Sofinancira program
Evropske unije
Erasmus+

NAVODILA ZA UPORABO CODING4GIRLS PRISTOPA

Erasmus+ no. 2018-1-SI01-KA201-047013

2020

Sofinancira program
Evropske unije
Erasmus+

O3 – Izobraževalne vsebine za učitelje

O3/A2 – NAVODILA ZA UPORABO CODING4GIRLS PRISTOPA UČENJA Z UPORABO IGER

Podatki o dokumentu

Rezultat: O3/A2 – Navodila za uporabo CODING4GIRLS pristopa učenja z uporabo iger

Intelektualni rezultat: O3 – Izobraževalne vsebine za učitelje

Odgovorni partner: South-West University "Neofit Rilski" (Bolgarija)

Sodelujoči

South-West University "Neofit Rilski", Bolgarija

Daniela Tuparova, Boyana Garkova, Ivanichka Nestorova, Rositsa Georgieva

University of Thessaly – Grčija

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

University of Rijeka – Hrvaška

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

EU-Track – Italy

Michela Tramonti, Alden Meirzhanovich Dochshanov and Luigi Tramonti

Virtual Campus - Portugalska

Carlos V. Carvalho, Rita Durão, Carolina Novo, Hanna Schiff, Sandra Cruz, Chiara Zanchetta

Univerza v Ljubljani - Slovenija

Jože Rugelj, Mateja Bevčič, Špela Cerar, Tadeja Nemanič, Matej Zapušek

Governorship Of Istanbul European Union and Foreign Affairs Department - Turčija

Ahu Şimşek, K.Fatih Mutlu, Abdurrahman Saygın

Izjava o omejitvi odgovornosti

Projekt CODING4GIRLS je financiran iz programa Erasmus+ Evropske unije.

Za vsebino tega dokumenta so odgovorni izključno avtorji. Vsebina ne odraža nujno mnenja Evropskih skupnosti. Evropska komisija ne prevzema odgovornosti za kakršnokoli uporabo informacij, ki jih ta dokument vsebuje.

KAZALO

1. UVOD

Projekt Coding4Girls obravnava odprto inovativno izobraževanje in usposabljanje za digitalno dobo. Osredotoča se na programiranje, ki je v družbi, usmerjeni v tehnološki razvoj, zelo pomembno in zahtevno. Poleg tega je usposobljenost za načrtovanje algoritmov, kodiranje in programiranje vedno bolj pomembna, saj je povezana z logičnim sklepanjem in sposobnostmi za reševanje problemov. Računalniške veščine so zelo zaželene, saj sodijo med kompetence, ki jih zahtevajo delodajalci v inovacijsko usmerjenih sektorjih, ki bodo v prihodnjih letih ključni za trajnostno gospodarsko rast, ki bo privedla do večje zaposlenosti in posledično do socialne kohezije.

Vendar pa je trenutno na področju računalništva veliko pomanjkanje usposobljene delovne sile in podjetja se soočajo s težavami pri iskanju zaposlenih z zahtevanimi kompetencami. Cilj programa CODING4GIRLS je zapolniti to vrzel z učinkovitim poučevanjem računalništva v osnovni in srednji šoli, ko mnogi dijaki izgubijo zanimanje za računalništvo. Na tem področju je problem tudi zelo majhno zanimanje deklet, saj jih kljub razmeram na trgu delovne sile ne zanima študij računalništva in nadaljevanje poklicne poti na tem področju.

Cilj projeta CODING4GIRLS je priprava sodobnih aktivnih oblik vključujočega izobraževanja mladih s spodbujanjem enakih možnosti za dekleta in fante za študij in za nadaljevanje poklicne poti na področju računalništva. Pri tem se partnerji v projektu lotevajo tega problema z odpravljanjem napačnih predstav in negativnega odnosa do do poklicev s področja računalništva pri učencih, učiteljih in starših, tako da pokažejo na njegovo povezanost z realnim življenjem, kjer se sodobne digitalne tehnologije pojavljajo na vseh področjih. Ob tem je pomembno poudariti tudi dejstvo, da se je takšna poklicna pot primerna tudi za ženske, ki so na tem področju povsem enakovredne moškim.

V projektu so partnerji pripravili učna gradiva in ustrezne didaktični pristope, ki bodo pomagali pri gradnji temeljnih znanj učencev, ki jim bodo omogočila uspeh v prihodnjih aktivnostih v akademskem okolju v terciarnem ali drugem nadaljnjem izobraževanju ali pri delu v poklicih s področja računalništva.

Projekt CODING4GIRLS pa naslavlja tudi razvoj učiteljevih kompetenc in profila učiteljskega poklica tako, da jim pomaga pri prodobivanju kompetenc za vodenje in pomoč učencem pri učinkovitem usvajanju želenih znanj in veščin s področja računalništva. Učiteljem rezultati

projekta pomagajo tudi pri tem, da pri svojem pedagoškem delu zagotvljajo enakost spolov v učnih aktivnostih in na bodoči poklicni poti.

Ta uporabniški priročnik za učitelje je del gradiv, razvitih v okviru projekta. Učitelji bodo v njem našli informacije o glavnih konceptih, ki temeljijo na metodologiji Coding4Girls. Mednje sodijo didaktične igre in snovalsko razmišljanje, učni pristopi pri poučevanju programiranja, ki temeljijo na igrah, učenje s programiranjem iger, primeri uporabnih programskih orodij in okolij ter značilnosti igralnega okolja Coding4Girls z dvema komponentama - učiteljevo in učenčevo. Vodič predstavlja tudi z učne liste za učne ure in njihove prilagoditve na metodologijo Coding4Girls ter na v projektu razvito učno okolje, ki podpira učenje s snovanjem iger in vključuje elemente igrifikacije.

Predstavljen je tudi položaj deklet v računalništvu in njihov interes za igranje računalniških iger s posebnim poudarkom na didaktičnih igrah.

.

2. OSNOVNI KONCEPTI

2.1. Didaktične igre

Igra je pojem s širokim pomenom, zato je težko izpostaviti najpomembnejše značilnosti, zaradi katerih določeno dejavnost lahko imenujemo igra. Za Thortona je najpomembnejša značilnost igre interaktivnost. Johnston igro vidi kot dejavnost, ki ima pravila, ki jih morajo upoštevati vsi udeleženci, enega ali več ciljev, interakcije in dinamične vizualne elemente (Johnston & de Felix, 1993). Malone (1981) v svoji teoriji izpostavlja fantazijo, radovednost, izziv in nadzor kot najpomembnejše sestavine vsake igre. Najobsežnejšo opredelitev je naredil Garris s sodelavci (2002), ki trdijo, da so te komponente tekmovanje, izziv, socialna interakcija, pogovori in domišljija.

Prenskyjeva teorija učenja z uporabo iger je ena od najvplivnejših. Trdi, da mora igra vsebovati nasledenje elemente: pravila, cilje in naloge, rezultate in povratne informacije, konfliktne situacije (tekmovanje, izziv, nasprotovanje), interakcijo in domišljijski okvir (Prensky, 2001). Avtorji knjige *"Serious games"* igro opredeljujejo kot prostovoljno dejavnost, ločeno od resničnega življenja, v namišljenem svetu, ki do resničnega življenja lahko ima neko povezavo ali pa tudi ne, pritegne pa vso pozornost igralca in se igra v skladu z uveljavljenimi pravili, ki jih morajo upoštevati vsi igralci (Michael & Chen, 2006).

Ena od najbolj izpostavljenih lastnosti igre je interaktivnost, ki se nanaša na interakcijo z računalnikom, z nasprotnikom (ki ga predstavlja živi igralec ali računalniški program) ali z drugimi igralci ekipe.

Računalniške igre se glede na število igralcev, ki se istočasno ukvarjajo z igro, delijo na igre z enim igralcem, igre z več igralci in na množične igre. Glede na žanr poznamo arkadne, akcijske in vojne igre, igre igranja vlog, strateške, družabne, športne, simulacijske in pustolovske igre. Lestvice priljubljenosti računalniških iger kažejo, da so najpogosteje igrane igre z več igralci, strateške in akcijske igre.

Da bi resnično razumeli moč igre, si najprej poglejmo značilnosti otroške igre. Otroška igra je znana kot ena najpomembnejših dejavnosti, ki pomaga pridobiti znanje in razviti pomembne veščine za celo življenje. Skozi igro otrok izboljšuje svoje intelektualne sposobnosti z odkrivanjem prvih osnovnih konceptov iz resničnega sveta in med seboj

ustvarja smiselne povezave. Jean Piaget igro vidi kot vključevanje novih znanj v obstoječo kognitivno strukturo in utrjevanje na novo naučenega vedenja. Freud poudarja čustvene koristi igre in trdi, da zmanjšuje objektivno in nagonsko tesnobo tako, da pomaga otroku pri reševanju čustvenih težav. Socialni konstruktivisti menijo, da je igra pomembna zaradi razvoja socialnih veščin.

Učenje z igranjem iger je ena najpogostejših dejavnosti v zgodnjem otroštvu, vendar so pozitivni učinki igranja in iger na učenje v formalnem izobraževanju pogosto spregledani. Raziskave, ki so jih opravili na področju učenja z igranjem iger, kažejo, da na motivacijo pozitivno vpliva, če je učna snov predstavljena v obliki računalniške igre. Igra učencem pomaga usvojiti učno snov, hkrati pa je vir zabave. Če imamo v učnem procesu ti dve komponenti, imamo sproščenega in motiviranega učenca, ki je zato bolj pripravljen na učenje. Druge pedagoške koristi so sodelovalno, eksperimentalno in aktivno učenje, problemsko učenje in avtetične učne situacije. Ker je danes uporaba IKT v formalnem izobraževanju vedno bolj pogosta, nam to daje priložnost, da razvijamo in potem v učnem procesu uporabimo kakovostna učna gradiva, ki spodbujo učenje.

Igre lahko uporabljamo za oglaševanje, simulacije, usposabljanje in izobraževanje. Učencem omogočajo, da doživijo situacije, ki jih v resničnem svetu ne morejo zaradi različnih razlogov, kot so varnost, stroški ali čas. Take igre morajo imeti opredeljene učne cilje in naj bi pomembno pripomogle k razvoju novih znanj ali različnih spretnosti igralcev.

Za oblikovanje dobre izobraževalne igre moramo biti sposobni oblikovati in izdelati dobro programsko opremo, so pa take igre mnogo več kot le programi. Za njihovo učinkovito uporabo v izobraževanju je potrebno pripraviti tudi ustrezna navodila za učence in učitelje ter pogosto tudi različna dodatna učna gradiva.

Potencialna vrednost iger za učenje se zdi velika, vendar ostaja odpor do uporabe iger v šolah. Zato moramo usposobiti učitelje, da bodo znali izbrati ustrezne učne pristope in primerne kakovostne igre ter jih opogumiti, da jih bodo vključili v svoje poučevanje, saj za to že obstojajo ustrezne praktične izkušnje in znanstvena potrditev skladnosti uporabe iger s splošno sprejetimi teorijami učenja in poučevanja.

Rieber, Smith in Noah so že leta 1998 ugotovili, da obstajata dva načina uporabe iger v izobraževanju: učenje z igranjem iger in učenje z izdelovanjem iger. Učenje z igranjem iger je tradicionalen pristop, kjer učencem ponujamo že izdelane igre. Pri učenju z izdelovanjem iger

pa predpostavljamo, da je snovanje in izdelava igre samo po sebi pot do učenja, ne glede na to, ali se igra izkaže za zanimivo drugim. Ideja »učenja s snovanjem« temelji na predpostavki, da je aktivno sodelovanje v procesu načrtovanja in izdelave najboljši način, da se učimo. Ta pristop je vedno bolj uporaben zaradi vedno boljšega dostopa do orodij in okolij za računalniško programiranje iger.

2.2. Učenje z igrami

Vedno več je razlogov, ki učitelja spodbujajo k uporabi iger (Gee, 2007). V formalnem izobraževanju doživljamo prehod s tradicionalnega didaktičnega modela, ki je osredotočen na poučevanje, na model, osredotočen na učenca. Ta poudarja aktivno vlogo učenca v procesu učenja. Spremenilo se je tudi osredotočenje z učnih ciljev z nižjih taksonomskih ravni, kot je na primer priklic informacij, na višje ravni, kot sta iskanje in uporaba informacij (Rugelj, 2016).

Prensky (2001), Gee (2003) in Whitton (2009) so učenje, ki temelji na igranju računalniških iger, opredelili kot proces učenja z uporabo digitalnih iger. Igre lahko zagotovijo motivacijo za učenje in tako povečajo možnost, da bodo učenci dosegli želene učne rezultate. Učenje je opredeljeno kot pridobivanje znanja in veščin z izkušnjami ali prakso (Pivec in Kearney, 2007). Skoraj vse študije o na igranju temelječega učenja kažejo, da so učenci zelo motivirani, ko so učna gradiva predstavljena v obliki računalniške igre in da to pozitivno vpliva na pospešitev in intenzivnost učnega procesa (Zapušek & Rugelj, 2013). Učenci potrebujejo motivacijo, da se osredotočijo na to, česar se je treba naučiti, vendar za kakovostno učenje to ni dovolj. Primerjava učnih rezultatov učencev, ki so se učili z računalniškimi igrami, in učencev z drugo vrsto učnega gradiva kaže, da med njimi ni bistvene razlike. To se običajno dogaja zaradi neprimerne zasnove iger. Igre so lahko zelo privlačne za učence, če pa se samo zabavajo in se ne učijo, uporaba iger v izobraževanju nima posebnega smisla. Torej moramo ugotoviti, kateri elementi lahko naredijo računalniško igro didaktično.

Gross (2003) trdi, da morajo imeti digitalne igre za izobraževalne namene natančno opredeljene učne cilje ter spodbujati razvoj pomembnih strategij in veščin za povečanje kognitivnih in intelektualnih sposobnosti učencev.

Po ugotovitvah Maloneja (1981b) in Garrisa s sodelavci (2002) so elementi, ki prispevajo k didaktični vrednosti digitalnih iger, ustrezni čutni dražljaji (vizualna in zvočna predstavitev

učnega gradiva), fantazija (kontekst, predstavljen v zgodbi igre), izziv (zahtevna ali spodbudna situacija) in radovednost (želja po spoznavanju ali učenju). Ti elementi morajo biti vključeni v igro v obliki strukturiranih ciljev in pravil, privlačne zgodbe, takojšnje povratne informacije, visoke stopnje interaktivnosti, izzivov in tekmovanja ter naključnih elementov presenečenja in bogatega okolja za učenje.

Igro je mogoče ustvariti za učenje, saj vključuje mentalno (in včasih fizično) stimulacijo in razvija praktične veščine - igralca prisili, da se odloči, izbere, določi prioritete, reši probleme. Takojšnja nagrada in povratne informacije so najpomembnejši motivacijski dejavnik, ne glede na to, ali je to prevedeno kot "dobitek" v igri (več življenjske moči, dostop do novih ravni itd.) ali kot nevrološki impulzi za igralca (sreča, občutek dosežka, ...).

Igre so lahko družabna okolja, ki včasih vključujejo velike porazdeljene skupnosti. Predpostavljajo (Baptista in vaz de Carvalho, 2010) zmožnosti za samoučenje (igralci so pogosto dolžni poiskati informacije za obvladovanje same igre), omogočajo prenos učenja iz drugih resničnosti in so že po naravi izkustveni z vključevanjem več čutov.

Van Eck (2006) trdi, da so igre lahko učinkovito učno okolje - ne zato, ker so zabavne, ampak zato, ker so poglobljene, od igralca zahtevajo pogoste pomembne odločitve, imajo pametne cilje, se prilagajajo vsakemu igralcu posebej in lahko vključujejo socialno mrežo igralca.

Če analiziramo model učenja na osnovi iger, ki so ga definirali Garris, Ahlers in Driskell (2002), lahko ogotovimo, da je po njihovem mnenju glavna značilnost izobraževalne igre, da je učna vsebina skrita v igri. Učenci se igrajo in zabavajo, niso pa zavestno pozorni na "učni" del izkušnje, čeprav jim igra nenehno predstavlja nove koncepte, ki si jih morajo prilagoditi, da bodo lahko v igri uspešni. Motivacijo bi po njihovem morali spodbujati z oblikovanjem iger, ki spodbuja ponavljanje ciklov v okviru igre. Od igre pričakujemo, da bo izvabila iz igralca želena obnašanja ali aktivnosti, ki temeljijo na čustvenih in kognitivnih reakcijah, ki izhajajo iz interakcije igralca z igro in iz povratnih informacij, ki jih dobiva pri igranju.

Gee (2003) trdi, da značilnosti računalniških iger z visokim učnim potencialom lahko uvrstimo v dve kategoriji: "ne-igrive"značilnosti, ki se lahko pojavijo tudi v kontekstih brez iger, in "igrive" značilnosti iger. Ob tem se moramo zavedati, da omenjene ne-igrive funkcije ne delujejo dobro za učenje, če so ločene od igre.

Ne-igrivi elementi z velikim učnim potencialom so:

- Empatija za zapletene sisteme - pogled na zapleten sistem od znotraj, da bi razumeli, kako njegove spremenljivke medsebojno delujejo.
- Simulacije izkušenj in priprave na akcijo

  V video igrah igralci vidijo navidezni svet s takega zornega kota, ki jim ponuja možnost izvedbe dejanj, ki jim omogočajo dosego določenih ciljev.
- Porazdeljena inteligenca z ustvarjanjem pametnega orodja

Dobre video igre "porazdelijo inteligenco" med resničnimi osebami in umetno inteligentnimi virtualnimi liki, da bi prikazale makro in mikro pogled na situacijo.

- Timsko delo

  Dobre video igre lahko naučijo sodelovanja in timskega dela v šoli in na delovnem mestu. Igralci medsebojno delujejo ne glede na njihove značilnosti v resničnem svetu, v skladu s svojimi funkcionalnimi identitetami v igri. Prav tako se lahko odločijo, da bodo svojsocialni položaj, kulturno identiteto in spol uporabili kot strateške vire, vendar niso prisiljeni v vnaprej določene spolne, kulturne ali razredne kategorije.
- Situacijski pomen

  Dialog in izkušnje so bistvenega pomena, da lahko ljudje besede povežejo z dejanskimi dejanji, funkcijami in reševanjem problemov. Ker so video igre simulacije izkušenj, lahko umeščajo jezik v določen kontekst.
- Odprtost: združevanje osebnega in družbenega

  V dobrih odprtih igrah si igralci oblikujejo cilje, ki temeljijo na njihovih lastnih željah. Kombinacija osebnih ciljev in ciljev v igri ustvarja stanje visoke motivacije.

Značilnosti "dobre igre", ki se nanaša na učinkovito učenje, so naslednje:

- Motivacija

  Video igre močno spodbujajo igralce, zato je pomembno razumeti vire te motivacije, če želimo zagotoviti dobre temelje za učenje.
- Vloga neuspeha

  Cena neuspeha se zniža in je pogosto videti kot način, kako se naučiti osnovnega vzorca. Te značilnosti neuspeha v igrah omogočajo igralcem, da tvegajo, v realnem življenju pa bi to pogosto bilo "predrago".
- Tekmovanje in sodelovanje

Številni igralci, vključno z mladimi, uživajo v tekmovanju z drugimi igralci, vendar morda v šoli tekmovanja ne vidijo kot prijetnega in motivirajočega. Konkurenco v video igrah igralci vidijo kot "družabno", tako pri igranju kot o zmagi in porazu.

- Zasnova iger, ki se nanaša na:
    - Interaktivnost - igralec ne samo pasivno uporablja svoje znanje, ampak ima tudi nadzor nad vsebino;
    - Prilagajanje - temelji na učnih stilih in zagotavlja več poti do uspeha;
    - Močne identitete - dobre igre igralcem ponujajo identitete, ki sprožijo intenzivno zavzetost igralca in so jasno povezane s funkcijami, spretnostmi in cilji, ki naj bi jih dosegli v virtualnem svetu;
    - Dobro razvrščeni problemi - pri konektivisitičnih pristopih k učenju je zaporedje ključnega pomena za učinkovito učenje na kompleksnih področjih;
    - Ustrezna stopnja zahtevnosti - prilagajanje izzivov tako, da lahko vrsta igralcev igro doživlja kot zahtevno, a izvedljivo;
    - Cikeličen proces učenja - ponavljajoči se cikli razširjene prakse in preizkusi obvladovanja;
    - "Zahtevno", a "pošteno" - igra mora biti zahtevna, vendar zastavljena tako, da vodi učenca do uspeha.
    - Gameplay elements should be initially simple and easy to learn and become more complex the more the player comes to master them.

Računalniške igre, ki se uporabljajo v izobraževalnih okoljih, dokazano povečujejo motivacijo. Toda visoko motivirani učenci niso dovolj za kakovostno učenje. Potrebujemo tudi dobre učne materiale, da bodo učenci dejansko pridobili novo znanje iz virov, predstavljenih v obliki računalniške igre.

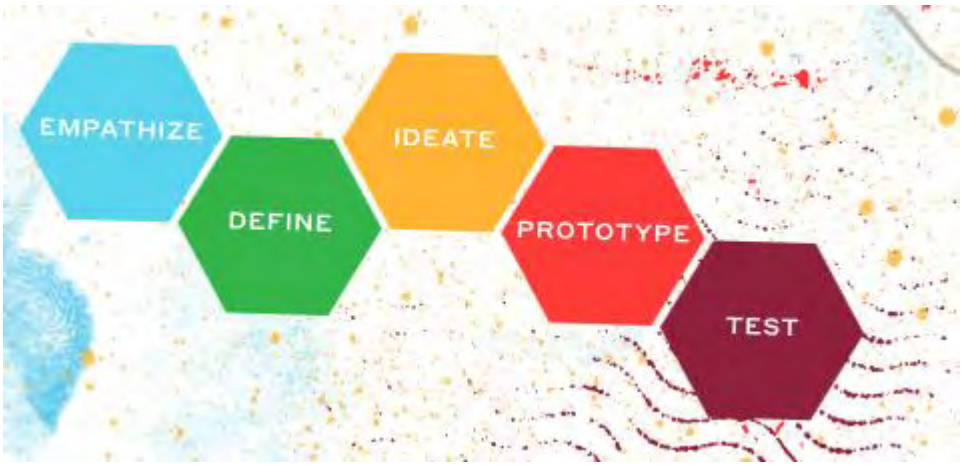
Kljub pozitivnemu vplivu igre na motivacijo, učitelji še vedno oklevajo uporabiti didaktičnih iger pri poučevanju. V naši raziskavi so izrazili mnenje, da so igre lahko preveč zamudne za uporabo v učilnici. Zato jih pogosto uporabljajo predvsem kot učno dejavnost na domu ali kot motivacijo pri uvodnih urah.

2.3. Metodologija snovalskega razmišljanja

Ideje o snovalskem razmišljanju (angl. design thinking) so se pojavile konec šestdesetih let, toda kot koncept se je na univerzi Stanford pojavilo "snovalsko razmišljanje" šele konec osemdesetih let. V poslovnem svetu se je metoda razširila, potem ko jo je promoviral Tim Brown, ki je dejal: "Snovalsko razmišljanje je k človešku usmerjen pristop k inovacijam, ki črpa iz orodij oblikovalca za vključevanje potreb ljudi, tehnoloških zmožnosti in zahtev za poslovni uspeh. "(IDEO Design thinking, 2008). Na začetku 21. stoletja je snovalsko razmišljanje postalo izjemno priljubljeno in IT-podjetja so začela uporabljati ta pristop pri razvoju svojih izdelkov. Od leta 2005 na primer SAP uporablja snovalsko razmišljanje" kot filozofijo reševanja problemov in kot inovativen pristop, osredotočen na končnega uporabnika " (Innovation Starter, 2015). Kot rezultat tega pristopa vodi razvoj novih izdelkov do želenega rezultata. Podjetja, kot so P&G, IBM Design, GOOGLE, AMAZON in Cisco, integrirajo snovalsko razmišljanje v svoje celotno poslovanje in organizacijo (Naiman, 2019).

