći događaje), gdje moraju paziti da objekt ne hoda po travi. Nakon toga, programiraju crtanje olovkom i mijenjanje boje olovke koristeći događaje. Također, moraju isprogramirati početni događaj u kojem se raščisti put i djevojčica daje upute za početak igre. **Cilj:** Učenici će se upoznati s crtanjem pokretom tipke. Uz to će naučiti kako koristiti uvjete kojima će spriječiti da se objekt kreće po

	cijelome ekranu.
Trajanje	30 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Game-design Based Learning – igra Rješavanje problema
Oblici poučavanja	Frontalni rad Individualni rad
Razrada aktivnosti	Učenicima je potrebno dati sljedeće materijale: • sliku pozadine • Snap! projekt u kojem je definiran objekt djevojčice • kôd za kretanje u jednom smjeru Učenicima se prezentira problem: Djevojčica odluči pomoći princezi pronaći njezinu mačku i princu pronaći njegovog psa pokazivanjem (crtanjem) puta do njihovih životinja. Kako ne bi došlo do zabune, putovi trebaju biti različitih boja i ne smiju se sijeći.  [Korak 1] Učenici trebaju urediti pozadinsku sliku – labirint. Za uvođenje uvjeta “ako dodiruje boju” ili pozadina (trava) mora biti jednobojna ili put mora imati jednobojni okvir, kao u našem slučaju. Kako bi izbjegli navedene “probleme” s pronalaženjem prikladne pozadinske slike, dajemo im gore priloženu pozadinu. [Korak 2] Učenici na početku već imaju objekt djevojčice. Trebaju pronaći još

četiri objekta i smjestiti ih u labirint. Za sve objekte trebaju namjestiti prikladne dimenzije (koje trebaju biti manje od širine putova u labirintu).

[Korak 3]
Nakon toga moraju osmisliti kretanje djevojčice u četiri smjera korištenjem tipki. Pretpostavimo da to znaju napraviti iz prethodnih aktivnosti. Ipak, dajemo im kôd za jedan smjer koji im služi kao pomoć za izradu kôda za preostala tri smjera.

Kako bi kretanje djevojčice bilo što realnije (da se okreće lijevo i desno) kliknemo na *gledaj samo lijevo-desno* u bloku za smjer.

[Korak 4]
U idućem koraku učenici moraju spriječiti kretanje djevojčice po livadi. To rade tako da dodaju blok s uvjetom ako dodiruje smeđu boju. Ako djevojčica dodiruje smeđu boju (kraj puta), pomiče se 10 koraka unatrag. Ti koraci nam nisu vidljivi i čini se da djevojčica ostaje na istoj poziciji.

Učenici dodaju ovaj kôd ispod gore navedenog kôda za svaku strelicu, npr.:

[Korak 5]
Nadalje, učenici programiraju crtanje s blokovima *olovku digni* i *olovku pritisni* korištenjem događaja *kad pritisnem tipku.*

Kada je tipka "D" pritisnuta i djevojčica se kreće, ona crta liniju. Kada je tipka "E" pritisnuta, crtanje prestaje.

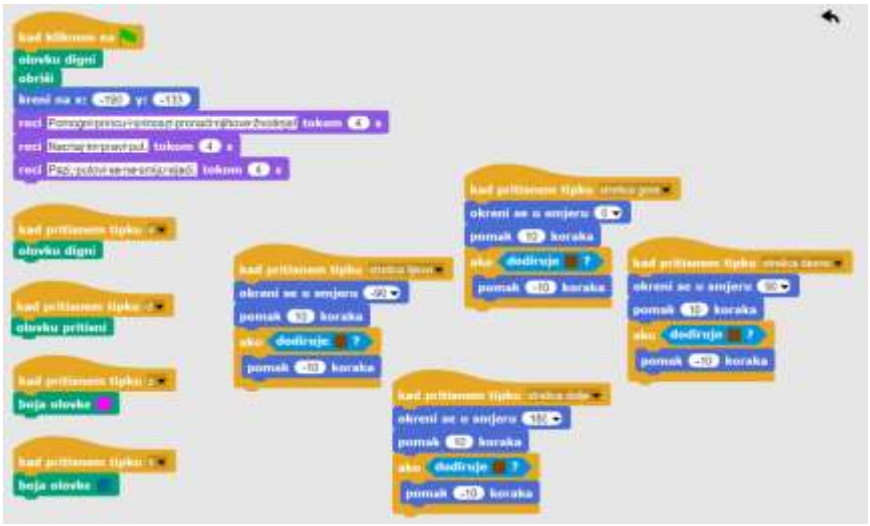
Na sličan način, učenici postavljaju boju olovke pritiskom tipke.

[Korak 6]
Na kraju, učenici programiraju događaj *kad kliknem na zelenu zastavicu*, gdje dodaju upute koje djevojčica na početku kaže. Nakon što odigraju igru nekoliko puta, učenici će vidjeti da je korisno dodati sljedeće blokove: *olovku digni* (za slučaj da ostane spuštena nakon prethodne igre), *obriši* (briše sve nacrtane putove iz prethodnih igara) i *kreni na x, y* (djevojčica uvijek započinje na tim koordinatama, koje su na putu i nisu na travi).

Kako bi odredili početne koordinate djevojčice, kliknemo na nju mišem i odvučemo ju na mjesto s kojeg želimo da započne igru. Nakon toga kliknemo na blok *Kretanje,* gdje možemo pronaći *položaj x* i *položaj y*. Klikom na *položaj x* saznajemo x koordinatu djevojčice, dok klikom na *položaj y* saznajemo y koordinatu.

<table>
<tr>
<td></td>
<td>


[Cijeli kôd]



[Dodatni zadatak]
Učenici mogu upotpuniti igru dodatnim zadacima po želji ili koristiti sljedeće prijedloge:

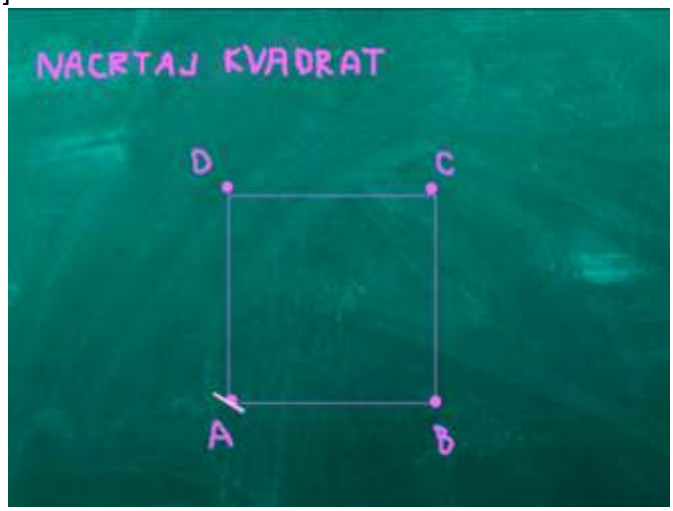
Postaviti početne coordinate za princa i princezu i napisati kod za njihovo kretanje. Postaviti odgovarajuću veličinu objekata. Crtati stazu do svojih životinja.
Uključiti dodatne objekte.
Svaki object treba crtati drugačijom bojom.
Prilagoditi početne upute.
Dodati upute igraču za pomicanje objekata (npr. tipkama W, S, A and D) i crtanje staze (npr. tipkom 3).

</td>
</tr>
<tr>
<td>Alati i materijali za učitelje</td>
<td>Igra izrađena alatom Snap!:
https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals
Primjer igre s dodatnim zadacima:
https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals%20%2B%20Add.%20Task
Lajovic, S. (2011). Scratch. Nauči se programirati in postani</td>
</tr>
</table>

	računalniški maček. Ljubljana: Pasadena. Vorderman, C. (2017). Računalniško programiranje za otroke. Ljubljana: MK.
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Helping%20Prince%20and%20Princess%20to%20find%20their%20animals%20-%20Part

Scenarij učenja 8 – Crtanje s kredom

Naziv scenarija	Crtanje s kredom
Potrebno predznanje iz programiranja	Dodavanje teksta objektu (sprite-u) Crtanje s olovkom (olovka gore, olovka dolje, postavljanje boje) Kretanje s koracima Petlje Događaji
Ishodi učenja	Koncepti programiranja: ● Petlja ponavljaj ● Okretanje za 90 stupnjeva ● Okretanje u smjeru ● Mijenjanje pozadine Ishodi učenja: ● Učenik će koristiti petlju *ponavljaj* kad se isti blokovi ponavljaju 2-4 puta ● Učenik će koristiti okretanje za 90 stupnjeva kod crtanja različitih oblika (kvadrat, pravokutnik, slovo „T“) ● Učenik će objasniti značenje *okreni se u smjeru 90* ● Učenik će moći promijeniti pozadinu pomoću događaja *kad pritisnem tipku*
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis**: Igrač dobije tri različite pozadine i mora spojiti točke u tri različita oblika – kvadrat, pravokutnik i slovo „T“. **Zadatak:** Učenici odabiru pozadinu „pločaKvadrat“ i počinju crtati kvadrat. Početna pozicija je točka „A“. Kod crtanja kvadrata, ponavljaju određene korake 4 puta, stoga umjesto da pišu kôd 4 puta, mogu koristiti petlju *ponavljaj 4.* Nakon toga crtaju pravokutnik, također koristeći petlju *ponavljaj*. U ovom slučaju *ponavljaj 2*. U zadnjem zadatku trebaju spojiti točke u obliku slova „T“, te trebaju pronaći broj potrebnih koraka. Mogu koristiti petlju *ponavljaj* tako gdje je to moguće. **Cilj:** Učenici će se upoznati s crtanjem različitih oblika pomoću kôda. Naučit će koristiti petlju *ponavljaj* kako bi smanjili veličinu kôda i promijeniti pozadinu.
Trajanje	60 min
Strategija i metode učenja i poučavanja	Aktivno učenje Game Based Learning – igra Rješavanje problema

Oblici poučavanja	Frontalni rad Individualni rad /Rad u paru
Razrada aktivnosti	Za učenike su unaprijed pripremljene: ● tri pozadine sa svim točkama koje trebaju spojiti ● objekt „kreda“ Učenicima se prezentira problem: Kreda želi nacrtati kvadrat, pravokutnik i spojiti točke u obliku slova „T“ ali ne zna kako se kretati i kako se okrenuti. Napiši kôd i pokaži kredi kako da to radi! [Korak 1]  Učenici počinju s ovom pozadinom. Pišu kôd za crtanje kvadrata. Počevši od točke „A“, kreće se X koraka do točke „B“, okrene za 90 stupnjeva lijevo, kreće X koraka do točke „C“, okrene za 90 stupnjeva lijevo, kreće X koraka do točke „D“, okrene za 90 stupnjeva lijevo, kreće X koraka do točke „A“ (i okrene za 90 stupnjeva lijevo).

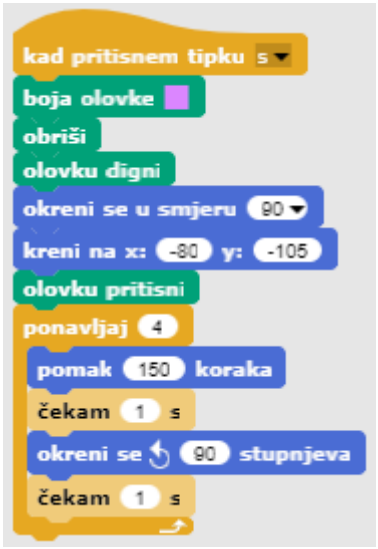
Lakše je koristiti *okreni se 90 stupnjeva* budući da uvijek možemo koristiti okretanje za 90 stupnjeva (ovisi samo da li želimo da se okrene lijevi ili desno). Drugi način je koristiti *okreni se u smjeru 0, 90, 180, -90*, ali je malo kompliciranije jer moramo razdvojiti 4 mogućnosti i ne možemo koristiti petlju *ponavljaj.* Blok *čekaj 1 s* je dodan zbog toga da vidimo crtanje (sve korake). Bez tog bloka, cijeli kôd se izvrši u sekundi. Učenici bi trebali isprobati napisati kôd bez tog bloka kako bi razumjeli značenje.

Pitamo učenike kako bi skratili kôd, ako je to moguće. Postoji li dio koji se može ponoviti? Odgovor je da postoji. Umjesto da piše isti kôd 4 puta, u programiranju koristiti petlju *ponavljaj.*

Ako želimo vidjeti crtanje, moramo staviti blok *olovku pritisni* ispred petlje *ponavljaj.*

Ako ne želimo da se kreda rotira kod okretanja, u bloku za smjer odaberemo *ne rotiraj.*

[Korak 2]
Za pokretanje kôda, učenici koriste blok za događaj npr. *Kad pritisnem tipku s.* Također mogu postaviti *boju olovke*, i, kao što već znaju iz prethodnih aktivnosti, sljedeće blokove: *olovku digni* (u slučaju da je ostala dolje od prethodnog igranja), *obriši* (briše crtež prethodnog igranja) i *kreni na x,y* (da kreda uvijek krene s te pozicije). Ponekad se dogodi da zaustavimo program tijekom reprodukcije i da je objekt rotiran u „neobičnom smjeru“. To je problem kod ponovnog pokretanja igre. Ako je objekt rotiran pogrešno, ići će na primjer dolje, a ne desno u prvom koraku. Kako bi izbjegli taj problem, dodajemo blok *okreni se u smjeru 90*.

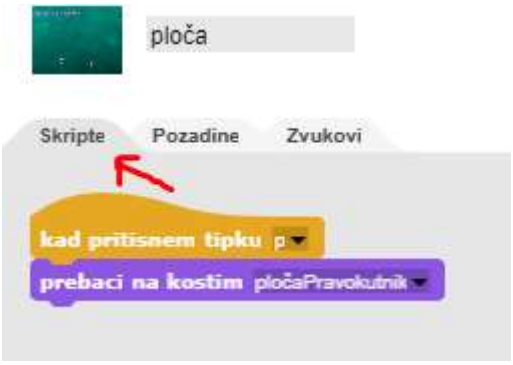


[Korak 3]
Nakon crtanja kvadrata, želimo nacrtati pravokutnik. Znači da moramo promijeniti pozadinu. To ćemo učiniti u dva koraka:

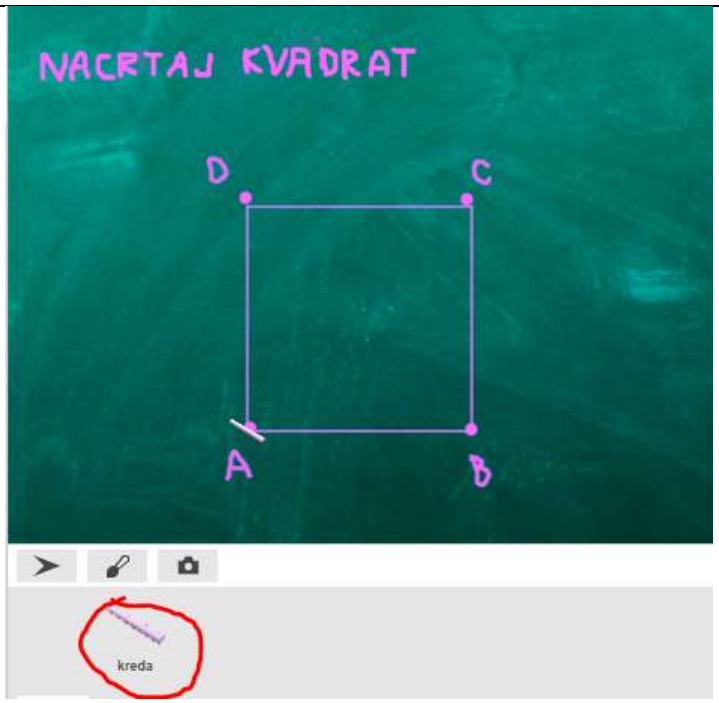
a) Kliknemo na pozadinu (nazvanu *ploča,* s desne strane ekrana)

Klikom na *Pozadine* možemo vidjeti sve tri potrebne pozadine (pločaKvadrat, pločaPravokutnik, pločaT) spremne za ovu aktivnost.

Za pisanje kôda, učenici trebaju kliknuti na gumb *Skripte.* Da bi program izmijenio pozadinu koristimo blok s događajem *kad pritisnem tipku p* i *prebaci na kostim pločaPravokutnik.*

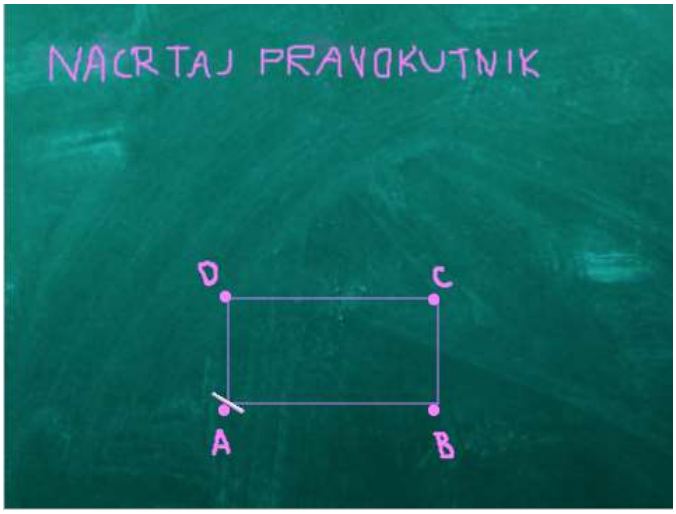


b) Kliknemo natrag na kredu.



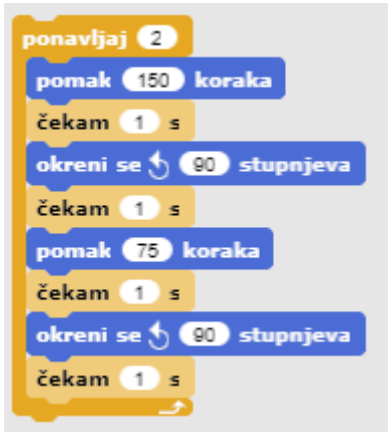
Ispod kôda iz [Korak 2] učenici dodaju blok, u kojem kažu igraču što da učini kako bi se pozadina promijenila, tj. da pritisne tipku „P“.

[Korak 4]



Nakon pritiska na tipku „P“, pozadina se mijenja u ovu na slici. Slično kao i prije, treba spojiti točke i nacrtati pravokutnik. Učenici mogu kopirati kôd za kvadrat i izmijeniti ga kako bi program nacrtao pravokutnik. Trebaju promijeniti petlju *ponavljaj.* U ovom slučaju se

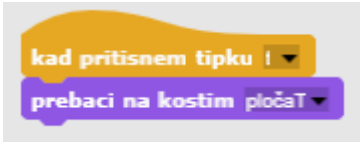
petlja ponavlja 2 puta.



[Korak 5]

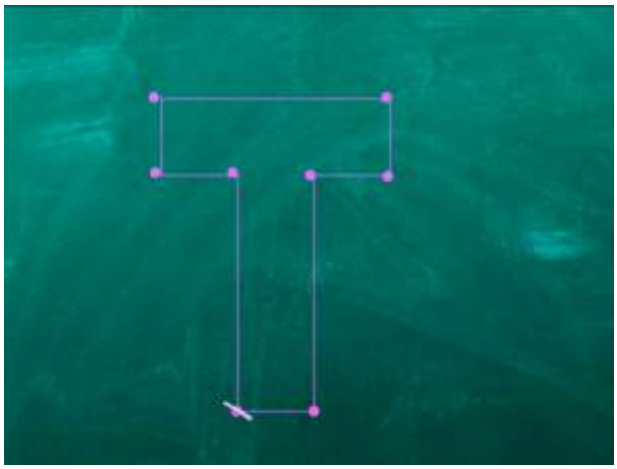
Nakon crtanja pravokutnika, učenici će spojiti točke u oblik slova „T“. To znači da će promijeniti pozadinu. U ovom koraku zapravo ponavljaju [Korak 3] i mijenjaju slovo „P“ u slovo „T“ i stavljaju kostim (PločaT).

a) Kliknu na pozadinu (naziva *ploča*, s desne strane zaslona), gdje napišu kôd za promjenu pozadine. To će učiniti sa blokovima *kad pritisnem tipku t* i *prebaci na kostim pločaT.*



b) Kliknu natrag na kredu i ispod kôda iz [Korak 4] dodaju blok, u kojem kažu igraču što da učini kako bi se promijenila pozadina tj. da pritisne tipku „T“.

[Korak 6]

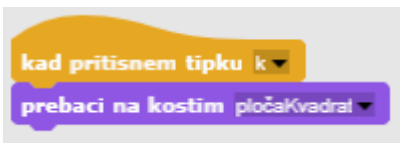


Nakon pritiskanja tipke „T“, pozadina se promijeni u ovu sliku. Slično kao i prije, trebaju spojiti točke i nacrtati slovo „T“. Učenici mogu

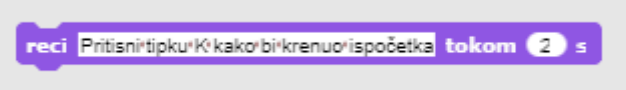
kopirati kôd za kvadrat i izmijeniti ga. Učenici trebaju promijeniti početne koordinate, koje nisu jednake kao i prije. Iz prethodnih aktivnosti znaju kako odrediti točne koordinate. Nakon toga napišu kôd za crtanje slova „T“. Učenici trebaju odrediti broj koraka. Jedno od mogućih rješenja je:

[Korak 7]
Nakon što smo promijenili pozadinu, ne možemo se vratiti na prvu pozadinu da bi nacrtali kvadrat. Stoga učenici trebaju dodati još jedan dio kôda. Ponavljaju [Korak 3/5].

a) Kliknu na pozadinu (naziva *ploča*, na desnoj strani zaslona), gdje napišu kôd za promjenu pozadine. To će učiniti s blokovima *kad pritisnem tipku k* i *prebaci na kostim pločaKvadrat.*



b) Kliknu natrag na kredu i ispod kôda iz [Korak 6] dodaju blok, u kojem kažu igraču što treba učiniti kako bi se promijenila pozadina, tj. da pritisne tipku „K“.



[Konačan kôd]

[Dodatni zadatak]

Učenici mogu uključiti u igru dodatne zadatke po želji ili koristeći sljedeće prijedloge:

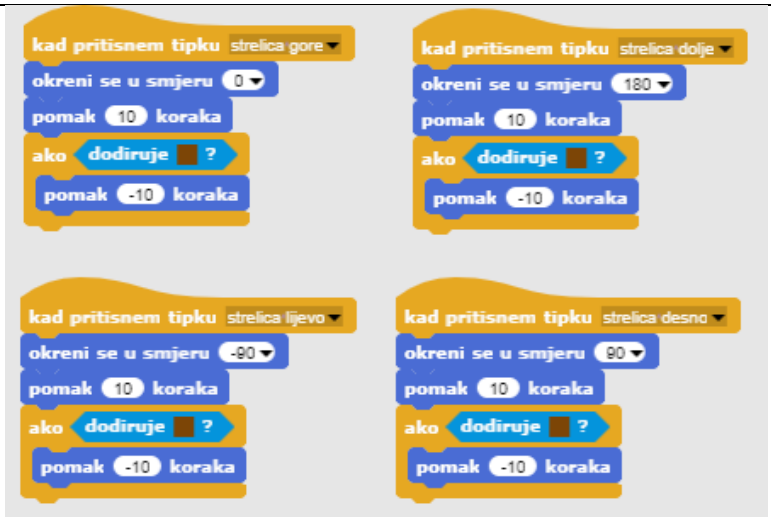
- Dodati novu pozadinu i nacrtati neke točke.

	● Napisati kod kojim se povezuje točke (koristeći novu ili postojeću pozadinu).
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Drawing%20with%20a%20chalk%20-%20Part

Scenarij učenja 9 – Skupljanje otpadaka i čišćenje parka

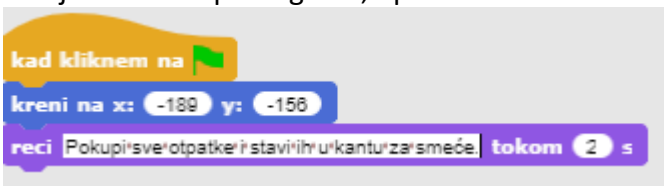
Naziv scenarija	Skupljanje otpadaka i čišćenje parka
Potrebno predznanje iz programiranja	Postavljanje početnih koordinata Postavljanje veličine objektu (sprite-a) Dodavanje teksta objektu Kretanje objekta pomoću strelica koristeći događaje
Ishodi učenja	Koncepti programiranja: ● Varijable ● Prikazivanje i sakrivanje objekta ● Kopiranje objekta ● Kopiranje bloka kôda ● Uvjeti Ishodi učenja: ● Učenik će koristiti varijable za brojanje pokupljenih otpadaka ● Učenik koristiti sakrivanje objekta kada se on dotakne i prikazuje objekta na početku ● Učenik može kopirati objekta (npr. od jedne boce napraviti 4 boce) ● Učenik može kopirati blok kôda (iz objekta boce u objekt papira) ● Učenik može koristiti uvjete za provjeru je li objekt prikazan i jesu li svi otpaci pokupljeni
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Park je pun otpadaka i djevojčica je odlučila počistiti ga. Kada pokupi sve otpatke, mora ih baciti u kantu za smeće. **Zadatak:** Učenici započinju s postavljanjem početnih koordinata lika djevojčice. Igra završava kada djevojčica pokupi sve otpatke i baci ih u kantu za smeće. Kako bi to napravili, učenici moraju koristiti varijable za prebrojavanje bodova (1 pokupljeni otpadak = 1 bod). Kada djevojčica dotakne otpadak, pokupi ga, otpadak se sakriva i broj bodova se poveća za 1. Kada djevojčica pokupi sve otpatke, odlazi do kante za smeće. Ako ode do kante za smeće prije nego pokupi sve otpatke, dobiva od kante poruku da se vrati kada pokupi sve otpatke. **Cilj:** Učenici će naučiti kako koristiti varijable i kako kopirati blok kôda ili čak cijeli objekt.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Rješavanje problema

	Game Based Learning – izrada igre
Oblici poučavanja	Frontalno poučavanje Individualni rad
Razrada aktivnosti	Učenicima je potrebno dati sljedeće materijale: ● sliku pozadine ● Snap! projekt u kojem je definiran objekt djevojčice (s kôdom za kretanje), objekt boce, objekt papira i objekt kante za smeće Učenicima se prezentira problem: Djevojčica se želi prošetati i uživati u parku. Kada stigne u park, vidi da je park pun otpadaka. Odluči pokupiti sve otpatke. Kada to učini, napokon može leći na travu i uživati u čistom parku.  [Korak 1] Postavljena je slika pozadine kao i objekt djevojčice s kôdom za kretanje pomoću strelica i uvjetom za dodir smeđe crte.



Učenici trebaju postaviti početne koordinate djevojčice koristeći blok *kreni na x, y.* Koordinate izaberu učenici, jedino je bitno da se djevojčica nalazi na stazi.

Također moraju dodati i uputu igraču, npr.:



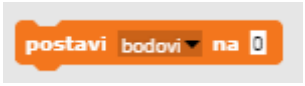
[Korak 2]
Za prebrojavanje broja pokupljenih otpadaka, koristit ćemo varijable.
Što je varijabla? Varijabla se može zamisliti kao kutija u koju spremamo informacije.
U našem slučaju, varijablu možemo zamisliti kao kutiju koja se zove bodovi. Kada djevojčica pokupi otpadak, on se sprema u varijablu *bodovi.* Ova varijabla broji koliko otpadaka je djevojčica pokupila.
Kako stvorimo varijablu?

Odaberemo narančasti blok *Varijable*, zatim kliknemo na gumb *Napravi varijablu*, u skočnom prozoru *Ime varijable* upišemo kako će se zvati naša varijabla i kliknemo OK. Pojavi se blok *bodovi*.

Nakon što se napravi, varijabla će bit prikazana na sceni. Ako ne

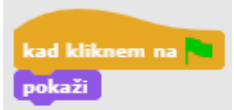
želimo da se varijabla prikazuje na sceni, može se isključiti kvačicu kraj bloka varijable.

Na početku igre, vrijednost varijable mora biti 0 jer još nije pokupljen niti jedan otpadak. Ispod kôda iz [Korak 1] učenik dodaje blok *postavi __ na 0.* Klikom na padajući izbornik odabire odgovarajuću varijablu, koja je u ovom slučaju *bodovi.*



[Korak 3]
Učenici trebaju napisati kôd za objekt boce. Ideja je da objekt nestane (odnosno sakrije se) kada dotakne djevojčicu. Dakle, kôd će započeti kada objekt dodiruje djevojčicu.
Zatim je potrebno razmisliti u kojem slučaju djevojčica pokupi otpatke. Ako kažemo da se otpadak sakrije kada je pokupljen, možemo ga pokupiti samo ako je i dalje na pozornici → prikazuje se. Ako je objekt (boca) i dalje na pozornici, pokupimo ga i „stavimo u kutiju varijable". Prije smo imali 0 elemenata u varijabli bodovi, sada imamo 1. Možemo vidjeti da sakupljanjem otpadaka mijenjamo vrijednost varijable (*bodovi*) za 1. Kada je otpadak pokupljen, sakrijemo ga.

Sada možemo provjeriti je li kôd ispravan.
Kliknemo na zelenu zastavicu i pokupimo bocu. Boca mora nestati i moramo imati 1 bod. Ako želimo ponovno igrati igru, ponovno kliknemo na zelenu zastavicu. Što se dogodi? Gdje je sada boca?
Boca je sakrivena jer smo je prije sakupili. Dakle, moramo napraviti da se boca prikazuje na početku igre. To učinimo tako da odaberemo blok *pokaži.*



[Korak 4]

Ako želimo imati više boca u igri, kopiramo objekt boce: desni klik na objekt i odaberite dupliciraj.

Mišem kliknite na novu bocu i odvucite je negdje unutar labirinta. Postupak dupliciranja se ponovi tako da se dobiju 4 boce.

[Korak 5]
Sada je potrebno napraviti kôd za objekt papira. Moguće je duplicirati kôd za bocu tako da se napravi desni klik mišem na blok kôda i odvuče na objekt papira tako da se mišem klikne na objekt *papir.*

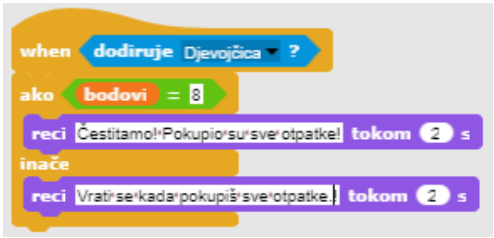
Ovaj korak se ponavlja kako bi se duplicirao blok kôda *kad kliknem na zelenu zastavu – pokaži.*

Također se ponovi i [Korak 4] te se duplicira cijeli objekt papira kako bi bilo više otpadaka papira u labirintu.

[Korak 6]
Posljednje što učenici trebaju napraviti je napisati kôd za kantu za smeće. Objekt za kantu za smeće je već napravljen, učenici ga mogu postaviti bilo gdje unutar labirinta.
Ovaj kôd će se također aktivirati kada djevojčica dotakne kantu za smeće.

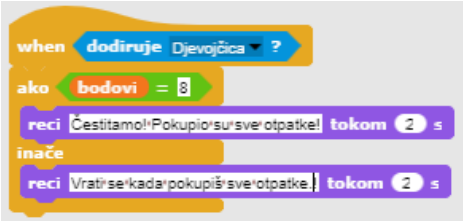
Kanta za smeće treba provjeriti ako su svi otpaci pokupljeni. Pošto imamo varijablu bodovi, ovo će biti jednostavno za napraviti. Recimo

da imamo 8 komada otpadaka u igri, potrebno je provjeriti ako je broj bodova jednak 8. Ako je, svi otpaci su pokupljeni. Ako je broj bodova manji od 8, djevojčica nije pokupila sve otpatke. Koristiti ćemo *ako-inače* izjavu kako bi to definirali u programu i dodat ćemo tekst igraču s informacijom je li pokupio sve otpatke ili nije.



[Cijeli kôd]
Djevojčica

Boce / Papiri

	Kanta za smeće  **[Dodatni zadatak]** Učenici mogu uključiti u igru dodatne zadatke po želji ili koristeći sljedeće prijedloge: • Dodati novu vrstu otpada (npr. bio otpad). • Dodati što će reći kanta za smeće (npr. "Pokupili ste X boca, Y papira i Z lubenica). • Ako igrač pokupi svo smeće, kanta za smeće može čestitati igraču. • Ako igrač ne pokupi svo smeće, kanta za smeće ga može upozoriti na to te navesti koje smeće je ostalo.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park Primjer igre s dodatnim zadacima: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20%2B%20Add.%20Task Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Picking%20up%20trash%20and%20cleaning%20the%20park%20-%20Part

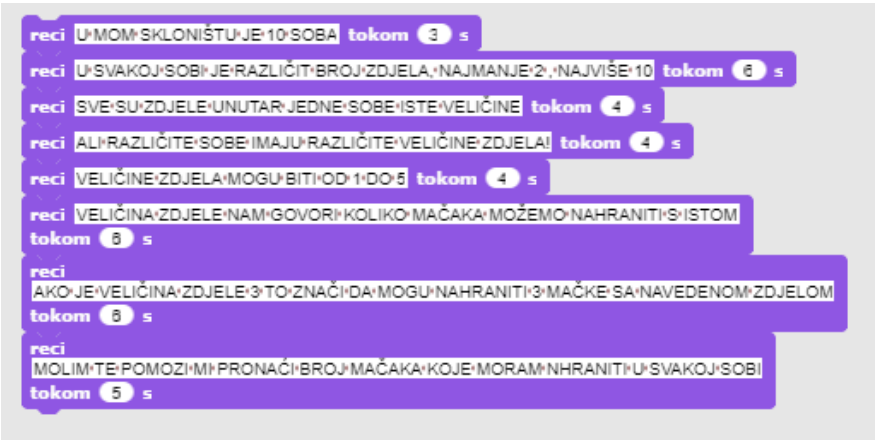
Scenarij učenja 10 – Hranjenje mačaka

Naziv scenarija	Hranjenje mačaka
Potrebno predznanje iz programiranja	Postavljanje pozadine Postavljanje objekata Dodavanje teksta objektu Korištenje varijabli
Ishodi učenja	Koncepti programiranja ● Varijable ● Petlje ● Slučajni brojevi ● Spajanje stringova ● Operatori: logički, aritmetički ● Unos Ishodi učenja: ● Učenik će moći prepoznati situaciju u kojoj će koristiti petlju za ponavljanje n puta ● Učenik će moći razlikovati dodjeljivanje vrijednosti u svakoj iteraciji petlje i prije same petlje ● Učenik će moći koristiti blok unos da bi dobio broj od „igrača“ ● Učenik će moći koristiti aritmetičke operatore za generiranje pravog odgovora ● Učenik će moći koristiti naredbu ako-onda za provjeru ispravnosti unosa i daje odgovarajuće odgovore ● Učenik će moći koristiti varijablu koja će brojati točne odgovore
Cilj, zadaci i kratki opis aktivnosti	**Kratak opis**: Programirajte igru u kojoj će igrač morati izvršiti deset izračuna množenja i računati odnosno brojati točne odgovore. **Zadatak**: Programirajte aktivnost u kojoj će Marta, čuvarica skloništa, neprestano pitati igrača za broj mačaka koje mora nahraniti u određenoj sobi. Broj mačaka ovisi o broju i veličini zdjela. Navedene veličine se za svaku sobu moraju dodijeliti nasumično. Također, moramo imati brojač koji će brojati točne odgovore. Prvo, čuvarica skloništa mora objasniti zadatak za igrače te onda započinje igra. Igra završava kada ona zatraži broj mačaka 10 puta. Svaki puta mora odgovoriti je li broj koji se upiše točan ili nije. Nakon aktivnosti mora dati sažetak u kojem će napisati koliko je igrač bio uspješan

	odnosno reći će koliko je puta igrač odgovorio ispravno odnosno koliko je puta pogriješio. **Cilj:** Učenici će biti upoznati sa konceptom dodjele slučajne vrijednosti varijabli unutar petlje te će razlikovati kada navedeno napravimo van petlje. Naučit će kako dobiti, testirati i izbrojati ispravne unose igrača.
Trajanje	45 min
Strategija i metode učenja i poučavanja	Aktivno učenje Suradničko učenje Rješavanje problema Game Based Learning-izrada igre
Oblici poučavanja	Frontalno poučavanje Rad u paru/Individualni rad/Grupni rad
Razrada aktivnosti	Učenicima se prezentira problem: Čuvarica skloništa pokušava nahraniti svoje mačke u 10 različitih soba. U svakoj sobi je slučajni broj zdjela (2-10) različitih veličina (1-5), ali unutar pojedine sobe sve su zdjele jednake veličine. Veličina zdjele nam govori koliko mačaka može jesti iz iste, npr. ako je veličina zdjele 3 to znači da iz nje mogu jesti 3 mačke. Pomozite pronaći broj mačaka koje mora nahraniti u svakoj sobi. [KORAK 1] Prvo upućujemo učenike da izaberu zanimljivu pozadinu za igru. Ako želimo uštedjeti na vremenu možemo im odmah ponuditi pozadinu.  [KORAK 2] Zatim, moramo odabrati novi objekt koji će predstavljati čuvara mačjeg skloništa. 

[KORAK 3]
Za pohranu potrebnih vrijednosti potrebne su nam 3 varijable: 1) za spremanje broja točnih odgovora, 2) za dodjelu slučajne vrijednosti za broj zdjela unutar svake sobe (2-10) i 3) za dodjelu slučajne vrijednosti veličine zdjela (1-5). Brojač točnih odgovora morati će biti postavljen na 0 dok za preostala dva brojača nećemo trebati dodijelit točnu vrijednost prije petlje. Razlog tome je što ćemo im u svakoj iteraciji petlje dodijeliti novu slučajnu vrijednost. Također, želimo brojati sobe, ali nam ne treba posebna varijabla za to. Upotrijebiti ćemo istu varijablu kao i u petlji, postaviti ćemo ju na 1 a zatim će se povećavati u svakoj iteraciji za 1 sve dok taj broj ne poprimi vrijednost 10. Navedena petlja također služi i za brojanje soba.

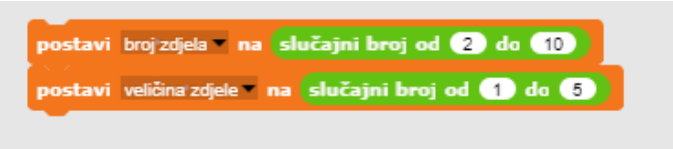
[KORAK 4]
Zatim, moramo programirati upute za igrače. To radimo pomoću Izgled (reci) i čekaj određeni broj sekundi (n).



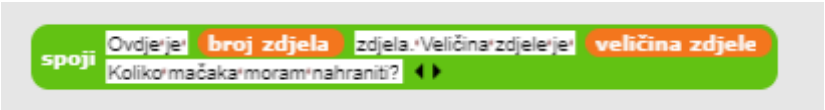
[KORAK 5]
Razgovaramo sa učenicima o tome koje radnje će se dogoditi u svakoj sobi i hoće li biti iste. Odnosno koje će se naredbe koristiti tj. smjestiti u petlju. Prvo ćemo morati nasumično odrediti vrijednost (1-10) za broj zdjela i za veličinu zdjele (1-5). Nakon toga morati ćemo pitati igrača koliko mačaka možemo nahraniti u pojedinoj sobi. Nadalje, morati ćemo ispitati točnost odgovora i dati odgovarajući odgovor (povratnu informaciju) i zapamtiti ga ako je točan (brojati točne odgovore). Na kraju svake iteracije

(ponavljanja) morati ćemo povećati broj soba za 1.

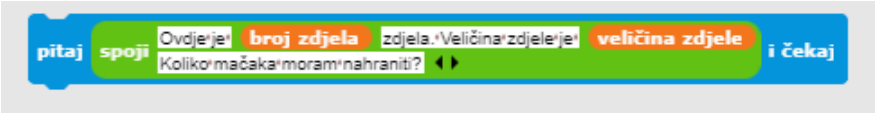
[KORAK 6]
Za dodjeljivanje slučajne vrijednosti za broj zdjela i njihovu veličinu upotrijebit ćemo Varijable/ postavi [opcija] vrijednost na Operatori / slučajni broj od [n] do [m].



[KORAK 7]
Želimo pitati igrače za broj mačaka koje je potrebno nahraniti i to koristeći Osjetila/ pitaj [string] i čekaj, jer će se u protivnom prikazati nekoliko sekundi, a zatim ažurirati novim retkom teksta. Na taj način igrači mogu lako zaboraviti koliko zdjela/veličina je u trenutnoj sobi. Kako bi napravili rečenicu koja se sastoji od kombinacije teksta i referenci na varijable koristimo blok Operatori/spoji [string1][string2]. Navedeni blok morat ćemo proširiti tako da stane cijela rečenica.



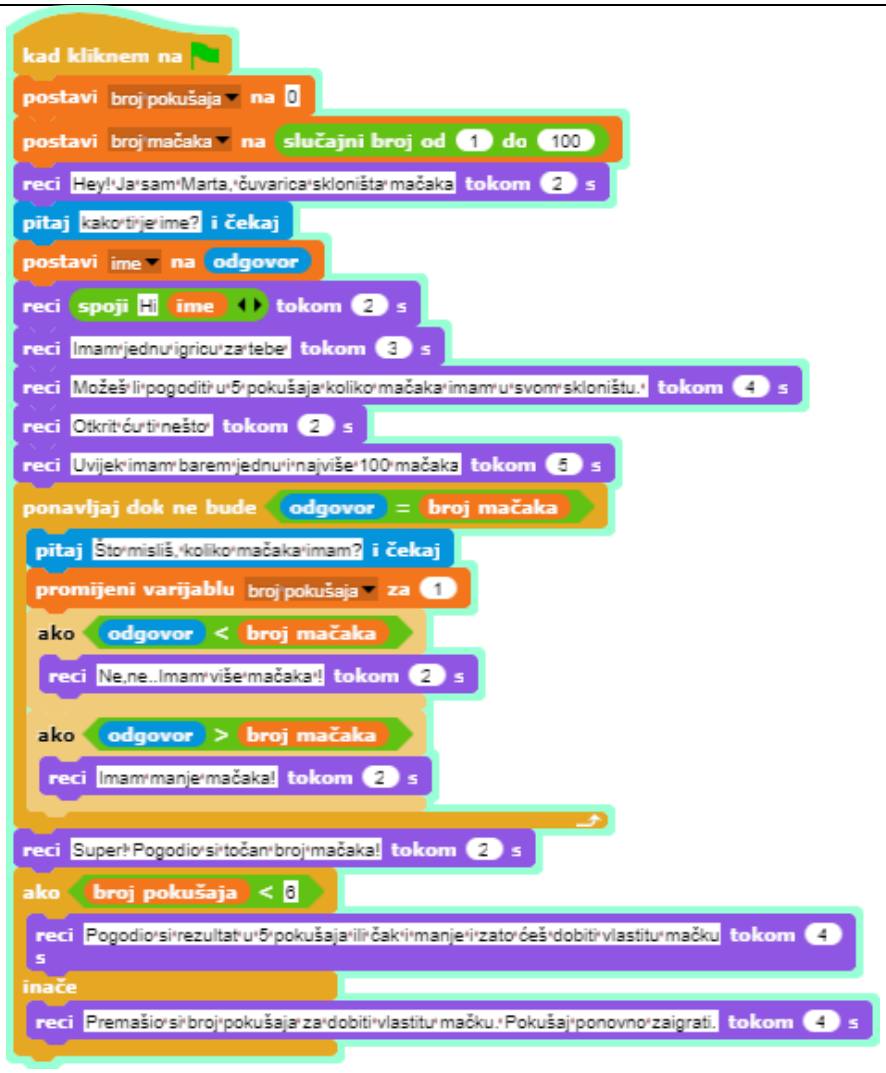
[KORAK 8]
Cijelu prethodnu rečenicu moramo staviti unutar bloka Osjetila/ pitaj [string] i čekati da bi dobili odgovor igrača.



[KORAK 9]
Nakon što igrač odgovori moramo provjeriti ispravnost odgovora. Moguće su dvije situacije, igrač može odgovoriti točno ili netočno, pa ćemo koristiti naredbu ako-onda. Točan odgovor je vrijednost umnoška broja zdjela i veličine zdjele. Moramo provjeriti je li odgovor igrača jednakom tom broju. Ako je odgovor točan, povećavamo brojač točnih odgovora za 1 i dajemo odgovor. Ako je odgovor netočan dajemo samo odgovor. Ne moramo brojati netočne odgovore jer ih možemo izračunati uz pomoć brojača točnih odgovora.

[KORAK 11]
Sada moramo odabrati neku petlju. Kao što je već rečeno najbolje je koristiti petlju jer varijabla koju koristimo za ponavljanje replicira prebrojavanje soba.
[KORAK 12]
Kada se petlja zaustavi, igra je gotova. Na kraju dajemo informacije o postignuću igrača. Broj točnih odgovora spremljen je u brojaču točnih odgovora; broj netočnih odgovora možemo naknadno izračunati.