Koncept, ki so ga razvili na Univerzi Stanford, opredeljuje snovalsko razmišljanje kot "vodilno metodologijo za kreativno reševanje problemov in ustvarjanje inovacij, ki jo vsak dan uporablja na tisoče podjetij in organizacij po vsem svetu. Cilj je pri mladostnikih razviti veščine 21. stoletja - timsko delo, reševanje problemov, kreativnost, empatijo, samozavest, potrpljenje, koncentracijo in eksperimentiranje. "(Karakehayova, 2019).

Stanfordski model je sestavljen iz 5 stopenj (Slika 2.1).



Slika 2.1 Stopnje Stanfordskega modela snovlaskega razmišljanja

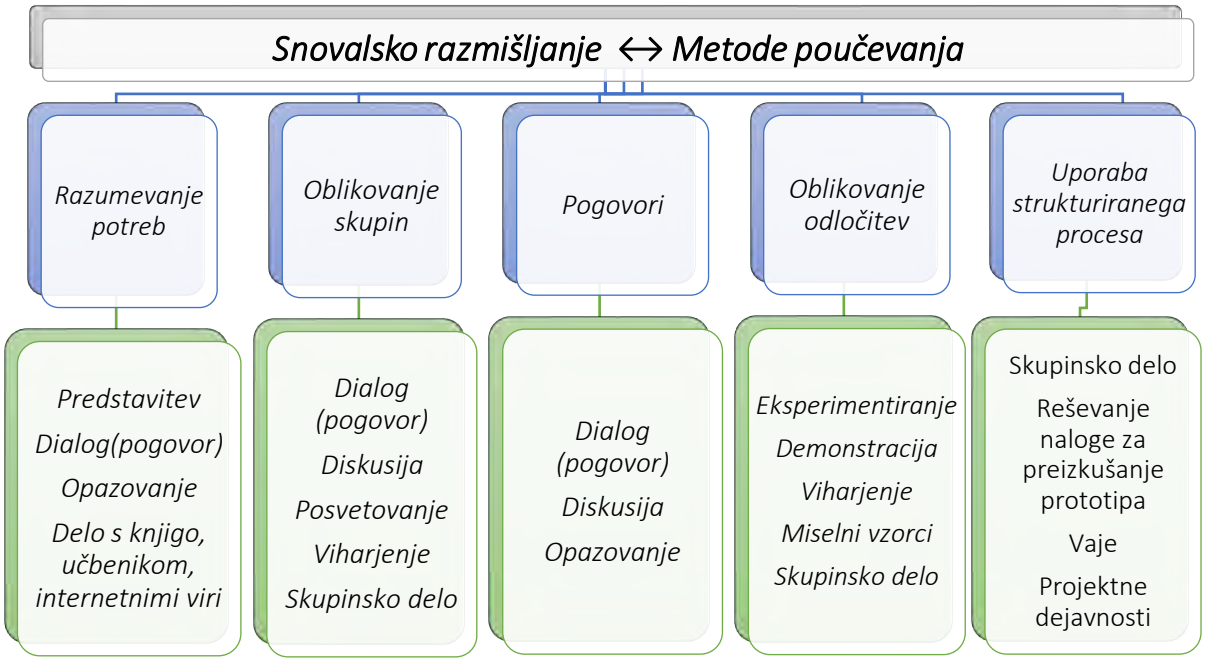
Glavne stopnje in značilnosti snovalskega ramišljanja lahko povzamemo na naslednji način (Tabela 2.1.) (Georgieva & Tuparova, 2020).

Table 2.1 Stopnje in značilnosti SR

Praksa	**Načela**	**Stopnje**		
		Kratek postopek	**Razširjen postopek**	**School Stanford**
Razvijanje globokega empatičnega razumevanja potreb uporabnikov	Odkrivanje - spoprijemanje z novim izzivom. Kako to rešiti?	Opazovanje in proučevanje vedenja in izkušenj uporabnikov, raziskovanje in definiranje problema z empatijo.		Sočustvovanje
Oblikovanje heterogenih timov	Tolmačenje - Kaj sem se naučil in kako to tolmačiti?	V kratkem času ustvarite veliko idej.		Definiranje
Pogovori na osnovi dialoga	Ideja - vidim priložnost, kako jo oblikovati v idejo?	Izbor in klasifikacija idej		Oblikovanje idej
Ustvarjanje odločitev s poskusi	Eksperimentiranje - imam idejo, kako to zgraditi?	Izdelava prototipov - izdelek, ki od uporabnikov prejme hitre povratne informacije		Prototip
Uporaba strukturiranega in podprtega procesa	Evolucija - poskusil sem nekaj novega, kako to razviti?	Testiranje / Povratne informacije - kako izboljšati prototip, če je potrebno		Test
			Testiranje, če ideja deluje	
			Učenje in izboljšanje na podlagi povratnih informacij	

Snovalsko razmišljanje koristi ne samo podjetjem, temveč tudi nevladnemu, izobraževalnemu in zdravstvenemu sektorju. Vse več izobraževalnih ustanov se usmerja k snovalskemu razmišljanju. Postopoma se povečujejo razpoložljivost dobrih praks za uporabo oblikovalskega mišljenja v izobraževanju. (Georgieva & Tuparova, 2020). Pri uporabi tega

pristopa mora učitelj vzpostaviti povezavo med posameznimi stopnjami v modelu snovalskega razmišljanja in ustreznimi učnimi metodami, ki jih je mogoče uporabiti. Snovalsko razmišljanje zahteva uporabo širokega nabora učnih metod.



Slika 2.2. Metode poučevanja in snovalsko razmišljanje (Georgieva & Tuparova, 2020)

2.4. Teorije o učenju in učenje z igranjem iger

Veliko izobraževalnih računalniških iger je bilo v preteklosti oblikovanih v skladu z vedenjsko teorijo učenja (Rugelj & Lapina, 2019). Izvajale so se v obliki *programiranega poučevanja*. Učenci dobivajo dražljaje v obliki vprašanja ali drugih vrst nalog ali problemov, ki jih je treba rešiti. Učenci se odzovejo z izbiro enega od ponujenih odgovorov. Če je odgovor pravilen, igra prinese nekakšen pozitiven odziv v obliki pozitivne reakcije lika ali vesele melodije, ki spodbuja pozitivna čustva. Tak primer povezovanja akcije in reakcije vzpostavi povezavo med vprašanjem in pravilnim odgovorom. V primeru napačnega odgovora je pripravljena reakcija v obliki negativnih dražljajev in povezava je oslabljena. Igre in kvizi imajo vgrajen koncept vaje in so tipični predstavniki iger, ki temeljijo na behaviorizmu. Primerni so za učenje osnovnih aritmetičnih operacij ali za podporo zapomnjevanju fraktografskih podatkov, torej učnih ciljev na najnižjih ravneh Bloomove taksonomije.

Teorija kognitivnega učenja poudarja kognitivno aktivnost učenca in oblikovanje ustreznih mentalnih modelov. Učenci se naučijo temeljnih konceptov od svojega učitelja ali ustrezno oblikovanih učnih virov, nato pa uporabijo dedukcijo za oblikovanje novega znanja. Uganke in strateške igre kot okolje za odločanje so primeri kognitivističnega pristopa. Najnaprednejše oblike iger, ki temeljijo na kognitivni teoriji, temeljijo na inteligentnih tutorskih sistemih, kjer se algoritmi strojnega učenja uporabljajo za modeliranje učenca in strokovnega znanja, da lahko učencu ponudijo prilagojeno učno gradivo.

Konstruktivizem je alternativni pogled, po katerem učenci sami gradijo svoje znanje; zato imamo več različnihindividualno izdelanih predstavitev znanja, ki so enako veljavna. Učenje je po tej teoriji aktiven proces konstruiranja (namesto pridobivanje znanja), kjer se novo znanje gradi rekurzivno na znanju, ki ga uporabnik že ima. V procesu gradnje se informacije, pridobljene iz okolja s čutili kombinirajo z obstoječim znanjem i dolgoročnega spomina učenca. Tako se ustvarijo novi miselni modeli, ki so osnova za nadaljnjo konstrukcijo znanja. Konstruktivistično učenje poudarja učenje z odkrivanjem in poizvedovanjem, pri čemer morajo biti učenci v okolju (ki ga je mogoče modelirati z računalniško igro), kjer gradijo svoje znanje. Tri temeljna načela opredeljujejo konstruktivistični pogled na učenje:

- vsaka oseba oblikuje svojo predstavitev znanja,
- učenje se dogaja, ko učenci pri raziskovanju odkrijejo neskladje med svojo trenutno predstavitvijo znanja in novo pridobljenimi izkušnjami,
- učenje poteka v družbenem kontekstu in interakcija med učenci in njihovimi vrstniki je nujen del učnega procesa.

Učna gradiva omogočajo učenje, ki temelji na podpori konstrukciji znanja in ne prenosu znanja po principih vedenjske teorije učenja. Računalniške igre v obliki simulacij so izdelane na osnovi različnih scenarijev iz resničnega življenja. Predstavljajo model abstraktne resničnosti, v katerem učenec doživlja določeno vlogo. Naloga učitelja je zagotoviti usmerjanje in povratne informacije, ko se učenec uči in gradi uporabne mentalne modele. Igre lahko vodijo do sprememb v znanju in veščinah igralca. Shute in Ke (2012) sta ugotovila, da obstaja velika podobnost med bistvenimi elementi dobre igre in značilnostmi produktivnega učenja. Zasnova iger nas lahko veliko nauči o učenju, sodobna teorija učenja pa nas lahko nauči o oblikovanju boljših iger (Shute, Rieber & Van Eck, 2012). Marshall McLuhan, kanadski filozof teorije komunikacije, ki je svetovni splet predvidel že v šestdesetih

letih prejšnjega stoletja, ko je govoril o globalni vasi, je izjavil: "Kdor razlikuje med igro in učenjem, ne pozna niti enega niti drugega."

Whitton in Moseley (2012) sta predlagala okvir dobre prakse pri oblikovanju izobraževalnih iger na osnovi značilnosti aktivnega učenja. Po njunih smernicah bi moralo okolje igre podpirati aktivno učenje s spodbujanjem raziskovanja, reševanja problemov in poizvedovanja, ustvarjati angažiranost z jasnimi in dosegljivimi cilji, biti primerno za učni kontekst, podpirati in zagotavljati možnosti za razmislek, zagotavljati enake možnosti za vse učence , nuditi stalno podporo s postopnim uvajanjem vedno večje zahtevnosti in podpirati učenca z različnimi oblikami pomoči in namigi.

Številne študije so pokazale, da lahko izobraževalne igre podpirajo učenje z motiviranjem, privabljanjem in zabavo (Hijon-Neira idr., 2015). Učenje programiranja zahteva veliko kompetenc, kot so logično razmišljanje, reševanje problemov in sposobnost razumevanja abstraktnih konceptov. Zaradi tega se mnogi učenci težko učijo računalniškega programiranja. To dejstvo lahko privede do nizke motivacije za učenje v uvodnih tečajih programiranja. Da bi učitelji izboljšali motivacijo in izboljšali odnos učencev do programiranja, iščejo spodbudne pristope k učenju.

Programiranja se najbolje naučimo s praktičnim delom in če naj bi se učenci učili učinkovito, morajo vsaj del teh praktičnih aktivnosti izvajati sami ali v sodelovanju z vrstniki. Ključna vloga učitelja je navdušit učence za take aktivnosti in jih potem vse čas motivirati (Feldgen & Clua, 2004).

Učenci v uvodnih tečajih računalniškega programiranja snujejo in razvijajo programe (Rugelj & Lapina, 2019). To je tipičen pristop aktivnega učenja. Če pri tem uvedemo projektno delo v skupinah, ki se je izkazalo za zelo učinkovit način učenja programiranja (Nančovska Šerbec, Kaučič & Rugelj, 2008), lahko dejansko govorimo o trialoškem pristopu k učenju. Trialoško učenje vsebuje take oblike učenja, pri katerih učenci v sistematičnem procesu skupaj razvijajo, spreminjajo ali ustvarjajo skupne artefakte (v našem primeru računalniški program) (Kafai, 1995). Osredotoča se na interakcijo, ki se pojavi pri ustvarjanju konkretnih predmetov, ne le med ljudmi ("dialoški pristop") ali znotraj človekovega uma ("monološki" pristop) (Paavola in Hakkarainen, 2005).

3. DEKLETA, IKT IN DIDAKTIČNE IGRE

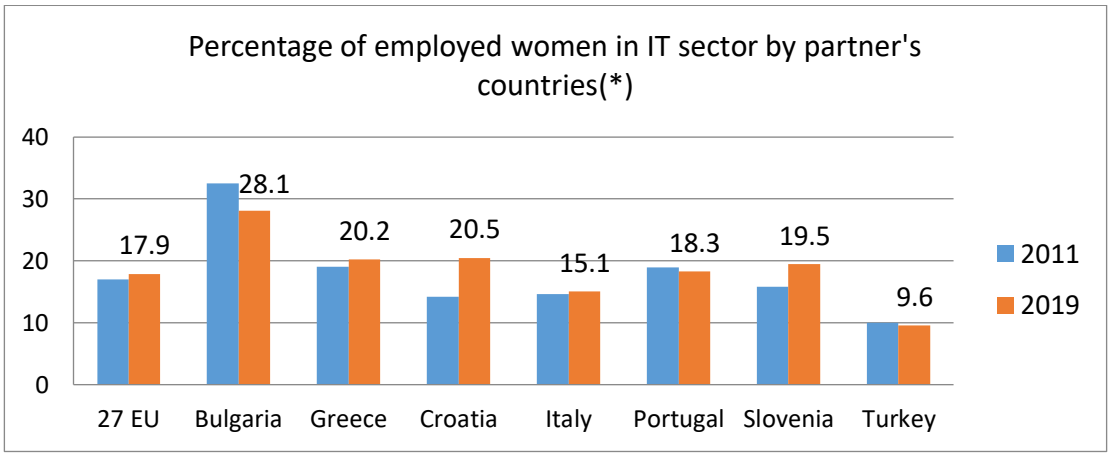
3.1. Položaj žensk v računalništvu

Digitalna družba svojim članom postavlja vedno večje zahteve po različnih kompetencah. Vse večje so potrebe po ljudeh, ki niso samo digitalno pismeni, temveč imajo tudi znanja za ustvarjanje in vzdrževanje programske opreme na različnih področjih človeške dejavnosti. Mediji nenehno ugotavljajo pomanjkanje usposobljenih strokovnjakov na področju informacijskih in komunikacijskih tehnologij (IKT), saj ta sektor neprestano raste.

Leta 2019 je v vsej Evropski uniji (EU) na področju računalništva delalo približno 7,8 milijona strokovnjakov za IKT (Eurostat, 2020). To je samo 3,9% vseh zaposlenih v EU.

Po podatkih Eurostata za obdobje 2007–2019 se število žensk v IT sektorju v 27 državah članicah EU v povprečju zmanjšalo z 22,5% na 17,9% zaposlenih.

Trenutno stanje zaposlovanja žensk v sektorju IKT v državah partnericah projekta Coding4girls je predstavljeno na sliki 3.1.



Slika 3.1 Odstotek žensk zaposlenih na področju računalništva v 2011 in 2019

3.2. Odnos deklet do izobraževalnih iger

Rezultati študije, ki so jo izvedli Vermeulen idr. (2011) kažejo, da razlike med spoloma dosledno prisotne, ampak predhodne izkušnje pomembno vplivajo na ugotovitve (Rugelj & Lapina, 2019). Ženske igrajo igre bolj pogosto, ampak krajši čas kot moški. Raje imajo abstraktne, krate in enostavne igre, kot so kratkočasne (npr. Tetris) in igre na družbenih

omrežjih, medtem ko moški pogosteje igrajo »glavne« žanre. Ta kategorija se nanaša na igre, ki temeljijo na spretnostih in veščinah, ki se igrajo dlje časa in običajno vključujejo visokokvalitetno tridimenzionalno grafiko, na primer strelske, borilne, akcijsko-pustolovske, športne, dirkaške, strateške, igre igranja vlog in masovne večigralske spletne igre (angl. MMOG). Ostali žanri iger so platformne, pustolovske, simulacijske, zabavne, izobraževalne, klasične in kratkočasne igre. »Glavne« žanre iger igra več moških kot žensk. Študija je pokazala, da ženske rade igrajo miselne igre, medtem ko dirkaške, plesne, simulacijske in virtualne igre igrajo tako ženske kot moški. Druga študija je pokazala, da imajo ženske raje igre za zabavo (kot so glasbene in plesne igre) in klasične »retro« igre.

Alserri idr. (2018) so opravili obširen pregled člankov povezanih z učinkovitostjo elementov izobraževalnih iger, kot so motivacijski elementi igralcev digitalnih iger, učinkoviti elementi izobraževalnih iger in elementi ženskih preferenc. Rezultate so uporabili za oblikovanje konceptualnega modela za vključevanje v izobraževalne igre glede na spol. V okviru Coding4Girls so ta model uporabili za povečanje motivacije deklet za programiranje skozi primerno izbiro iger, ki jih bodo ustvarjali in implementirali učenci s pristopom učenja s snovanjem iger.

Avtorji so navedli, da je študija pokazala, da je motivacija za igranje specifičnih tipov digitalnih iger odvisna od spola. Stereotio, da ženske ne igrajo računalniških iger, ne drži več. Razlike med spoloma v preferencah za igranje digitalnih iger, ki so jih odkrili v tej raziskavi, so zelo podobne kot že predstavljene razlike, ki so jih zaznali Vermeulen idr. Ženske imajo raje raziskovalne in kreativne igre kot moški. Igrajo bolj pogosto, ampak krajši čas kot moški, in njihove preference za igralne žanre so prav tako različne. Ugotovili so, da moški za igranje računalniških iger porabijo več časa kot ženske. Ženske imajo raje miselne igre, družabne igre z nagradami, ki jih ponujajo v igrah, izobraževalne igre, simulacijske igre kot tudi sodelovalne in raziskovalne igre, navidezno življenje, navidezne svetove in igre za zabave. Poleg tega ženske rade igrajo pustolovske igre, ampak pred igranjem rade opazujejo, kako drugi igrajo te igre, preden jih igrajo same. Motivacijski elementi za igranje iger pri ženskah so izziv, eskapizem, zabava, družabne interakcija, motivacija, fantazija, tekmovanje in vznemirljivost. Tako moški kot ženske imajo radi dirkaške, simulacijske in virtualne igre.

Phan idr. (2012) so v svoji študiji o razlikah med igralci iger moškega in ženskega spola prišli do zelo podobnih zaključkov o uporabi, preferencah in vedenju v računalniških igrah.

Tuparova, Tuparov, Veleva (2019 ECGBL) so v svoji študiji ugotovili, da obstajajo razlike v preferencah fantov in deklet do lastnosti in elemente iger, kot so grafično oblikovanje, zvok, možnost igranja igre na različnih napravah (računalnik, tablica, pametni telefon, ipd.), možnost vključitve več igralcev, možnost igranja preko spleta, možnost preklopa na bolj zahtevne nivoje igre. Za dekleta je najpomembnejša lastnost igre možnost preklopa na bolj zahtevne nivoje igre, medtem, ko sta za fante najpomembnejša logika in zaplet/scenarij igre. Najmanj pomembna lastnost za dekleta je zvok v igri, medtem ko so za fante najmanj pomembne nagrade v igri. Kar zadeva napravo, ki se uporablja za igranje, je ne glede na spol najpogosteje uporabljen pametni telefon. Dekleta kot drugi najpogosteje uporabljeno napravo izberejo prenosnik, ki mu sledi tablični računalnik. Za fante sta druga izbira namizni računalnik in prenosnik. Avtorji so opazili, da fantje kažejo večje zanimanje za igre z zgodbo kot dekleta, dekleta pa večje zanimanje za spominske igre kot fantje. Neodvisno od spola je zanimanje za naslednje žanre iger: igre z družbenimi elementi, pozornost/koncentracija, igre za hitrost reakcij, igre vlog, miselne igre, pustolovske igre. Najmanjše zanimanje tako med dekleti kot fanti je za miselne igre.

Zaradi specifičnosti projekta Coding4Girls je potrebno upoštevati, da pristop temelji na učenju s snovanjem iger in pripraviti naloge, ki so zanimive in motivacijske za dekleta. Programsko okolje, v katerem je programiranje uporabljeno kot način kreiranja animiranih filmov (to je pripovedovanje zgodb) ali preprostih iger, je lahko primerno za učenje deklet, saj lahko dobijo idejo za zgodbo ali preprosto igro, ki bi jo želele izdelati (Wolz idr., 2009). Obe sta naravno zaporedni in verjetno ne bosta takoj zahtevala poznavanja naprednih programerskih konceptov, sta obliki lastnega izražanja in omogočata dekletom priložnost za eksperimentiranje z različnimi vlogami. Prav tako lahko prijatelji, ki ne znajo programirati, hitro razumejo in cenijo animirano zgodbo ali igro, kar omogoča priložnost za pozitivno povratno informacijo. Kelleher, Pausch in Kiesler (2007) so uporabili v obeh primerih uporabili Storytelling Alice in Generic Alice. Trdijo, da so številne o otrocih programerjih pokazale, da ko so dekleta in fantje, ki imajo podobne izkušnje z računalniškim programiranjem, enako zainteresirani in učinkoviti pri učenju programiranja. Toda njihova uspešnost programiranja je v korelaciji z časom, ki ga posvetijo programiranju in njihovimi predhodnimi programerskimi izkušnjami.

4. UČENJE PROGRAMIRANJA PREKO IGER

V tem delu se bomo osredotočili na predstavitev in primerjavo različnih pristopov in platform/igralnih okolij za poučevanje programerskih veščin za otroke.

4.1. Pristopi za poučevanje programiranja skozi igre

Trenutno je na voljo širok nabor pristopov za poučevanje programiranja za programerje začetnike.

Te pristope lahko razdelimo na:

- Učenje programiranja s snovanjem iger predstavlja uporabo programskih jezikov za učenje programiranja s snovanjem in kodiranjem iger.
- Učenje programiranja v igralnih učnih okoljih.
- Učenje programiranja s snovanjem iger v igralnem okolju. Metodologija Coding4Girls sodi v to skupino okolij za učenje programiranja.

Tak primer je uporaba vizualnih programskega okolja za programiranje z delčki. Ti so uporabniku prijazni, preprosti za uporabo (princip povleci in spusti), preprečujejo sintaktične in logične napake (Ouhabi idr., 2015) ter predstavljajo vsebino na zelo intuitiven način. Scratch (Resnick idr., 2009) (Scratch, n.d.), Snap! (Weintrop idr., 2015) (Snap!, n.d.) ali Alice 2/3 (Alice, n.d.) so znani in uspešni primeri programskih jezikov in okolij za programiranje z delčki. Tabela 4.1 prikazuje primerjavo njihovih lastnosti.

Table 4.1 Lastnosti najbolj znanih jezikov in okolij za programiranje z delčki

	PLATFORMA	CILJNA SKUPINA	JEZIK(I)	BREZPLAČNO/ PLAČLJIVO
Scratch	Splet (z možnostjo prenosa aplikacije na računalnik)	8-16	Okolje za programiranje z delčki	Brezplačno
Snap!	Splet (z možnostjo prenosa	12-20	Okolje za programiranje z delčki	Brezplačno

	aplikacije na računalnik)			
Alice	Windows, Mac OS X, Linux	12-20	Okolje za programiranje z delčki	Brezplačno
Tynker	Splet, iOS	7+	Okolje za programiranje z delčki, JavaScript, Python	Plačljivo

Za učenje programiranja se lahko uporabljajo tudi računalniške igre, bodisi s spodbujanjem učencev za razvoj in izdelavo lastnih video iger (učenje s snovanjem iger) bodisi z omogočanjem igranja izobraževalnih iger, ki so namenjene usvajanju učnih ciljev povezanih s programiranjem (učenje z igranjem iger). Računalniške igre, namenjene izobraževalnim dejavnostim lahko ustvarijo motivacijsko in zabavno učno okolje, saj vključujejo aktivnosti, ki dosegajo izobraževalne standarde, učne cilje, podajajo povratno informacijo in omogočajo doseganje visokih učnih rezultatov (Georgieva & Tuparova, 2019).