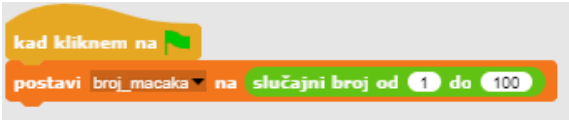
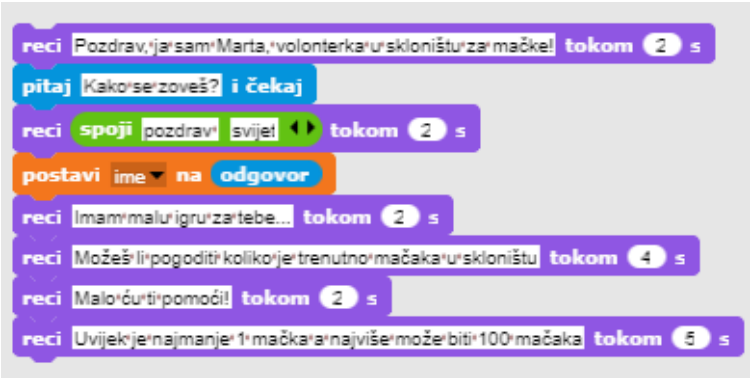
[Cijeli kod]

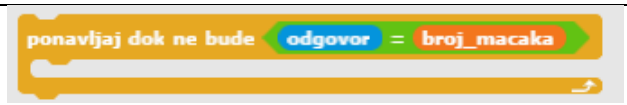
	
Alati i materijali za učitelje	Cijela aktivnost u Snap!-u: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_2 Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen u Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cat_feeding_template

Scenarij učenja 11 – Pogađanje broja mačaka u skloništu

Naziv scenarija	Pogađanje broja mačaka u skloništu
Potrebno predznanje iz programiranja	Dodavanje pozadine. Dodavanje novog objekta. Definirati što objekt govori.
Ishodi učenja	Koncepti programiranja: ● Nasumične vrijednosti ● Varijable ● Unos vrijednosti od strane korisnika ● Petlja *ponavljaj dok* ● Uvjeti ● Operatori uspoređivanja ● Brojač Ishodi učenja: ● Učenik će pohraniti nasumičnu vrijednost u varijablu ● Učenik će koristiti blok za unos podataka kako bi pohranio pokušaj pogađanja broja mačaka ● Učenik će koristiti uvjete (naredbu *ako*) i operatore uspoređivanja za provjeru vrijednosti varijable i ispis odgovarajućeg teksta ● Učenik će koristiti petlju *ponavljaj dok* kako bi igraču omogućio više pokušaja pogađanja i provjerio upisanu vrijednost ● Učenik će implementirati brojač u petlji za prebrojavanje broja pokušaja pogađanja igrača ● Učenik će postaviti uvjet za završetak izvođenja petlje da bi definirao kraj igre
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Djevojčica Marta volontira u skloništu za mačke. Igrač mora pogoditi broj mačaka koji se trenutno nalazi u skloništu **Zadatak:** Sklonište za mačke u kojem djevojčica volontira uvijek ima između 1 i 100 mačaka. Na početku se zadaje nasumičan broj mačaka koji igrač mora pogoditi. Marta pita igrača da pogodi trenutni broj mačaka. Nakon što igrač unese broj, djevojčica daje jedan od odgovora: 1) ako je upisani broj manji od stvarnog broja, kaže: “broj mačaka je veći”, 2) ako je upisani broj veći od stvarnog broja, kaže: “broj mačaka je manji”, 3) ako je upisani broj točan, kaže: “Odlično, pogodio si točan broj!”. Igrač može pogađati 5 puta.

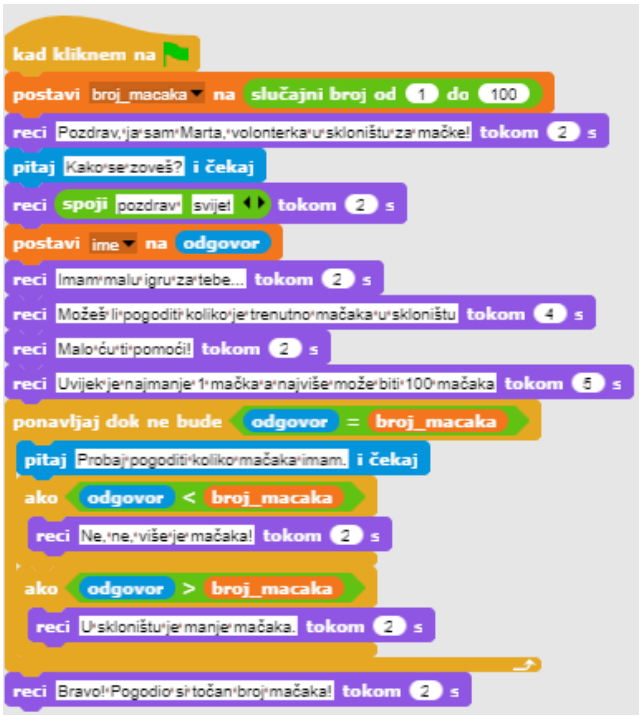
	Ako pogodi, igrač dobiva mačku, u suprotnom je ne dobiva. **Cilj:** Učenici će se upoznati s petljom *ponavljaj dok* i kako postaviti uvjet koji zaustavlja igru. Također će naučiti kako koristiti varijable u različitim situacijama: za pohranjivanje nasumične vrijednosti, kao brojač ili za pohranjivanje vrijednosti koju upiše igrač.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Rješavanje problema Game Based Learning – izrada igre
Oblici poučavanja	Frontalno poučavanje Individualni rad / rad u paru / grupni rad
Razrada aktivnosti	Učenicima se prezentira problem: Djevojčica Marta, volonterka u skloništu za mačke, želi da pogodiš točan broj mačaka u skloništu. Broj može biti između 1 i 100. Kada igrač upiše broj, Marta odgovori je li broj veći, manji ili je to točan broj mačaka. Ako igrač pogodi broj u manje od 5 pokušaja, dobiva mačku. U suprotnom, djevojčica zatraži da se igra ponovno. [Korak 1] Prvi zadatak je napraviti zanimljivu pozadinu igre. Učenici mogu sami nacrtati ili koristiti slobodne slike s interneta. Kako bi se uštedjelo na vremenu, pozadina je unaprijed pripremljena.  [Korak 2] Potrebno je odabrati novi kostim za zadani objekt koji će biti volonterka u skloništu.  [Korak 3] Raspravljamo s učenicima kako je ova igra zanimljiva ukoliko se igra

	više puta pošto se broj mačaka postavlja nasumično. Kako bi postavili nasumičan broj, potrebno ga je spremiti u varijablu. Varijabla je sada jedini način da se zapamti određena vrijednost u Snap!-u (pretpostavljamo da još uvijek nisu upoznati s konceptom liste). To se mora dogoditi kada se igra pokrene (Upravljanje → Kada kliknem na zelenu zastavu).  [Korak 4] Djevojčica treba pitati igrača njegovo ime kako bi ga pozdravila. To je moguće napraviti koristeći blok *pitaj i čekaj* koji se nalazi unutar skupine *Osjetila*. Igračev odgovor se spremi u već definiranu varijablu *odgovor* (nalazi se u skupini *Osjetila*). Kako bi Marta pozdravila igrača, potrebno je povezati ime koje se spremilo u varijablu *odgovor* s pozdravom. To se napravi pomoću bloka *spoji[riječ1][riječ2]* koji se nalazi u skupini *Operatori*. Za prikaz teksta koristi se blok *reci tokom 2 sekunde* iz skupine *Izgled*. Isti blok se koristi i za pisanje uputa igraču. Učenicima je potrebno naglasiti da trebaju biti pažljivi oko vremena koliko se tekst prikazuje.  [Korak 5] S učenicima se diskutira kako nije moguće predvidjeti koliko puta će igrač pogađati kako bi došao do točnog broja. Igrač može pogoditi jako brzo i pogoditi u prvom pokušaju, možda će mu biti potrebno 5 pokušaja ili čak i više. Iz tog razloga je potrebno odabrati pravu petlju za ovaj zadatak. Volonterka treba uzastopno pitati i dati odgovarajući odgovor sve dok igrač ne pogodi točan broj. Jedina petlja koju možemo koristiti je petlja *ponavljaj dok ne bude.* Uvjet je relativno jednostavan - moramo ponavljati dok odgovor igrača, koji je spremljen u varijabli *odgovor*, nije jednak vrijednosti koja je spremljena u varijabli *broj_mačaka.*



[Korak 6]
Zatim, moramo pitati učenike koje naredbe moramo uključiti u tijelo petlje. Koja se aktivnost ili naredba ponavlja dok igrač ne pogodi točan broj? Prvo, potrebno je pitati igrača da upiše broj, zatim moramo dati odgovor ovisan o vrijednosti upisanog broja.

[Korak 7]
Posljednja stvar za objasniti ili diskutirati s učenicima je što uzrokuje kraj izvršavanja petlje. Kada je igračev odgovor jednak broju mačaka, oba uvjeta u tijelu petlje će biti pogrešna i petlja će otići u iduću iteraciju provjeravajući uvjet petlje. Ovog puta, uvjet će biti istinit, pa će se petlja prekinuti i izvest će se naredbe koje dolaze nakon petlje. Drugim riječima, kada se petlja završi znamo da je igrač pogodio točan broj i možemo ispisati poruku igraču u skladu s time.



[Korak 9]
Kako je broj pokušaja za pogađanje ograničen, moramo napraviti novu varijablu koja će imati ulogu brojača i postaviti početnu vrijednost na 0. S učenicima diskutiramo o važnosti inicijalizacije varijable i razlike između postavljanja vrijednosti i povećanja vrijednosti (kada postavimo vrijednost varijable, prethodna vrijednost se izgubi). Svaki put kada igrač upiše broj želimo brojač povećati za 1.

[Korak 10]
Nakon točnog odgovora, moramo provjeriti vrijednost varijable brojača kako bi odlučili hoće li igrač dobiti mačku ili ne. Pošto Snap! ima samo logički operator manje (<), a nema operator manje ili jednako, uvjet je je li *brojač_mačke < 6.* Ovo je također dobar primjer korištenja *ako-inače* uvjeta jer razlikujemo dva slučaja.

[Cijeli kôd]

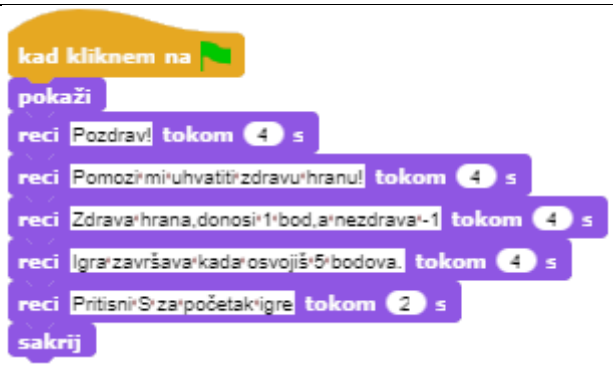
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter Lajovic, S. (2011). Scratch. Nauči se programirati in postani računalniški maček. Ljubljana: Pasadena. Vorderman, C. (2017). Računalniško programiranje za otroke. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen u alatu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=cats_in_a_shelter_template

NAPREDNI SCENARIJI UČENJA

Scenarij učenja 12 – Hvatanje zdrave hrane

Naziv scenarija	Hvatanje zdrave hrane
Potrebno predznanje iz programiranja	Dodavanje teksta objektu Prikazivanje i skrivanje objekta Korištenje točke za određivanje smjera Korištenje nasumičnih vrijednosti Korištenje varijabli za brojenje bodova Korištenje petlje *ponavljaj* Korištenje petlje *zauvijek* Korištenje uvjeta
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Petlje ● Točka za smjer ● Nasumične vrijednosti Ishodi učenja: ● Učenik će moći koristiti varijablu za sprječavanje pokretanja igre prije nego djevojka završi s govorom (izborno) ● Učenik će moći koristiti uvjet *ako* kako bi provjerio (uz pomoć varijable) može li se hrana početi kretati ● Učenik će moći koristiti petlju ponavljaj za kretanje hrane sve dok je broj bodova manji od 5 ● Učenik će moći koristiti točku u smjeru 180 (prema dolje) za objekte koji se kreću prema dolje ● Učenik će moći koristiti nasumične vrijednosti za određivanje broja koraka ● Učenik će moći koristiti nasumične vrijednosti za pomak na nasumičnu poziciju ● Učenik će moći koristiti nasumične vrijednosti za pomak na poziciju x (nasumična vrijednost), y (fiksna vrijednost) (izborno)
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Djevojka hvata hranu. Mora biti oprezna, samo zdrave namirnice donose bodove! **Zadatak:** Učenici trebaju programirati dva različita objekta, djevojku koja daje upute, govori što treba učiniti za početak igre i broji bodove; i hranu koja nasumično pada s vrha zaslona. Uz to, učenici

	mogu dodati varijablu i uvjet *ako* za sprječavanje kretanja hrane sve dok djevojčica ne prestane govoriti. **Cilj:** Učenici će naučiti kako nasumično pomicati za X koraka i odabrati položaj i također kako koristiti varijable i uvjete za sprječavanje drugih događaja.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Game Based Learning – izrada igre Rješavanje problema
Oblici poučavanja	Individualni rad / Rad u paru
Razrada aktivnosti	Učenicima se prezentira problem: Djevojka treba uhvatiti hranu. Svaka zdrava namirnica donosi 1 bod, dok svaka nezdrava oduzima 1 bod. Igra počinje s nekoliko uputa koje daje djevojčica. Tada ona nestaje i pojavljuje se hrana. Kad igrač sakupi 5 bodova, hrana nestaje i djevojka se ponovo pojavljuje.  [Korak 1] Ova aktivnost namijenjena je za individualni rad ili rad u paru. Učitelj daje sugestije, objašnjava teže dijelove i pomaže kada je potrebno. Učenici biraju pozadinu i dodaju glavni lik, npr. djevojku. Djevojka na početku daje upute, a zatim nestaje. Kao što smo vidjeli u prethodnim aktivnostima, dobro je napisati blok *prikaži kad kliknem zastavu* (kada se ponovo igra, ako se objekt ne pokaže). Kod glasi, na primjer:



Na ovaj objekt ćemo se vratiti kasnije. Sad napišemo kod za voće.

[Korak 2]
Učenici dodaju novi objekt, zdravu hranu, npr. jabuku.
Najprije, programiraju pokret, odozgo prema dolje, pa odabiru sljedeće blokove:

Ako ne žele da njihova jabuka bude naopako, učenici mogu odabrati treću opciju *ne okreći se* u bloku smjera.

Da bi igra bila zanimljivija, broj koraka može biti nasumično odabran, tako da brzina ne bude uvijek ista. na primjer:

Sljedeći korak je razmišljanje o tome što se događa kad jabuka dođe do dna zaslona?
U ovom slučaju učenici mogu koristiti blok *dodiruje rub* u kombinaciji s *uvjetom ako*. Ako jabuka dodirne rub, pomaknut će se na neki nasumični položaj. Blokovi za kretanje nude nam sljedeći blok:

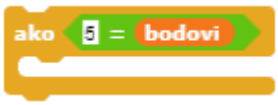
Ova naredba će nasumično odabrati bilo koje x i y koordinate i jabuka se može pojaviti bilo gdje na zaslonu (pogledajte crvene točke na slici).

Ako želimo da se jabuka uvijek pojavi na vrhu zaslona, vrijednost y se može fiksirati, a samo x vrijednost se odabire nasumično.

Sljedećim kodom jabuka će se uvijek pojaviti na vrhu zaslona (pogledajte crvene točke na slici).

[Korak 3]
Učenici sada mogu kreirati varijablu, *bodovi*, koju će koristiti za brojanje. Bodovi na početku moraju biti postavljeni na 0 (na sprite-u djevojke).

[Korak 4]
Ako želimo da se jabuka pomiče po nekom obrascu, potrebna nam je petlja. Učenici mogu koristiti petlju *ponavljaj dok* i postaviti uvjet. Na primjer, žele da igra završi kada osvoje 5 bodova. Dakle, uvjet će biti *bodovi = 5* i petlja će se ponavljati dok je uvjet lažan. Kad je uvjet istinit, ako igrač dosegne 5 bodova, petlja će se zaustaviti.



[Korak 5]

Ne želimo da se jabuka prikaže na početku, nego nakon što djevojčica da upute. Učenici mogu programirati jabuku tako da se pokaže kad se pritisne tipku. Naravno, učenici trebaju dodati blok *pokaži* prije petlje ponavljaj i blok sakrij se nakon toga. Cijeli kod zasada izgleda ovako:

[Korak 6]

Što se događa kad se klikne na jabuku (ili se unese miš)?

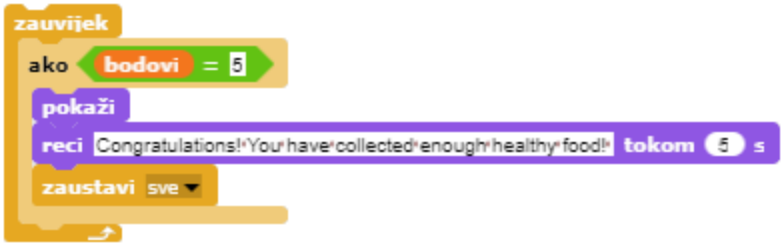
Jabuka se mora nestati, pribrojiti bodove, promijeniti položaj i ponovo se pokazati. Bodovi će se mijenjati za 1, a za poziciju učenici mogu koristiti isti kod kao i prije.

[Korak 7]

Vrativmo se djevojci.

Djevojka se sada mora ponovo pojaviti i reći, npr. Čestitam!

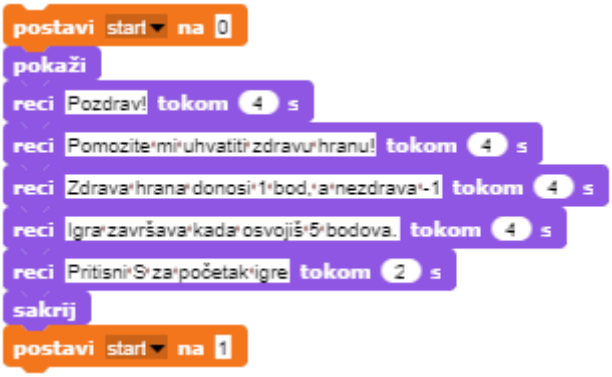
Trebat će nam trebati petlja *zauvijek* koja će provjeravati jesmo li osvojili 5 bodova. Ako smo osvojili 5 bodova, djevojka će pokazati i reći nešto. Nakon toga dodat ćemo blok *zaustavi sve*. Neka učenici sami dokuče što znači taj blok (bez zaustavljanja, djevojka će zauvijek govoriti "Čestitam ...").



[Korak 8]

Igrajući igru ponovo, kada učenici već znaju sve upute (od [Korak 1]) sigurno će ih htjeti preskočiti. Učenici mogu pritisnuti tipku S prije nego što igra počne, ali djevojka će još uvijek govoriti.

Da bismo to spriječili, možemo stvoriti drugu varijablu (nazvanu *start*), koju na početku moramo postaviti na 0. Tada, nakon djevojčinih uputa, varijabla *start* će se promijeniti u 1.



Sada moramo programirati jabuku da se pokrene samo ako je varijabla *start* jednaka 1, što će učenici učiniti uvjetom if. Uz to, učenici neće moći pokrenuti igru prije nego što djevojčica prestane govoriti.

Kod jabuke je sada:

[Korak 9]

Učenici sada mogu duplicirati objekt jabuke i mijenjati mu kostim (ako žele). Kod će biti isti.

Jedina promjena je s nezdravom hranom, gdje će učenici klikom na nju izgubiti jedan bod.

[Konačni kod]
Djevojka

Jabuka

[Dodatni zadatak]

Učenici mogu dopuniti igru po želji ili koristiti sljedeće prijedloge:

- Dodati novi object – zdjelu (koristeći sliku koju ćete pronaći online ili koristeći priloženu sliku zdjele).
- Promijeniti igru tako da objekt zdjela hvata hranu.
- Postaviti početnu pozivi zdjele i napisati kod za njeno pomicanje.
- Promijeniti pravila – neka igra završava kada igrač prikupi 20 bodova (igrač pobjeđuje) ili kada pokupi 3 nezdrave namirnice (igrač gubi).

	● Dodati još objekata koji predstavljaju hranu kako bi igra bila zanimljivija. ● Promijeniti kostim zdjele kada igrač prikupi 5, 10 ili 15 bodova.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food Primjer igre s dodatnim zadacima: https://snap.berkeley.edu/project?user=mateja&project=Catching%20healthy%20food%20%2B%20Add.%20Task Lajovic, S. (2011). Scratch. Nauči se programirati in postani računalniški maček. Ljubljana: Pasadena. Vorderman, C. (2017). Računalniško programiranje za otroke. Ljubljana: MK.
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=C4G12_Catching%20healthy%20food%20-%20Part

Scenarij učenja 13 – Pričam ti priču

Naziv scenarija	Pričam ti priču
Potrebno predznanje iz programiranja	Prikazivanje i sakrivanje objekata Uvjeti Upotreba *reci* Upotreba *pričekaj … sekundi*
Ishodi učenja	Koncepti programiranja: ● Pomicanje i promjena veličine ● Prikazivanje poruka i pričanje ● Sastavljanje strukture priče ● Mijenjanje pozadine prizora Ishodi učenja: ● Učenici će moći planirati dijaloge i aktivnosti u priči ● Učenici će moći emitirati poruke i prikazivati govor (dijalog) ● Učenici će moći pomicati i mijenjati veličinu objekata ● Učenici će moći prikazivati i sakrivati objekte ● Učenici će moći proširiti kod objekata
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Zec priča priču o Alisi u zemlji čudesa. Započinje s par rečenica, a u pozadini je prizor Alise. Priča započinje u šumi. Alisa šeta i pita se „Gdje sam?“ (Da bismo shvatili kako se Alise kreće, postupno se njezina veličina smanjuje). Alisa dolazi do raskrižja i ugleda mačku na drvetu. Započinje razgovor između Alise i mačke. **Zadatak:** Učenici moraju eksperimentirati s kratkim primjerom priče o susretu Alise i mačke na temelju sinkronizacije dijaloga koristeći blok za čekanje. Nadalje, pregledavaju drugu verziju priče pomoću prikazanih poruka. Upisuju se naredbe za razmjenu poruka. Učenici dovršavaju kôd, prema tekstu sa slike. Zadatak postaje kompleksniji mijenjanjem scenske pozadine i pomicanjem Alise kroz šumu prije njezinog susreta s mačkom. **Cilj:** Učenici će naučiti kako ispričati priču, kako koristiti poruke i kako promijeniti pozadinu pozornice.
Trajanje	90 minuta
Strategija i metode učenja i poučavanja	Game Based Learning – igra Aktivno učenje Rješavanje problema

Oblici poučavanja	Frontalni rad Rad u paru Individualni rad
Razrada aktivnosti	1. Učitelj razgovara sa učenicima o Alisi u zemlji čudesa i pokazuje sliku Alise koja se susrela s mačkom. Objašnjava da se Alisina priča može ispričati uz alat Snap!. Učenici imaju zadatak pokrenuti projekt i pogledati kodove objekata. https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 S učenicima se vodi rasprava oko sljedećih pitanja: ● Tko prvi počinje razgovarati? ● Kada se Alisa uključuje u razgovor, a kada Mačka? ● Zašto u dijalogu likova nema sinkronizacije? Odgovor leži u netočnom proračunu vremena u kojem se prikazuju poruke likova koji razgovaraju i nedostatku vremena da lik završi svoje odgovore. 