Ideja o uporabi iger za učenje programiranja izvira iz dejstva, da se učenci z njihovo uporabo bolj zavzeto in motivirano učijo računalniškega mišljenja in programerskih veščin v zabavnih in domačih okoljih, preden te veščine prenesejo na učenje programskega jezika (Kazimoglu idr., 2012). Prav tako po Kazimoglu idr. (2012) izobraževalne igre za učenje programiranja in računalniškega mišljenja ne bi smele biti ustvarjene samo za zabavo, ampak bi morale upoštevati tudi učni načrt in veščine, pokazati razlike med različnimi programskimi koncepti in spodbujati dobre programerske prakse (z igrifikacijo, kot je doseganje največjega števila točk).

Izobraževalne igre so lahko uporabne tudi za programiranje s preoblikovanjem neprijetnih operacij v zanimive izkušnje - Shabanah idr. (2010) so ustvarili Algorithm Game Designer, ki je namenjen ustvarjanju algoritmičnih iger, da bi povečali zanimanje za sisteme za vizualizacijo algoritmov. Grivokostopoulou idr. (2016) so, da bi študentom pomagali z vizualizacijami in animacijami algoritmov umetne inteligence razumeti delovanje teh algoritmov, njihove funkcije in kako posamezna odločitev temelji na specifičnih parametrih, razvili igro za poučevanje algoritmov umetne inteligence, ki temelji na igri Pacman.

S hitrim razvojem in vpeljevanjem vse hitrejših internetnih povezav in večanjem razpoložljivosti domačih računalnikov so se začele pojavljati platforme za spletne igre kot na primer CodeComabt in LightBot. Combéfis idr. (2016) so pregledali glavne vrste spletnih platform za poučevanje programerskih veščin in prišli do zaključka, da lahko naslednji dejavniki naredijo izobraževalno igro bolj motivacijsko in uspešno: 1) proces podajanja povratnih informacij in ocenjevanja, 2) estetika, 3) sodelovalni in večigralski vidiki, 4) vodenje skozi igro, 5) brez negativnih posledic, 6) glasba, 7) srednji nivo zahtevnosti izzivov za ohranjanje interesa igralcev.

Miljanovic idr. (2018) so raziskovali, kakšne učne cilje pokrivajo različne izobraževalne igre. Pregledali so 49 izobraževalnih iger za učenje programiranja in ugotovili, da se večina izobraževalnih iger osredotoča na reševanje problemov in temeljne programerske koncepte, nekaj iger na podatkovne strukture, razvojne metode in snovanje programske opreme.

Obstajajo tudi igre, ki sicer niso namenjene učenju programerskih veščin, ampak lahko na posreden, a učinkovit, način pomagajo pri učenju ključnih spretnosti računskega mišljenja, kot so abstrakcija (npr. Tetris), dekompozicija (npr. Sudoku), algoritemsko razmišljanje (npr. miselne igre), evalvacija (npr. spominske igre) in generalizacija (npr. igre z vzorci) (Franković idr., 2018).

Pri učenju s snovanjem iger se pogosto kombinira uporabo vizualnih programskih jezikov in njihovih okolij za programiranje z delčki z načrtovanjem in kodiranjem iger. V raziskavi, v kateri so primerjali uporabo programerskih okolij Scratch in Pascal (proceduralni jezik z ukazno vrstico ali urejevalnikom besedila) pri programerjih začetnikih, so ugotovili, da so tisti učenci, ki so uporabljali Scratch, veliko bolj motivirani za nadaljnje učenje računalništva (Ouahbi idr., 2015). Raziskava je prav tako pokazala, da je snovanje iger in zgodb povzročilo, da so bili učenci bolj ustvarjalni in samostojni pri učenju programiranja. Weintrop idr. (2015) so v svoji raziskavi primerjali uporabo Snap!-a in Jave pri srednješolcih in ugotovili, da je pri programiranju z delčki strmejša krivulja učenja kot pri učenju programiranja v Javi, kar je v skladu s prepričanjem, da je pristop k učenju programiranja z uporabo okolij za programiranje z delčki (kot sta Scratch in Snap!) primernejši za programerje začetnike. V skladu z ugotovitvami raziskave so bili razlogi za to lažje branje delčkov, zaradi njihove oblike, ki pomaga pri razumevanju, kako jih je mogoče uporabiti, lažje jih je sestavljati in služijo kot pomoč pri pomnjenju.

Integrirani pristop k poučevanju programiranja in njegov vpliv na dosežke 14-letnih učencev obravnavajo Tuparova, Nikolova, Tuparova (2020). Ta pristop združuje dve stopnji:

- Uporabo obstoječih izobraževalnih računalniških iger: za pridobivanje znanja in spretnosti, motivacijskost učnih aktivnosti, razvoj algoritmičnega razmišljanja, za eksperimentiranje z obstoječimi izobraževalnimi računalniškimi igrami za raziskovanje pravil iger, elementov uporabniškega vmesnika in njihovih lastnosti in dogodkov
- Snovanje in razvoj izobraževalnih računalniških iger z uporabo matematičnih modelov iger, razvojem grafičnega uporabniškega vmesnika (GUI), kodiranje, testiranje in preverjanje.

Za implementacijo pristopa je bilo uporabljeno programsko okolje C#. Model je preizkušen na 126 študentih s področja informatike, razdeljenih v dve skupini - eksperimentalno in kontrolno skupino. Dosežki študentov so bili merjeni s praktično nalogo in testom na papirju. Rezultati kažejo, da med dosežki fantov in deklet v eksperimentalni skupini ni bilo razlike. Toda avtorji so ugotovili, da so dekleta v eksperimentalni skupini dosegla višje rezultate kot dekleta v kontrolni skupini.

4.2. Okolje za poučevanje programiranja z uporabo iger

V nadaljevanju bomo predstavili nekaj primerov uporabe različnih okolij, s katerimi lahko realiziramo uporabo iger pri učenju programiranja.

4.2.1. Učenje s snovanjem in izdelavo iger

Scratch!

Slika 4.1 Primer skripte v okolju Scratch

Scratch je programsko okolje za blokovno programiranje, ki omogoča učencem, da sestavljajo programe tako, da združujejo bloke, ki predstavljajo različne programske konstrukte in pri tem dobivajo vizualne povratne informacije (Weintrop idr., 2015). Ta način jim omogoča spoznavanje osnov programiranja brez ukvarjanja s sintakso programskega jezika (Ouahbi idr., 2015).

Scratch lahko, zlasti pri mlajših učencih, uporabimo za uvod v programiranje, prav tako pa kot način za olajšanje prehoda na druga programska okolja, s čimer uvajamo in spodbujamo zanimanje za računalništvo (Wolz idr., 2008). Rezultati raziskave, ki sta jo izvedla Meerbaum-Salant s sodelavci (2013) kažejo, da lahko večina učencev s pomočjo programa Scratch doseže soliden nivo razumevanja konceptov pri računalništvu. Prav tako so ugotovili, da je pri obravnavanju tem, ki zahtevajo večjo stopnjo abstrakcije prihajalo do težav, ki pa jih je bilo možno premostiti, če je učence v procesu spremljal učitelj.

Spletna stran programa Scratch zagotavlja podporo mreži uporabnikov iz celega sveta. Uporabniki se preko nje združujejo v skupnost programerjev in si med seboj delijo svoje projekte (Meerbaum-Salant idr., 2013). Scratch je na voljo v več kot 50 jezikih, vključno z bolgarščino, hrvaščino, angleščino, grščino, italijanščino, portugalščino, slovenščino in

turščino. Program si lahko prenesemo in namestimo na računalnik ter ga uporabljamo brez internetne povezave.

Snap!

Snap! je programsko okolje za blokovno programiranje. Njegova zasnova temelji na programu Scratch, vendar vključuje nekaj dodatnih funkcionalnosti. Snap! je, podobno kot Scratch, na voljo kot spletna različica, ki pa jo je mogoče uporabljati preko brskalnika tudi brez internetne povezave. Poleg nabora funkcionalnosti, ki jih vsebuje Scratch, pa Snap! ponuja dodatne možnosti uporabe razrednih seznamov, procedur, sličic, izgledov, zvokov in nadaljevanj, kar ga naredi bolj primernega za naprednejše uporabnike. Na voljo je v več kot 40 jezikih, vključno z bolgarščino, hrvaščino, angleščino, grščino, italijanščino, portugalščino, slovenščino in turščino.

Slika 4.2 Snap! Uporabniški vmesnik

Alice

Slika 4.3 Alice 3 Urejevalnik kode

Alice je programsko okolje za blokovno programiranje. Zasnovan je bil z namenom, da bi motiviral učence za programiranje s spodbujanjem njihove ustvarjalnosti preko ustvarjanja animacij, interaktivnih zgodb in preprostih 3D iger (Kelleher idr., 2007). Orodje vsebuje urejevalnik Virtual World Editor (Urejevalnik navideznih svetov), ki omogoča ustvarjanje navideznega sveta, v katerega lahko učenci vstavljajo 3D predmete in nanje dodajajo različne funkcije oz. metode. Ko je navidezni svet ustvarjen, pa lahko učenec sestavi programsko kodo, s katero ustvari logiko igre/interaktivne zgodbe oz. animacije (Sykes, 2007). Program Alice je primeren za učence, ki nimajo programerskega predznanja, saj jim omogoča, da vidijo kako se program izvaja, medtem ko sestavljajo programsko kodo (Cooper idr., 2000).

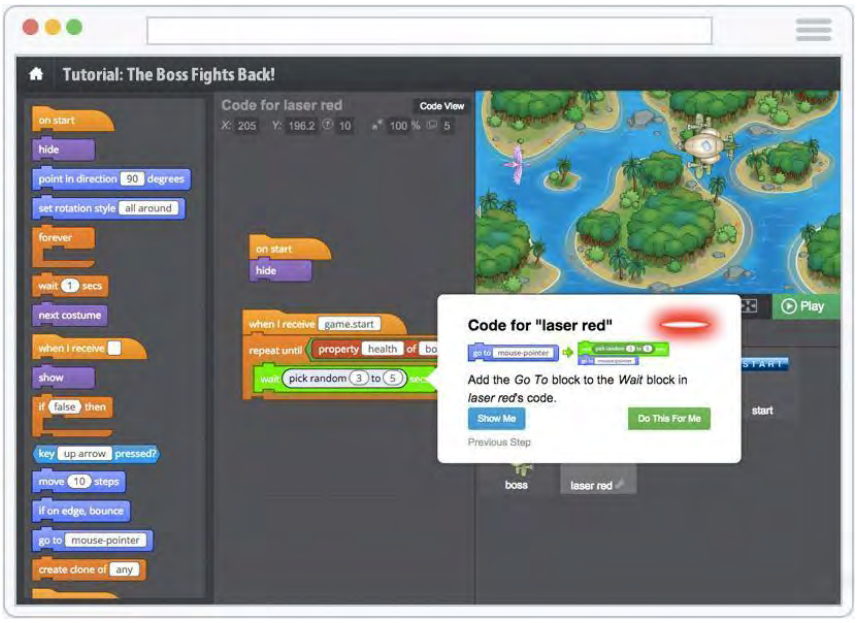
Rezultati neke druge študije, ki se je osredotočila na program Storytelling Alice (Pripovedovanje zgodb Alice - programsko okolje, ki temelji na različici programa Alice 2 in ima dodana orodja za kreativno pisanje zgodb), so pokazali, da sta bila osnovna in razširjena različica programa enako učinkoviti pri učenju programerskih konceptov pri dekletih, vendar pa so bila dekleta pri različici z dodanimi orodji za kreativno pisanje zgodb bolj motivirana za delo in so posvetila več časa programiranju. To pa je imelo pozitiven vpliv pri programiranju.

Alice 3 je najsodobnejša različica orodja, ki je na voljo v 13 jezikih, vključno z angleščino, portugalščino, grščino, slovenščino in bolgarščino, angleščino, španščino, portugalščino in nemščino.

Tynker

Tynker je izobraževalna platforma za programiranje, ki temelji na vizualnem programskem jeziku tipa povleci in spusti. Za razliko od prej predstavljenih programov je Tynker plačljiv program. Eden od vidikov, ki ga razlikuje od okolja Scratch je, da h kodiranju pristopa na način izdelovanja igre, v kateri morajo uporabniki napisati programe za premikanje in interakcijo glavnega lika ter izvajanje nalog. To naredi tako, da predstavi probleme, ki jih morajo igralec rešiti, s tem pa omogoči, da otroci pri tem začnejo prepoznavati vzorce (Geist, 2016). V program je vključen vodič, ki podaja navodila korak za korakom, tako da se lahko učenec nauči kako uporabljati različne programerske koncepte. Prav tako omogoča vizualizacijo blokov v kodi JavaScript, kar olajša razumevanje delovanja bloka v tekstovnem programskem jeziku.

Tynker ponuja več kot 23 tečajev programiranja, 11 tečajev za iPad in več kot 2000 različnih programerskih aktivnosti. Podjetje ponuja individualne in družinske naročniške sheme (za do 4 člane). Na voljo je samo v angleščini.

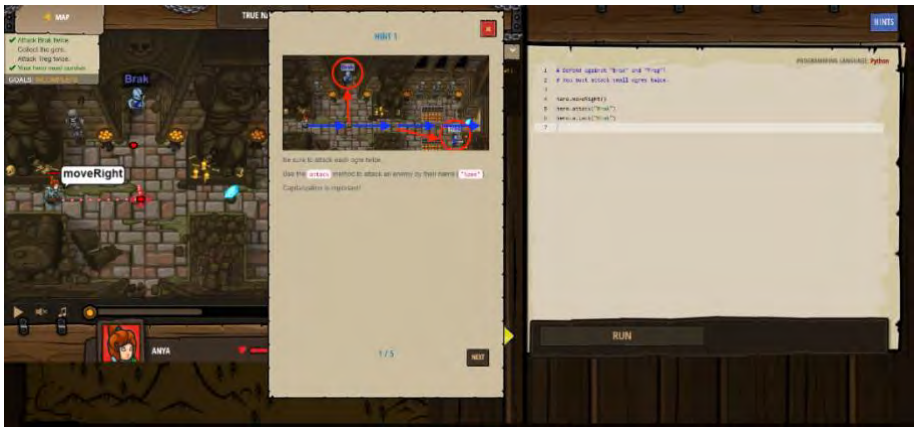


Slika 4.4 Uporabniški vmesnik programa Tynker

4.2.2. Učenje programiranja s pomočjo iger

V nadaljevanju bomo predstavili nekaj platform, ki temeljijo na igrah in so namenjene razvoju programerskih sposobnosti.

Code Combat



Slika 4.5 Igranje igre Code Combat

Code Combat je spletna pustolovska igra. Avtorji igre ponujajo naročniške sheme za posamezne učence, kot tudi za celotne razrede. Za učitelje so na voljo materiali, ki jih lahko uporabljajo pri pouku. Igra ima več kot sto brezplačnih in plačljivih nivojev. Na voljo je v več kot 50 jezikih, vključno z bolgarščino, hrvaščino, angleščino, grščino, italijanščino, portugalščino, slovenščino in turščino.

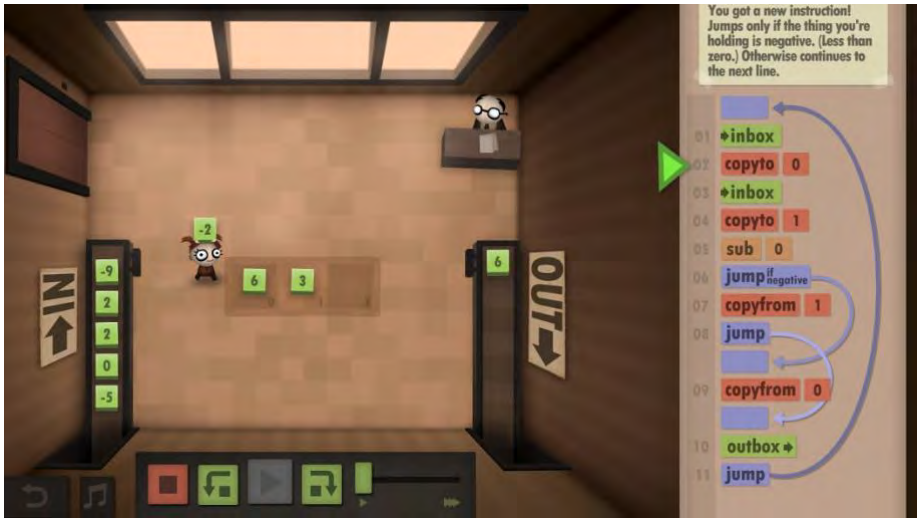
Učenci morajo v igri pisati svojo programsko kodo (v programskem jeziku JavaScript ali Python) in s tem doseči, da se glavni lik v igri premakne, komunicira in rešuje različne naloge. Programiranje torej ni blokovno, saj morajo učenci napisati svojo programsko kodo. Med igranjem ima učenec na voljo namige, koncepti pa so podani postopoma. Igra je razdeljena v pet različnih svetov, v katerih se igralec med napredovanjem skozi igro, seznanja z različnimi koncepti: 1) Kithgard Dungeon (Temnica Kithgard) – sintaksa, metode, parametri, nizi, zanke in spremenljivke; 2) Backwoods Forest (gozd Backwood) – pogojni stavek, Boolova logika, relacijski operatorji, funkcije, lastnosti objektov, upravljanje z dogodki in vhodnimi podatki; 3) Sarven Desert (puščava Sarven) – aritmetika, števci, while zanka, ukaz break, ukaz continue, tabele, primerjave nizov, iskanje ekstremov - minimalna/maksimalna vrednost; 4) Cloudrip Mountain (Gora Cloudrip) – literali, oddaljene metode, klicanje funkcij, for zanka, kompleksne

funkcije, risanje, modularna aritmetika; 5) Kelvintaph Glacier (ledenik Kelvintaph) – napredne tehnike.

Po Miljanoviču in sodelavci (2018) izobraževalna vsebina igre vključuje različne konceptualne vidike načrtovanja algoritmov in reševanja problemov, temeljne koncepte programiranja (kot npr. sintaksa in semantika, spremenljivke, osnovni podatkovni tipi, izrazi in prirejanje, vhod in izhod, pogojni stavki, iteratorji, funkcije in parametri ter rekurzijo) in osnovne podatkovne strukture (kot npr.: seznami, heterogeni agregati in abstraktni podatkovni tipi).

Human Resource Machine

Human Resource Machine (Stroj za človeške vire) je računalniška igra, v kateri moramo reševati različne uganke. Temelji na vizualnem programskem jeziku. Igralec programira pisarniškega uslužbenca in s tem rešuje naloge ter napreduje do težjih ugank (Johnson idr., 2016). To naredi tako, da na način povleci in spusti iz ukazov sestavlja zaporedja potez. Pri napredovanju skozi igro odklepa bolj kompleksne ukaze, ki mu omogočajo izvajanje kompleksnih operacij. S pomočjo vgrajenega vizualnega programskega jezika in možnosti primerjanja različnih algoritmov se lahko z igro naučimo temeljnih programerskih konceptov.

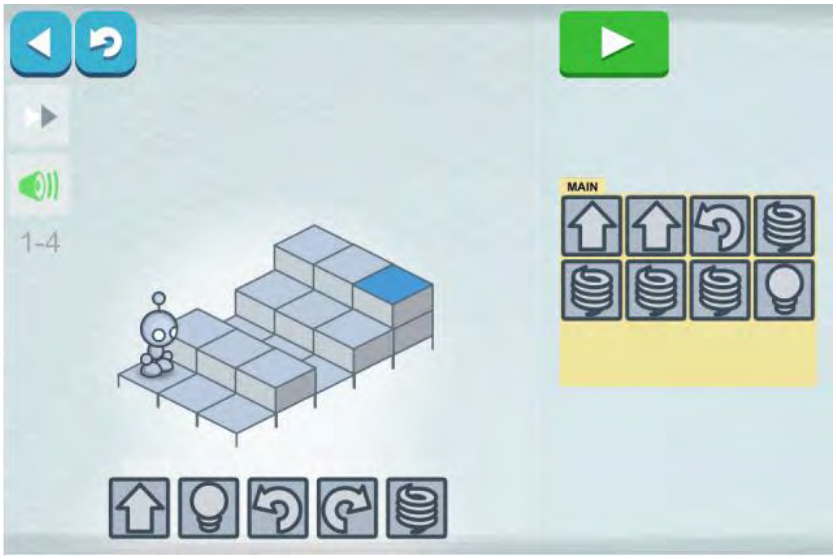


Slika 4.6 Igranje igre Human Resource Machine

LightBot

LightBot je igra, ki temelji na reševanju ugank in je po načinu igranja podobna igri Human Resource Machine. Za napredovanje preko nivojev moramo uporabljati logično mišljenje.

Igralec nadzira robota, s katerim osvetljuje ploščice, na ta način pa mora rešiti 40 stopenj. Ukazi, ki jih lahko uporablja v igri, so prikazani z ikonami.



Slika 4.7 Uporabniški vmesnik igre Lightbot

Čeprav v igri ni eksplicitnega učenja programskega jezika, lahko igralci z njeno pomočjo pridobijo znanje o zaporednem izvajanju ukazov in implementaciji algoritmov s poudarkom na veščinah reševanja problemov (Miljanovic idr., 2018). Kljub temu pa se lahko učenci z igro naučijo različnih temeljnih konceptov programiranja, kot so: pogojni stavek, zanke, iteracije, rekurzija in strategije za razhroščevanje. Prostor za ploščice je omejen, zato mora igralec upoštevati tudi učinkovitost kode.

Študija, ki jo je izvedel Mathrani s sodelavci (2016) je pokazala, da so uživali pri igranju in da se je izkazala za učinkovit način učenja programskih konstruktov, kot so: funkcije, procedure, pogoji in rekurzija. Večina učencev, ki so bili vključeni v raziskavo se je strinjala, da je takšen pristop zanje učinkovit način za usvajanje konceptov, ki bi jih sicer težje razumeli.

May's Journey

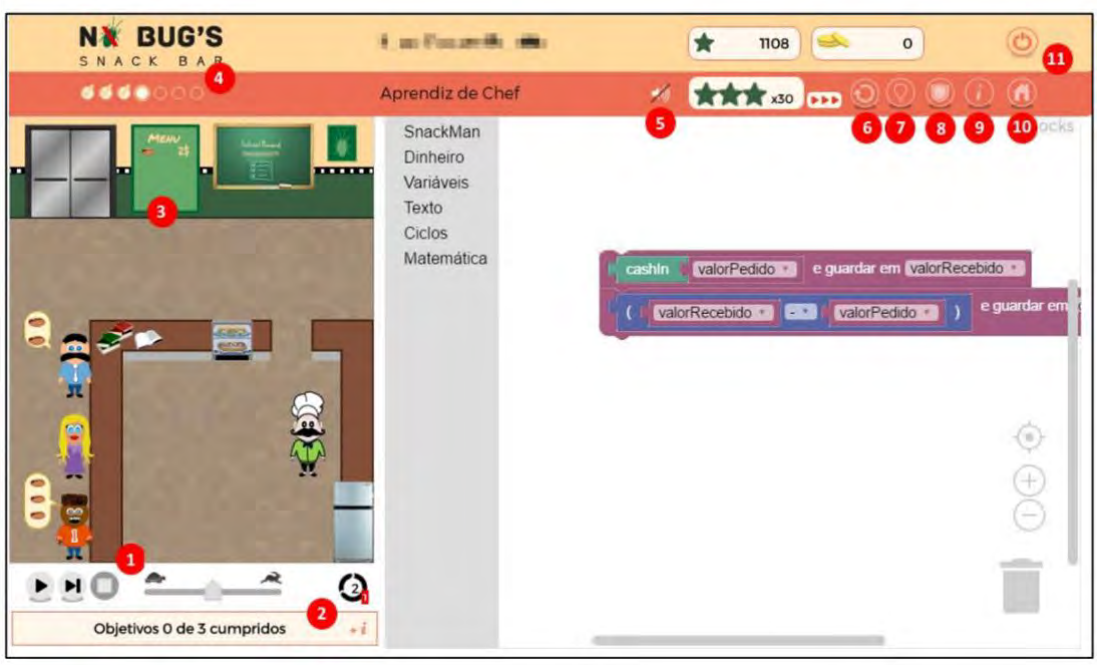
May's Journey (May-ino potovanje) je edina igra s tega seznama, ki je neposredno namenjena osnovnošolkam. Gre za 3D igro, ki temelji na reševanju ugank. Igralec mora s pomočjo uporabe programskega jezika, ki je podoben Javi, najti pot iz labirinta. Igra je zasnovana z namenom, da bi pritegnila zanimanje deklic za računalništvo. Razvijalci so v igro vključili psevdokodo, ki omogoča enostavnejši prehod med vizualnimi in tekstovnimi programskimi jeziki. Učna vsebina, ki je vključena v igro vsebuje: osnovne ukaze, logiko zaporednega izvajanja, zanke, spremenljivke, operatorje za primerjanje in Boolovo logiko, operacije na celih številih in nizih (Jemmali, 2016).

Slika 4.8 Igranje igre May's Journey

V igri se izmenjujeta dve fazi: 1) faza raziskovanja, ki je narejena kot običajna igra ter 2) faza programiranja. V fazi programiranja se uporabniški vmesnik razdeli na dve podokni, igralec v eno piše ukaze, v drugem pa lahko opazuje izvajanje svojega programa. Igralec dobi namige za pisanje kode tudi med fazo raziskovanja. Glavna junakinja v zgodbi živi v svetu, ki propada in je ločena od svojega prijatelja. Njena naloga je razreševanje skrivnosti, s čimer postopoma popravlja svet v igri. Deli skrivnosti se razkrivajo postopoma, kar pritegne k igranju. Ciljna skupina igre so deklice v osnovni šoli, zato je tudi glavni lik oblikovan tako, da ponazarja deklico teh let, kar pripomore, da se deklice med igranjem bolj poistovetijo z glavno junakinjo. Tudi grafična podoba igre je prilagojena ciljni publiki, saj so avtorji igre uporabili toplo barvno shemo in manj agresivne ilustracije. Igro je ocenjevala testna skupina, ki je izpostavila kvaliteto zgodbe.

No Bug's Snack Bar

No Bug's Snack Bar (Okrepčevalnica brez žuželk) je resna igra, ki je nastala v raziskovalne namene. Temelji na sistemu povleci in spusti ter se osredotoča na veščine reševanja problemov. Učni cilji, ki jih igra obravnava se nanašajo na manipuliranje spremenljivk, zaporedje ukazov, pogoje, iteratorje in strategije za razhroščevanje (Vahldick, 2017).



Slika 4.9 Igranje igre No Bug's Snack Bar

Glavni lik v igri dela v okrepčevalnici, igralec pa ga upravlja s pomočjo kode. Na ta način rešuje različne naloge. Ena od inovacij, ki so vključene v igro, je orodje, ki omogoča učiteljem spremljanje napredka učencev. Prav tako vsebuje sistem igrifikacije, ki omogoča, da učenec med napredovanjem preko igre pridobiva točke, ki odražajo njegovo uspešnost pri reševanju, na ta način pa se trudi najti bolj optimalne rešitve za podane probleme. V igro je vključen asistent, ki ponuja namige. Ta je povezan z administrativnim sistemom, tako da lahko učitelj ugotovi, kdo so učenci, ki potrebujejo pomoč, poleg tega pa omogoča, da jim pošlje njim prilagojene namige. Prisotnost učitelja je pri igri zato ključna.

Robot ON!

Robot On! (Robot Prižgan!) je še ena raziskovalna igra, ki temelji na reševanju ugank. Namenjena je dodiplomskim študentom, ki se učijo programskega jezika C++. V igri prevzamemo vlogo znanstvenika, ki ima za cilj aktivirati t.i. “Mech Suit” (robotsko obleko). To lahko naredimo tako, da rešimo niz nalog. Ob vsaki rešeni stopnji igre, se aktivira nov robotski sistem. Igra omogoča učiteljem, da ustvarijo svoje stopnje, v katerih lahko uporabijo poljuben programski jezik. Igralcu so na voljo orodja različnih barv, na katere se nanašajo deli kode iste barve. Igralec se seznani z različnimi orodji s pomočjo vodičev (Miljanovic idr., 2016).

Slika 4.10 Igranje igre Robot ON!

Igralec med napredovanjem v igri dobi naslednja orodja: 1) orodje za aktivacijo (za učenje nadzora nad vrstnim redom izvajanja ukazov), 2) orodje za komentiranje in od-komentiranje (za učenje načina kako se koda izvaja), 3) orodje za poimenovanje (za učenje koncepta spremenljivke) in 4) orodje za preverjanje (za učenje toka podatkov).

Educational Pacman Game

Educational Pacman Game (Izobraževalna igra Pacman) je raziskovalna resna igra, ki je namenjena poučevanju iskalnih algoritmov. Študentom omogoča, da s pomočjo grafičnega prikaza opazujejo kako se obnašajo različni iskalni algoritmi (Grivokostopoulou idr., 2016).

Slika 4.11 Izobraževalna igra Pacman

Igra ponuja dva načina za igranje:

1) Izobraževalni način: študent si lahko za poljuben algoritem prebere njegov besedilni opis in psevdokodo ter prikaže diagram poteka. Igra mu omogoča, da se algoritma nauči s pomočjo vizualizacije na različnih labirintih.

2) Način igranja: študent mora pod različnimi pogoji reševati stopnje v labirintu, pri tem pa je omejen s časom. Stopnje so narejene tako, da mora študent uporabiti točno določen algoritem za iskanje, s katerim premika Pacman-a po labirintu. Nalogo začne reševati iz naključne pozicije, njegov cilj pa je, da pobere češnjo ali nek drug predmet, ki ga naredi močnejšega. To lahko naredi tako, da lik Pacmana premika po določenem algoritmu.

CMX

CMX je množična večigralska spletna igra vlog, ki je namenjena učenju programiranja. V igri sta dve ekipi – krekerji in hekerji, ki med seboj tekmujeta pri iskanju gesel za globalno tovarno, ki proizvaja strupene odpadke. Igralcem pri tem pomagajo t.i. sensei (sensei je učitelj borilnih veščin), ki so jim v pomoč pri učenju programskega jezika C. Igra vključuje tri ravni učiteljev sensei, ki jih je možno doseči z odklepanjem gesel v prejšnji stopnji. Pri študentih, ki so igrali to igro, je bilo mogoče opaziti povišanje nivoja razumevanja programskega jezika C, saj so ustrezno odgovarjali na teoretična vprašanja, s pomočjo orodij, ki delujejo na način povleci in spusti so sestavljali izvršljive programe, prav tako pa so pisali programe v jeziku C (Malliarakis, 2014).

Slika 4.12 Igra CMX

Table 4.2 Primerjava okolij za poučevanje programiranja z uporabo iger

IGRA	PLATFORMA	STAROST	JEZIK(I)	UČNI CILJI	TIP IGRE
Code Combat	spletna	9+	JavaScript, Python	sintaksa, metode, parametri, nizi, zanke in spremenljivke, pogojni stavki (if/else), Boolova logika, relacijski operatorji, funkcije, lastnosti objektov, dogodkovno programiranje, obdelava vhodnih podatkov, aritmetika, števci, while zanka, ukaz break, ukaz continue, tabele, primerjave nizov, iskanje minimalnega/maksimalnega elementa, literali objektov, oddaljene metode, klici, for-zanke, kompleksne funkcije, risanje, modularna aritmetika	Komercialna igra (*Zastonj*)
Human Resource Machine	Windows, Mac OS X, Linux, iOS, Android	9+	Vizualni programski jezik	Vhod in izhod, pogoji, iteracije Primerjava algoritmov	Komercialna igra (*Plačljiva*) Reševanje ugank
Lightbot	iOS, Android, Windows, Mac OS X	9+	Vizualni programski jezik	Pogoji, iteracije in rekurzija. Strategije za razhroščevanje.	Komercialna igra (*Plačljiva*) Žanr: Reševanje ugank
May's Journey	Windows	12-18	Poljuben programski jezik (zgleduje se po objektno orientiranih jezikih kot je npr. Java)	Osnovni ukazi, logika zaporedja ukazov, zanke, spremenljivke, pogojni stavki, primerjave in Boolova logika, operacije nad celimi števili, operacije nad nizi	Raziskovalna igra Reševanje ugank
No Bug's Snack Bar	Web-based	18+	Vizualni programski jezik	Delo s spremenljivkami, Zaporedje akcij, Pogoji in iteracije, Strategije za razhroščevanje	Raziskovalna igra
Robot ON!	Windows	18+	C++	Sintaksa, semantika, spremenljivke, osnovni podatkovni tipi, izrazi, prirejanje vrednosti, pogoji in iteracije, razumevanje programa.	Zastonjska/ Raziskovalna igra

Educational Pacman Game	Windows	18+	Igra za učenje iskalnih algoritmov pri umetni inteligenci	Algoritmi (osnovni besedilni opis algoritmov, pripadajoča grafična ponazoritev z diagramom poteka in psevdokodo)	Raziskovalna igra
CMX	Windows	18+	C	Teorija o tabelah, izhodi spremenljivk po izvedbi programa, ki vključuje tabelo, sintaksa programov s tabelami, zapisovanje podatkov v tabelo, računanje vsote podatkov v tabeli, iskanje maksimalne/minimalne vrednosti v tabeli.	Raziskovalna igra MMORPG

5. CODING4GIRLS IGRALNA PLATFORMA ZA UČITELJE IN UČENCE

5.1. Coding4Girls igralna platforma in snovalsko razmišljanje

Idealna kombinacije IKT in izobraževalnih iger je prikazana kot rezultat projekta *O2-Promocija razvoja programerskih veščin med dekleti skozi izobraževalne igre*, katerega cilj je razvijanje programerskih večin mladih, zlasti deklet, v osnovnih in srednjih šolah z uporabo programske opreme, ki smo jo razvili v okviru tega projekta.

Ta programska oprema je bila zasnovana za premagovanje obstoječe vrzeli med udeležbo moških in žensk v računalniškem izobraževanju in karierah na področju računalništva z uvedbo metodoloških učnih posegov, ki računalništvo naredijo privlačnega za vse. Ta programska oprema in pripravljene učne dejavnosti so bile razvite s ciljem spodbujanja vključenosti deklet v računalništvo. Zasnovani so bili po pristopu snovalskega razmišljanja (angl. design thinking), razumljenim kot ustvarjalni proces, ki učencem pomaga, da skupaj s sovrstniki oblikujejo izdelke, ki so jim pomembni.

Ta postopek oblikovanja je zelo učinkovit zaradi strukturiranega okvira za prepoznavanje izzivov, zbiranje podatkov, ustvarjanje potencialnih rešitev, izboljšanje idej in preskušanje rešitev.

Ta proces je po naravi rekurziven in zahteva interakcijo. Vsaka stopnja v procesu zahteva ponovni pregled in uporabo skozi celotno učno izkušnjo, da se spodbudi eksperimentiranje, izvedljivost rešitve in refleksijo.

Aktivnosti s snovalskim razmišljanjem omogočajo visoko stopnjo sodelovanja znotraj in izven razreda. Učenci so neposredno ukvarjajo z zbiranjem podatkov, izgradnjo znanja, komunikacijo in predstavitvijo.

V projektu Coding4Girls oblikovan pristop razmišljanja izkorišča glavne prednosti učenja z uporabo iger, da spodbudi in motivira učence v njihovem učnem procesu. Uporablja nekatere pomembne elemente igre (npr. točke in izzivi) in scenarije, ki s povečanjem stopnje vključenosti, motivacije in doseženih rezultatov naredijo posamezno aktivnost bolj zabavno.

Še posebej pomembna je uporaba izzivov, ki učencem pomagajo, da vidijo širšo sliko preden oblikujejo podrobne rešitve. Ti jih spodbujajo in izzivajo k podjetniškemu razmišljanju o digitalnih tehnologijah in k razmisleku o možnostih njihove uporabe za reševanje problemov iz realnega sveta.

Programska oprema, ki je bila razvita v okviru projekta, je sestavljena iz dveh različnih medsebojno povezanih delov: platforme za učitelje in igralnega okolja za učence.

5.2 Platforma za učitelje

Platforma za učitelje je spletna platforma, kjer lahko učitelji ustvarjajo in pripravljajo različne predmete za razvoj programerskih veščin svojih učencev z uporabo programskega okolja Snap!.

Znotraj platforme ima vsak učitelj na voljo tako zasebno kot javno območje. V prvem lahko pripravi svoje predmete, ki jih oblikuje na podlagi potreb svojih učencev. Javno območje pa je repozitorij vseh ustvarjenih predmetov, ki so jih ustvarili ostali učitelji. Cilj je deljenje, uporaba in pridobivanje idej na podlagi učnih aktivnosti, ki so jih že pripravili ostali kolegi (Slika 5.1). Vsak učitelj lahko personalizira vse javno dostopne predmete v skladu s specifiko njegovega razreda.

Slika 5.1 Platforma za učitelja in nekaj programerskih predmetov, ki so na voljo na njej

Ti predmeti so sestavljeni kot prostor za združevanje tematsko povezanih dejavnosti, ki so povezane s krovnim izzivom. Izziv je učencem predstavljen na samem začetku predmeta, kjer

lahko vsi skupaj sodelujejo v viharjenju možganov in skupaj izdelajo okvirne rešitve. To je zelo pomembna faza učnega procesa, kjer lahko učenci ustvarijo novo idejo in se kreativno potopijo v postopek reševanja problemov.

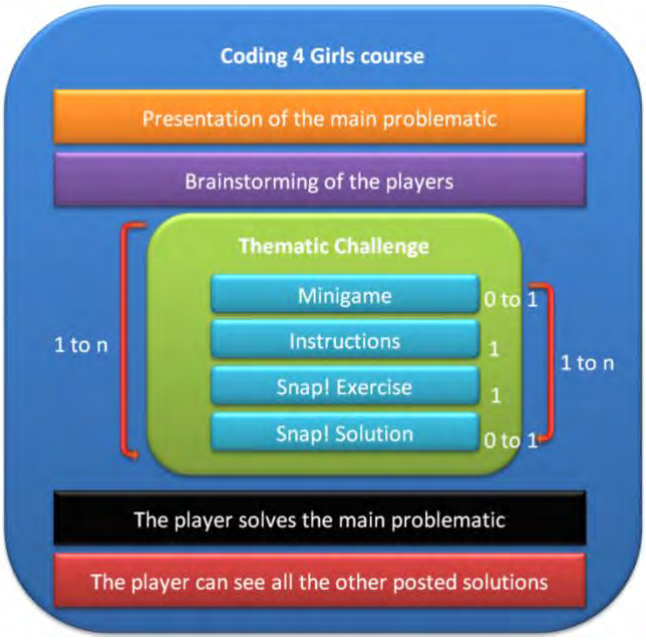
Kanvas za viharjenje možganov lahko pripravijo učitelji znotraj platforme za učitelje. Učencem tu zastavijo nekaj ključnih vprašanj ali kakšno pripombo ali refleksijo, ki jo učenci uporabijo med diskusijo s svojimi vrstniki. Programska oprema omogoča možnost priprave post-it listkov na virtualni deski, na katere se lahko doda besedilo, slike ali video kot prikazuje Slika 5.2:

Slika 5.2 Območje za viharjenje možganov v igralnem okolju za učence namenjeno diskusiji in sodelovanju med učenci

Vsi post-it listi imajo ikono ključavnice, ki je lahko odprta, da se vsebino listka lahko ureja, ali zaprta, da se vsebine liska ne da spreminjati. Samo učitelji imajo možnost zaklepanja in odklepanja listkov, učenci lahko le vidijo ali lahko listek urejajo ali ne.

Po koncu viharjenja možganov, učenci po korakih dobijo specifične zaporedje aktivnosti, s katerimi se seznanijo z orodji, ki jih potrebujejo za rešitev krovnega izziva. Te programerske aktivnosti so predstavljene na kanvasu s programskim okoljem Snap!, tako skozi delno izdelane naloge, ki spodbujajo učence k rešitvi problema, kot tudi s predstavitvijo končne rešitve.

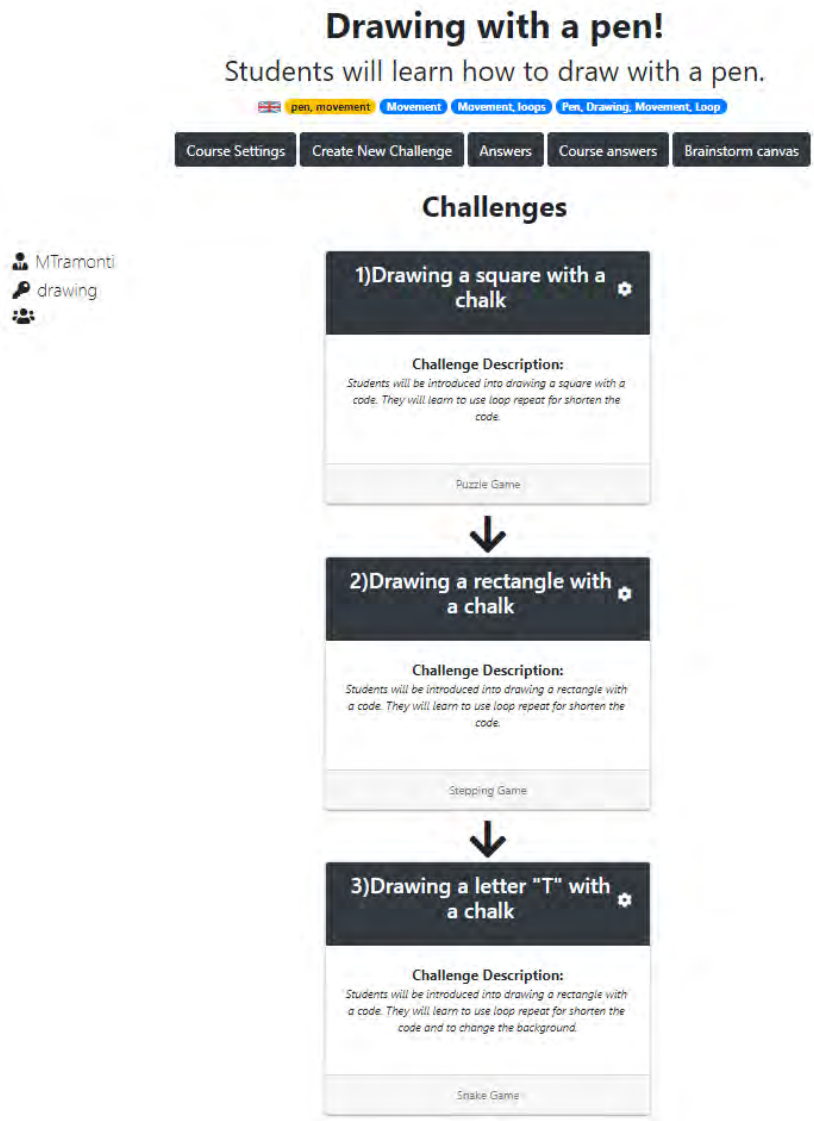
Slika 5.3 prikazuje strukturo posameznega Coding4Girls predmeta, ki je objavljen na platformi za učitelje.



Slika 5.3 Primer strukture Coding4Girls predmeta

Vsak skupek aktivnosti v predmetu se imenuje izziv. Ti so učencem predstavljeni v specifičnem vrstnem redu, ki ga določi učitelj (**Error! Reference source not found.**). Na primer, če želi učitelj ustvariti predmet o osnovnih programerskih znanjih, je lahko prva tema Boolova algebra, druga pogojni stavki in zadnja zanke. Tako lahko učitelj pripravi prilagojene učne korake za svoje učence na platformi za učitelje in učenci bodo znotraj igralnega okolja za učence sledili tem korakom do končnega cilja.

Seveda morajo učenci pred končno rešitvijo igrati in rešiti vse izzive po vrstnem redu, ki jim ga je določil učitelj.

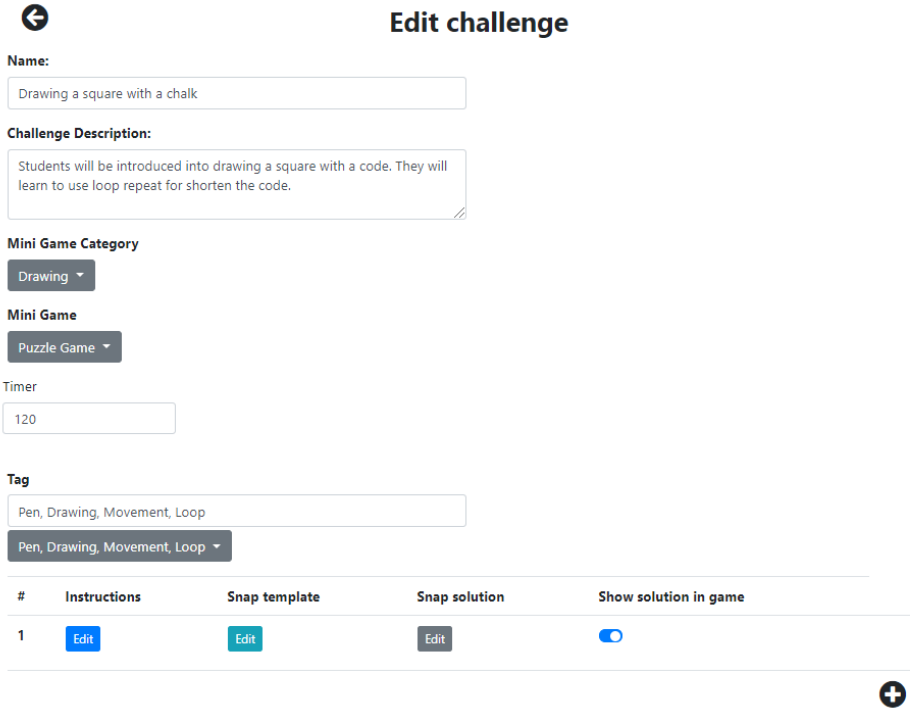


Slika 5.4 Primer izziva v predmetu v platformi za učitelje

Vsak izziv je lahko povezan z mini-igro, ki je bila razvita v okviru projekta. Cilj teh je, da učencem pomagajo bolje razumeti, kateri koncept spoznavajo v okviru tega izziva.

Na primer, če je izziv povezan s spoznavanjem zank, se lahko uporabi mini-igro 3 v vrsto.

Tako ima lahko vsak izziv poleg reševanja naloge na Snap! kanvasu, dodano mini-igro, ki jo učenci igrajo, ko preberejo navodila, ki jih je za njih prej pripravil učitelj v platformi za učitelje (Slika 5.5).



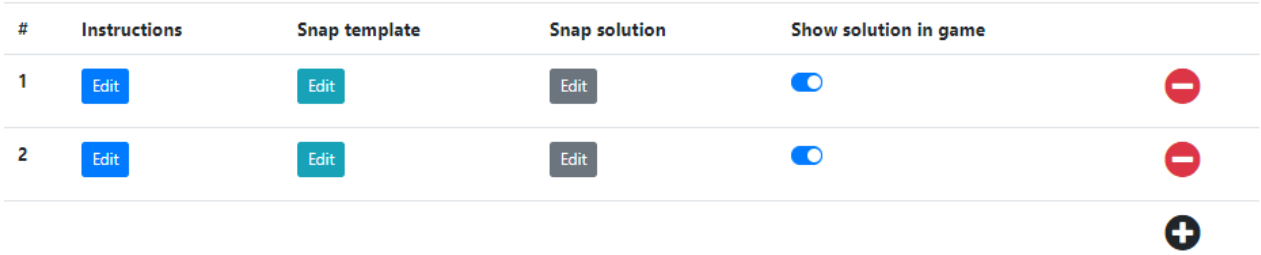
Slika 5.5 Urejanje izziva v platformi za učitelje

Vse igre, ki so na voljo (trenutno jih je 11) učencem, so povezane z in poenostavljajo dejanske programerske koncepte, ki so obravnavani v predmetih in scenarijih C4G. Izzivi so lahko sestavljeni tudi brez mini-iger. To je le možnost, ki je učiteljem na voljo.

Vsak izziv je sestavljen iz treh delov:

1. Navodila, kjer lahko učitelj poda nekaj namigov ali razlag o nalogi, ki jo morajo učenci opraviti.
2. Predloge v Snap!-u, kjer učenci poskusijo rešiti zadano nalogo z uporabo programskih delčkov odprtokodnega programskega okolja Snap! v igralnem okolju za učence.
3. Rešitev naloge v Snap!-u, kjer se lahko učitelj odloči, ali se bo pokazala končna rešitev naloge.