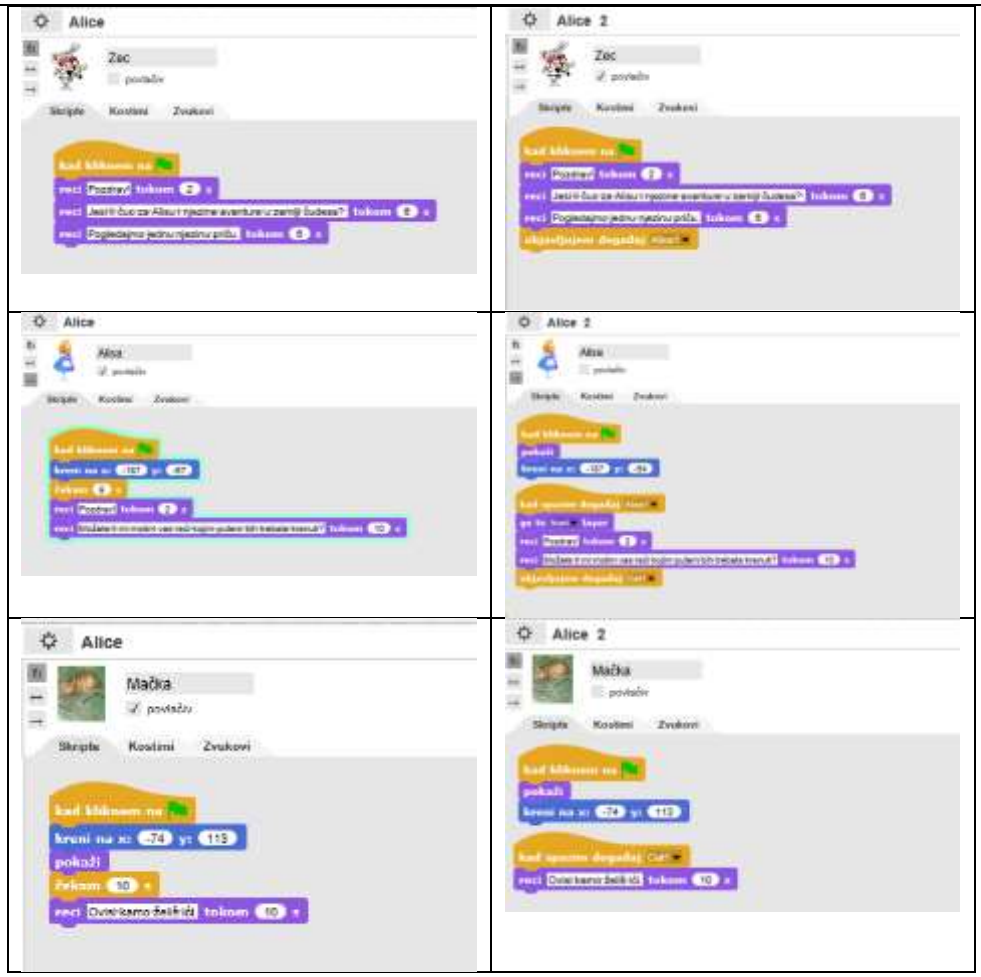
Objekt	Aktivnost	Početak	Završetak	Trajanje
Zec	Reci: Pozdrav! Jesi li čuo za Alisu i njezine avanture u zemlji čudesa? Pogledajmo njezinu priču.	0	14	14
Alisa	Reci: Možete li mi molim vas reći kojim putem bih trebala krenuti?	9	21	12
Mačka	Reci: Ovisi kamo želiš ići.	10	20	10

Zaključak je da sinkronizacija s blokom *pričekaj*… i drugim blokom može dovesti do pogreške u ponašanju likova koji razgovaraju.

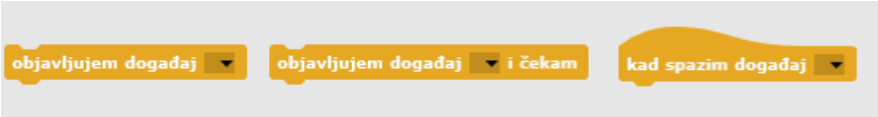
2. Učitelj pokreće i pregledava se projekt https://snap.berkeley.edu/project?user=ddureva&project=Alice_2 te se postavlja pitanje koje su dosad nepoznate naredbe?

Nadalje, uspoređuju se kodovi dvaju projekata: Alice_1 i Alice_2:

Alice_1	Alice_2

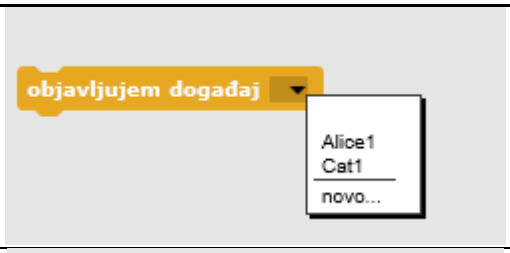


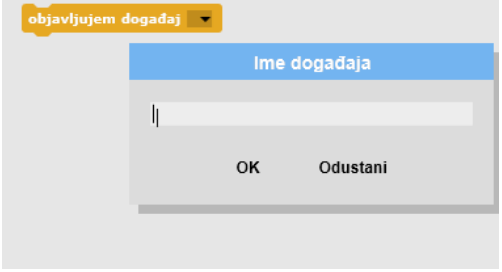
Uvode se blokovi za emitiranje poruka, odnosno događaja:



Raspravlja se o tome da emitirane poruke (događaji) šalju se svima, ali mogu ih primiti samo neki likovi.
Objavljujem događaj … i *čekam* zahtijeva da svi koji su primili poruku izvrše svoje radnje, a zatim se nastavljaju aktvnosti objekata.

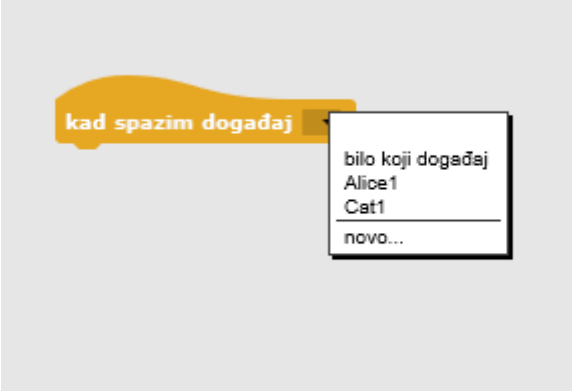
Učitelj pokazuje kako imenovati događaj i kako se koristi u slučaju "*Kad spazim događaj*"…





Upisati ime te pritisnuti OK.

Upotreba u događaju (*Kad spazim događaj..):*



Poruka koju treba primiti od objekta je odabrana s popisa događaja.

Raspravlja se kako dovršiti priču u slici te kako imenovati poruke.

Učenici dovršavaju priču radom u paru.

Učitelj naglašava da pričanje često zahtijeva promjenu scenskih pozadina. Cilj je napraviti priču cjelovitijom tako što će započeti priču sa Zecom na uvodnoj pozadini, premještajući radnju u šumu gdje Alisa hoda i pita se" Gdje sam?" . Njena se veličina postupno smanjuje kako se odmiče sve dalje. Zatim se nađe na raskrižju i vidi mačku te započinje razgovor između njih dvoje.

Učitelj prikazuje gotov projekt:
https://snap.berkeley.edu/project?user=ddureva&project=Alice_2

- Komentiraju se promjene u scenama i radnjama likova. "*Kada se prizor mijenja? Kada se Alisa pojavljuje i koje su njene aktivnosti? Kada se mačka pojavljuje i koje su njene aktivnosti? "*

- Raspravlja se o scenama u projektu Alice_2. Postoje 3 scene, jedna je već korištena. *Koju scenu koristiti za početak? Što treba učiniti da se objekti Alisa i Mačka ne bi prikazali na početku projekta? Kako promijeniti scensku pozadinu?*

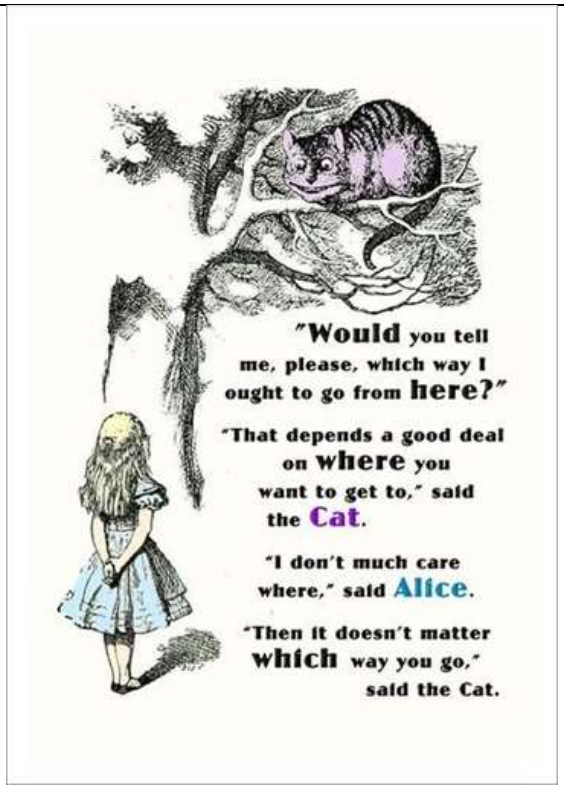
Dok je Alisa na putu u šumu, ona šeće, tj. udaljava se pa se njezina veličina smanjuje za -10%. To se ponavlja 5 puta pomoću petlje za ponavljanje.

Kad stigne do raskrižja, prizor (pozadina) se mijenja s porukom "Susret s Mačkom". Ovu poruku istovremeno prima i Zec te se i njegova veličina smanjuje na 80% i nastavlja pričati sa smanjenom veličinom.
U ovoj fazi, Mačka nije prikazana jer je prisutna kao dio ukrasa na drvetu. Pojavljuje se na Cat1 poruci. Učitelj može objasniti da je mačka izrezana iz ukrasa pomoću vanjskog grafičkog uređivača.
Nakon objavljivanja poruke Zeca, priča se nastavlja.

3. Učitelj komentira da, kako bi ispričala priča, prvo se mora izmisliti zaplet. Za opis scenarija priče može se koristiti dodatna tablica. Učitelj odlučuje hoće li učenicima dati gotovu tablicu ili djelomično dovršenu te ju učenici vođeni ilustracijom, mogu dovršiti.

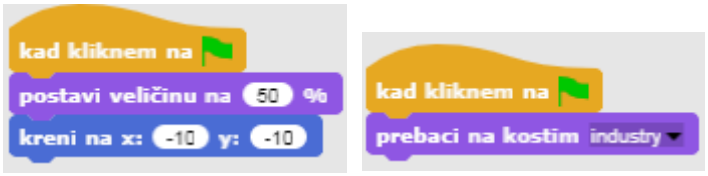
4. Učenici bi trebali opisati napravljeni scenarij te u paru dovršiti priču na projektu Alice_2.

Alati i materijali za učitelje	Aktivnost izrađena alatom Snap! https://snap.berkeley.edu/project?user=ddureva&project=Alice

Alati i materijali za učenike	 https://snap.berkeley.edu/project?user=ddureva&project=Alice_1 https://snap.berkeley.edu/project?user=ddureva&project=Alice_2

Scenarij učenja 14 – Crtanje

Naziv scenarija	Crtanje
Potrebno predznanje iz programiranja	Dodavanje objekta Korištenje pokazivača smjera Korištenje varijablu za brojanje bodova Korištenje naredbe ponavljanja (petlje) Korištenje uvjetovanja
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjetovanja ● Petlje ● Pokazivač smjera ● Operatori Ishodi učenja: ● Učenici će moći koristiti opciju olovke za crtanje ● Učenici će moći koristiti emitiranje za kontrolu objekta ● Učenici će moći mijenjati vrijednost varijable prilikom crtanja novih stabala ● Učenici će moći koristiti petlje za crtanje dijelova stabala ● Učenici će moći koristiti uvjete za promjenu pozornice ● Učenici će moći koristiti pokazivač smjera (okreni se u smjeru) za crtanje krošnje stabala ● Učenici će moći koristiti petlje ponavljanja za crtanje ● Učenici će moći koristiti operator > za promjenu pozornice
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Klima se jako promijenila, zrak je radi industrije jako zagađen. Kako bi se poboljšala kvaliteta zraka potrebno je posaditi stabla! **Zadaci:** Kako bi se poboljšala kvaliteta zraka, učenici moraju programirati objekt pomoću kojeg će crtati dvije vrste različitih stabala – borove i hrast te gumbe koje simboliziraju te vrste stabala. Klikom na gumb, crta se određena vrsta stabla. **Cilj:** Učenici će naučiti kako se crta pomoću Snap!-a, promijeniti boju i debljinu olovke te kako koristiti varijable i uvjete koji uzrokuju novi događaj.
Trajanje	45 minuta

Strategija i metode učenja i poučavanja	Game Based Learning – igra, aktivno učenje, rješavanje problema
Oblici poučavanja	Rad u paru / Individualni rad
Razrada aktivnosti	Učenicima se prezentira kako treba izgledati igra: Na početku igre na pozornici se prikazuje industrija koja uzrokuje klimatske promjene i varijabla koja prikazuje kvalitetu zraka. Potrebno je posaditi stabla kako bi se poboljšala kvaliteta zraka. Crtaju se dvije različite vrste stabala, bor i hrast. Kada se nacrta bor, zrak se poboljša za 3 boda, a crtanjem hrasta bor se poboljša za 2 boda. Kada kvaliteta zraka dođe na 10 bodova, pozadina se promijeni na livadu. [Korak 1] Ova aktivnost zamišljena je za individualan rad ili rad u paru. Učiteljica daje prijedloge, objašnjava teže dijelove i pomaže kada zatreba. Učenici za pozadinu postavljaju i prikazuju sliku industrije koja zagađuje zrak te dodaju objekt olovke. Pošto je objekt prevelik, potrebno ga je smanjiti tako da se njegova veličina postavi na 50% i zadaje se početna pozicija olovke, npr. x=-10, y=-10. Primjer koda:  [Korak 2] Objekt olovke bi trebao primiti poruke „oak“ i „pine“ i crtati različita stabala kao odgovor na poruku. Označite objekt olovke i dodajte kod tako da kada objekt primi poruku „pine“ crta bor. Olovku je potrebno rotirati za 90 stupnjeva kako bi krošnja bila u obliku trokuta te postaviti boju olovke na neku nijansu zelene boje.  Za crtanje krošnje bora potrebno je pomaknuti objekt 40 koraka i nakon svakog koraka okrenuti se ulijevo za 120 stupnjeva.

Ovo kretanje je potrebno ponoviti 3 puta.

Nakon što se završilo s crtanjem krošnje, treba nacrtati deblo. Objekt olovke se pomakne za 22 koraka i boja olovke se promijeni u smeđu.

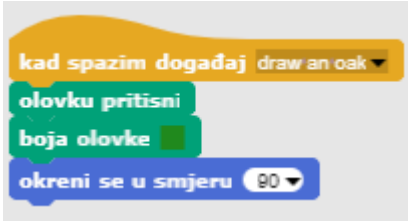
Olovka se okrene udesno za 90 stupnjeva i pomakne 10 koraka.

Sve se ponovi 3 puta.

Kada se završi sa crtanjem potrebno je podignuti olovku kako ne bi napravila crtu prilikom pomicanja objekta olovke. Također, olovka se premješta na novu slučajnu poziciju.

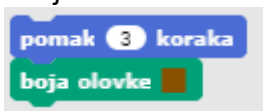
[Korak 3]
Na sličan način je potrebno omogućiti crtanje hrasta.

Označite objekt olovke i dodajte kod tako da kada objekt primi poruku „oak“ crta hrast. Kako bi krošnja bila u obliku kruga, olovka treba biti spuštena i zarotirana za 90 stupnjeva a njena boja zelena.

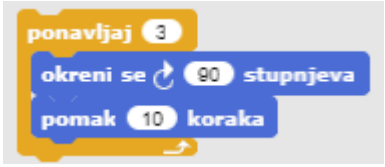


Za crtanje krošnje hrasta pomaknite se za 1 korak, okrenite ulijevo za 3 stupnja te to ponovite 120 puta.

Nakon krošnje, treba nacrtati i deblo. Olovku je potrebno pomaknuti na sredinu krošnje za 3 koraka te postaviti smeđu boju.

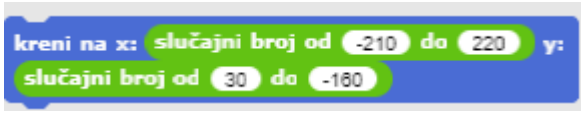


Zatim objekt okrenite 90 stupnjeva udesno te pomaknite za 10 koraka. Taj dio je potrebno ponoviti 3 puta.

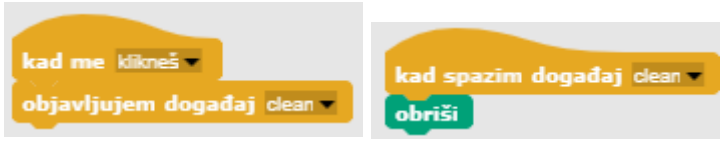


Na kraju je potrebno podignuti olovku kako objekt ne bi ostavljao trag prilikom pomicanja.

Nakon što je hrast nacrtan, olovku treba pomaknuti na novu slučajnu poziciju.



[Korak 4]
Nakon toga je potrebno omogućiti da se klikom na objekt za brisanje maknu sva nacrtana stabla. Kada se klikne na objekt „X“ šalje se poruka o brisanju. Nakon što primi tu poruku, olovka obruše drveće.



[Korak 5]
Napravite novu varijablu „clean air“ koja će pokazivati kvalitetu zraka. Na pozornici postavite da je na početku igre je vrijednost varijable jednaka 0.

Svaki puta kad se nacrta bor zrak se poboljša za 2 jedinice. Dodajte kod objektu olovka kojim će se povećati vrijednost varijable „clean air“ za 2 svaki puta kada se klikne na bor.

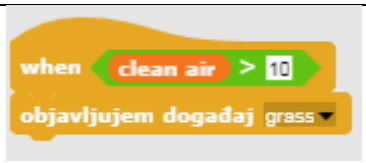
Svaki puta kad se nacrta hrast zrak se poboljša za 3 jedinice. Dodajte kod objektu olovka kojim će se povećati vrijednost varijable „clean air“ za 3 boda svaki puta kada se klikne na hrast.

[Korak 6]
Dodajte kod koji omogućuje da kada varijabla „clean air“ poprimi vrijednost 10, pozornica se promijeni u travu.
Na pozornici dodajte novu pozadinu „grass“ (iz materijala).

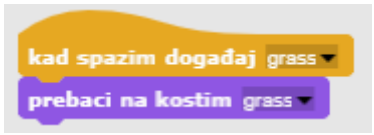
Označite objekt olovka i dodajte blok „When“ iz bloka „Upravljanje“.

Iz bloka „Operatori“ dodajte operator >.

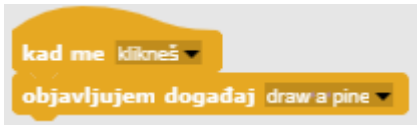
Napravite da objekt pošalje poruku „grass“ kada je varijabla „clean air“ veća od 10.



Označite pozornicu i dodajte kod da kada primi poruku „grass“ promijeni kostim na „grass“.

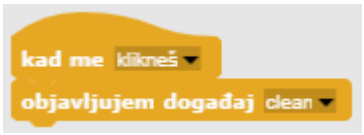


[Završni kod]
Bor (pine)

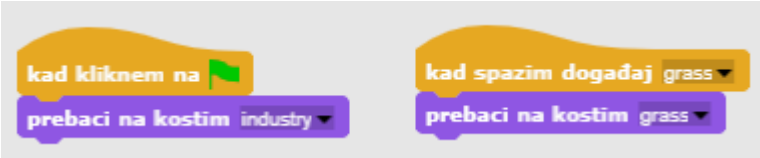


Hrast (oak)

X



Olovka

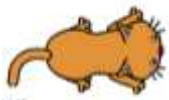
	Pozornica  [**Dodatni zadatak**] Možete dopuniti ovu igru dodavanjem životinja koje će se pojaviti kada zrak više nije zagađen.
Alati i materijali za učitelje	Snap! projekt “Drawing”: https://snap.berkeley.edu/project?user=tadeja&project=Improve%20the%20climate
Alati i materijali za učenike	Programski jezik Snap!: https://snap.berkeley.edu/ Slike: grass.png, industry.png

Scenarij učenja 15 – Uhvati miša

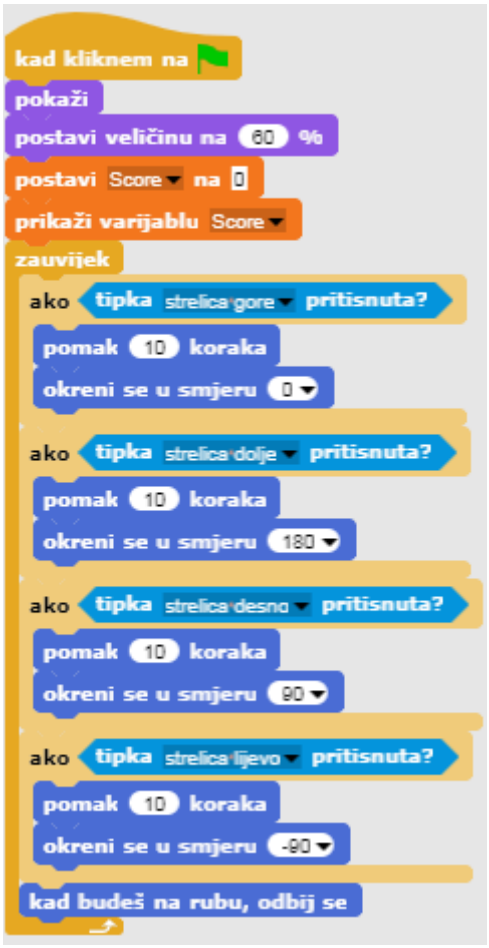
Naziv scenarija	Uhvati miša
Potrebno predznanje iz programiranja	Dodavanje pozadine. Dodavanje objekata. Promjena kostima objektima. Dodavanje zvukova. Kretanje strelicama s tipkovnice uvažavajući ograničenja. Logički izrazi za dva različita stanja. Korištenje uvjeta.
Ishodi učenja	Koncepti programiranja: ● Beskonačna petlja ● Slučajni broj ● Brojilo ● Mjerač vremena Ishodi učenja: ● Učenik će moći koristiti beskonačnu petlju za pomicanje objekta. ● Učenik će moći koristiti slučajne brojeve za određivanje položaja objekta, pomicanje objekta za slučajan broj koraka i okretanje objekta za slučajan broj stupnjeva. ● Učenik će moći koristiti brojač za brojanje miševa i koristi konačnu vrijednost kako bi rezimirao koliko je igrač bio uspješan. ● Učenik će moći koristiti mjerač vremena za određivanje kraja igre.
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Programiranje igre u kojoj će igrač (mačka) morati uhvatiti miša. **Zadatak:** Programiranje aktivnosti u kojoj će mačka uhvatiti miša. Igrač mačku pomiče strelicama na tipkovnici, a miš se kreće nasumično. Kada mačka dodirne miša, miš će se sakriti i pojaviti na nasumičnom mjestu. Također moramo imati i brojač koji će brojati koliko je puta mačka uhvatila miša. Za završetak igre također nam treba mjerač vremena. Na kraju igre djevojka mora rezimirati koliko je uspješan igrač bio, izgovorit će koliko je puta igrač uhvatio miša. **Cilj:** Učenik će se upoznati s konceptom dodjeljivanja više varijabli slučajnim vrijednostima. Naučit će kako koristiti blok *Operatori / slučajni broj od [x] do [y].*
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje, zajedničko učenje, rješavanje problema, učenje zasnovano na igrama.

Oblici poučavanja	Frontalni rad Rad u paru / grupni rad
Razrada aktivnosti	**Motivacija – uvod** Nastavnik pokazujući igru motiviram učenike. Razgovaram s učenicima kako bi započeli s programiranjem te igre. Zajedno s učenicima određujem redoslijed koraka, na primjer: 1. Odrediti pozadinu i dodati objekt; 2. Programirati mačku da se kreće tipkama sa strelicama; 3. Programirati miša da se kreće nasumično; 4. Programirati miša da se sakrije (i pojavi nasumično) kada dodirne mačku; 5. Brojač programa; 6. Dodati mjerač vremena i odredite kraj igre; 7. Dodati djevojčicu i Programirati da rezimira koliko je uspješan igrač bio; 8. Programirati djevojčicu da skoči kad dodirne miša; 9. Dodati zvuk mačke / miša; 10. … Učenici mogu pomoći u koracima ili sami smisliti svoja pravila igre (ali moraju slijediti podebljane korake).  Uvedimo operator za dodjeljivanje slučajnih varijabli  Učenici programiraju sljedeće zadatke u parovima/grupama uz pomoć učitelja.

[Korak 1]
Prvi korak je određivanje pozadine dane igrice. Učenici besplatno traže sliku na internetu. Zatim dodaju nove objekte - mačku i miša.

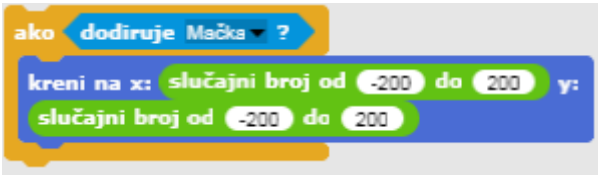


[Korak 2]
Učenici programiraju mačku da se kreće pomoću strelica na tipkovnici. Tu moraju odrediti što se događa ako je mačka na rubu.

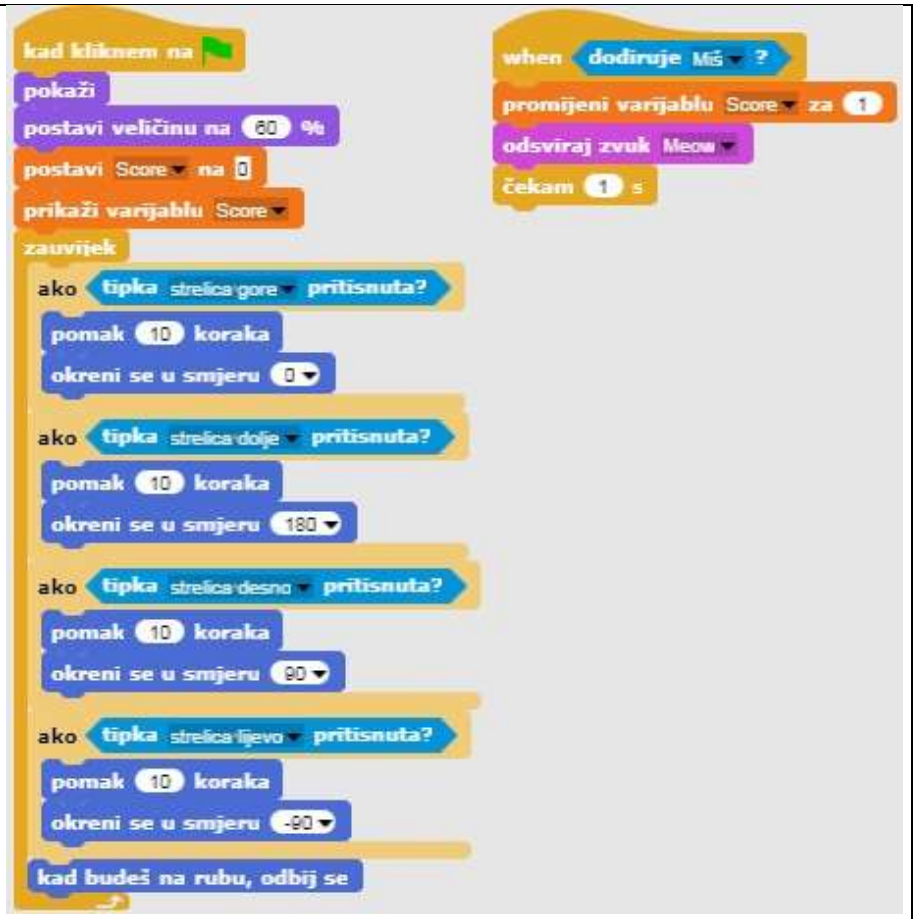


[Korak 3]
Učenik mora napraviti program u kojem se miš kreće nasumično. U našem slučaju, ideja je da miš beskonačnom petljom uzima slučajan broj koraka i zaokreće se za slučajan broj stupnjeva. Učenik to radi koristeći blok *kretanje / pomak [x] koraka* i blok *kretanje / okreni se [x] stupnjeva* u koji umeće *operator slučajan broj od [x] do [y]*.

[Korak 4]
Sljedeći korak je programirati miša da se sakrije kada ga mačka dotakne. Ideja je da se miš sakrije i pojavi na slučajnoj lokaciji kada ga mačka dotakne. U ovom slučaju, igra ne završava nakon što prvi puta mačka uhvati miša. Učenici mogu dodati svoja vlastita pravila. U svakom slučaju moraju koristiti operator *slučajan broj od [x] do [y].*



[Korak 5]
U slučaju da želimo znati broj uhvaćenih miševa, moramo dodati brojač. Učenik radi novu varijablu – score i dodaje ju u kod mačke. Score na početku igre uvijek mora biti jednak nuli. Učenici to rade pomoći bloka *varijable / postavi [varijabla] na [x].* Ukoliko želimo da se rezultat prikazuje tijekom igranja, učenici moraju dodati blok *prikaz varijable [varijabla].* Zatim učenici dodaju novi kontrolni blok (*upravljanje / when*) kako bi provjerili dodiruje li mačka miša. Ako mačka dodiruje miša, rezultat se povećava za 1 (*varijable / promijeni [score] za [x]*).

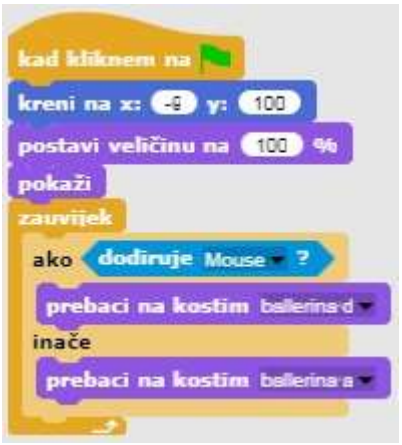


[Korak 6]
Učenik definira kada igra završava. To rade dodajući mjerač vremena. Nakon nekog vremena (npr. 30 sekundi) miš i mačka nestanu, varijabla Score je skrivena i igra je završena.

Učenik dodaje taj kod u skriptu mačke i miša.

[Korak 7]
Učenik mora dodati naredbu kojom će djevojčica rezimirati koliko je igrač bio uspješan. Ako igrač ne uhvati niti jednog miša, djevojčica kaže: “Nisi uhvatio niti jednog miša!”. Inače kaže: “Čestitam! Ulovio-la si *x* miševa!”.

[Korak 8]
Učenik može dodati bilo koji element u svoju igricu. Na primjer, djevojčicu koja skače svaki puta kada dotakne miša.



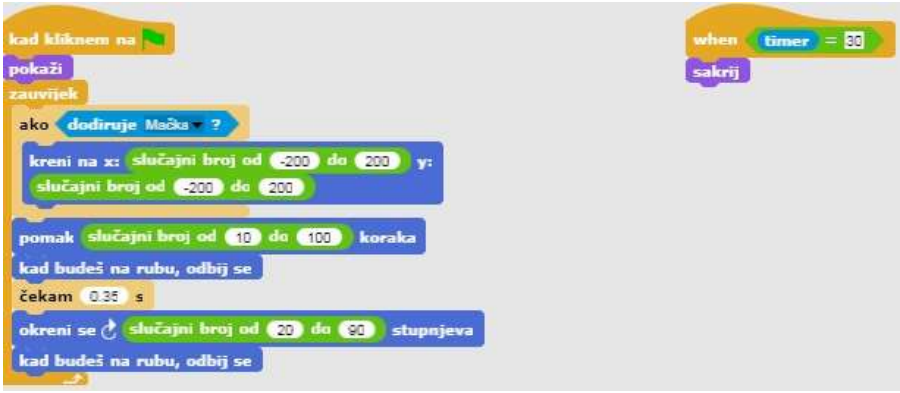
[Korak 9]
Učenici dodaju zvuk. Na primjer, mogu dodati zvuk mačke. Zvuk svira kada je miš ulovljen.

Učenici mogu doraditi kod i omogućiti sljedeće:

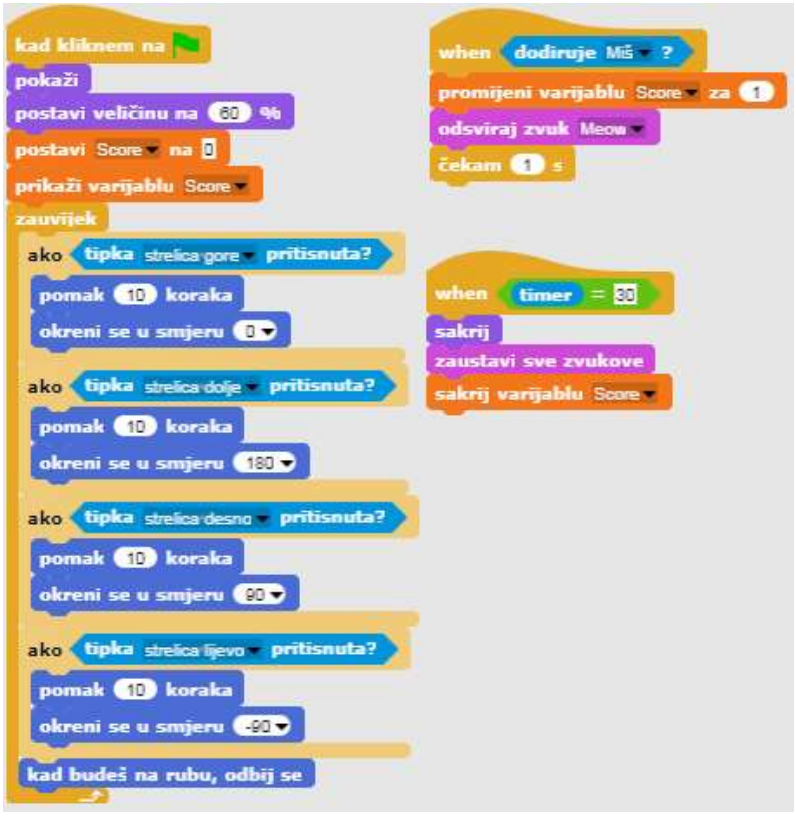
- Miš se pomiče od 20 do 60 koraka
- Kada mačka dotakne miša, miš se pomakne na lokaciju x=100
- Miš se zauvijek okreće za 90 stupnjeva
- ....

[Završni kod]

Miš

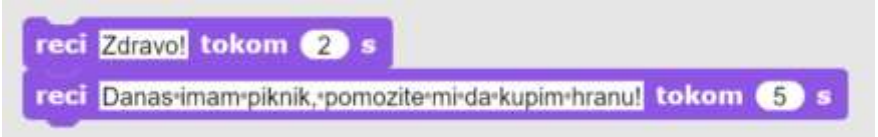


Djevojčica

	Mačka  Pozadina  **[Dodatni zadatak]** Učenici mogu upotpuniti igru dodatnim elementima po želji. Na primjer, mogu dodati djevojčicu koja će skočiti svaki puta kada dotakne miša.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse Stranica s besplatnim slikama: https://pixabay.com/ Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen u alatu Snap!: https://snap.berkeley.edu/project?user=tadeja&project=Catch%20the%20mouse_0 Stranica s besplatnim slikama: https://pixabay.com/

Scenarij učenja 16 – Kupnja hrane za piknik

Naziv scenarija	Kupnja hrane za piknik
Potrebno predznanje iz programiranja	Dodavanje teksta objektu Pokazivanje i skrivanje objekata Korištenje operatora Korištenje varijabli Korištenje stringova Korištenje uvjeta
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Operatori Ishodi učenja: ● Učenik će koristiti varijable za postavljanje cijene za različite objekte ● Učenik će mijenjati vrijednost varijabli jer se budžet mijenja kada igrač kupuje hranu ● Učenik će koristiti naredbu *ako* za provjeru dostupnosti novca ● Učenik će koristiti operatore za spajanje *tekst - vrijednost varijabli - tekst* ● Učenik će koristiti operatore za usporedbu cijena i budžeta ● Učenik će koristiti operatore (oduzimanje) za promjenu vrijednosti varijablama
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Djevojčica ide na izlet i treba joj pomoć oko kupnje hrane. Ima 15 kuna i ne može potrošiti više. Kada nešto kupi, vrijednost budžeta se promijeni. Ako joj je budžet prenizak ne može kupiti odabranu hranu. **Zadatak:** Učenici moraju programirati tri različita objekta: djevojčicu, hranu (koju mogu duplicirati uz male promjene) i gumb za završetak. Djevojčica daje upute, govori koliko novaca igrač ima i na kraju (klikom na gumb za završetak) govori koliko je zdravih i nezdravih proizvoda igrač kupio. Prelaskom miša preko odabranog proizvoda ispisuje se njegova vrijednost te klikom na proizvod, ako igrač ima dovoljno novca, vrijednost budžeta se mijenja. Inače se hrana ne može kupiti. **Cilj:** Učenici će naučiti kako raditi sa varijablama: postavljanje

	različitih početnih vrijednosti, korištenjem uvjeta za usporedbu vrijednosti varijabli, promjenom vrijednosti varijabli, korištenjem varijabli za brojanje (ne) zdrave hrane. Osim toga, ponovit će dodavanje i spajanje teksta te naredbu *ako*.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Game Based Learning – izrada igre Rješavanje problema
Oblici poučavanja	Individualni rad/rad u paru
Razrada aktivnosti	Djevojčica je u trgovini i želi kupiti hranu za piknik. Ima 15 kuna. Prelaskom miša preko proizvoda može vidjeti cijenu hrane i kupiti je klikom. Hranu može kupiti samo dok ima dovoljno novaca. Klikom na gumb *Finish*, djevojčica govori koliko je zdravih i nezdravih proizvoda igrač kupio.  [Korak 1] Aktivnost je osmišljena za individualni rad ili rad u paru. Učitelj daje upute, objašnjava neke teže dijelove i pomaže kad je potrebno. Učenici biraju pozadinu i dodaju glavni objekt, npr. djevojčicu. Djevojčica daje upute na početku, npr .: 

[Korak 2]

Za ovu igricu trebat će nam nekoliko varijabli:

- *budžet*, za postavljanje dostupnog novca,
- *završi*, za završetak igre,
- *zdrava_hrana*, za računanje koliko je zdrave hrane igrač kupio,
- *nezdrava_hrana*, za brojanje koliko je nezdrave hrane igrač kupio,
- varijabla za svaku hranu, npr. *lubenica_cijena*, za određivanje cijene pojedine hrane.

Na početku je varijabla budžeta postavljena na npr. 15 (Kn). Ostale tri varijable postavljene su na 0. Ovaj se kôd može dodati prije koda iz [Korak 1].

[Korak 3]

Učenici dodaju objekt (hranu) i biraju njegov kostim.
kôd hrane (lubenice) treba tri kontrolna događaja:

a) *Kad se klikne zelena zastavica:* postaviti i prikazati cijenu hrane. Neka je cijena varijable razumno određena (naravno, ne 0, ali veća od 1).

b) *Prelaskom miša preko proizvoda:* prikazati igraču koliko košta proizvod. Učenici mogu upotrijebiti blok *Izgleda-razmišljaj* koristeći spajanje *tekst - vrijednost varijable - tekst*, npr .:

c) *Kad se klikne na određeni proizvod*

Postavljamo sljedeća pitanja:

1) U kojem slučaju igrač može kupiti proizvod, a u kojem ne?
2) Što se događa s budžetom kada igrač kupi hranu?
3) Kako računamo kupljene proizvode?
4) Što se događa s hranom na polici kada ju igrač kupi?

1) Igrač može kupiti proizvod ako ima dovoljno novaca. Na taj način učenici moraju usporediti dvije varijable: *budžet* i *lubenica_cijena*. Ako lubenica košta više nego što je imao, ne može ju kupiti. Učenici mogu dodati neki tekst kako bi rekli igraču da ne može kupiti ovaj proizvod.

2) Ako igrač ima 15 Kn i kupi lubenicu za 4 Kn, sada ima 15 - 4 = 11 Kn. Dakle, vrijednost budžeta je sada: *budžet - lubenica_cijena*.

Učenici mogu i ovdje dodati neki tekst.

3) Prebrojavanje broja kupljenih proizvoda ostvarit će se promjenom varijable zdrava_hrana u 1.

4) Kad se hrana klikne, ona se sakrije.

Jedno moguće rješenje je:

[Korak 4]

Da bi na policama bilo više hrane, učenici mogu duplicirati objekt lubenice. Recimo da će druga hrana biti torta. Kôd iz [Korak 3] tada treba neke izmjene. Učenici moraju:

- promijeniti kostim
- napraviti novu varijablu: *torta_cijena*
- postaviti *torta_cijena* na neku vrijednost
- promijeniti u kodu svaki blok *lubenica_ cijena* s *torta_cijena*
- promijeniti odgovor o kupnji torte
- zamijeniti *promjeni varijablu zdrava_hrana za 1* u *promijeni nezdrava_hrana za 1.*

Na primjer, kôd za kolač kada se klikne može biti:

[Korak 5]

Kada igrač završi kupnju, klikne na gumb *finish*. Da bismo rekli programu da je igrač kliknuo gumb (*finish*), varijablu *završi* postavimo na 1.

[Korak 6]

Na kraju se vraćamo u objekt da mu djevojčica kaže koliko je zdravih i nezdravih proizvoda kupio.
To će se učiniti provjerom je li igrač pritisnuo gumb *finish* – blokom *kada*. Ako je odgovor potvrdan, tada je vrijednost varijable *završi* 1, a djevojčica govori, npr. "Odabrali ste X zdrava proizvoda i Y nezdrava proizvoda".

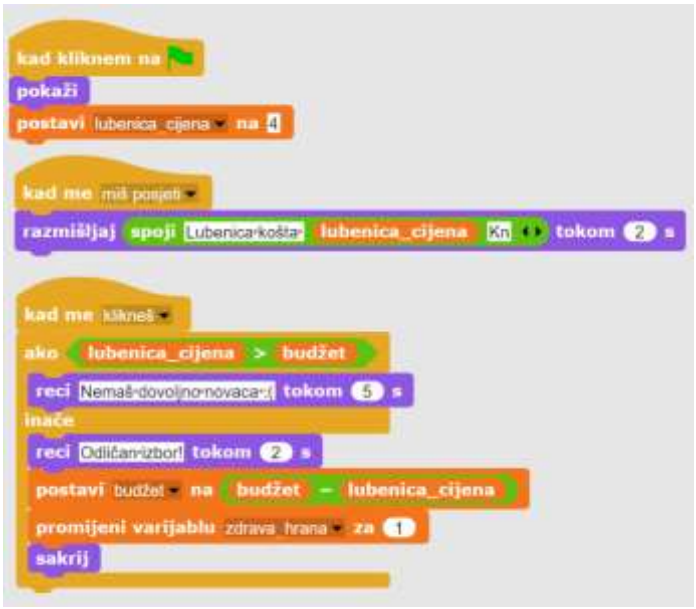
[Korak 7]

U bilo kojem trenutku tijekom igre, igrač može provjeriti njegov budžet mišem – postavljanjem miša na djevojčicu. Na primjer, ona može reći / razmišljati nešto poput:

[Cijeli kôd]

Djevojčica

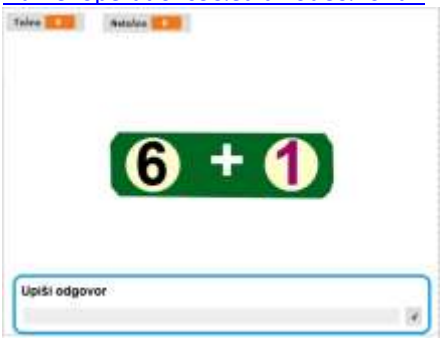
Hrana



	Gumb za završetak  **[Dodatni zadatak]** Učenici mogu upotpuniti igru po želji ili prema sljedećim prijedlozima: ● Omogućiti kupnju svake namirnice 3 puta. ● Igraču na početku dodijeliti više novca. ● Ispisati poruku na kraju igre kojom će se prikazati što je igrač kupio.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic Primjer igre s dodatnim zadacima: https://snap.berkeley.edu/project?user=mateja&project=Buying%20food%20for%20a%20picnic%20%2B%20Add.%20Task Lajovic, S. (2011). *Scratch. Nauči se programirati in postani računalniški maček.* Ljubljana: Pasadena. Vorderman, C. (2017). *Računalniško programiranje za otroke.* Ljubljana: MK.
Alati i materijali za učenike	

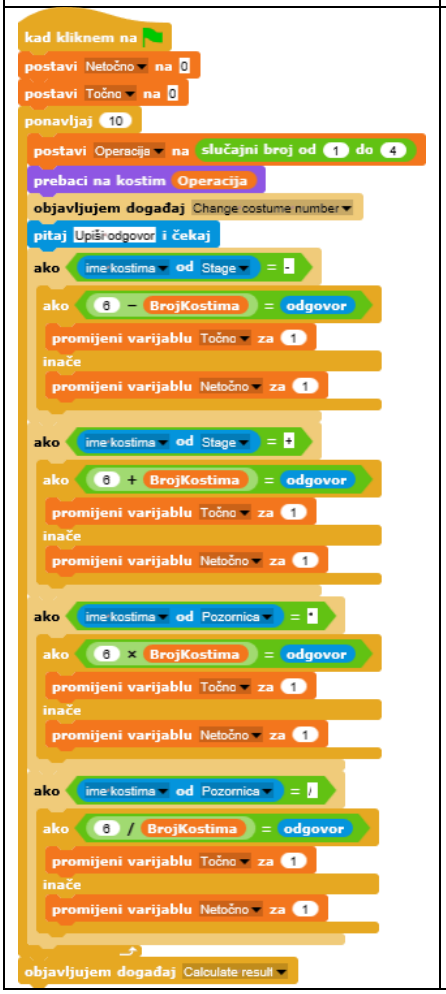
Scenarij učenja 17 – Operacije

Naziv scenarija	Operacije
Potrebno predznanje iz programiranja	Primjenjivanje varijabli za brojanje bodova i za promjenu kostima pozornice i objekta Korištenje slučajnog broja za postavljanje izgleda pozornice i kostima objekta Korištenje petlje *ponavljaj* Korištenje uvjeta Primjena operacija uspoređivanja Korištenje *osjetila* za dijalog (pitaj…i čekaj) Korištenje bloka *objavljujem događaj.*
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Petlja ● Blokovi *osjetila* ● Objavljivanje događaja Ishodi učenja: ● Učenici će moći primjenjivati varijable za brojanje bodova, za promjenu izgleda pozornice i za kostime objekta ● Učenici će moći inicijalizirati varijable za brojanje bodova ● Učenici će moći koristiti uvjete i logičke operacije ● Učenici će moći primjenjivati objavljivanje događaja za promjenu objekta i za računanje konačnog rezultata.
Cilj, zadaci i kratki opis aktivnosti	**Opis aktivnosti:** Za vrijeme igranja igrice, provjeravamo je li igrač usvojio aritmetičke operacije u Snap!-u. Pravila su sljedeća: 10 puta se bira aritmetička operacija slučajnim odabirom, prvi operand je broj 6, dok je drugi operand slučajni broj od 1 do 3. Igrač mora upisati točan odgovor. Broj točnih i netočnih odgovora se broji. Na kraju igrice izvještava se o konačnom rezultatu. **Zadaci:** Učenici trebaju izabrati scenarij, izgled pozornice i kostim objekta; osmisliti potrebite varijable, odrediti potrebite blokove. Na kraju je potrebno izraditi kod za pozornicu i za objekt. ***Dodatni zadatak bi mogao biti:*** Ovisno o rezultatu, dodijeliti objektu da kaže: „Bravo za tebe!“ ili „Nisi dobro usvojio aritmetičke operacije u Snap!-u“.