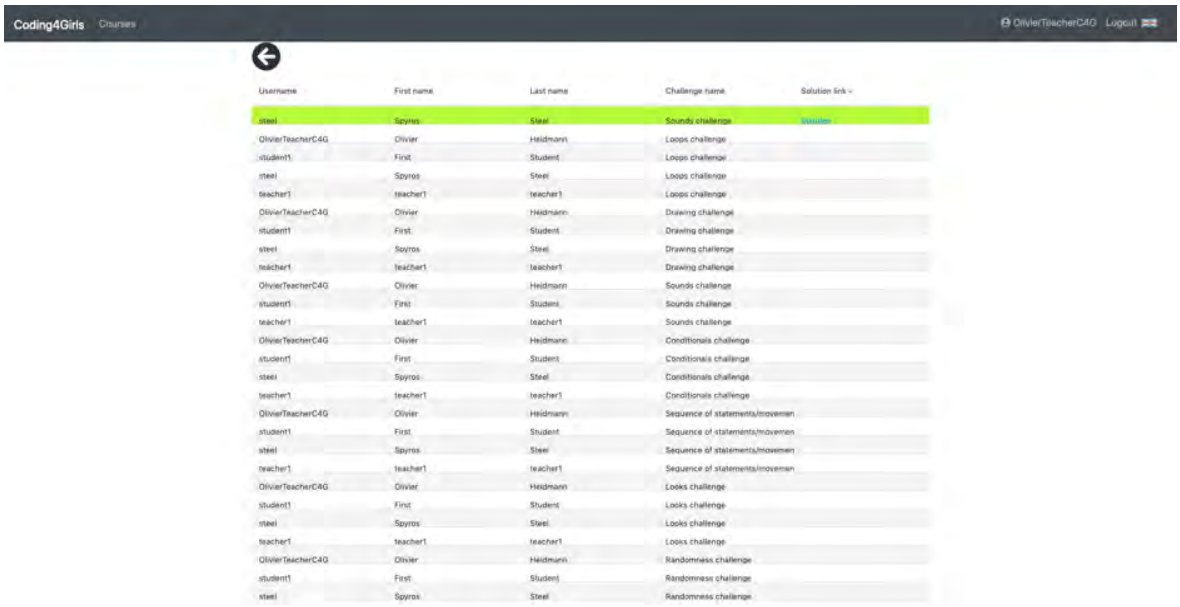
Če je izziv bolj kompleksen z vidika programskega znanja, je lahko razdeljen v več nalog (Slika 5.6). Na ta način bodo učitelji lahko izziv razdelili na več delov in tako svoje učence korak za korakom vodili skozi različne faze reševanja tega izziva.



Slika 5.6 Razdelitev kompleksnejše naloge na več delov

Poleg tega je učiteljem na platformi za učitelje na voljo seznam vseh rešitev v Snap!-u, ki so jih oddali njihovi učenci. Seznam je prikazan v tabeli, ki prikazuje podatke o uporabnikih in njihove rešitve za vse izzive znotraj predmeta. Vsaka vrstica v tabeli vsebuje uporabniško ime, ime, priimek, imej izziva in morebitno povezavo do rešitve.

Če rešitev še ni bila oddana, bo stolpec »Povezava do rešitve« prazen. V nasprotnem primeru bo vrstica označena in v zadnjem stolpcu se bo pojavila povezava z napisom »Rešitev«. Po kliku na povezavo se bo prikazala uporabnikov rešitev (Slika 5.7).



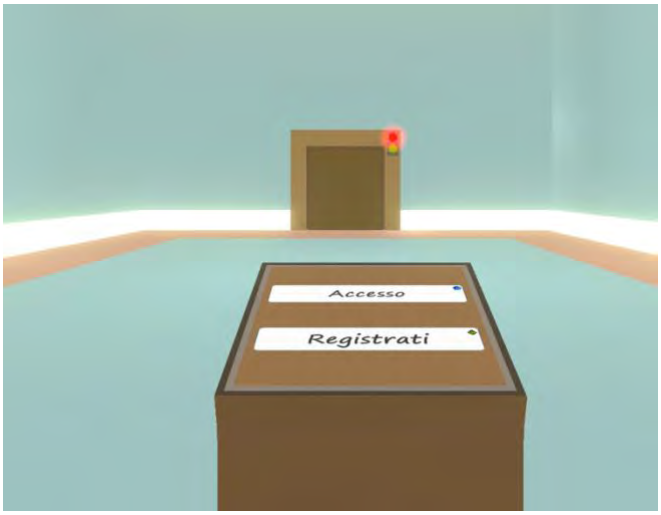
Slika 5.7 Rešitve izziva po posameznih uporabnikih z povezavo na posamezno rešitev

Na ta način ima učitelj vedno nadzoruje in spremlja napredek in rezultate svojih učencev.

5.3. Igralno okolje za učence

Učenci bodo do izzivov dostopali preko igralnega okolja za učence, ki je narejeno v Unity 3D in ga je potrebno prenesti na računalnik. V tem programskem okolju lahko učenci na zabaven, privlačen in igriv način odkrijejo in dokončajo predmete, ki jih je zanje pripravil učitelj.

Predmeti, ki uporabljajo elemente pristopa snovalskega razmišljanja, predstavljajo učencem krovni izziv za reševanje in orodja za njegovo postopno reševanje. Kot je opisano v prejšnjem poglavju, te izzive pripravijo učitelji in jih objavijo znotraj platforme za učitelje, učenci pa do njih dostopajo znotraj igralnega okolja za učence.



Slika 5.8 Prva soba za dostop do igralnega okolja za učence

Učenci se lahko pridružijo predmetu, če v terminal na portalu vnesejo pravilno kodo (Slika 5.8). Koda je unikatna in jo pridobi učitelj, ki je predmet pripravil na platformi za učitelje. Po vnosu uspešni prijavo v predmet, je na nad portalom vidno ime predmeta. To hkrati sporoča, kateri predmet je trenutno izbran (Slika 5.9).

Slika 5.9 Druga soba z dvema terminaloma: enim za prvi vpis v predmet in enim za izbor med predmeti, v katere je uporabnik že vpisan

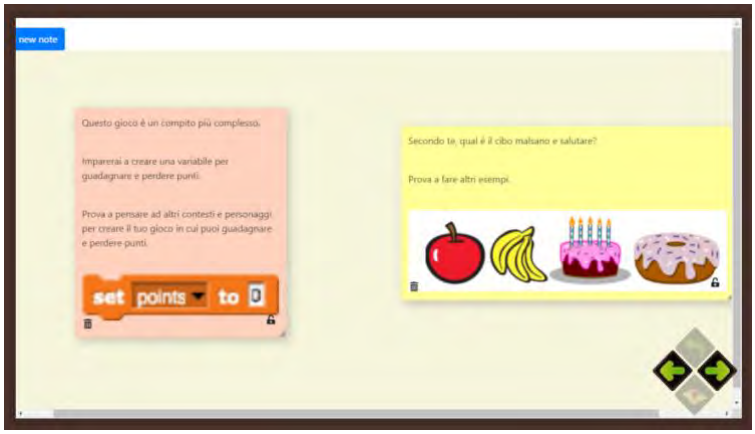
Učence spodbujamo k oblikovanju in kodiranju iger, ki naslavljajo določene potrebe ali probleme (odvisno od učiteljeve izbire). Pri tem uporabljamo pristop z velikim razponom težavnosti (angl. low floor high ceiling), ki omogoča učencem, da začnejo z lažjimi problem dokler ne pridobijo znanja za rešitev bolj zahtevnih problemov. Učitelji ustvarijo in predstavijo učencem delno rešene (angl. half-baked) scenarije, v katerih je rešitev delno pripravljena in učence spodbudi k izdelavi končne rešitve problema. Učitelji lahko za ta namen s programsko opremo Coding4Girls (C4G) pripravijo manjše in obvladljive module, ki so prilagojeni potrebam in znanju njihovih učencev.

V skladu s pristopom snovalskega razmišljanja je projektna ekipa pripravila nekaj predmetov in učnih scenarijev, ki so zasnovani tako, da učence izzovejo k rešitvi specifičnih programerskih izzivov. Trenutno je na voljo 21 učnih scenarijev, zbranih v poročilu “Zbirka učnih scenarijev za učenje programiranja na osnovi oblikovanja iger”, ki so bili prilagojeni za pristop snovalskega razmišljanja sledeč strukturi programskega okolja C4G.

Scenariji imajo različne nivoje zahtevnosti od preprostih do naprednih za bolj sposobne učence in so na voljo v angleščini, italijanščini, hrvaščini, grščini, bolgarščini, portugalščini, slovenščini in turščini.

Naloge je mogoče reševati individualno ali sodelovalno s spodbuditvijo skupinske diskusije v razredu ali virtualno v igralnem okolju za učence. Programsko okolje ponuja območje za

viharjenje možganov, kjer učenci razpravljajo in delijo svoje ideje z objavljanjem in urejanjem večpredstavnostnih post-it listkov (Slika 5.10). Vsak učenci, ki so vključeni v predmet lahko objavijo svoje prispevke, ki so vidni vsem vpisanim v ta predmet.



Slika 5.10 Območje za viharjenje možganov v učnem scenariju Kupovanje hrane za piknik v igralnem okolju za učence

Slika 5.11 prikazuje skupino izzivov v igralnem okolju za učence znotraj predmeta, ki ga je ustvaril učitelj na platformi za učitelje. Številke od 0 do 13 predstavljajo izzive. V tej skupini, izziv številka 0 predstavlja osnovna navodila, ki so podana na začetku igre. Ko jih učenec odpre, bo videl splošna navodila o izzivu, povezano predlogo v Snap!-u, ki mu sledi območje za viharjenje možganov. Ostali izzivi, oštevilčeni od 1 do 13, predstavljajo naloge, ki jih je pripravil učitelj na platformi za učitelje.

Slika 5.11 Skupina izzivov v igralnem okolju za učence

Z izbiro enega od njih, se bodo na dnu zaslona prikazale podrobnosti, z imenom izziva na levi, imenom povezane mini-igre na sredini in gumbom za začetek izziva na desni.

Slika 5.11 prikazuje imena izziva, ki ustreza temi, ki jo je potrebno usvojiti, v tem primeru »Pogojni izziv«. Poleg teme izziva sistem prikaže ime 3D mini-igre, ki je povezana s tem izzivom, t. j. »Poišči svojo pot«.

Učitelj se lahko odloči, katero mini-igro (opcijsko) bodo njegovi učenci igrali z izbiro ene od obstoječih mini-iger, kot na primer 3 v vrsto, Najdi svojo pot, Shramba/Frnikole, Igra korakov, Igra zvoka, Igra kača, Igra prič, Igra ujemanja vzorca, Igra korakov, Igralni avtomat in Več vprašanj.

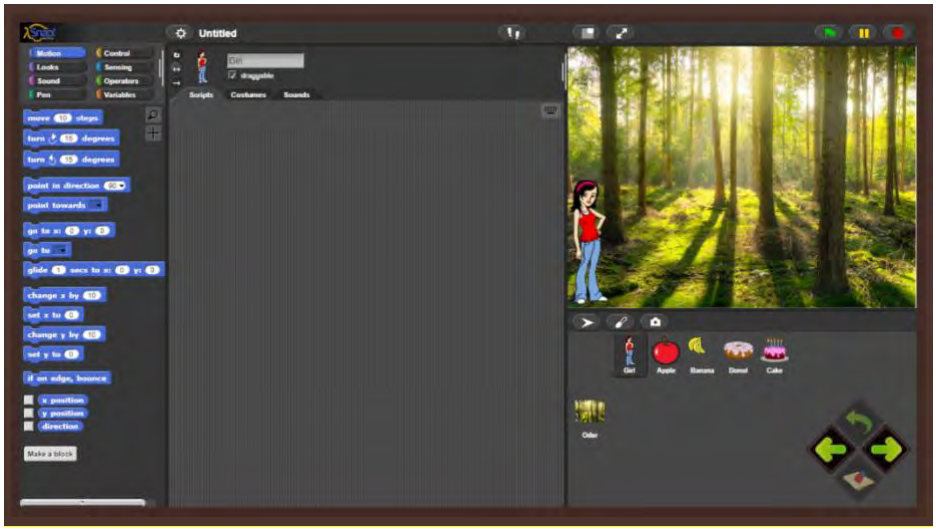
Slika 5.12 Primer mini-igre – Igralni avtomat

Vsak izziv v igralnem okolju za učence je sestavljen iz: uvodne mini-igre, ki ilustrira programski koncept (opcijsko), strani z navodili za nalogo, ki jih je potrebno rešiti v Snap!-u, dveh Snap! kanvasov – enega za reševanje naloge in drugega za prikaz rešitve naloge.

Stran z navodili je napisana v HTML-ju (Slika 5.13), običajno obogatena s sliko ali videom, za predstavitev konteksta in specifičnih ciljev učne naloge, ki jo je potrebni rešiti v Snap!-u.

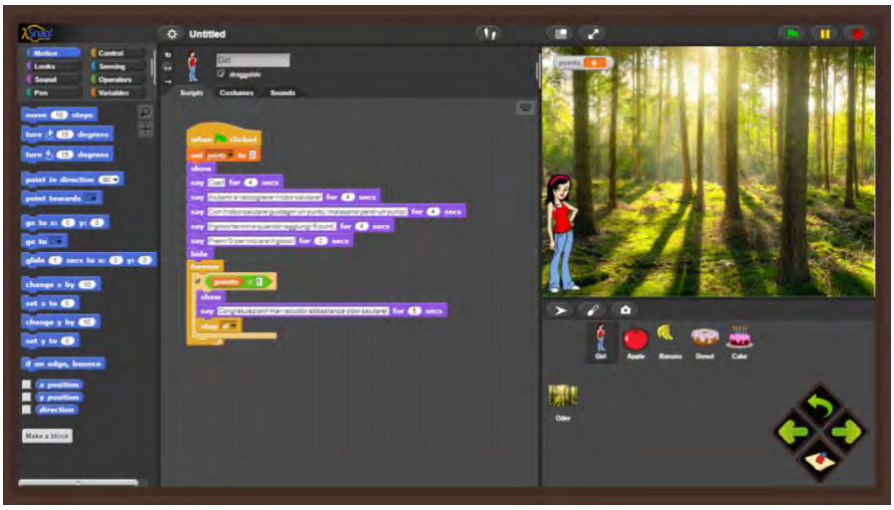
Slika 5.13 Primer strani z navodili v igralnem okolju za učence

Tej strani bo sledil kanvas s Snap!-om (Slika 5.14), na katerem je prikazana predloga, ki jo pripravi učitelj, in vsebuje programersko nalogo ali problem, ki ga morajo učenci izvesti ali rešiti.



Slika 5.14 Primer predloge, ki jo pripravi učitelj, za reševanje programerske naloge v Snap!-u v igralnem okolju za učence

Drugi kanvas s Snap!-om (Slika 5.15) bo prikazoval predlagano rešitev izziva. Učitelj se lahko odloči, ali bo rešitev pokazal ali ne.



Slika 5.15 Primer izgleda kanvasa s Snap!-om, v igralnem okolju za učence, ki prikazuje predlagano rešitev izziva, ki jo je za učence pripravil učitelj

Ker je vsak predmet sestavljeni iz več izzivov, se bodo ti koraki ponovili tolikokrat kolikor je izzivov, ki jih določi učitelj. To omogoča razdelitev ene kompleksne programerske naloge na več preprostejših korakov, kjer odgovor in/ali rešitev prejšnje naloge postane predloga za naslednjo aktivnost.

Na koncu je predmet sestavljen iz določenega števila izzivov, ki predstavljajo proces postopnega poučevanja programiranja.

Ko učenci rešijo vse izzive v predmetu (Slika 5.3), jih okolje vrne nazaj na začetni programerski problem in učenci dobijo nalogo, da povzamejo svoje novo pridobljeno znanje. Na koncu predmeta , lahko učenci vidijo tudi vse rešitve tega problema, ki so jih predlagali ostali učenci, kar omogoča primerjavo različnih rešitev.

C4G pristop in programsko okolje omogoča učencem razvoj, izboljšanje in poglabljanje programerskih veščin skozi personalizirane učne aktivnosti, ki združujejo zabavo in učenje.

6. PRIMERI UČNIH UR (UČNI SCENARIJI)

6.1. Učni scenariji

V okviru projekta Coding4Girls smo pripravili 22 učnih scenarijev. Opisujejo različne učne dejavnosti, s katerimi je mogoče uvajati pristop poučevanja z uporabo, snovanjem in izdelovanjem resnih iger. Učni scenariji so na voljo v angleškem jeziku, prav tako pa vseh nacionalnih jezikih partnerjev, ki so sodelovali pri projektu - bolgarščini, hrvaščini, grščini, italijanščini, portugalščini, slovenščini in turščini. Vsi so na voljo na spletni strani projekta. (https://www.coding4girls.eu/results_03.php)

Pripravljeni učni scenariji natančno povzemajo informacije, ki bodo učiteljem pomagale pri vključevanju resnih iger in metod učenja s snovanjem v njihovo učno prakso. Učni scenariji temeljijo na aktivnih oblikah učenja in uporabi izobraževalnih iger ter vključujejo informacije o tem kaj in kako morajo deklice in dečki izdelati pri vsaki učni aktivnost za to, da pridobijo veščine pri programiranju.

Na voljo so naslednje informacije:

- Koncepti, ki jih obravnava določena učna aktivnost
- Specifični učni cilji
- Pričakovani učni rezultati
- Opis (korak za korakom) uporabe CODING4GIRLS učnega pristopa z izdelovanjem iger
- Metode za ocenjevanje pridobljenega znanja
- Vprašanja za začetek razprave med učenci v okviru razrednega sodelovanja

Učitelji lahko uporabljajo scenarije in igre v predlaganem vrstnem redu ali pa jih prosto izberejo glede na njihove želje in potrebe. Prav tako morajo biti pozorni na prilagoditev nekaterih scenarijev glede na znanje učencev o konceptih, ki jih obravnavajo pri matematiki, kot npr. koordinatni sistem, koordinate, negativna števila, ulomki, itd..

Učni scenariji zajemajo generično funkcionalnost predlagane resne igre, vključno s procesi interakcije z uporabniki, ustvarjanjem povratnih informacij in opisom vseh učnih aktivnosti, ki se bodo izvajale ob predlagani resni igri.

Učni scenariji si sledijo po težavnosti, od osnovnih z enim konceptom do naprednejših z več koncepti programiranja. Naslednja tabela predstavlja predlagani vrstni red dejavnosti.

Tabela 6.1 Seznam učnih scenarijev, ki so nastali v okviru projekta Coding4Girls

OSNOVNI UČNI SCENARIJI		
1	**Uvod v okolje Snap! vmesnik** Spoznavanje vizualnega programskega okolja Snap!	UL
2	**Lik oživi** Iskanje programskih blokov, njihovo povezovanje, premikanje lika, lik spregovori	UL
3	**Premikanje po odru** Ustvarjanje smiselnega zaporedja blokov	UL
4	**Menjava obleke in obrat**	UL
5	**Zvoki na kmetiji** Dodajanje, vključevanje, snemanje in predvajanje zvokov	UL
6	**Kameleon na počitnicah** Spoznavanje dogodkov, prepoznavanje barv, logične vrednosti, preverjanje in reagiranje na različni stanji v igri	UL
7	**Pomagaj princu in princeski najti svoje živali** Uporaba pogojev, risanje	UL
8	**Risanje s kredo** Uporaba zank, obračanje, spreminjanje ozadja	UL
9	**Pobiranje smeti in čiščenje parka** Spoznavanje koncepta spremenljivk, podvajanje likov, bloki kodi	UL
10	**Nahrani mucke** Uporaba spremenljivk (zunaj/znotraj zanke), zanke, naključna števila, združevanje nizov, operatorji, vhod	UL
11	**Mačje zavetišče** Uporaba naključnih števil, shranjevanje uporabnikovega vnosa, pogoji, operatorji za primerjanje, števec.	UL

NAPREDNI UČNI SCENARIJI		
12	**Lovljenje zdrave hrane** Uporaba spremenljivk, pogoji, zanke, obračanje v smer, naključnost	UL
13	**Sestavi zgodbo**	SWU
14	**Onesnažen zrak**	UNIRI
15	**Ulovi miš** Uporaba zank, pogojev, spremenljivk	UL
16	**Kupovanje hrane za piknik** Uporaba spremenljivk, pogoji, operatorji	UL
17	**Računanje**	SWU
18	**Recikliranje**	SWU
19.1	**Zaigraj na klavir 1**	SWU
19.2	**Zaigraj na klavir 2**	UNIRI
20	**Test**	SWU
21	**Enostavni Pacman** Uporaba premikanja, ki temelji na dogodkih, zaznavanje barv, logične vrednosti, preverjanje in reagiranje na različni stanji v igri.	UL

Vseh 22 učnih scenarijev je predstavljenih v prilogi 1. V prilogi 2. so podane kode za vse igre, ki so predstavljene v javno dostopnem delu okolja Coding4Girls.

6.2. Primeri uporabe spletnega okolja z elementi igrifikacije, razvitega v okviru projekta Coding4Girls

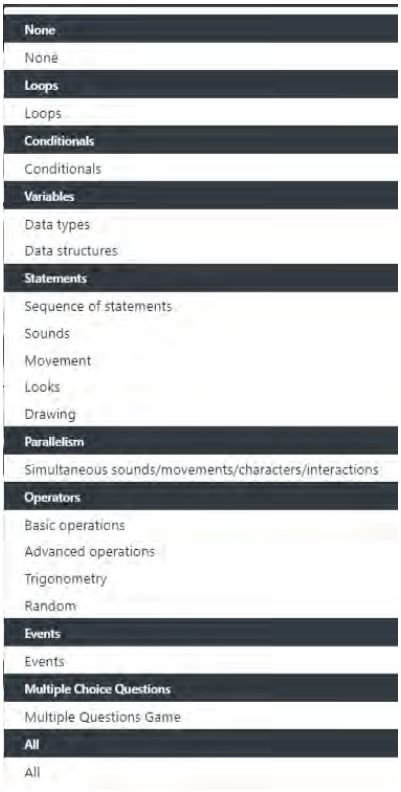
Platforma za učence je didaktična videoigra, narejena z okoljem Unity 3D, ki je na voljo za operacijske sisteme Windows, Mac OS X ter Linux. Najprej se prikaže uvodna scena, na kateri je zapisana izjava o odgovornosti, prikazana pa sta tudi logotipa projekta in Erasmus+.

Po uvodni sceni se igralec znajde v veliki sobi, imenovani *lobby*. Igro gleda skozi oči nevidnega lika, kar mu daje občutek, kot da se sam sprehaja po 3D igri. Sredi sobe se nahaja

terminal, preko katerega se učenec prijavi na C4G strežnik s svojim uporabniškim računom in geslom, lahko pa se tudi na novo registrira. Po uspešni prijavi se odprejo vrata v naslednjo sobo, kjer se lahko učenec pridruži novemu predmetu ali pa si izbere že ponujen predmet. Nato stopi skozi vijolični portal in igra se nadaljuje v predmetu, ki ga je učitelj ustvaril.

Predmet je sestavljen iz enega ali več izzivov v programskem okolju Snap!, ki jih lahko učitelj po želji poveže z mini igro. Mini igre dodatno ponazorijo vidike in korake, ki jih morajo učenci v izzivu usvojiti.

Mini igre ponazarjajo naslednje osnovne programerske koncepte: Zanke (Loops), Pogojne stavke (Conditionals), Spremenljivke (Variables), Izjave (Statements), Vzporednost (Parallelism), Operatorje (Operators) in Dogodke (Events). Poleg tega se lahko učitelj odloči, da bo mini igro zamenjal s kvizom, lahko pa mini igre ne izbere in se tako odloči le za izzive v okolju Snap!



Slika 6.1 Kategorije mini iger v platformi za učitelje

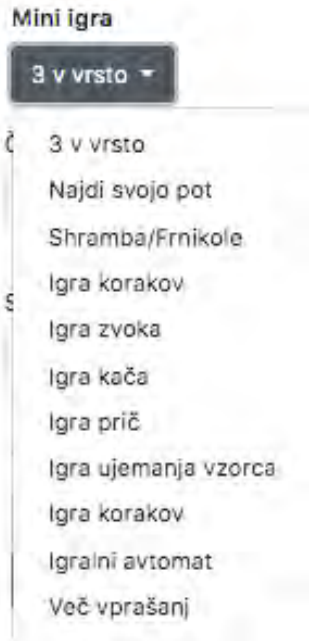
V vsakem izzivu čaka učence aktivnost v Snap!-u in pripadajoča mini igra, v kolikor se je učitelj odločil zanjo (Slika 6.2).

Slika 6.2 Prikaz izzivov v platformi za učence

Vključitev mini igre v posamezen izziv še dodatno ponazori programerski koncept, ki se skriva v Snap! aktivnosti. Če želi učitelj mini igro vključiti v izziv, to stori tako, da jo izbere s seznama obstoječih mini iger. Lahko pa se tudi odloči, da mini igre ne bo vključil v izziv.

Učitelj mini igro vključi v izziv v kolikor v njej prepozna dodano vrednost in predvideva, da bo igra nudila učencem pomoč pri razumevanju, vizualizaciji koncepta, mehaniki ter pri vključenosti in motivaciji učencev na njihovi učni poti.

Trenutno je na voljo 11 mini iger, od 3 v vrsto pa do Več vprašanj.



Slika 6.3 Seznam mini iger v platformi za učitelje

Na začetku mini igre se igralec znajde v okolju, kjer igra poteka. Lokacije mini iger so različne, vse od peščenih plaž pa do zasneženih gora.

Slika 6.4 Primer lokacije mini igre

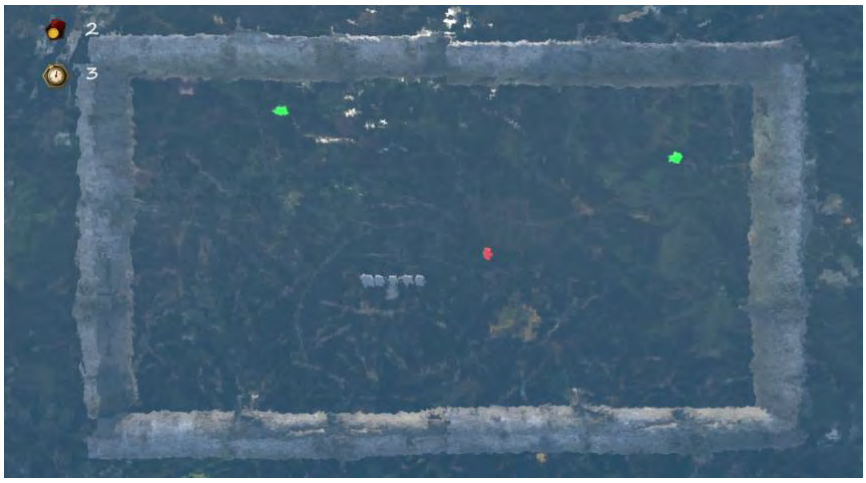
Med razpoložljivimi mini igrami so 3 v vrsto, Najdi svojo pot, Shramba/Frnikole, Igra korakov, Igra zvoka, Igra kača, Igra prič, Igra ujemanja vzorca, Igra korakov, Igralni avtomat in Več vprašanj.

Vsaka mini igra ima različno težavnostno stopnjo, obsega različne starostne skupine in pokriva različne programersk koncepte, ki se jih učenci učijo pri predmetih.

Ena izmed preprostejših iger je Igra kača, ki se odvija na reki. Igralec mora slediti leseni puščici, ki ga vodi do rdeče oznake. Nato se igra prične (Slika 6.5).

Slika 6.5 Primer okolja za Igro kača

Igra kača (Slika 6.6) se odvija v vodi. Kačo upravljamo s štirimi smernimi tipkami, cilj igre pa je pojesti čim več hrane. Kača se poveča tako, da poje zelen košček hrane, igre pa je konec, če se ugrizne v svoj rep, zaleti v rob ali poje rdeč košček hrane. Igralec dobi točke, ko kača poje zelen košček hrane.



Slika 6.6 Igra kača

Igra kače prikazuje različne programerske koncepte, kot so zanka, premikanje, spreminjanje grafike itd.

Igra 3 v vrsto poteka v zasneženenm gorskem okolju, po katerem se lahko igralci prosto sprehajajo in raziskujejo.

Slika 6.7 Okolje za igro 3 v vrsto

Igra 3 v vrsto predstavlja tipično igro, kjer moramo kvadratke preurediti tako, da so trije kvadratki enake barve v isti vrsti. Kvadratki nato izginejo, če so v vrsti trije ali več kvadratkov enake barve.

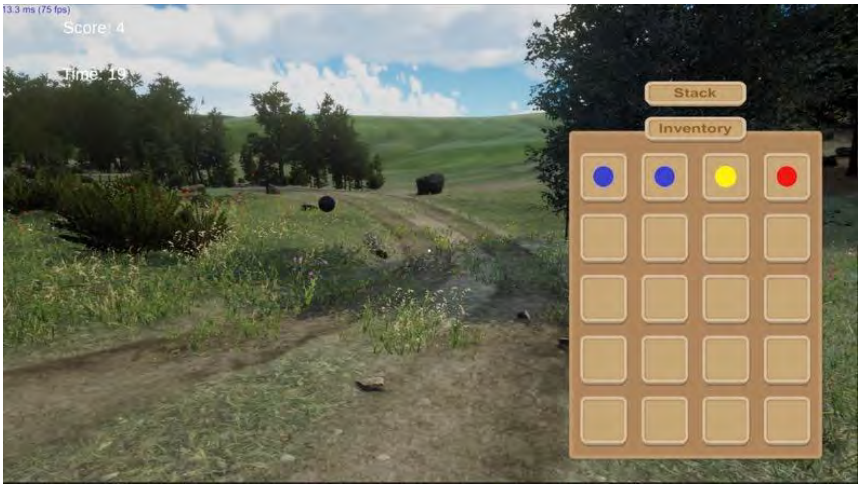
Slika 6.8 Igra 3 v vrsto

Igralni avtomat ali igra s kockami se odvija v temnem gozdu. Interaktivni del igre poteka v zapuščeni koči. Na starem stolu se nahaja lesena igralna plošča z dvema kockama.

Igralec in računalnik izmenično mečeta kocki. Zmagovalec je tisti, ki doseže večjo vsoto pik na kockah.

Slika 6.9 Igralni avtomat

Igra Shramba/Frnikole se odvija na polju blizu gozda. Na polju se nahajajo pisane krogle, ki jih igralec zbira glede na navodila, zapisana na lesenem znaku. Navodila poda učitelj preko platforme pri ustvarjanju predmeta.



Slika 6.10 Igra Shramba/Frnikole

Igra korakov se odvija na plaži ob sončnem zahodu. Igralec mora odgovoriti na vprašanje, zapisano na veliki skali na sredini plaže. Na vprašanje odgovori tako, da stopi na kamen, na katerem je zapisana prva črka pravilnega odgovora.

Na drugi skali se izpiše črka, na katero je igralec stopil. Če igralec stopi na pravilen kamen, se črka izpiše v zeleni barvi, če stopi na napačen pa v rdeči.

Slika 6.11 Igra korakov

Igra kviza Več vprašanj je postavljena na polju ob pečini. Vprašanje se pojavi na pečini zgoraj, vsako polje pa vsebuje možne odgovore ter tudi slike, v kolikor jih učitelj doda.

Ikona s kovancem prikazuje trenutni rezultat (glede na število pravilnih odgovorov), spodnja ikona z uro pa odšteva sekune do konca izziva.

S klikom na gumb s puščico spodaj se premaknemo na naslednje vprašanje.

Slika 6.12 Kviz Več vprašanj

Igra prič oz. ugank se odvija na skalnatem območju. V igri mora igralec najprej najti bele table, nato pa na njih povezati belo in rdečo piko.

Slika 6.13 Tabla v Igri prič

Igra ujemanja vzorca poteka na bregovih reke. Po reki potujejo škatle s števili. Igralec mora izbrati škatle, da lahko reši določeno matematično operacijo. S klikom na škatlo in ustrezno matematično operacijo mora priti do števila, ki je zapisano na veliki modri tabli.

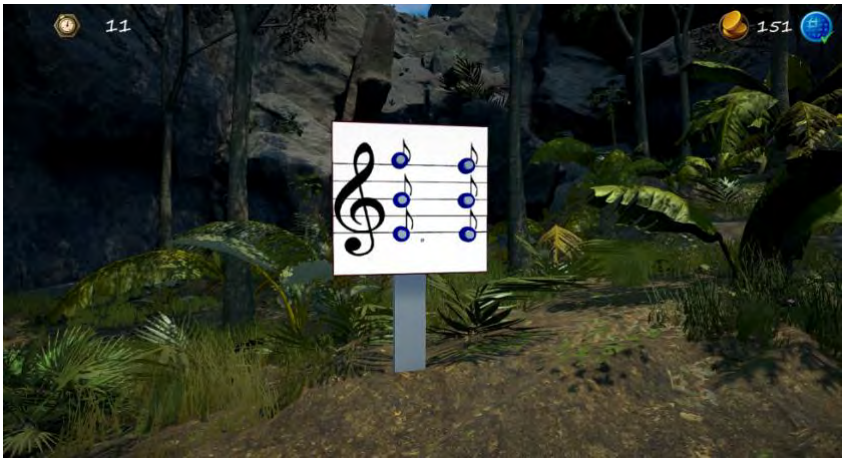
Številke na škatlah in na tabli se generirajo naključno, vedno pa obstaja kombinacija, s katero račun rešimo.

Učitelj v platformi izbere, katere operacije želi v mini igri. Izbira lahko med:

- Osnovnimi operacijami (seštevanje, odštevanje, množenje in deljenje),
- Naprednimi operacijami (potenciranje, kvadriranje, računanje po modulu),
- Trigomonetrijo (sinus, kosinus, tangens) in
- Boolean vrednostmi (IN, ALI, eksluzivni ALI).

Slika 6.14 Igra ujemanja vzorca

Naslednja je Igra zvoka, ki poteka v tropskem gozdu. Igralec mora najti pet tabel in nato rešiti uganko, pri čemer mora pozorno poslušati zvoke, ki jih sliši iz džungle.

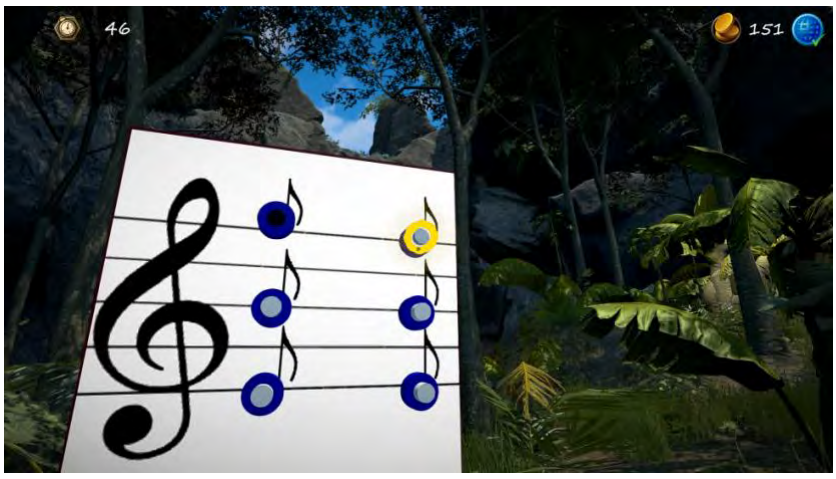


Slika 6.15 Tabla v igri Zvoka

Vsak stolpec na tabli predstavlja določen zvok, vrstica pa višino tona. Note oz. gumbi na vrhu table predstavljajo visoke, na dnu pa nizke tone. Ko se igralec približa tabli, zasliši oglašanje ptic, ki se ponavljajo v zanki, vsako ponovitev pa ločijo 3 sekunde tišine.

Igralec mora biti pozoren na vrstni red oglašanaj ptic in ga ustrezno označiti na tabli.

Spodaj je prikazan primer (Slika 6.16), ko igralec najprej sliši nizek, nato pa visok ton. V prvem stolpcu mora tako označiti nizek, v drugem pa visok ton.



Slika 6.16 Prikaz odgovorov učenca na table v igri Zvoka

Določen gumb izberemo tako, da kazalec miške usmerimo na noto, ki se nato obarva v rumeno. Če smo izbrali pravilno, se nota pobarva v zeleno, v nasprotnem primeru pa v črno.

Zvok je pogosto pomemben pri vsaki igri, pri tej pa je ključnega pomena, saj se morajo igralci osredotočiti na glasove in ločiti med visokimi, srednjimi in nizkimi toni.

Spletno okolje je namenjeno vsem učencem, predvsem pa učencem med 10. in 16. letom starosti. Namen okolja je spodbuditi enakopravnost pri izzivih s področja programiranja tako v razredu kot tudi izven njega. Okolje prikazuje programerske koncepte skozi 3-dimenzionalno virtualno okolje, ki ga lahko učenci raziskujejo skozi mini igre, sodelovalno okolje za izmenjavo idej s sošolci, skupno iskanje potencialne rešitve ter vpogled v rešitve sošolcev in učitelja, s čimer pridobijo drugačen pogled na dani problem in možno rešitev.

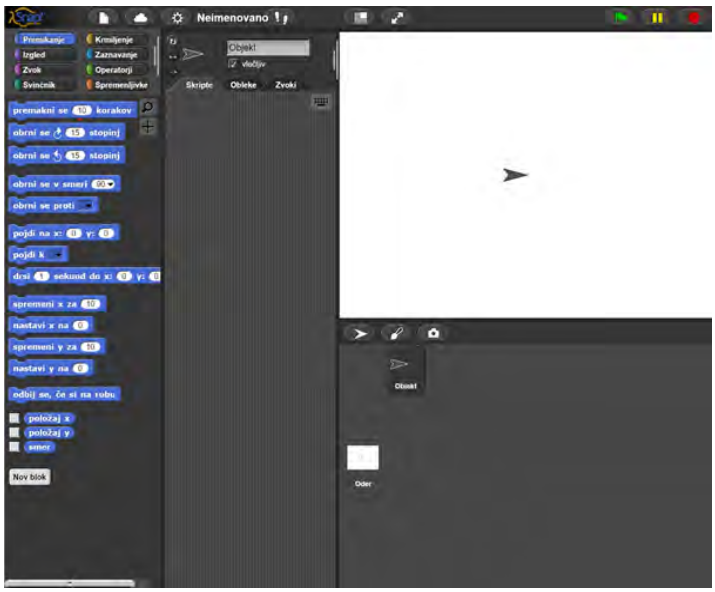
Spletno okolje dopušča številne razširitve in izboljšave, kot je npr. dodajanje novih mini iger, ki bi pomagale ponazoriti še druge programerske koncepte.

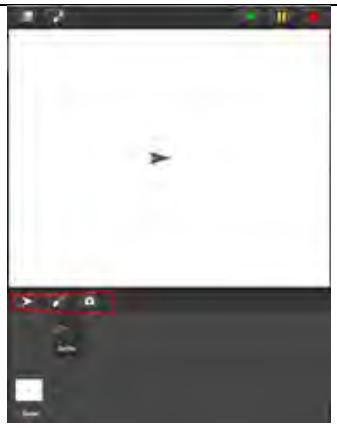
PRILOGA 1. UČNI SCENARIJI

Osnovni učni scenariji

Učni scenarij 1 – Uvod v okolje Snap!

Naslov učnega scenarija	Uvod v okolje Snap!
Pričakovano programersko predznanje	/
Učni cilji	Splošni učni cilji: ● seznanitev z vizualnim programskim okoljem Snap! Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenec zna dodati nov lik ● Učenec zna liku dodati novo obleko in jo urediti ● Učenec zna nastaviti središče vrtenja lika tako, da zgleda njegovo vrtenje čim bolj naravno ● Učenec zna na oder dodati novo ozadje in ga urediti
Cilji, naloge in kratek opis aktivnosti	Učenec doda nov lik, ki mu doda obleko. Obleko uredi in izbriše enega od njih. Učenec ustvari novo ozadje in izbriše neželjenjega **Cilji: Ob koncu učne ure bo učenec narisal svoj izbrani lik in okolje v katerem živi, resnično ali izmišljeno, da bi ga v nadaljevanju lahko uporabil v igri. Študije so pokazale, da je na ciljno skupino ustvarjanje lastnih likov deluje motivacijsko.**
Trajanje aktivnosti	45 minut
Učne strategije in metode	Demonstracija Individualno delo
Učne oblike	Frontalno delo Individualno delo

Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Ob koncu učne ure bodo učenci narisali svoj najljubši lik in okolje, v katerem živi, resnično ali izmišljeno, da bi ga kasneje lahko uporabili v igri. [1. korak] Pokažite učencem, kje lahko najdejo Snap! (https://snap.berkeley.edu/). Pokažite jim različne dele uporabniškega vmesnika: predel, kjer najdejo različne delčke kode, predel, kjer sestavljajo svoj program,/ spreminjajo obleke/dodajo zvok, oder, na katerem je lik, seznam likov.  [2. korak] Nov lik lahko ustvariš s klikom na enega od treh gumbov:



Ker boste narisali svoj lik, kliknete na čopič in odprl se bo urejevalnik slik, ki je podoben Risarju.

Naloga za učence: Narišite svoj prvi lik. Na voljo imate 10 minut.

Ko narišejo lik, morajo preveriti, da je središče vrtenja njihovega lika na pravem mestu. Za to naj uporabijo orodje  .

Naloga za učence: določite točko, okoli katere se bo vrtel vaš lik.

[3. korak]

Za urejanje lika izberite zavihek obleke, ki je viden le, če je trenutno izbran lik. Po desnem kliku na obleko, ki jo želite urediti, se odpre spustni seznam, v katerem izberete uredi. Obleko lahko v istem seznamu tudi podvojite ali izbrišete.

[4. korak]

Za uvoz že predpripravljene obleke, kliknite na ikono, ki izgleda kot list papirja, in izberite Obleke…

Tudi tokrat velja, da se ta možnost pojavi le, če je trenutno izbran lik.

Naloga za učence: izberite si eno obleko in jo dodajte svojemu liku.

[5. korak]

Zdaj ko imate svoje like, jim boste dodali še ozadje. Najprej pod odrom namesto na lik, kliknite na Oder. Da bi dodali novo ozadje, izberite zavihek Backgrounds:

Naloga za učence: nariši svoje ozadje.

	Naloga za učence: dodaj še eno od vnaprej pripravljenih ozadij, da boš imel dve. Najdeš jih na podoben način kot prej obleke za lik Naloga za učence: poišči način, kako lahko urediš ozadje in kako ga izbrišeš. (Refleksija in evalvacija) Ali so učenci uspeli narisati svoj lik in ozadje? So imeli kakšne težave? Kako so jih reševali?
Učni pripomočki, sredstva za učitelja	https://snap.berkeley.edu/
Učni pripomočki za učenca	Navodila za učenca (C4G1_NavodilaZaUcenca.docx)

Učni scenarij 2 – Lik oživi

Naslov učnega scenarija	Lik oživi
Pričakovano programersko predznanje	/
Učni cilji	Splošni učni cilji: ● Učenec ve, kje v programskem vmesniku lahko najde delčke in kako jih poveže v zaporedje ● Učenec zna sestaviti zaporedje delčkov za premikanje lika ● Učenec zna uporabiti delček reci Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Sestaviti ustrezno zaporedje delčkov
Cilji, naloge in kratek opis aktivnosti	**Učenci izvejo, kje so shranjeni programski delčki in kako najdejo ustreznega, katere kategorije delčkov poznamo in kako jih povežemo v smiselno zaporedje**
Trajanje aktivnosti	45 minut
Učne strategije in metode	Demonstracija Individualno delo
Učne oblike	Frontalno delo Individualno delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) To uro boste spoznali, kako z bloki likom naročimo naj se premikajo po odru in govorijo. Učencem lahko pokažete primer programa, ki ga bodo znali ob koncu ure narediti. [1. korak] Najprej na zaslonu poiščite, kje najdete delčke kode, ki jih lahko uporabite za sestavljanje svojega programa. Kje se nahajajo?

Na levi strani, nad delčki, lahko vidite, da imate različne kategorije delčkov:

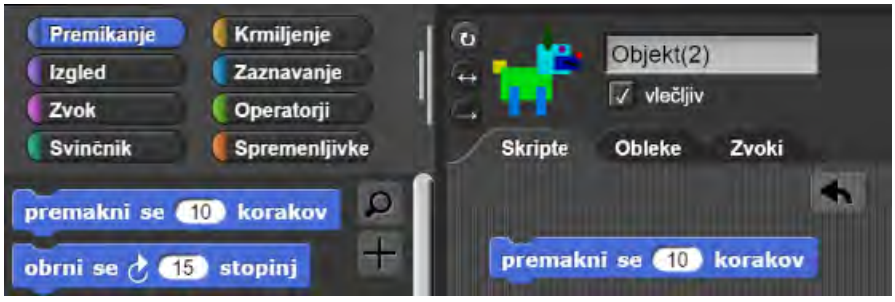
premikanje, izgled, zvok, svinčnik, krmiljenje, zaznavanje, operatorji in spremenljivke.

Ti delčki so obarvani z različnimi barvami in s tem povezani v skupine, na primer vsi delčki za premikanje so modre barve.

Naloga za učence: Najprej poiščite delček za premikanje in dvakrat kliknite nanj. Kaj se je zgodilo?

[2. korak]

Da se bo lik premikal naprej, moramo najprej povleči in izpustiti delček premakni se 10 korakov z levega dela okna na del, ki je namenjen sestavljanju skript, to je delčkov programa:



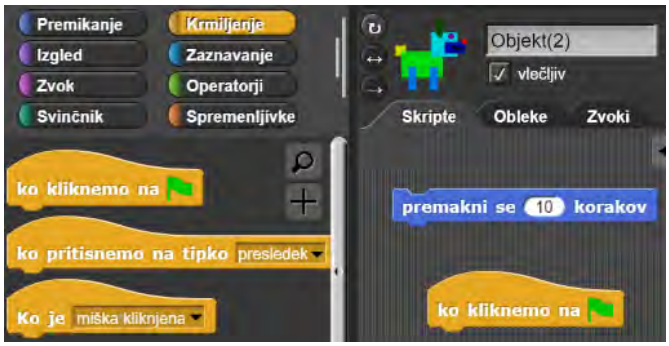
Če sedaj dvakrat kliknete na delček premakni se 10 korakov, se bo vaš lik premaknil za 10 korakov.

[3. korak]

Program v Snap!-u se običajno začnejo s klikom na zeleno zastavico.

Naloga za učence: poglejte v različne kategorije in poiščite delček, s katerim se bo program začel izvajati, ko kliknemo na zeleno zastavico. Povlecite ga med skripte.

Rešitev:



Če želite, da bo progam ukaze izvajal v pravilnem vrstnem redu, morate delčke med seboj povezati kot pri sestavljenki. Tako:

Zdaj se bo vaš lik vsakič, ko pritisnete na zeleno zastavico premaknil za 10 korakov, vedno s točke, na kateri se v tistem trenutku nahaja.

[4. korak]

Če je na delčku bel prostor, lahko spremeniš številke ali črke napisane na njem.

Naloga za učence: Poskrbite, da se bo lik premaknil za 30 korakov naenkrat, in ne 10 kot se je doslej.

[5. korak]

Poskrbi, da bo tvoj lik nekaj rekel. Kje boste našli delček reci? Poskusite v čem je razlika med delčkom 

in 

. O razliki se pogovori s sošolcem.