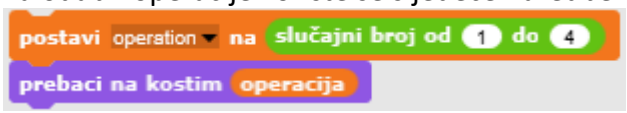
	Cilj: Učenici će proširiti znanje o varijablama, slučajnim brojevima, petljama, objavljivanju događaja.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje (diskusija, eksperiment s unaprijed pripremljenom igrom) Učenje temeljeno na igrama Učenje rješavanjem problema
Oblici poučavanja	Frontalni rad Individualni rad Rad u parovima
Razrada aktivnosti	1. Nastavnik predstavlja problem u vezi potrebe igre da utvrdi jesu li aritmetičke operacije u Snap!-u usvojene i demonstrira projekt. https://snap.berkeley.edu/snap/snap.html#present:Username=ddureva&ProjectName=operations3&editMode&noRun  2. Nastavnik komentira kako oblikovati uvjet zadatka. Slučajnim odabirom se 10 puta izabire aritmetička operacija sa prvim operandom brojem 6 i drugim operandom slučajnim brojem od 1 do 3. Igrač mora upisati točan odgovor. Točni i netočni odgovori se broje. Na kraju igrice izvještava se o konačnom rezultatu. 3. Objašnjavaju se varijable. Objašnjava se njihovo definiranje, inicijaliziranje i mijenjanje. 4. Ponovo se prolazi kroz naredbe slučajnog broja, aritmetičke i logičke operacije, te kroz objavljivanje događaja. 5. Raspravlja se o tome da li se glavni kod postavlja na skriptu od pozornice ili od objekta. U ovom primjeru, glavni kod je zapisan u skripti od pozornice, dok kod koji je zapisan u skripti od objekta ima za ulogu mijenjanje kosima i računanje konačnog rezultata.

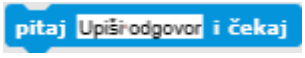

1
number2
povlačiv
Skripte
Kostimi
Zvukovi
kad spazim događaj Change costume number
postavi CostumeNumber na slučajni broj od 1 do 3
prebaci na kostim BrojKostima
kad spazim događaj Calculate result
reci spoji Konačan rezultat Točno – Netočno tokom 10 s


1
Broj
povlačiv
Skripte
Kostimi
Zvukovi
Kostim zero
Slike uvoziš povlačenjem s druge web stranice ili s računala
svg
1
1
svg
2
2
svg
3
3

Scena	Izgled pozornice
	

Kod za scenu sadrži inicijalizaciju varijabli za točan i netočan odgovor.
Za odabir operacije koriste se sljedeće naredbe:



Izbor kostima objekta se izvršava objavljivanjem događaja objektu Broj. Izabrani broj kostima se pohranjuje u varijablu pod imenom BrojKostima koja je definirana za sve objekte projekta radi čega se ista varijabla može primjenjivati unutar koda za pozornicu.
Nakon što je izgled pozornice i kostim objekta izabrano slučajnim načinom, postavlja se pitanje igraču da upiše ispravan odgovor s obzirom na operaciju naredbom: 

.

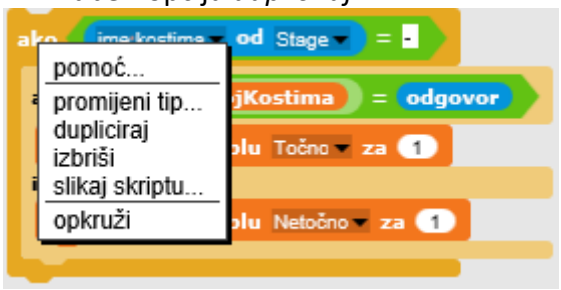
Upisani odgovor se uspoređuje s ispravnim rezultatom ovisno o izabranoj operaciji.

Upotrijebljene su sljedeće naredbe:
Ako (uvjet)
Onda
Ukoliko je operacija „-„ izabrana, provjerava se odgovora li razlika broja 6 i broja „BrojKostima“ igračevom upisanom rezultatu. Ukoliko se rezultati podudaraju, vrijednost varijable *Točno* se povećava za jedan, u suprotnom se povećava vrijednost varijable *Netočno.*

Isto vrijedi za ostale operacije.

S ciljem izbjegavanja ponovnog pisanja koda za preostale operacije, učenike se može naučiti kako kopirati kod (dio koda):

1. Desni klik na dio koda koji želiš kopirati
2. Izaberi opciju *dupliciraj*

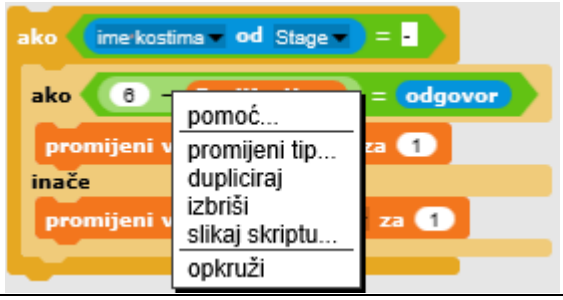


3. Koristi se mišem da dupliciranu kopiju koda smjestiš na odgovarajuće mjesto.

Nastavnik može biti diskretan i zadati učenicima da samostalno istraže kako kopirati dio koda.

Promjena operacija.

1. Desnim klikom na znak operacije. Prikazati će se izbornik.



	2. Izaberi *promijeni tip.* Prikazati će se lista operacija.  3. Izaberi operaciju. Napomena: Ukoliko učenikove godine i znanje o aritmetičkim operacijama dopuštaju, onda se zadatak može proširiti tako da se uvede potenciranje (^) i dijeljenje s ostatkom (mod). Učenici rade u timovima tako da postave izgled pozornice i kostime objekta. Ukoliko se radi o zaostatku s vremenom, učenicima se može dati polugotov projekt u kojem su pozornica i izgled objekta unaprijed postavljeni.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ddureva&project=operations3
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ddureva&project=operations_half

Scenarij učenja 18 – Recikliranje

Naziv scenarija	Recikliranje
Potrebno predznanje iz programiranja	Prikazivanje i skrivanje objekta Korištenje varijabli za brojanje bodova Korištenje petlje zauvijek Korištenje uvjeta Korištenje operacija za usporedbu Korištenje osjetljivosti boja
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Petlje ● Usmjeravanje u smjeru ● Raspoznavanje blokova naredbi ● Restrukturiranje koda Ishodi učenja: ● Učenici će moći koristiti naredbu *čekaj dok* i logičke operatore kako bi završili igru ● Učenici će moći koristiti naredbu *čekaj dok* i blokiranje za promjenu pozornice ● Učenici će moći koristiti varijable za brojanje bodova ● Učenici će moći koristiti uvjete i logičke operacije ● Učenici će moći usporediti kod sličnih objekata ● Učenici će moći restrukturirati kod ● Učenici će moći koristiti naredbe za pozicioniranje objekta
Cilj, zadaci i kratki opis aktivnosti	**Zadatak:** Učenici moraju istražiti kod pozadine i objekata, usporediti kod objekata otpadnog papira i stakla, dodati nove objekte i skripte, promijeniti postojeću skriptu pozornice s obzirnom na novo dodane objekte. **Dodatni zadatci:** ● Promijeniti položaj objekata otpada slučajnim odabirom koordinata objekta ● Smanjiti broj razina (pozornica) i izdvojiti robota kao zaseban objekt (robot je dio pozadine pozornice) **Cilj:**

	Učenici će poboljšati prethodno stečeno znanje te će proširiti scenarij igre sa novim objektima, kodom i promjenom koda s obzirom na nove objekte. Učenici će moći restrukturirati kod.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Game Based Learning – igra Razgovor Demonstracija Rješavanje problema
Oblici poučavanja	Frontalni rad Rad u paru Individualni rad
Razrada aktivnosti	Učitelj postavlja problem razdvajanja otpada i komentira boje kanti za različite vrste smeća – plava za papir, zelena za plastiku. Učitelj zadaje učenicima da odigraju igru i opišu je riječima: Koliko pozornica gledaju i koliko objekata postoji (likova)? Kako igra počinje? Koji objekt traži ime igrača? Koliko varijabli se koristi i kako se zovu? Što se događa kada se papir stavi u spremnik za staklo, a što kada se stavi u spremnik za papir?  [Korak 1] **Ažuriranje proučenih naredbi** Poziva se naredba za dijalog s korisnikom (igračem). Dodaje se komentar o promjeni scena – Scena 1 s robotom, Scena 2 sa školom i smećem te Scena 3 s robotom i natpisom ‘Bravo!’. Raspravlja se o

mogućim naredbama za promjenu scene.

Raspravlja se da provjeru ispravnog odlaganja otpada u spremnike treba provesti pomoću uvjetnog bloka i blokova s uvjetima iz skupine Sensing (dodir).

Daje se usmeni opis: ako komad papirnatog otpada dodirne spremnik za papir, otpad se sakrije (odložen je u odgovarajući spremnik) te se dodaje 1 bod za skupljeni papirnati otpad. Ako papirnati otpad dotakne spremnik za staklo, pojavljuje se poruka: "Ovo nije spremnik za papir". Isto se događa sa staklenim otpadom.

[Korak 2]

Proučavanje koda scena i objekata

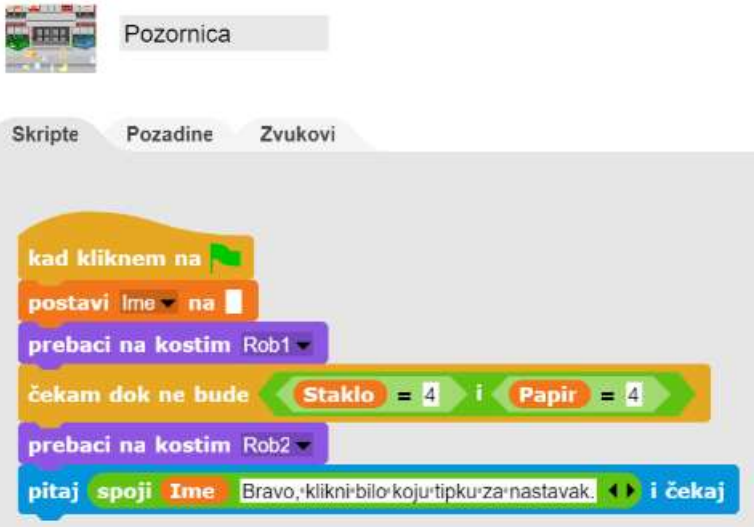
Nakon rasprave o mogućim rješenjima problema, raspravlja se o kodu za scenu i objekte.

Kod scene komentira se s naglaskom na:

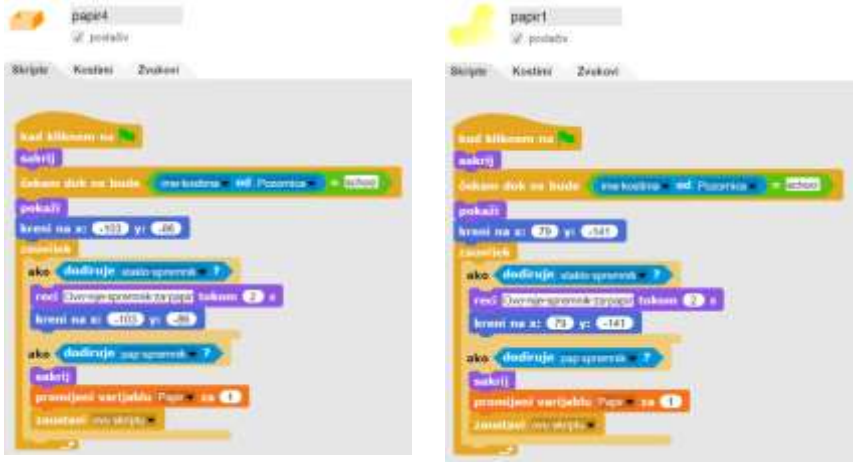
- Postavljanje početne vrijednosti varijable imena i korištenje

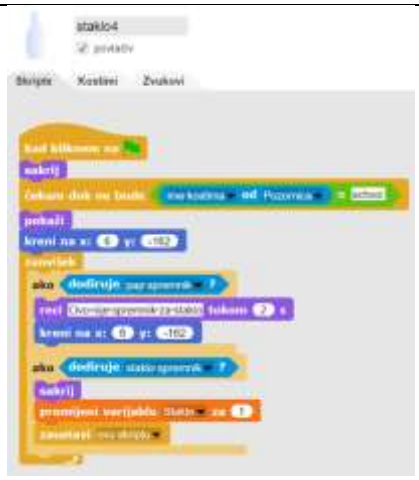
iste u dijalogu sa korisnikom;

- Promjena scenskog prizora (pozornice) i uvjet za završetak igre.



Kada gledate kod objekata, preporučljivo je prikazati ih na jednoj stranici ili po dva ispisana koda za papirnati i stakleni otpad. Radi se usporedba između sličnih i različitih elemenata u kodu.





[Korak 3]
Postavljanje zadatka za dovršetak igre s dva nova objekta – papirnati otpad i stakleni otpad, dodijelimo im kod te promijenimo scenu (pozornicu) i kod spremnika za otpad. Raspravlja se o dva nova objekta.
Opcije – Umnožite postojeće objekte i uredite ih u Snap!-u, kreirajte nove u grafičkom uređivaču ili pretražite besplatne slike na internetu i uvezite ih u igru.
Potrebno je komentirati i promjene koda scene s obzirom na završetak igre. Treba se prokomentirati i postavljanje početnih vrijednosti varijabli u kodu scene, a ne u kodu spremnika, te napraviti potrebnu prilagodbu.

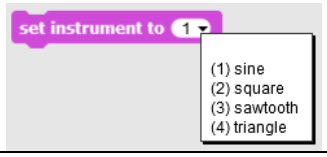
Po potrebi, učitelj može otežati zadatak:

- Otpad treba rasporediti na bilo koje prikladno mjesto prilikom pokretanja igre. Treba napomenuti da koordinate unutar kojih se otpad može rasporediti moraju biti ograničene, tako da otpad bude raspoređen na realnom mjestu. Na primjer, koordinate su ograničene crvenim pravokutnikom.

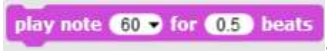
	 • Uvođenje novog objekta za robota i smanjenje broja elemenata na pozornici. Napisati odgovarajući kod za robota kako bi mogao sudjelovati u razgovoru sa korisnikom (igračem) umjesto objekta plavog spremnika.
Alati i materijali za učitelje	Igra izrađena u alatu Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling
Alati i materijali za učenike	Igra izrađena u alatu Snap!: https://snap.berkeley.edu/project?user=ddureva&project=recycling

Scenarij učenja 19.1 – Sviranje klavira

Naziv scenarija	Sviranje klavira
Potrebno predznanje iz programiranja	Korištenje varijabli za brojanje bodova Korištenje događaja *Kada sam pritisnut* Korištenje petlje za ponavljanje Korištenje uvjeta Korištenje emitiranih događaja za promjenu scenarija/uređivanje pozornice i za upravljanje aktivnostima objekata
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Petlje ● Emitiranje događaja ● Zvukovi ● Programiranje glazbe Ishodi učenja: ● Učenici će moći koristiti varijable za brojanje bodova ● Učenici će moći inicijalizirati varijable za brojanje bodova ● Učenici će moći koristiti uvjet za procjenu postignutih bodova ● Učenici će moći koristiti događaj emitiranja za promjenu scenarija / uređivanje pozornice / i za aktivnosti objekata ● Učenici će moći koristiti blokove iz grupe *Zvuk* da bi komponirali melodije ● Učenici će moći prepoznati potrebu za korištenje petlje za ponavljanje da bi se smanjio broj blokova u skriptama ● Učenici će moći proširiti funkcionalnost igre
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Uđimo u prekrasan svijet kraljice Marije. Ona poziva igrača u svoju palaču da posluša neku glazbu. U plesnoj dvorani njezin mali dinosaurus Dino svira klavir. U igri Dino svira nekoliko glazbenih tonova i igrači moraju prepoznati o kojem tonu se radi. Ako pogode, dobivaju jedan bod za ispravan odgovor, a ako ne znaju, oduzme im se jedan bod za pogrešan odgovor. Nakon prepoznavanja tonova postavlja se složeniji zadatak: Dino svira melodiju, a igrač mora prepoznati o kojoj se pjesmi radi. Igrač dobiva 5 bodova za pravilno prepoznatu pjesmu.

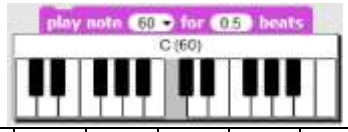
	Zadatak: Učenici koriste djelomično gotovu datoteku s scenografijom / scenskom dekoracijom / i kostima objekata. Moraju planirati potrebne varijable, odrediti koji su blokovi potrebni; upoznati se s blokovima grupe *Zvuk* i načinom sviranja nota. Stvaraju skripte za reprodukciju nekoliko melodija. **Cilj:** Učenici će upoznati kodiranje i sviranje melodija te će poboljšati svoje prethodno stečeno znanje o varijablama, petlji, uvjetnim, emitiranim i ostalim događajima.
Trajanje	90 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje (diskusije, eksperimentiranje s prethodno pripremljenom igrom) Game Based Learning - igra Rješavanje problema
Oblici poučavanja	Individualni rad / Rad u paru / Frontalni rad s cijelim razredom
Razrada aktivnosti	Učitelj postavlja zadatak stvaranja igre. Raspravlja se o sredstvima pomoću kojih se zadatak može dovršiti. Zaključeno je da trenutno nisu svjesni dostupnih resursa za pisanje koda za sastavljanje melodije. Učitelj demonstrira dio igre skladajući napjev. https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1  Učitelj pokazuje kôd i objašnjava kako se mogu koristiti naredbe grupe *Zvuk*. U programu Snap! mogu se koristiti zvukovi iz ugrađene biblioteke, kao i datoteke s računala ili glazbeni tonovi koji se sviraju na različitim instrumentima. Za odabir alata koristite naredbu: 

Učenici testiraju zvuk raznih instrumenata.
Učitelj objašnjava način postavljanja glazbenih tonova:

Koristi se naredba 

. U njemu prvi broj postavlja ton, a drugi broj opisuje koliko dugo će se ton reproducirati.

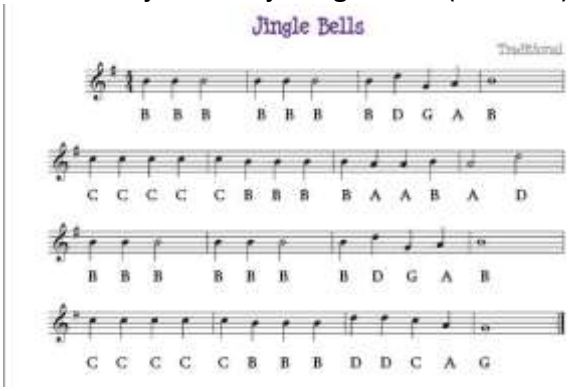
Kada kliknete strelicu pokraj prvog broja, pojavljuje se tipkovnica za klavir i iz nje se može odabrati ton. Ova klavirska tipkovnica obuhvaća dvije oktave.



C	C#	D	Eb	E	F	F#	G	G#	A	Bb	B	C
60	61	62	63	64	65	66	67	68	69	70	71	72

Trajanje svake bilješke postavlja se brojevima 1 - cijela nota, 0,5 - polovinka, 0,25 - četvrtinka. (Za učenike ne znaju, decimalni se broj može prikazati u obliku razlomaka: ½, ¼, 1/8 itd.)

Prema navođenju učitelja, učenici mogu eksperimentirati sa naredbama.
Raspravlja se o scenariju melodije Jingle Bells (Zvončići).



Zadatak je postavljen tako da se smanji broj redaka u kodu koji se ponavljaju. Raspravlja se o naredbi koja će se koristiti (petlja za ponavljanje). Učenici su podijeljeni u timove koji su potrebni za kreiranje igre postavljene na početku predavanja. Svaki tim raspravlja o scenariju igre i opisuje plan igre (detaljan opis objekata i aktivnosti na pozornici). Može se dodati uvjet da dinosaurus može plesati tijekom sviranja. (Dinosaurus ima nekoliko kostima za koje su unaprijed pripremljene datoteke).

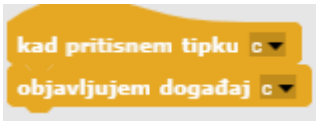
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_1 https://snap.berkeley.edu/project?user=ddureva&project=PlayAPiano
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ddureva&project=Play_a_Piano_Half_backed

Scenarij učenja 19.2 – Sviranje klavira

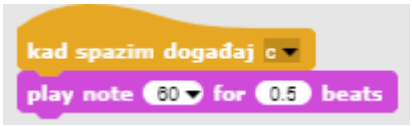
Naziv scenarija	Sviranje klavira
Potrebno predznanje iz programiranja	Korištenje petlji za ponavljanje Korištenje varijabli Korištenje uvjeta
Ishodi učenja	Koncepti programiranja: ● Uvjeti ● Petlje Ishodi učenja: ● Učenik će moći koristiti petlje za ponavljanje za reprodukciju glazbe ● Učenik će moći koristiti kôd kako bi objekti reagirali na unos ● Učenik će moći dodati zvukove objektu ● Učenik će moći koristiti kôd za promjenu kostima objektu
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Učenik mora odsvirati pjesmu na klaviru u skladu s danim notama. **Zadaci:** Učenici trebaju programirati tipke za klavir - svaka tipka mora svirati određeni ton. Na pozornici se moraju pokazati dva različita gumba, jedan za prikaz nota, a drugi za reprodukciju melodije. **Cilj:** Učenici će naučiti svirati melodiju i mijenjati kostim klikom na objekt.
Trajanje	45 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Game Based Learning - Igra Rješavanje problema
Oblici poučavanja	Individualni rad / Rad u parovima

Razrada aktivnosti	Na početku se na pozornici prikazuje klavir. Pored klavira trebaju biti dva gumba. Klikom na prvi gumb trebaju se prikazivati note i riječi pjesme, a klikom na drugi gumb treba svirati melodija koju je potrebno ponoviti. Uz to, pored glasovira bi trebao biti gumb "X", koji će ponovno pokrenuti projekt. [Korak 1] Preuzmite datoteku s materijalima. Datoteka sadrži pozadinu i sve objekte potrebne za ovaj zadatak. Na pozornicu umetnite praznu pozadinu iz materijala.  Dodajte novi objekt-tipku C i uvezite kostim "C" iz materijala.  Učinite isto za tipke D, E, F, G, A, B i stavite ih jednu za drugom.  Dodajte novi objekt – crnu tipku 1 i uvezite kostim "black_key" iz materijala. Kopirajte ove objekte 4 puta da biste dobili 5 crnih tipki i imenujte ih black key 2 do black key 5. Crne tipke moraju biti između svake dvije bijele tipke, osim E, F.  Poništite gumb "draggable", tako da se tipke ne mogu pomicati tijekom reprodukcije.

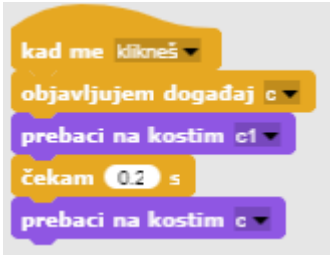
[Korak 2]
Omogućite reprodukciju tonova pritiskom na tipke. Za tipku "C" dodajte šešir "Kad pritisnem tipku" i dopustite mu da emitira poruku "c".



Za stvaranje zvuka kad pritisnete tipku, dodajte šešir na blok "Kad spazim događaj c" i odsviraj ton 60 sa 0.5 otkucaja.



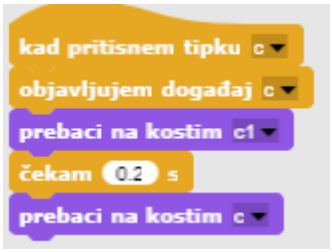
Kako biste istaknuli koja se tipka pritisne, kostim tog objekta treba privremeno promijeniti. Uvezite c1 kostim u objekt C. U bloku "Kad me klikneš" promijenite kostim u c1 za 0,2 sekunde, a zatim se vratite u kostim c.



[Korak 3]
Ponovite korak 2 za ostale bijele tipke. Definirajte da ostale tipke sviraju sljedeće tonove:
D - 62
E - 64
Ž - 65
G - 67
A - 69
B - 71

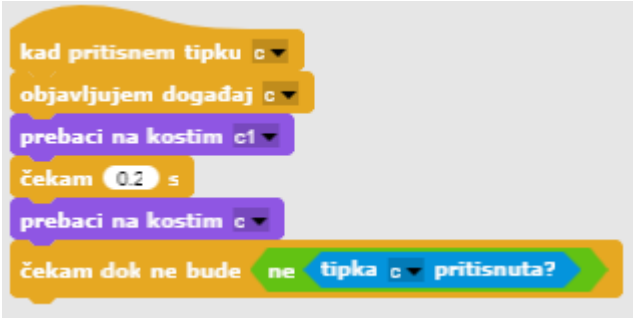
[Korak 4]
Za reprodukciju klavira pomoću tipkovnice, dodajte blok "Kad

pritisnem tipku c" na tipku c objekta i kopirajte ostatak koda iz bloka "Kad me klikneš".



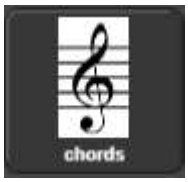
Opazite da ako je pritisnuta tipka c na tipkovnici, zvuk će se ponavljati sve dok držite pritisnutu tipku. To se događa jer se poruka "c" ponovno emitira. Kako biste prestali emitirati poruku, na kraju koda dodajte blok "čekam dok ne bude" s kontrolne ploče.

Za završetak emitiranja poruke koristite operatora "ne" i dodajte im blok "tipka c pritisnuta".



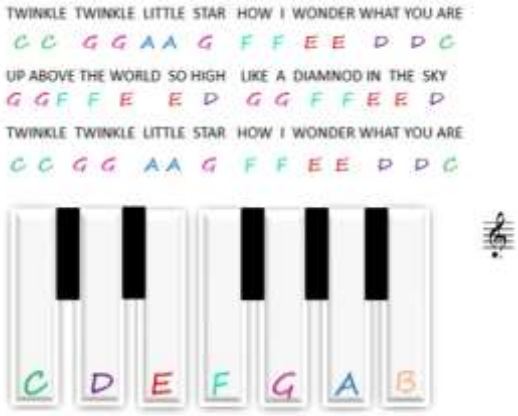
Učinite isto i za preostale bijele tipke.

[Korak 5]
Napravite novi objekt i uvezite sliku violinskog ključa kao kostim. Ovo će biti gumb za prikaz riječi i bilješki koje se trebaju svirati.



Da biste prikazali bilješke, omogućite emitiranje poruke "chords" kad se klikne gumb.

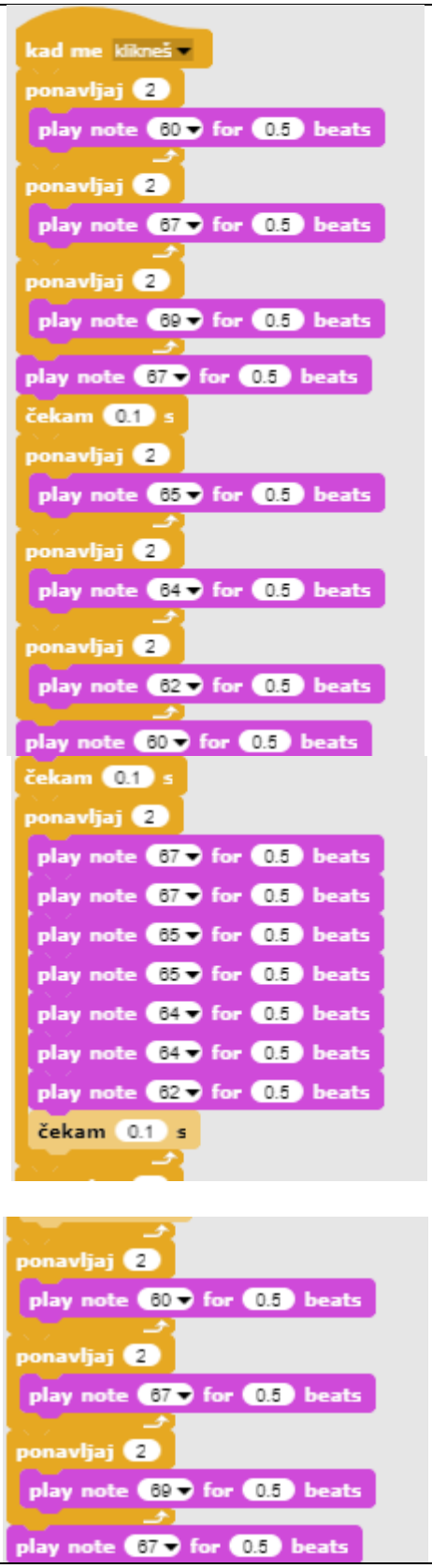
Uvezite novi kostim „chords" za pozornicu.

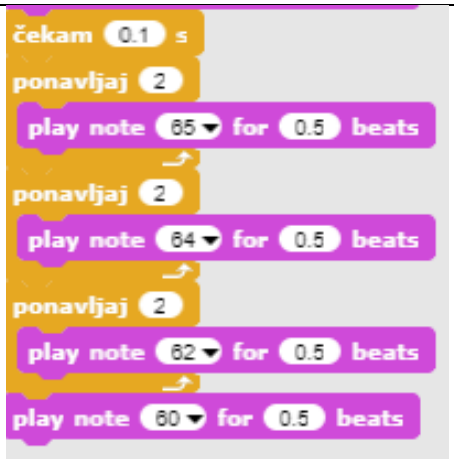


Dodajte kôd koji omogućava pozornici da promijeni kostim u "chords" kad primi poruku "chords".

[Korak 6]
Napravite novi objekt i uvezite sliku tipke s notama kao kostim. Ovo će biti gumb za reprodukciju pjesme koju je potrebno ponoviti.

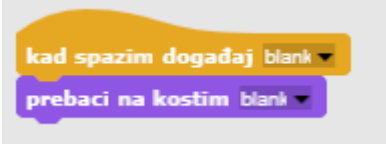
Dodajte blok "Kad me klikneš", a nakon toga bilješke koje će se reproducirati. To je ista pjesma koja je prikazana u notama.


kad me klikneš
ponavljaj 2
play note 60 for 0.5 beats
ponavljaj 2
play note 67 for 0.5 beats
ponavljaj 2
play note 69 for 0.5 beats
play note 67 for 0.5 beats
čekam 0.1 s
ponavljaj 2
play note 65 for 0.5 beats
ponavljaj 2
play note 64 for 0.5 beats
ponavljaj 2
play note 62 for 0.5 beats
play note 60 for 0.5 beats
čekam 0.1 s
ponavljaj 2
play note 67 for 0.5 beats
play note 67 for 0.5 beats
play note 65 for 0.5 beats
play note 65 for 0.5 beats
play note 64 for 0.5 beats
play note 64 for 0.5 beats
play note 62 for 0.5 beats
čekam 0.1 s
ponavljaj 2
play note 60 for 0.5 beats
ponavljaj 2
play note 67 for 0.5 beats
ponavljaj 2
play note 69 for 0.5 beats
play note 67 for 0.5 beats

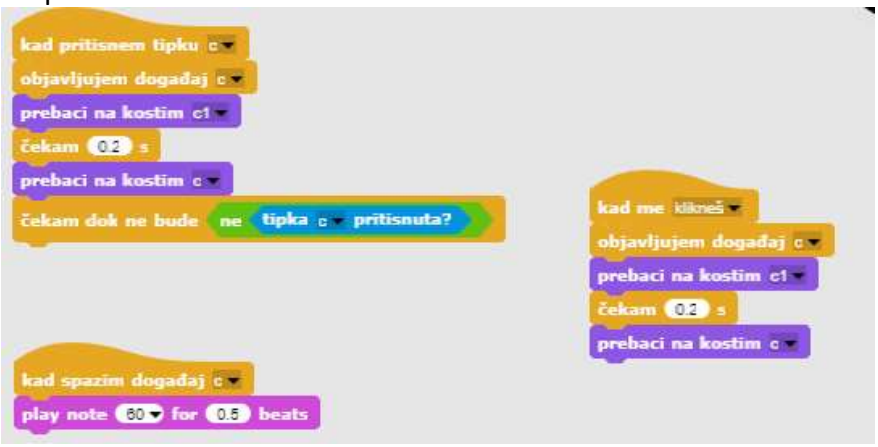


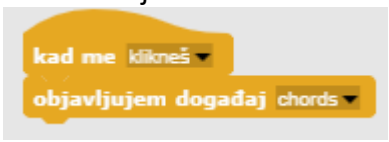
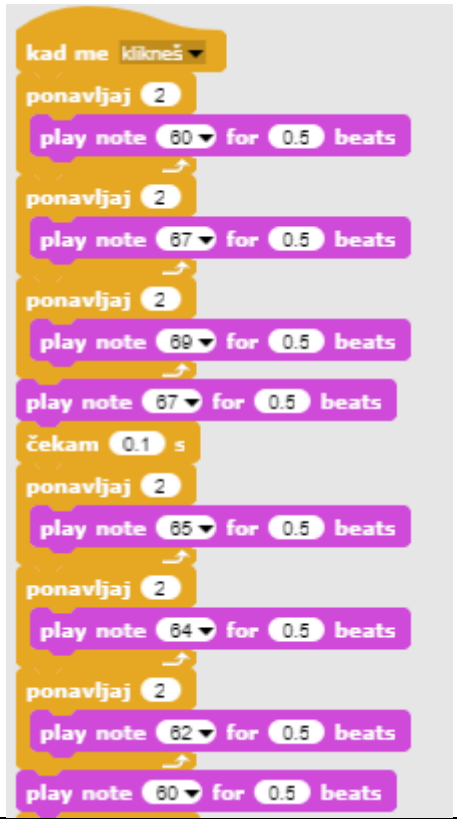
[Korak 7]
Stvorite novi gumb X koji će resetirati projekt (bez napomena).
Stvorite novi objekt - resetirajte, odaberite kostim "X" i postavite njegovu veličinu na 50%. Omogućite emitiranje "prazne" poruke kada pritisnete gumb.

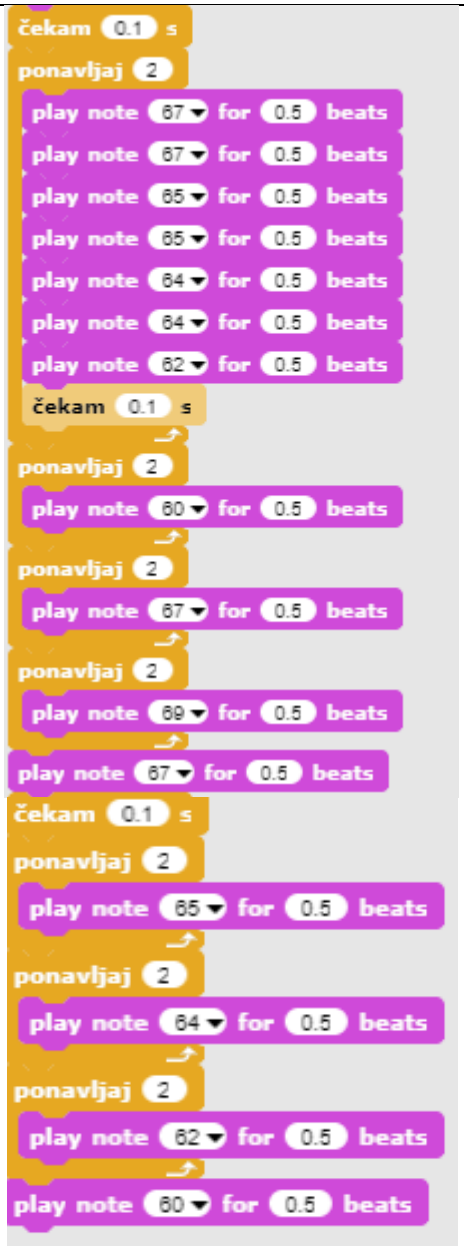
Dodajte blok "Kad spazim događaj" na pozornicu da promijeni kostim u "prazno" nakon primitka poruke "prazno".



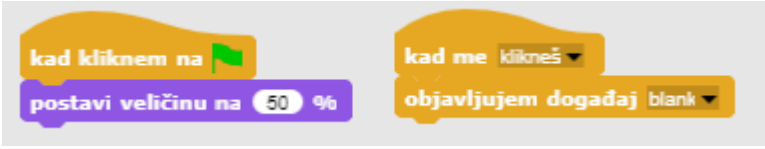
[Konačni kod]
C tipka



	Violinski ključ  Nota 


čekam 0.1 s
ponavljaj 2
play note 67 for 0.5 beats
play note 67 for 0.5 beats
play note 65 for 0.5 beats
play note 65 for 0.5 beats
play note 64 for 0.5 beats
play note 64 for 0.5 beats
play note 62 for 0.5 beats
čekam 0.1 s
ponavljaj 2
play note 60 for 0.5 beats
ponavljaj 2
play note 67 for 0.5 beats
ponavljaj 2
play note 69 for 0.5 beats
play note 67 for 0.5 beats
čekam 0.1 s
ponavljaj 2
play note 65 for 0.5 beats
ponavljaj 2
play note 64 for 0.5 beats
ponavljaj 2
play note 62 for 0.5 beats
play note 60 for 0.5 beats

X


kad kliknem na
postavi veličinu na 50 %
kad me klikneš
objavljujem događaj blank

	Pozornica  **[Dodatni zadatak]** Učenici mogu upotpuniti igru po želji ili koristeći sljedeće prijedloge: • Duplicirajte objekt Nota, promijenite joj poziciju i napišite program za drugu pjesmu. • Dodajte pozadinu s notama za drugu pjesmu.
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20a%20Piano
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ifrankovic&project=Play%20Piano Slike: Slike objekata: • a.png, a1.png • b.png, b1.png • c.png, c1.png • d.png, d1.png • e.png, e1.png • f.png, f1.png • g.png, g1.png • black_key.png, violin_key.png, note.png Pozadine: blank.png, notes.png

Scenarij učenja 20 – Test

Naziv scenarija	Test
Potrebno predznanje iz programiranja	Prikaz i skrivanje objekata Korištenje varijabli za brojanje bodova Korištenje petlje zauvijek Korištenje uvjeta Korištenje operacija za usporedbu Upotreba blokova Očitanja Promjena faze
Ishodi učenja	Koncepti programiranja: ● Varijable ● Uvjeti ● Petlja ● Blokovi Očitanja Ishodi učenja: ● Učenik će koristiti uvjet za procjenu odgovora - točno ili netočno ● Učenik će koristiti blokove za promjenu pozornice u kostimu ● Učenik će koristiti varijable za brojanje bodova ● Učenik će koristiti logičke operacije ● Učenik će koristiti vanjski grafički uređivač za pripremu složenih pozadina pozornica.
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Pomozite svom učitelju da testira vaše znanje o Snap!-u stvaranjem testa za provjeru znanja o naredbama. **Zadatak:** Učenici moraju istražiti primjer igre, izabrati iz „polugotovih“ igara, pronaći ili osmisliti vlastiti objekt (*sprite*) koji će postavljati pitanja ili osmisliti pozadinu početne pozornice i pozadinu pozornice s odgovarajućim pitanjima, izmijeniti i proširiti skripte u testu s obzirom na pitanja. **Cilj:** Učenici će poboljšati svoja stečena znanja i proširit će scenarij igre s novom pozadinom, kodom i promjenom koda s obzirom na nove pozornice.
Trajanje	90 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje (rasprava, eksperimentiranje s unaprijed pripremljenom igrom) Game Based Learning – izrada igre Rješavanje problema

Oblici poučavanja	Frontalno poučavanje Individualni rad/ rad u paru
Razrada aktivnosti	1. Učitelj postavlja problem potreban za stvaranje igre-testa za provjeru znanja programiranja. 2. Zadaje učenicima da igraju igru i opisuju riječima: Koliko scena mogu uočiti, a koliko objekata (likova)? Kako igra počinje? Koliko se koristi varijabli, kako se imenuju, za što se koriste? Što se događa kada je odgovor točan/netočan? Kako su pitanja predstavljena u testu? Po izboru učitelja može se koristiti individualni rad ili rad u parovima. 3. Komentirajte algoritam postavljanja i odgovora na pitanja. ● Prelazak na scene (sadrži pitanje) ● Dodjela Abby kostima za postavljanje pitanja ● Abby kaže - Odgovorite Da ili Ne ● Igrač unosi odgovor - Da ili Ne ● Ako je odgovor točan, Abby kaže "*Točno*", a broj točnih odgovora se povećava; Inače Abby kaže "*Netočno*" i broj pogrešnih odgovora se povećava. 4. Komentirajte što se događa nakon što odgovorite na sva pitanja. ● Promjena kostima / pozadine na pozornici; ● Abby pokazuje broj ispravnih i pogrešnih odgovora i daje ukupnu procjenu. 5. Ispitivanje kodova u igri/Modificiranje postojećeg znanja Komentiraju se uvjetne naredbe, naredbe za vođenje dijaloga s korisnikom, za promjenu scena te kostima lika. Pregledavaju se kodovi svakog lika, a stvaranje varijable se komentira.


START
start2
povlačiv
Skripte
Kostimi
Zvukovi
kad kliknem na
pokaži
kad me pritisneš
sakrij
objavljujem događaj Start


Stage
Skripte
Pozadine
Zvukovi
kad kliknem na
prebaci na kostim room2

Komentiraju se situacije kada je točan odgovor *Da* i kada je točan odgovor *Ne*.

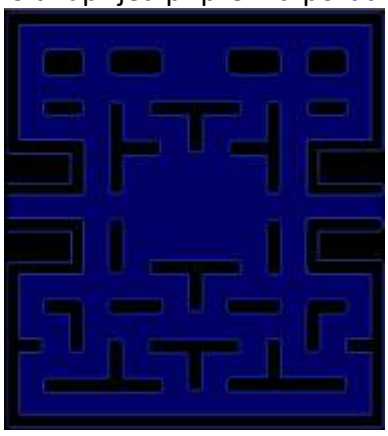
Kôd za ocjenjivanje detaljno je obrađen, kao i zašto se koristi varijabla *Ukupno*.

Raspravlja se o načinu oblikovanja scenske pozornice za pojedinačna pitanja.
U Snapu! nije moguće pisati tekst u kostimima i u sceni pa je potrebno koristiti vanjski grafički uređivač. Druga mogućnost je korištenje MS PowerPoint-a za stvaranje pitanja i izvoz odgovarajućeg tekstualnog okvira u grafičkom obliku.