[6. korak]

	Oba delčka reci ste našli v kategoriji Izgled. Glavna razlika med njima je, da pri delčku  programu ne poveš, kako dolgo naj počaka preden izvede naslednji ukaz ali da mora kadarkoli prekiniti z izvajanjem tega ukaza. [7. korak] Naloga za učence: Odprite program, ki ste ga ustvarili prejšnjo uro. Svoj lik povlecite na levi del odra in sestavite program, ki poskrbi, da se vaš lik premika z izbrane točke na levi proti desni strani odra. Po vsakem premiku naj nekaj pove. Narediti more več kot le en premik. Program večkrat poženite. Ali lik po vsakič vedno konča na isti točki? Poiščite delček, ki poskrbi, da se bo lik vedno najprej vrnil na začetno točko in ne bo pobegnil z zaslona. Namig za učitelja. Če lik pobegne z zaslona, ga pokličite nazaj tako, da ga poiščete med liki in z desnim miškinim gumbom kliknete nanj. V seznamu izberite prikaži. Delček, ki ste ga iskali je  . Za določitev pravega x in y, lahko najprej premaknete svoj lik na željeno točko in pod delčki v kategoriji Premikanje odkljukate okence pri položaj x in položaj y, da se vam izpiše trenutni položaj lika. Ti dve vrednosti vpišeš v prostorčka za x in y. (Refleksija in vrednotenje) Kolikokrat je moral vaš lik ponoviti zaporedje premikanja in govorjenja, da je opravil nalogo? Ste vsi v razredu naredili enako število ponovitev? Zakaj?

Učni pripomočki, sredstva za učitelja	Primer programa: https://snap.berkeley.edu/project?user=mateja&project=C4G2_KuzaGreDomov
Učni pripomočki za učenca	Učenci, ki prejšnjo uro niso izdelali svoje lika in ozadja, lahko uporabijo: ● Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G2_KuzaGreDomov_tmp Navodila za učence, ki delajo samostojno: ● Navodila za učenca (C4G2_NavodilaZaUcenca.docx) – navodila za učence so namenjena učencem, ki delajo samostojno.

Učni scenarij 3 – Premikanje po odru

Naslov učnega scenarija	Premikanje po odru
Pričakovano programersko predznanje	/
Učni cilji	Splošni učni cilji: ● Sestaviti smiselno zaporedje delčkov Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenec ve, kako postaviti lik na določeno mesto na odru ● Učenec zna premakniti li v smeri x in smeri y ● Učenec zna uporabiti zanko ponovi __krat ● Učenec ve, da je smer gibanja lika pri delčku pojdi __ korakov relativna glede na smer, v katero je lik obrnjen
Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Učenec spozna, kako premikati lik po odru v x in y smeri. Učenec sestavi preprost program za rešitev naloge. Nauči se obračanja lika v različne smeri in kako to vpliva na delček pojdi __ korakov.

	Naloge: ustvari program, ki bo premaknil lik v smeri x, ustvari program, ki bo premaknil lik v smeri y, ustvari program, ki bo združil gibanje v x in y smeri. **Cilji:** razlikovanje med premiknjem po odru v x in v y smeri ter uporaba zanke ponovi __ krat
Trajanje aktivnosti	45 minut
Učne strategije in metode	Demonstracija Individualno delo
Učne oblike	Frontalno delo Individualno delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Različnim živalim boste pomagali, da dosežejo svoj cilj. Za to jim boste mroali podati jasna navodila o tem, kako se morajo premikati po odru. [1. naloga] Odprite nalogo Ujemi žogo (https://snap.berkeley.edu/snap/snap.html#present:Username=mateja&ProjectName=C4G3_Ujemi_zogo). Naredite animacijo premikanja psa z levega dela odra proti žogi. Pri tem uporabite delčka  in  . Možna rešitev naloge:

Kot vidite se pri premikanju levo in desno po odru spreminja vrednost x. Ko je x 0, je vaš lik na sredini odra. Vse kar se nahaja levo od njega mora imeti pred številko oznako -, dlje stran kot je od sredine, večja je številka poleg znaka -. Desno od sredine so vrednost x večje od 0 in pred njih ne pišemo nobenega predznaka.

Namig: Če učenci poznajo decimalna števila, lahko čas čakanja zmanjšamo na 0.1 sekunde. Če učenci že poznajo negativna števila, lahko to uporabimo pri razlagi negativnih vrednosti x. Če učenci že poznajo koordinatni sistem, razlago vrednosti x navežemo na to temo.

[2. naloga]

Odprite nalogo Plezanje opice () in sestavite naredite animacijo opice, ki pleza na palmo, da bi dosegla banane. Pri tem uporabite

delčka 

in 

. Pazite, da bo opica vedno začela s plezanjem z istega mesta.

Dodatna naloga: Ko opica pride do banan, naj spleza nazaj na začetno mesto.

Možna rešitev osnovne naloge:

Kot lahko vidite, se vrednost y spreminja, ko se lik premika gor in dol po odru. Če je vrednost y 0, je lik na sredini odra. Vse kar je višje od sredine odra, ima vrednost y večjo od 0. Če se vaš lik nahaja nižje od sredine odra, je to podobno kot pri potapljanju: da si pod vodno gladino poveš s tem, da pred številko napišeš -, s številko pa poveš, koliko »metrov« pod vodno gladino si. V našem primeru pa ti številka za znakom - pove, koliko korakov pod sredino odra je lik. Če želiš, da se opica s palme vrne nazaj na začetno mesto, uporabite 

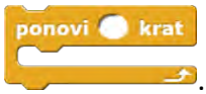
.

Namig: Učencih, ki že poznajo decimalna števila, lahko za boljšo animacijo pri čakanju uporabijo decimalno število (na primer 0.1). Če učenci že poznajo koordinatni sistem, razlago premikanja v y smeri navežite na to snov.

[3. naloga]

V obeh nalogah ste izmenično uporabljali dva enaka delčka. Kolikokrat ste morali **ponoviti** ta dva delčka z enakimi podatki?

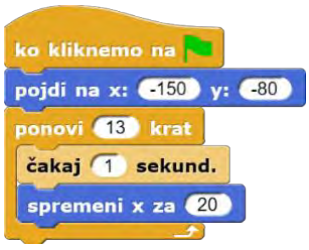
Obstaja krajši način za pisanje kode, s katerim računalniku poveš, kolikokrat naj ponovi enake ukaze. Delček, s katerim računalniku povemo, katere kaze naj ponovi in kolikokrat naj to stori, je zanka



.

Uporabite jo lahko, kadar želite enake ukaze večkrat zapored ponoviti v istem vrstnem redu. Poskusite svoja programa, ki ste ju napisali danes, spremeniti tako, da boste uporabili zanko ponovi __ krat. Ukazi, za katere želite, da se ponovijo, morate pripeti znotraj delčka in napisati, kolikokrat naj se to zaporedje ukazov ponovi.

Koda za psa:



Koda za opico:

*Naloga: Dopolni program Ujemi žogo teko, da se bo pes, ko pride do žoge obrnil in vrnil nazaj na začetno mesto in zalajal.

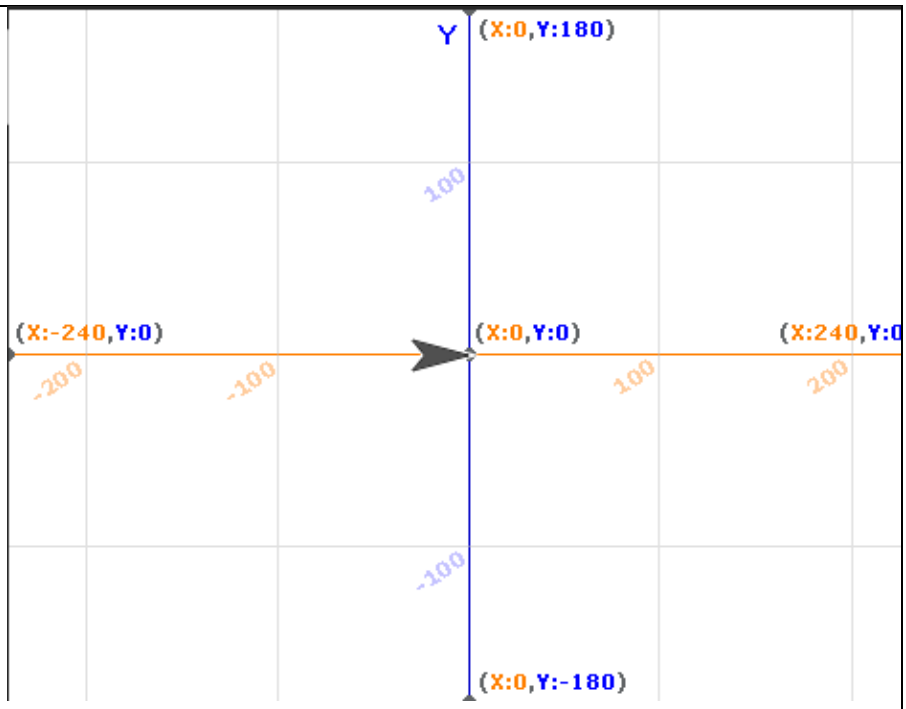
[Refleksija]

Kaj vam je bilo najbolj všeč? Če imate težave pri ugotavljanju, ali morate lik premakniti v x ali v y smer, si lahko pomagate z ozadjem XY Grid.

Spodnja slika prikazuje *oder*.

Za premikanje *levo/desno* se premikamo v smeri *x* (desno v +, levo v -),

za premikanje *gor/dol* pa se premikamo v smeri *y* (gor v +, dol v -).

	
Učni pripomočki, sredstva za učitelja	• Rešitev naloge Ujemi žogo: https://snap.berkeley.edu/project?user=mateja&project=C4G3_Ujemi_zogo_resitev • Rešitev naloge Plezanje opice: https://snap.berkeley.edu/project?user=mateja&project=C4G3_Plezanje_opice_resitev
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: Ujemi žogo https://snap.berkeley.edu/snap/snap.html#present:Username=mateja&ProjectName=C4G3_Ujemi_zogo • Predloga aktivnosti v Snap!-u: Plezanje opice https://snap.berkeley.edu/snap/snap.html#present:Username=mateja&ProjectName=C4G3_Plezanje_opice • Navodila za učenca (C4G3_NavodilaZaUcenca.docx)

Učni scenarij 4 – Menjava obleke in obrat

Naslov učnega scenarija	Menjava obleke in obrat
Pričakovano programersko predznanje	Premikanje po odru z ukazi pojdi _ korakov, spremeni x za _ in spremeni y za _ Poznavanje zanke ponovi __ krat

Učni cilji	Splošni učni cilji: ● Sestavljanje smiselnega zaporedja delčkov Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenci znajo liku zamenjati obleko in tako narediti animacijo ● Učenci znajo nastaviti različne načine obračanja lika in lik obrniti
Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Učenci spoznajo, kako naredijo animacijo lika s spreminjanjem njegove obleke. Naučijo se, kako določiti pravilen način obračanja lika glede na njegove lastnosti. **Naloge:** Ustvari programe, v katerih lik menjuje obleko. V vsakem programu nastavi ustrezen način obračanja lika. **Cilji:** znati liku zamenjati obleko in nastaviti ustrezen način obračanja lika.
Trajanje aktivnosti	45 minut
Učne strategije in metode	Demonstracija Individualno delo
Učne oblike	Frontalno delo Individualno delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) V tej uri se boste naučili, kako narediti tako animacijo lika, da bo izgledalo kot da lik pleše, hodi… [1. naloga] Odprite prazen projekt in kliknite na ikono, ki izgleda kot bel list papirja v levem zgornjem kotu in kliknite na Obleke. Izberite obleko ballerina a in kliknite na Import. Enako naredite še z oblekami ballerina b, ballerina c in ballerina d. Nato kliknite Prekliči.

V zavihku Obleke imate sedaj 4 obleke svojega lika. Ime lika lahko spremenite v Balerina. To naredite tako, da dvakrat kliknete na besedilo Objekt in vpišete poljubno ime lika.

Vrnite se na zavihek Skripte in napišite kodo, v kateri bo lik začel z obema nogama na tleh, nato pa bo plesal tako, da bo 15x spremenil svojo obleko. Pri tem boš uporabil delčka 

. Na koncu naj lik zamenja obleko tako, da bo svoj ples končala z obema nogama na tleh. Poskrbi, da bodo vidni vsi gibi v plesu in ne pozabi na začetek programa dodati delčka, ki programu sporoči, kdaj naj se začne izvajati!

Primer rešitve:

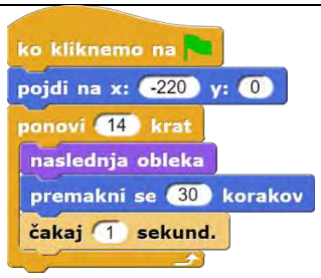
Naša balerina noče biti ves čas na istem mestu, zato se vsakič, ko zamenja obleko, še malo premakne. Dopolnite svojo kodo in si shranite program.

Možna rešitev:

[2. naloga]

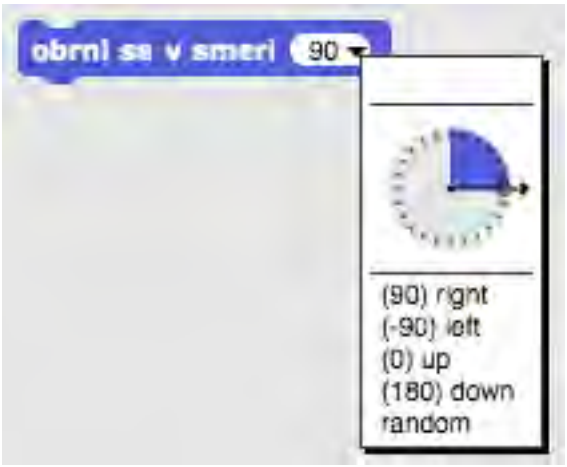
Odprite nov prazen projekt in uvozite vse obleke za lik avery walking. Dodajte ozadje, po katerem se bo Avery lahko sprehajala. Ustvarite animacijo, v kateri se Avery sprehaja od leve proti desni strani odra. Poskusite ugotoviti, kako Avery animirati tako, da bo izgledalo kot da naslednji korak kot v realnem življenju nadaljuje tam, kjer je prejšnjega končala. Ne pozabite shraniti svojega programa.

Možna rešitev:



[3. naloga]

Doslej smo sestavljali le programe, pri katerih se lik ni obračal. V tej nalogi pa imamo miško, ki jo moramo po cikcakasti poti pripeljati do sira, zato jo bomo morali na poti tudi obrniti v ustrezno smer. Za to lahko uporabimo delček obrni se v smeri _, kjer miši povemo, v katero smer naj gleda:



right – desno

left – levo

up – gor

down – dol

random – naključno

Druga možnost pa je, da miši povemo, za kakšen kot naj se obrne. To lahko naredimo z dvema ukazoma: obrni se v smeri urinega kazalca 

ali obrni se v nasprotni smeri od urinega kazalca 

. Cel krog meri 360°. Če želite miš obrniti v nasprotno smer, jo obrnete za 180°; če jo želite obrniti desno, jo obrnite za 90°v smeri urinega kazalca; če jo želite obrniti levo, jo obrnite za 90°v nasprotni smeri od urinega kazalca.

[Naloga] Odprite predlogo (https://snap.berkeley.edu/snap/snap.html#present:Username=mateja&ProjectName=C4G4_Najdi_sir) in napišite program, s pomočjo katerega se bo miš sprehodila do sira. Pri tem se lahko premika le po zeleni poti. Usmerite miš v pravo smer in za premikanje uporabite delček pojdi _ korakov. Da boste videli, kako se mi š počasi premika, uporabite ukaz čakaj 1 sekund.

Možna rešitev:

Poskusite napisati isti program še tako, da uporabite ukaza obrni se _ stopinj.

Možna rešitev:

[4. naloga]

Kot ste lahko opazili, se miš med premikanjem obrača v različne smeri. Včasih ne želimo, da se lik obrača v vse smeri – na primer ne želimo, da bi Avery hodila po glavi, ampak želimo, da se obrača le levo in desno. Zato da se bo vaš lik obračal na tak način kot si želite, moramo določiti ustrezen način obračanja. To naredite teko, da levo od lika izberete eno od naslednjih možnosti:

a. Krožna puščica pomeni prosto vrtenje in lik se lahko obrača v vse smeri (kot tvoja miš)
b. Puščica levo-desno pomeni, da se lahko lik obrača le levo ali desno (to lahko uporabiš pri psu, da ne hodi »po glavi«)
c. Puščica desno pa pomeni, da se lik ne vrti in bo vedno gledal v isto smer (to lahko uporabiš pri opici).

V programih, ki ste jih sestavili prejšnjo uro (s psom in opico) nastavite način obračanja lika. V programu s psom, nastavite premikanje tako, da se bo pes lahko sprehodil do žoge, se obrnil za 180° in se vrnil na izhodišče, ne da bi se obrnil na glavo. Podobno naredite še z Avery.

[Refleksija in preverjanje znanja]

Pri katerih od programov, ki ste jih sestavili do sedaj je smiselno uporabiti prosto vrtenje lika, pri katerih obračanje levo-desno in pri katerih ne vrti?

Se vam zdi lažje obračanje likov v določeno smer ali za določeno število stopinj? Ali je vedno vseeno, kateri način obračanja uporabimo?

Učni pripomočki, sredstva za učitelja	Primer celotne aktivnosti v Snap!-u: • Balerina: https://snap.berkeley.edu/project?user=mateja&project=C4G4_Balerina • Avery: https://snap.berkeley.edu/project?user=mateja&project=C4G4_Avery • Najdi sir: https://snap.berkeley.edu/project?user=mateja&project=C4G4_Najdi%20sir%20resitev
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u za Najdi sir: https://snap.berkeley.edu/snap/snap.html#present:Username=mateja&ProjectName=C4G4_Najdi_sir • Navodila za učenca (C4G4_NavodilaZaUcenca.docx)

Učni scenarij 5 – Zvoki na kmetiji

Naslov učnega scenarija	Zvoki na kmetiji
Pričakovano programersko predznanje	• Dodajanje ozadja; • dodajanje novega lika; • govorjenje lika.
Učni cilji	Splošni učni cilji: • dodajanje zvoka iz knjižnice orodja Snap!; • dodajanje zvoka iz računalnika; • snemanje zvoka; • predvajanje zvoka ob pritisku na tipko. Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • učenec doda zvok iz knjižnice orodja Snap! in ga doda v kodo tako, da se začne predvajati ob pristiku na določeno tipko; • učenec uvozi zvok iz računalnika in ga doda v kodo tako, da se začne predvajati ob pristisku na določeno tipko; • učenec posnema zvok in ga doda v kodo tako, da se začne predvajati ob pritistku na določeno tipko.

Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Učenci sestavijo enostavno igro, v kateri igralec s pritiskom na določeno tipko spozna oglašanje živali na kmetiji. **Naloga:** Sprva morajo učenci izbrati ozadje igre. Nato dodajo gospodarico kmetije, ki pove navodila igre: »Če želiš slišati psa, klikni na tipko P!«. Podobna navodila pove tudi za ostale živali na kmetiji. Učenec po navodilih gospodarice sestavi kodo igre. **Cilj:** Učenec spozna kako dodati in uporabiti zvok v orodju Snap!. Pri tem se bo naučil tudi kako uporabiti zvočni blok (*predvajaj_zvok_[zvok]*) in kontrolni blok (*ko_pritisnemo_na_tipko_[tipka]*).
Trajanje aktivnosti	45 minut
Učne strategije in metode	Aktivni pouk, učenje z ustvarjanje iger
Učne oblike	Frontalna oblika Individualno delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) **Motivacija-uvod** Za uvodno motivacijo učenci igrajo primer že ustvarjene igre Zvoki na kmetiji, pri čemer ne vidijo kodo igre. Učenci vidijo cilj učne ure in so motivirani za ustvarjanje svoje igre. 

[1. korak]

Učenci morajo najprej izbrat ozadje svoje igre. Ozadje mora vsebovati različne živali na kmetiji. Kot učitelji lahko izbiramo med naslednjimi možnostmi:

1. učencem ponudimo možnost, da sami narišejo ozadje (v tem primeru je priporočljivo, da risanje ozadja časovno omejimoa);
2. učenci na spletu poiščejo sliko kmetije, ki je prosto dostopna in ima dovoljenje za uporabo ali pa jim ponudimo spletne strani, ki vsebujejo omenjene slike;
3. učitelj učencem pripravi predlogo programa, ki že vsebuje ozadje (v tem primeru imajo vsi učenci enako ozadje).

Učenci so dodajanje ozadja spozanli v predhodnih urah, zato to naredijo individualno.

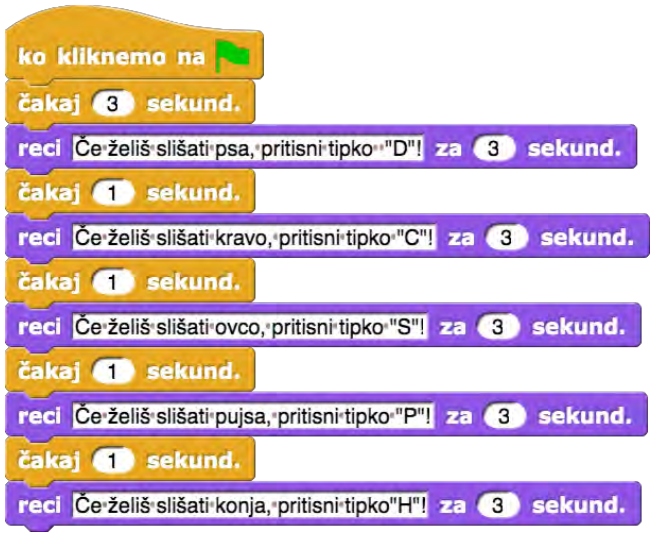
[2. korak]

V naslednjem koraku učenci dodajo gospodarico kmetije. Tudi v tem primeru se učitelj odloči za eno izmed možnosti:

1. učenci sami narišejo gospodarico;
2. učenci uvozijo prosto dostopno sliko;
3. učitelji pripravi predlogo programa, ki vsebuje gospodarico kmetije.

[3. korak]

Ko je dodana gospodarica kmetije, ji lahko dodamo kodo tako, da bo na začetku igre povedala navodila. Učenci pri tem uporabijo bloke kot so *Izgled/reci_[stavek]_za_[x] sekund* in *počakaj_[x]_sekund*. Učenci so te bloke spoznali v predhodnih učnih urah, zato lahko ta korak opravijo individualno.



Implementacija

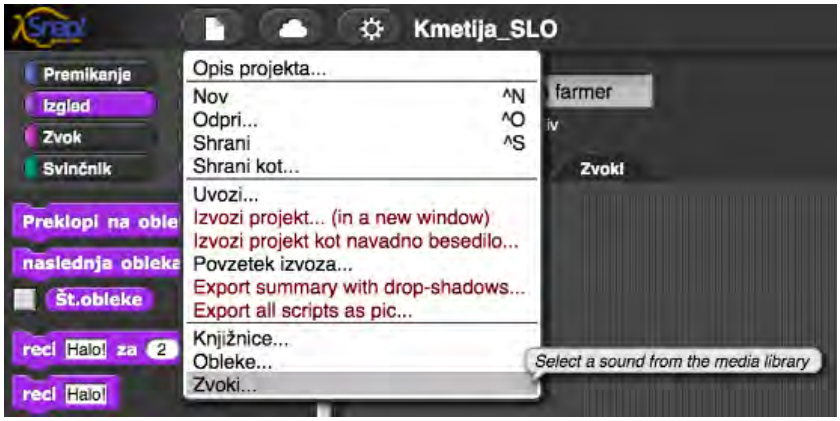
V prvih 3. korakih so učenci ponovili snov iz prejšnjih učnih ur. V nadaljevanju pa bodo spoznali kako v igro dodajati zvok. Za začetek jim učitelj frontalno pokaže tri načine dodajanja zvoka:

1. dodajanje zvoka iz knjižnice orodja Snap!;
2. uvoz zvoka iz računalnika na način polveči in spusti;
3. snemanje zvoka v orodju Snap!.

Vsi zvoki se uvozijo v zavihek *Zvoki*. Ko se učenci spoznajo z načini dodajanja zvoka, nadaljujejo z ustvarjanjem igre individualno s podporo učitelja.

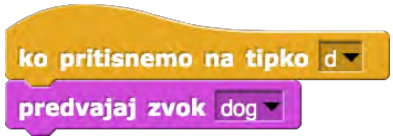
[4. korak]

Učenci najprej dodajo zvok psa iz knjižnce orodja Snap!. Knjižnico zvokov najdemo s klikom na ikono belega lista, kjer se na koncu spustnega seznama ponudi možnost *Zvoki*.