	Umetanje kostima u Snap! se može ponoviti. Proces izrade testa: 1. Podjela grupe na timove od 2 ili 3 učenika. 2. Objavljivanje teme za testna pitanja. Na primjer - korištenje varijabli, petlje, kretanje, očitanja, aritmetičke i logičke operacije. 3. Dizajniranje scena s pitanjima o temi od strane odgovarajućeg tima. Ako je potrebno, nastavnik savjetuje učenike o sadržaju pitanja. O pitanjima se raspravlja i svaki tim stvara scenu s najmanje dva pitanja. 4. Izrada koda. Učenicima je na raspolaganju biblioteka s kostimima pozornice i objekata. Ako žele, mogu stvoriti i vlastite slike/kostime..
Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=ddureva&project=test2
Alati i materijali za učenike	Polugotova igra izrađena alatom Snap!: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_en_tmp

Scenarij učenja 21 – Pojednostavljena igra PACMAN

Naziv scenarija	Pojednostavljena PACMAN igra
Potrebno predznanje iz programiranja	Dodavanje pozadine Dodavanje novog objekta Definirati što objekt govori Kretanje objekata Slučajni broj
Ishodi učenja	Koncepti programiranja: ● Kretanje objekta obzirom na događaj ● Osjetilo jedne boje ● Čitanje logičkih vrijednosti u logičkim izrazima ● Definiranje, razlikovanje, dinamičko ispitivanje i odgovaranje na dva različita stanja u igri Ishodi učenja: ● Učenik će moći omogućiti kretanje objekta tipkama sa strelicama koristeći događaje i uzimajući u obzir ograničenja ● Učenik će moći koristiti blok osjetilo boje kako bi dobio logičku vrijednost za očitavanje osjetila jedne boje. ● Učenik će moći prepoznati da stanje objekta može biti izraženo bojom koju objekt dodiruje ● Učenik će moći razlikovati dva različita stanja i izražava ih logičkim izrazima ● Učenik će moći koristiti petlju za ponovnu promjenu trenutnog stanja kada se položaj objekta dinamički mijenja ● Učenik će moći upotrebljavati ako – inače grananje za postavljanje različitih odgovora na osnovu trenutnog položaja objekta
Cilj, zadaci i kratki opis aktivnosti	**Kratki opis:** Programska igra u kojoj će glavni lik sakupljati nasumično postavljene zvijezde i pritom biti proganjan duhom. **Zadaci:** Učenici moraju programirati kretanje glavnog lika kako bi se on kretao unutar labirinta. Moraju provesti ograničenja kretanja kako se glavni lik ne bi mogao kretati kroz zidove. Zatim, moraju programirati objekt zvijezdu koja će se klonirati kada igra započne, a zatim pojaviti nasumično na novom mjestu svaki put kada ju glavni lik prikupi. Moraju pohraniti vrijednosti prikupljenih zvijezda te završiti igru kada igrač sakupi 20 zvijezda. Kako bi igra bila zanimljivija, učenici moraju isprogramirati zlog duha koji će se nasumično kretati po labirintu. Ako glavni lik dotakne duha, igra je

	gotova. **Cilj:** Ovom aktivnošću učenici će preispitati svoje znanje o kretanju unutar labirinta, koristeći blok osjetilo boja koje su naučili u prethodnim aktivnostima. Upoznat će se s konceptom kloniranja objekta s ograničenjima položaja, te stvaranjem jednostavnih likova, koji nisu igrači, sa vlastitim nasumičnim kretanjem.
Trajanje	90 minuta
Strategija i metode učenja i poučavanja	Aktivno učenje Suradničko učenje Rješavanje problema
Oblici poučavanja	Frontalni rad Rad u paru Individualni rad Grupni rad (svi učenici)
Razrada aktivnosti	Učenicima se prezentira igra koju trebaju izraditi: Igrač skuplja nasumično postavljene zvijezde dok ga progoni crveni duh. Ako se igrač i duh sudare, igra je gotova. Ako igrač sakupi 20 zvijezda, on pobjeđuje. [Korak 1] Upućujemo učenike da dizajniraju labirint, gdje je područje unutar kojeg se igrač može kretati jedne boje (npr. plave), a zidovi koji ograničavaju kretanje igrača druge boje (npr. crne). Kako bi uštedili na vremenu možemo unaprijed pripremiti pozadinsku sliku labirinta.  [Korak 2] Učenici trebaju nacrtati lik *pacman* i crvenog duha. Kao zvijezdu mogu nacrtati krug unutar Snap!-a:  [Korak 3]

Kako bi napravili kretanje pacman-a, možemo koristiti različite mogućnosti. Primjer ispod je jedan od njih. U njemu koristimo operacije upravljanja za određivanje događaja, tj. tipke koja je pritisnuta: lijevo, desno, gore ili dolje. Nakon svakog od događaja, moramo testirati dodiruje li lik boju područja unutar kojeg mu je dozvoljeno kretanje. U tom slučaju, najprije se okrene u smjeru kretanja i pomakne se. Ukoliko dotakne boju zida, mora se pomaknuti unatrag jer bi u protivnom zapeo u zidu zbog prvog slučaja.

[Korak 4]

Sljedeći korak je isprogramirati zvijezde. Zvijezde će biti sve jednake, ali će ih biti mnogo. U ovom slučaju, umjesto izrade više jednakih objekta (u našem slučaju 20), bolje je napraviti jedan objekt pa zatim izraditi njegove klonove. Na početku igre prvi klon će se pojaviti nasumično unutar labirinta. Kada ga zatim igrač pokupi, nestat će i novi će biti napravljen na drugoj nasumičnoj lokaciji. Kako bi izradili prvog klona na samom početku igre, stavljamo sljedeći kod na scenu skripte.

Kako bi sakrili originalni objekt i prikazali samo klonove, moramo ovo napraviti na samom početku programa.

Kako bi pronašli odgovarajuću nasumičnu lokaciju moramo postaviti određene uvijete. Ako je zvijezda kreirana na zidu, igrač je ne može doseći, što znači da je ne smijemo postaviti na zid. Strategija za postavljanje je sljedeća.

1. Moramo pronaći nasumičnu x,y poziciju klona zvijezde. x i y koordinata su unutar intervala [-140, 140]. Stoga izabiremo nasumičan broj iz intervala za obje koordinate.
2. Zatim provjerimo dodiruje li klon boju zida. U tom slučaju lokacija nije dozvoljena.
3. Ako je lokacija dozvoljena, moramo prikazati klona (prisjetite se, originalni objekt je skriven i klon bi isto bio skriven kada ne bi koristili blok za prikaz objekta) i u beskonačnoj petlji provjeriti dolazi li do preklapanja sa igračem.
4. Ako lokacija nije dozvoljena, izradimo novog klona (nadajući se da će novo odabrani nasumični brojevi biti takvi da je klon postavljen na dozvoljenu lokaciju) i obrišemo postojećeg.
5. Kako bi izbrojali prikupljene klonove moramo dati informaciju brojaču zvijezda koji mora biti definiran izvan klona, npr. na igraču. To može biti napravljeno na način da se prenese poruka da je došlo do preklapanja. Zatim je možemo izbrisati.

[Korak 5]
Nadalje programiramo duha. Duh se mora kretati nasumično kroz labirint i mora promijeniti smjer kada dođe do zida. Kako bi napravili njegovo kretanje nasumičnim, želimo da se kreće u nasumičnom smjeru nakon što dođe do zida. U Snap!-u smjer kretanja je izražen stupnjevima:

1. 0 stupnjeva – GORE

2. 180 stupnjeva – DOLE
3. 90 stupnjeva – DESNO
4. 270 stupnjeva – LIJEVO

Drugim riječima, ako slučajno odabrani broj od 0 do 3 pomnožimo sa 90 dobit ćemo nasumični smjer kretanja.
Duh se mora kretati dok se ne sudari sa pacman-om. Tada je igra gotova.

[Korak 6]
Sada trebamo napisati program kada igrač pobjeđuje u igri. To će se dogoditi kada igrač sakupi 20 zvijezda. Brojač zvijezda imamo unutar pacman skripte. Na samom početku postavimo inicijalnu vrijednost na 0, i zatim joj povećavamo vrijednost za 1 svaki puta kada klon pošalje poruku da je igrač sakupio zvijezdu. Ako brojač dođe do 20, pacman pobjeđuje i moramo zaustaviti igru.

Alati i materijali za učitelje	Igra izrađena alatom Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_clone Lajovic, S. (2010). Scratch. Nauči se programirati in postani računališki maček. Ljubljana: Pasadena. Vorderman, C. (2017). Računalniško programiranje za otroke. Ljubljana: MK.
Alati i materijali za učenike	Predložak izrađen alatom Snap!: https://snap.berkeley.edu/project?user=zapusek&project=pacman_template

PRILOG 2. KODOVI SCENARIJA UČENJA

OSNOVNI SCENARIJI UČENJA

Br.	Naziv	Kôd
1	**Uvod u sučelje alata Snap!** Učenik dodaje nove objekte, dodaje kostime objektima, uređuje kostime te ih briše. Učenik stvara nove pozadine na pozornici, uređuje ju, te neželjene briše.	**start_hr**
2	**Otkrijte Snap!: kretanje sprite** Učenici otkriva gdje pronaći programske blokove i povezati ih u niz. Učenici će naučiti kako micati objekt i omogućiti da objekt govori.	**disc_cro**
3	**Kretanje po pozornici** Učenici će vidjeti kako izraditi program u kojem će se objekt kretati u smjeru x, te izraditi program u kojem će se objekt kretati u smjeru y.	**kretanje**
4	**Mijenjanje kostima i okretanje** Učenik uči kako promijeniti kostim objekta te kako napraviti animaciju. Također uči kako se između njih može mijenjati različite vrste rotacije objekta.	**ples**
5	**Zvukovi s farme** Učenici će se upoznati kako programirati jednostavnu igru u kojoj igrač može prepoznati zvukove životinja pritiskom na određene tipke.	**farma**
6	**Kameleonov ljetni odmor** Učenici će biti upoznati s blokom naredbi osjeta boja i kako ga koristiti u logičkim izrazima da bi razlikovali dinamično promjenjiva stanja igre i dali prave odgovore.	**chameleon_cro**
7	**Pomaganje princu i princezi u pronalasku njihovih životinja** Učenici će se upoznati s crtanjem pokretom tipke. Uz to će naučiti kako koristiti uvjete kojima će spriječiti da se objekt kreće po cijelome ekranu.	**finding_hr**
8	**Crtanje s kredom** Učenici će se upoznati s crtanjem različitih oblika pomoću kôda. Naučit će koristiti petlju ponavljaj kako bi smanjili veličinu kôda i promijeniti pozadinu.	**kreda**
9	**Skupljanje otpada i čišćenje parka**	**park**

	Učenici će se upoznati kako koristiti varijable i kako kopirati blok kôda ili čak cijeli objekt.	
10	**Hranjenje mačaka** Učenici će biti upoznati sa konceptom dodjele slučajne vrijednosti varijabli unutar petlje te će razlikovati kada navedeno napravimo van petlje. Naučiti će kako dobiti, testirati i izbrojati ispravne unose igrača.	**hranjenje**
11	**Pogađanje broja mačaka u skloništu** Učenici će se upoznati s petljom ponavljaj dok i kako postaviti uvjet koji zaustavlja igru. Također će naučiti kako koristiti varijable u različitim situacijama: za pohranjivanje nasumične vrijednosti, kao brojač ili za pohranjivanje vrijednosti koju upiše igrač.	**pogodi**

NAPREDNI SCENARIJI UČENJA

Br.	Naziv	Kôd
12	**Hvatanje zdrave hrane** Učenici će naučiti kako nasumično pomicati za X koraka i odabrati položaj i također kako koristiti varijable i uvjete za sprječavanje drugih događaja.	**hvatanje**
13	**Pričam ti priču** Učenici će naučiti kako ispričati priču, kako koristiti poruke i kako promijeniti pozadinu pozornice.	**prica**
14	**Crtanje** Učenici će naučiti kako se crta pomoću Snap!-a, promijeniti boju i debljinu olovke te kako koristiti varijable i uvjete koji uzrokuju novi događaj.	**crtanje**
15	**Uhvati miša** Učenik će se upoznati s konceptom dodjeljivanja više varijabli slučajnim vrijednostima. Naučit će kako koristiti blok Operatori / slučajni broj od [x] do [y].	**miš**
16	**Kupnja hrane za piknik** Učenici će naučiti kako raditi sa varijablama: postavljanje različitih početnih vrijednosti, korištenjem uvjeta za usporedbu vrijednosti varijabli, promjenom vrijednosti varijabli, korištenjem varijabli za brojanje (ne) zdrave hrane. Osim toga, ponovit će dodavanje i spajanje teksta te naredbu ako.	**piknik**
17	**Operacije** Učenici će poboljšati svoje znanje o varijablama, slučajnim brojevima, petljama, emitiranju.	**operacije**
18	**Recikliranje** Učenici će poboljšati prethodno stečeno znanje te će proširiti scenarij igre sa novim objektima, kodom i promjenom koda s obzirom na nove objekte. Učenici će moći restrukturirati kod.	**recikliranje**
19.1	**Sviranje klavira_1** Učenici će upoznati kodiranje i sviranje melodija te će poboljšati svoje prethodno stečeno znanje o varijablama, petlji, uvjetnim,	**glazba**

	emitiranim i ostalim događajima.	
19.2	**Sviranje klavira_2** Učenici će naučiti svirati glazbu i mijenjati kostim klikom na sprite.	**klavir**
20	**Test** Učenici će poboljšati svoje znanje i proširiti scenarij igre s novom pozadinom, kodom i promjenom koda u odnosu na nove pozornice.	**test_cro**

REFERENCE

A project of the K-12 Initiative at Stanford University. (2009, 05 06). *Taking Design Thinking to School: Approaches to Integrating the Design Process in K-12 Teaching and Learning.* Retrieved 09 10, 2020, from Stanford Graduate School of Education: https://web.stanford.edu/dept/SUSE/taking-design/presentations/Taking-design-to-school.pdf

Alserri, S., Zin, N., & Wook, T. (2018). Alserri, S. A., Zin, N. A. M., & Wook, T. S. M. T.Gender-based Engagement Model for Serious Games. , , 8(4). . *International Journal on Advanced Science, Engineering and Information Technology, 8*(4), 1350-1357.

Baptista, R., & Vaz de Carvalho, C. (2010). Role Play Gaming and Learning. *Learning Technology, 12*(1).

Combéfis, S., Beresnevičius, G., & Dagiene, V. (2016). Learning Programming through Games and Contests: Overview, Characterisation and Discussion. *Olympiads in Informatics, 10*.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: A 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Doorley, S., Holcomb, S., Klebahn, P., Segovia, K., & Utley, J. (2018). *Design Thinking Bootleg.* Retrieved from https://static1.squarespace.com/static/57c6b79629687fde090a0fdd/t/5b19b2f2aa4a99e99b26b6bb/1528410876119/dschool_bootleg_deck_2018_final_sm+%282%29.pdf

Eurostat. (2020, 09 25). *ICT specialists in employment.* Retrieved 11 12, 2020, from eurostat Statistics Explined: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=ICT_specialists_in_employment#Number_of_ICT_specialists

Feldgen, M., & Clua, O. (2004). Games as a motivation for freshman students to learn programming. *34th ASEE/IEEE Frontiers in Education Conference*, (pp. 1079–1084).

Frankovic, I., Hoic-Bozic , N., & Nacinovic-Prskalo, L. (2018). Serious Games for Learning Programming Concepts. *8th Conference edition The Future of Education.* Florence, Italy.

Garris , R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation and learning. Simulation & Gaming. *An Interdisciplinary Journal of Theory, Practice and Research, 33*(4), 43-56.

Gee, J. P. (2003). *What Video Games Have to Teach Us About Learning and Literacy.* New York: Palgrave/Macmillan.

Gee, J. P. (2007). Are video games good for learning? *Curriculum & Leadership Journal, 5*(1).

Geist, E. (2016). Robots, Programming and Coding, Oh My! *Childhood Education, 92*(4), 298-304. doi:10.1080/00094056.2016.1208008

Georgieva, R., & Tuparova, D. (2019). Educatiomnal Computer Game-Based Environments for Teaching Programming and Developmnt of an Algorithmic Thinking - Comparative AnalysisITHMIC THINKING – COMPARATIVE ANALYSIS. *Mathematics and Education in Mathematics, Forty-eighth Spring Conferenceof the Union of Bulgarian Mathematicians* (pp. 150-156). Union of Bulgarian Mathematicians. Retrieved from http://www.math.bas.bg/smb/2019_PK/tom/pdf/150-156.pdf

Georgieva, R., & Tuparova, D. (2020). Design Thinking - helps in school eucation. *Anniversary International Scientific Conference "Synergeticsand ReflectioninMathematics Education"*, (pp. 341-347). Pamporovo.

Grivokostopoulou, F., Perikos, I., & Hatzilygeroudis, I. (2016). An Educational Game for Teaching Search Algorithms. *Proceedings of the 8th International Conference on Computer Supported Education (CSEDU 2016)* (pp. 129–136). Setubal. Portugal: SCITEPRESS - Science and Technology Publications, Lda.

Gross, B. (2003). The impact of videogames in education. 8(7). *First Monday, 8*(7).

Hijon-Neira, R., Velazquez-Iturbide, A., Pizarro-Romero, C., & Carrico, C. (2015). Serious games for motivating into programming. *Frontiers in Education Confere.*

IDEO Design thinking. (2008, 09 7). *Definitions of design thinking*. Retrieved 11 20, 2020, from https://designthinking.ideo.com/blog/definitions-of-design-thinking

Innovation Starter. (2015, 06 04). *What is design thinking?* Retrieved 11 20, 2020, from http://innovationstarterbox.bg/resources/kakvo-e-dizain-mislene/

Jemmali , C. (2016). May's Journey: A serious game to teach middle and high school girls programming. Worcester Polytechnic Institute. Retrieved from https://digitalcommons.wpi.edu/etd-theses/455

Johnson, C., & all, a. (2016). Game Development for Computer Science Education. *the 2016 ITiCSE Working Group Reports.*

Johnston, R. T., & de Felix, W. (1993). Learning from video games. *Computer in the Schools* , 199-233.

Kafai, Y. (1995). Making game artifacts to facilitate rich and meaningful learning. *Annual Meeting of the American Educational Research Association*, (pp. 1–20). San Francisco.

Karakehayova, M. (2019). Opportunities for Using Design Thinking in School Education. *Education and Development, 3*.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences, 47*, 1991-1999.

Kelleher , C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. *Proc. CHI 2007.*, (pp. 1455-1464).

Luka, I. (2014). Design thinking in pedagogy. 2014, 5,. *The Journal of Education, Culture, and Society, 5*, 63-74.

Malliarakis, C., Satratzemi, M., & Xinogalos, S. (2014). CMX: Implementing an MMORPG for learning programming, 8th European Conference on Games Based Learning. *8th European Conference on Games Based Learning - ECGBL 2014.* Berlin.

Malone, T. (1981). What makes computer games fun? *Byte, 6*(12), 258-277.

Malone, T. W. (1981b). Toward a theory of intrinsically motivating instruction. *Cognitive Science, 4*.

Mathrani, A., Christian, S., & Ponder-Sutton, A. (2016). PlayIT: Game Based Learning Approach for Teaching Programming Concepts. *Educational Technology & Society, 19*(2), 5-17.

Meerbaum-Salant , O., Armoni, M., & Ben-Ari, M. (2013). Learning computer science concepts with Scratch. *Computer Science Education, 23*(3), 239-264. doi:10.1080/08993408.2013.832022

Michael, D. R., & Chen, S. (2006). *Serious Games: Games That Educate, Train, and Inform.* Boston: Course Technology PTR.

Miljanovic, M., & Bradbury, J. (2016). Robot on!: a serious game for improving programming comprehension. *Proceedings of the 5th International Workshop on Games and Software Engineering.* Austin, Texas, USA.

Miljanovic, M., & Bradbury, J. (2018). A Review of Serious Games for Programming. *4th Joint International Conference on Serious Games.* Darmstadt, Germany.

Naiman, L. (2019, 06 10). *Design Thinking as a Strategy for Innovation*. Retrieved 11 21, 2020, from Creativity at work: https://www.creativityatwork.com/design-thinking-strategy-for-innovation/

Nančovska Šerbec, I., Kaučič, B., & Rugelj, J. (2008). Pair programming as a modern method of teaching computer science. *International journal on emerging technologies in learning, 3*, 45-49.

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment. , 191, . *Procedia - Social and Behavioral Sciences, 191*, 1479-1482. doi:doi:10.1016/j.sbspro.2015.04.224

Paavola, S., & Hakkarainen, K. (2005). The Knowledge Creation Metaphor – An Emergent Epistemological Approach to Learning.14. *Sci Educ, 14*, 535-546.

Phan, M., Jardina, J., Hoyle, S., & Chaparro, B. (2012). Examining the role of gender in video game usage, preference, and behavior. *Human Factors and Ergonomics Society 51*, (pp. 1496–1500.).

Pivec, M., & Kearney, P. (2007). Games for Learning and Learning from Games. *Informatica, 31*, 419-423.

Pivec, M., Koubek, A., & Dondi, C. (2004). *Guidelines for Game-Based Learning. Lengerich.* Pabst Science Publishers.

Prensky , M. (2001). *Digital Game-Based Learning.* New York: Mc-Graw-Hill.

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond , E., Brennan , K., . . . Kafai, Y. (2009). Scratch: Programming for All. *Communication of the ACM, 52*, 60-67.

Rieber , L. P., Smith , L., & Noah, D. (1998). The value of serious play. *Educational Technology, 38*(6), 29-37.

Rugelj, J. (2016). Serious computer games design for active learning in teacher education. (C. Carvalho , P. Escudeiro, & A. Coelho, Eds.) *Serious games, interaction, and simulation : revised selected papers. Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering, 161*, 94-102.

Rugelj, J., & Lapina, M. (2019). Game design based learning of programming. In J. Rugelj, & M. Lapina (Ed.), *Proceedings of the International Scientific Conference Innovative Approaches to the Application of Digital Technologies in Education and Research (SLET-2019), May 20-23, CEUR workshop proceedings. 2494.* Stavropol-Dombay, Russia: Aachen: CEUR-WS.

Shabanah, S., Chen, J., Wechsler, H., Carr, D., & Wegman, E. (2010). Designing Computer Games to Teach Algorithms. *Seventh International Conference on Information Technology: New Generations, ITNG 2010*, (pp. 1119-1126). Las Vegas, NV.

Shute, V. J., & Ke, F. (2012). Games, Learning, and Assessment. In D. Ifenthaler, D. Eseryel, & X. Ge, *Assessment in Game-Based Learning: Foundations, Innovations, and Perspective.* New York, USA: Springer.

Shute, V. J., Rieber, L. P., & Van Eck, R. (2012). Games ... and ... Learning. In R. A. Reiser, & J. V. Dempsey, *Trends and issues in instructional design and technology (3rd ed.).* (pp. 321-332). Boston: Pearson Education.

Sykes, E. (2007). Determining the Effectiveness of the 3D Alice Programming Environment at the Computer Science I Level. *Journal of Educational Computing Research, 36*(2), 223-244. doi:10.2190/J175-Q735-1345-270M

Tuparova, D., Nikolova, E., & Tuparova, E. (2020). Integrated game-based learning in an informatics secondary course: Is there a difference between girls' and boys' achievements? *CSEDU 2020 - Proceedings of the 12th International Conference on Computer Supported Education*, *1*, pp. 702-709. Retrieved 10 12, 2020, from https://www.scitepress.org/Link.aspx?doi=10.5220/0009818407020709

Tuparova, D., Tuparov, G., & Veleva, V. (2019 b). Girls' and boys' viewpoint on educational computer games. *European Conference on Games-based Learning*, (pp. 757–766).

Vahldick, A. (2017). Aperfeiçoamento das Competências de Resolução de Problemas na Aprendizagem Intodutória de Programação de Computadores usando um jogo sério digital. Faculdade de Ciências e Tecnologia da Universidade de Coimbra. Retrieved from http://hdl.handle.net/10316/79528

van Eck, R. (2006). Digital Game-Based Learning: It's Not Just the Digital Natives Who Are Restless. *EDUCAUSE Review, 41*(2).

Vermeulen, L., Van Looy, J., Courtois, C., & De Grove, F. (2011). Girls will be girls: a study into differences in game design preferences across gender and player types. *Under the mask: perspectives on the gamer conference*, (pp. 1-21).

Weintrop, D., & Wilensky, U. (2015). To Block or not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. *Interaction Design and Children*, (pp. 199-208).

Whitton, N. (2009). *Learning with Digital Games: A Practical Guide to Engaging Students in Higher Education.* New York: Routledge.

Whitton, N., & Moseley, A. (2012). *Using Games to Enhance Learning and Teaching: A Beginner's Guide.* New York: Routledge.

Wolz, U., Barnes, T., & Bayliss, J. (2009). Girls do like playing and creating games. *ACM SIGCSE Bulletin, 41*(1), 199–200.

Wolz, U., Maloney, J., & Pulimood, S. (2008). 'Scratch' Your Way to Introductory CS. *SIGCSE Bulletin, 40*, 298-299.

Zapušek, M., & Rugelj, J. (2013). Learning programming with serious games. *EAI Endorsed Trans. on Game-Based Learning, 13*, 1-12.