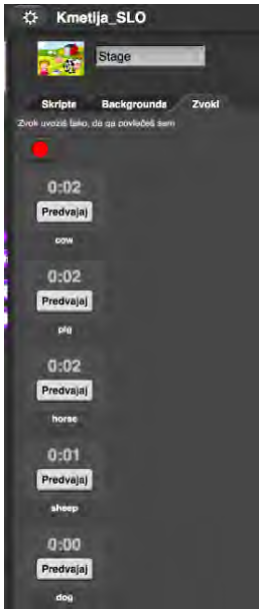
Odpre se knjižnica zvokov, v kateri učenci poiščejo zvok psa, ga označijo in za uvoz v program kliknejo na gumb *Import*.

Ko je zvok psa dodan pod zavihek *Zvoki*, ga je potrebno vključiti v kodo igre. Učenci morajo dodati kodo tako, da bo pes zalajal, ko bo igralec pritisnil na tipko »D«. Pri tem si pomagajo z blokoma *ko_pritisnemo_na_tipko_[tipka]* in *predvajaj_zvok_[zvok].*



[5. korak]

V naslednjem koraku učenci uvozijo zvoke še od preostalih živali. Ko imajo zvoke shranjene na svojem računalniku, jih enostavno primejo, povlečejo in spustijo v zavihek *Zvoki*.



Ko so vsi zvoki uvoženi, jih učenci dodajo v kodo pod zavihkom *Skripte*. Pri tem uporabijo bloka *ko_pritisnemo_na_tipko_[tipka]* in *predvajaj_zvok_[zvok].*

[6. korak]

Sedaj pa učenci dodajo še posneman zvok. Učenci posnemajo pozdrav gospodarice: »Dobrodošli na moji kmetiji!« in ga dodajo na začetek igre. V zavihku *Zvoki* najdejo rdeči gumb. S klikom nanj se odpre snemalnik zvoka. Za snemanje kliknejo na prvi gumb s krogcem, s klikom na drugi gumb s kvadratkom vstavijo snemanje, za predvajanje

posnemanega zvoka pa kliknejo na tretji gumb s trikotnikom. Ko so učenci zadovoljni s posnemanem zvokom, ga shranijo s klikom na gumb *Shrani*.

Ko je pozdrav gospodarice posneman, ga učenci dodajo v kodo gospodarice pod zavihkom *Skripte*. Pri tem uporabijo blok *predvajaj_zvok_[zvok].*

[Dodatne naloge]

Učenci lahko igri dodajo poljubne elemente, like (npr. kmeta, traktor, petelina ipd.) in jim doda zvoke.

Refleksija in vrednotenje

Učenci ponovijo in utrdijo pridobljeni znanje:

- kako so dodali zvok v kodo;
- katere bloke so uporabili pri dodajanju zvokov v kodo;
- katere kontrolne bloke so uporabili;

[Končna koda]

Gospodarica kmetije

Ozadje

Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Kmetija_SLO • Spletna stran s prostodostopnimi slikami: https://pixabay.com/ • Spletna stran s prostodostopnimi zvoki: 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.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Zvoki%20na%20kmetiji • Navodila za učenca (C4G5_NavodilaZaUcenca.docx)

Učni scenarij 6 – Kameleon na počitnicah

Naslov učnega scenarija	**Kameleon na počitnicah**
Pričakovano programersko predznanje	• predznanje ni potrebno
Učni cilji	Splošni učni cilji: • premikanje preko dogodkov, • zaznavanje ene barve, • Boolean vrednosti v logičnih izjavah, • definiranje, razlikovanje, dinamično preverjanje in odzivanje na dve različni stanji igre, Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • učenec implementira premikanje objekta s smernimi tipkami in pri tem uporabi dogodke, pri tem pa upošteva omejitve, • učenec uporabi blok za zaznavanje barve, da pridobi boolean vrednost, ki pove ali se objekt dotika neke barve,

	• učenec ve, da lahko določi stanje objekta preko barv, ki se jih ta dotika, • učenec razlikuje med dvema (enostavna različica) oz. petimi (zahtevnejša različica) različnimi stanji objekta in jih zna izraziti z logičnimi izjavami, • učenec ve, da se pozicija objekta v igri dinamično spreminja in uporabi neskončno zanko, da z njo periodično preverja njegovo trenutno stanje, • učenec uporabi pogojni blok "če-sicer", da glede na trenutno pozicijo objekta poda ustrezen odziv.
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Izdelajte preprosto igro, v kateri bo glavni objekt spreminjal svojo obleko glede na barvo ozadja na njegovi trenutni poziciji. Naloge: Učenci naj sprogramirajo kameleona, ki bo spreminjal svoj izgled (obleko) v dveh (enostavna različica): 1) ko bo plaval v morju, bo postal modre barve in rekel: "Kopam se v morju"; 2) ko bo na plaži bo rjave barve in bo rekel: "Sončnim se na plaži"; oz. petih (zahtevnejša različica) različnih situacijah: 1) ko bo plaval v morju, bo postal modre barve in rekel: "Kopam se v morju"; 2) ko se bo nahajal med morjem in plažo bo spremenil barvo v kombinacijo modre in rjave in rekel: "Nahajam se med morjem in plažo."; 3) ko bo na plaži bo rjave barve in bo rekel: "Sončnim se na plaži"; 4) ko se bo nahajal med plažo in gozdom bo spremenil barvo v kombinacijo rjave in zelene in rekel "Nahajam se med gozdom in plažo"; 5) v gozdu bo spremenil svojo barvo v zeleno in rekel: "Hladim se v senci dreves". **Učenci bodo spoznali blok za zaznavanje barve in se naučili kako ga lahko uporabijo v logičnih izrazih z namenom, da razlikujejo med dinamično spreminjajočimi stanji igre, ki so odvisna od trenutne pozicije glavnega lika (kameleona) in pri vsakem podali ustrezen odziv.**
Trajanje aktivnosti	45 minut

Učne strategije in metode	aktivno učenje, kolaborativno učenje, reševanje problemov
Učne oblike	frontalni način podajanja snovi individualno delo/delo v parih/skupinsko delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) **[Enostavna različica]** Kameleon se je odpravil na poletne počitnice. Rad se kopa v morju in sonči na plaži. Kot vsak kameleon, tudi on spreminja barvo tako, da se zlije z barvo okolice, nato pa še pove kje se trenutno nahaja. [1. korak] Učencem damo navodilo, da spremenijo izgled ozadja tako, da je razdeljen na dva dela iste barve. Prva polovica naj bo modra, ki bo predstavljala morje, druga pa rjava, ki bo predstavljala peščeno plažo. Lahko jim damo dodatno navodilo, da v sliko ozadja vključijo tudi druge elemente, da bi naredili ozadje bolj realistično. Vključijo lahko slike: valov, školjk, peščenih gradov, senčnikov, ipd. Opozorimo jih na to, da ne uporabijo predmetov, ki bi bili večji od glavnega lika in bi bili popolnoma druge barve od ozadja, saj v tem primeru blok za zaznavanje barve ne bo mogel ugotoviti, na katerem delu zaslona se nahaja kameleon. 

[2. korak]

Učenci naj narišejo ali poiščejo sliko kameleona na spletu (v primeru da sliko poiščejo, jih opozorimo, da mora imeti slika ustrezno licenco). Sliko kameleona naj pobarvajo modro (morje) oz. rjavo (plaža).

[3. korak]

Najprej morajo sprogramirati premikanje kameleona v štirih smereh z uporabo tipk. Kombinacijo tipk lahko določijo sami (npr. smerne tipke, WASD, ipd.). V tem predlogu učnega scenarija privzemamo, da so se to naučili v eni od prejšnjih aktivnosti. Učence na tem mestu le opozorimo, da morajo upoštevati omejitve pri premikanju oz. da se kameleon ne sme premakniti izven zaslona in da to dosežemo z uporabo bloka “odbij se, če si na robu”.

Gibanje lahko naredimo bolj realistično, če pred premikom kameleona obrnemo v smeri tega premika. To dosežemo z uporabo bloka “obrni se v smeri”, kjer med možnostmi izberemo ustrezno smer.

[4. korak]

Učence seznanimo s konceptom zaznavanja barve oz. barv, ki se jih glavni lik dotika. Z blokom “se dotika <barva>” lahko dobimo podatek v obliki boolean vrednosti (res oz. ni res) o tem ali se objekt dotika izbrane barve. To, da dobimo podatek v obliki boolean vrednosti nam

omogoča, da blok uporabimo v pogojnem bloku in s tem določimo ali oz. kdaj se bodo izvedli bloki v telesu pogojnega bloka.

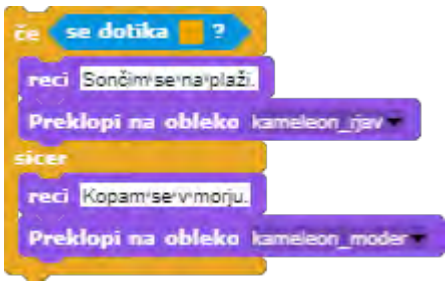
Z učenci se pogovorimo o tem katere so različne možnosti pozicije kameleona glede na barvo ozadja na sceni in kako jih lahko izrazimo preko bloka za zaznavanje barve.

Možnosti sta dve:

1. kameleon se dotika modre barve -> se dotika <modra>?
2. kameleon se dotika rjave barve -> se dotika <rjava>?

Ko se dotika določene barve moramo spremeniti njegov izgled (obleka) in uporabiti blok "reci" iz skupine ukazov "Izgled", da bo povedal kje se trenutno nahaja. Izgled objekta spremenimo tako, da preklapljamo med različnimi oblekami objekta. To naredimo tako, da uporabimo blok "Preklopi na obleko", ki ga najdemo v skupini ukazov "Izgled" in iz spustnega menija v bloku izberemo želeno obleko.

Kameleon se nahaja bodisi na modri podlagi, bodisi na rjavi podlagi, zato lahko za ločevanje uporabimo blok "če-sicer", saj če se ne dotika npr. rjave barve, posledično vemo, da se prav gotovo dotika modre, saj sta barvi samo dve:

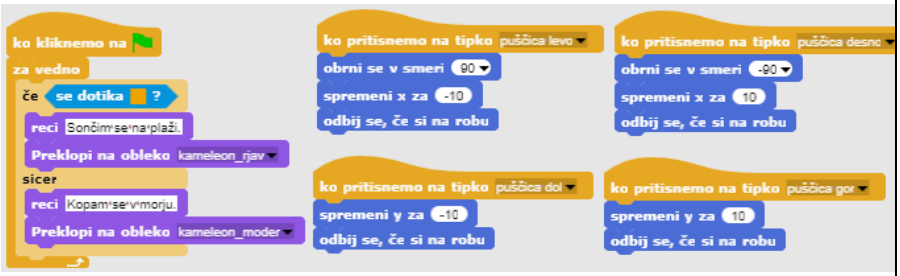


[5. korak]

Učencem povemo, da v situacijah, ko moramo nekatere ukaze izvrševati ves čas izvajanja programa, uporabimo neskončno zanko. Vse

kar bomo napisali pod neskončno zanko se bo periodično izvrševalo dokler se ne bo program zaključil. Preko diskusije z učenci razložimo, da je ta situacija točno taka, da jo moramo realizirati z neskončno zanko. Pozicija kameleona se namreč ves čas spreminja, prav tako pa ne moremo predvideti kako ga bo igralec premikal, zato moramo to kje se trenutno nahaja, ves čas preverjati.

[Končna koda]



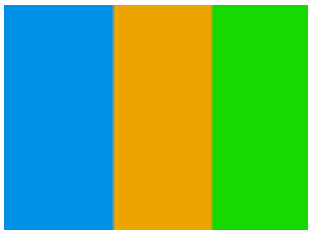
[Zahtevnejša različica]

Kameleon se je odpravil na poletne počitnice. Kot vsak kameleon, tudi on spreminja barvo tako, da se zlije z barvo okolice, nato pa še pove kje se trenutno nahaja. Rad se kopa v morju, sonči na plaži in ko postane prevroče se gre ohladit v senco dreves v gozdu.

[1. korak]

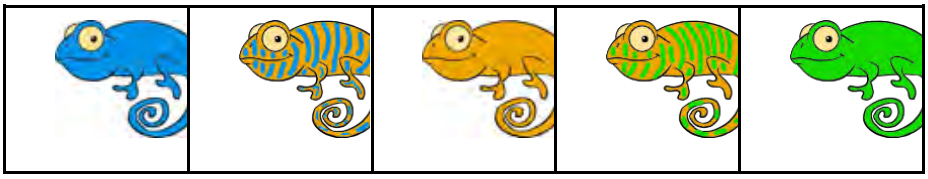
Učencem naročimo naj spremenijo izgled ozadja tako, da bo razdeljen na tri dele iste barve. Vsak del bo predstavljal različen del scene: modra bo predstavljala morje, rjava predstavlja plažo, zelena pa gozd. Učenci lahko po svoji izbiri na ozadje dodajo še dodatne predmete, kot npr.: valove, školjke, peščene gradove, senčnike, drevesa, ipd. da jo naredijo bolj realistično. Pri tem morajo biti pozorni, da ne uporabijo predmetov, ki bi bili večji od glavnega lika in bi bili popolnoma druge barve od ozadja, saj v tem primeru blok za

zaznavanje barve ne bo mogel ugotoviti, na katerem delu zaslona se nahaja kameleon.



[2. korak]

Učenci naj narišejo ali poiščejo sliko kameleona na spletu (v primeru da sliko poiščejo, jih opozorimo, da mora imeti slika ustrezno licenco). Sliko kameleona naj pobarvajo modro (morje), rjavo (plaža), zeleno (gozd), modro-rjavo (območje med morjem in plažo) in rjavo-zeleno (območje med plažo in gozdom), da bodo s tem ponazorili vse možnosti v katerih se lahko znajde kameleon.



[3. korak]

Najprej morajo sprogramirati premikanje kameleona v štirih smereh z uporabo tipk. Kombinacijo tipk lahko določijo sami (npr. smerne tipke, WASD, ipd.). V tem predlogu učnega scenarija privzemamo, da so se to naučili v eni od prejšnjih aktivnosti. Učence na tem mestu le opozorimo, da morajo upoštevati omejitve pri premikanju oz. da se kameleon ne sme premakniti izven zaslona in da to dosežemo z uporabo bloka “odbij se, če si na robu”.

Gibanje lahko naredimo bolj realistično, če pred premikom kameleona obrnemo v smeri tega premika. To dosežemo z uporabo bloka “obrni se v smeri”, kjer med možnostmi izberemo ustrezno smer.

[4. korak]

Učence seznanimo s konceptom zaznavanja barve oz. barv, ki se jih glavni lik dotika. Z blokom “se dotika <barva>” lahko dobimo podatek v obliki boolean vrednosti (res oz. ni res) o tem ali se objekt dotika izbrane barve. To, da dobimo podatek v obliki boolean vrednosti nam omogoča, da blok uporabimo v pogojnem bloku in s tem določimo ali oz. kdaj se bodo izvedli bloki v telesu pogojnega bloka.

Z učenci se pogovorimo o tem katere so različne možnosti pozicije kameleona glede na barvo oz. barve ozadja na sceni in kako jih lahko izrazimo preko bloka za zaznavanje barve.

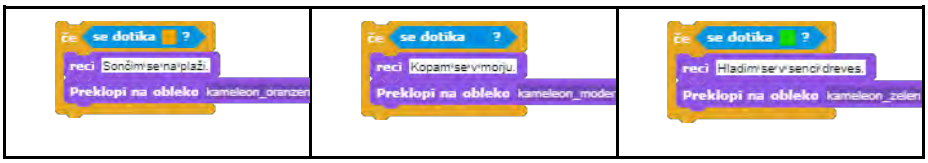
Ugotovimo, da imamo pet možnosti:

1. Kameleon se v celoti dotika modrega dela ozadja -> se dotika <modra>?
2. Kameleon se v celoti dotika rjavega dela ozadja -> se dotika <rjava>?
3. Kameleon se v celoti dotika zelenega dela ozadja -> se dotika <zelena>?
4. Kameleon je delno na modrem, delno na rjavem delu ozadja -> se dotika <modra>? IN se dotika <rjava>?
5. Kameleon je delno na rjavem, delno na zelenem delu ozadja -> se dotika <rjava>? IN se dotika <zelena>?

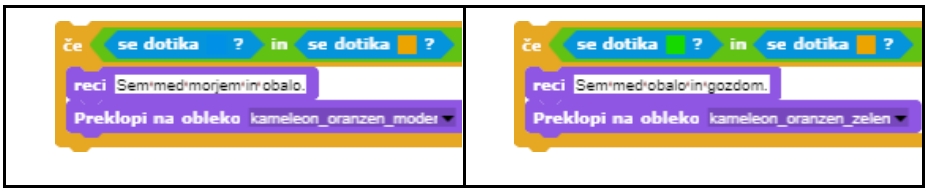
Ko se dotika določene barve moramo spremeniti njegov izgled (obleka) in uporabiti blok “reci” iz skupine ukazov “Izgled”, da bo

povedal kje se trenutno nahaja. Izgled objekta spremenimo tako, da preklapljamo med različnimi oblekami objekta. To naredimo tako, da uporabimo blok “Preklopi na obleko”, ki ga najdemo v skupini ukazov “Izgled” in iz spustnega menija v bloku izberemo želeno obleko.

Najprej to naredimo za enostavnejše situacije, ko je kameleon v celoti na delu ozadja iste barve.



Nato oblikujemo logično izjavo z uporabo operatorja IN, saj želimo preveriti ali se dotika dveh barv hkrati, torej ene IN druge.



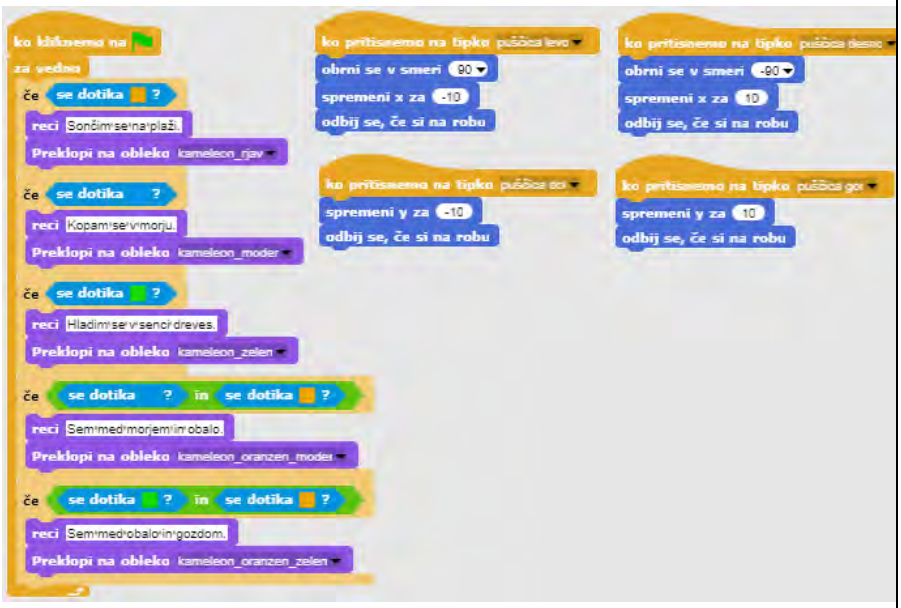
Če združimo zgornja bloka in jih postavimo pod dogodkovni blok za začetek programa (“ko kliknemo na zeleno zastavico”) ugotovimo, da se bosta pogojna bloka izvedla natanko enkrat. Učencem pojasnimo, da se bo pozicija kameleona zaradi poljubnega premikanja med igro spreminjala, zato ni dovolj, da le na začetku preverimo oba pogoja, ampak moramo to početi ves čas izvajanja programa.

[5. korak]

Učencem povemo, da v situacijah, ko moramo nekatere ukaze izvrševati ves čas izvajanja programa, uporabimo neskončno zanko. Vse kar bomo napisali pod neskončno zanko se bo periodično izvrševalo

dokler se ne bo program zaključil. Preko diskusije z učenci razložimo, da je ta situacija točno taka, da jo moramo realizirati z neskončno zanko. Učence prav posebej upozorimo na vrstni red preverjanja pogojev. Če bi postavili pogojni blok, ki preverja, če se kameleon nahaja na dveh barvah hkrati, pred blok, ki ugotavlja, če se dotika ene barve, bi se upošteval pogoj, ki je napisan kasneje.

[Končna koda]



[Učenci kodo prilagodijo]

Če želimo poenostaviti aktivnost, lahko pripravimo nekatere dele kode vnaprej v predlogi, ki jo učenci nato dopolnijo.

Učenci, ki so sledili priporočenemu sosledju učnih scenarijev, so se že naučili kako premikati objekt s tipkami. Zato predlagamo, da to kodo vključite v predlogo. Lahko pa kodo premikanja tudi prilagodijo po svoje, npr. namesto smernih tipk, za premikanje uporabijo kombinacijo WASD.

En od ključnih učnih ciljev v aktivnosti je razumevanje pomena neskončne zanke in prepoznavanje situacij, ki jih moramo reševati z njeno uporabo. V predlogo lahko vključimo del kode, ki se bo izvajala periodično, ostalo pa dopolnijo sami. Na ta način bodo še vedno imeli priložnost razumeti koncept neskončne zanke, čeprav ne bodo definirali vseh situacij sami, ampak le dopolnili manjkajoče. Koda v predlogi naj vključuje dve pomensko različni situaciji: 1) objekt je v celoti na eni barvi, 2) objekt se hkrati dotika dveh različnih barv.

Predlagan del kode v predlogi:

Orodja in viri za učitelje

- Enostavna različica v programu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_enostavni_SLO
- Zahtevnejša različica v programu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_SLO
- Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena.
- Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.

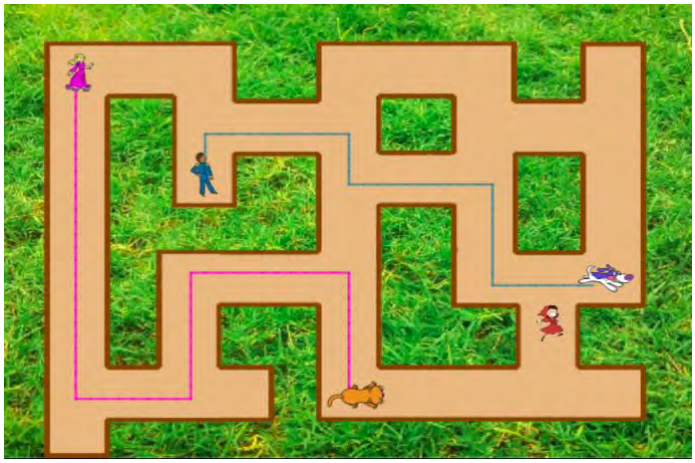
Viri/gradiva za učence	• Predloga le s kalemeonom in ozadjem: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_SLO_predloga • Predloga s slikovnim materialom za enostavno različico: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_enostavni_slike_SLO • Predloga s slikovnim materialom za zahtevnejšo različico: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_zahtevnejsi_predloga_slike_SLO • Delno rešena predloga za učence: https://snap.berkeley.edu/project?user=zapusek&project=kameleon_predloga_SLO

Učni scenarij 7 – Pomagaj princu in princeski najti svoje živali

Naslov učnega scenarija	Pomagaj princu in princeski najti svoje živali
Pričakovano programersko predznanje	Dodajanje besedila za lik Premikanje lika s tipkami in uporabo dogodkov Uporaba pogoja *če se lik dotika* Uporaba dogodkov
Učni cilji	Splošni učni cilji: • Pogoj *če se lik dotika* določene barve • Premikanje po koordinatah • Svinčnik dvignjen, spuščen • Barva svinčnika Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporabi pogojni stavek za preverjanje, ali se lik dotika določene barve in mu v primeru izpolnjenega pogoja ne dovoli nadaljevanja poti • Učenec nastavi začetne x in y koordinate lika • Učenec uporabi dvignjen / spuščen svinčnik za risanje črte / poti • Učenec spremeni barvo svinčnika glede na par, ki ga povezuje • Učenec ve, da mora na začetku izbrisati vse poti, ki so ostale od prej

Cilji, naloge in kratek opis aktivnosti	Kratek opis: Dekle mora pomagati princeski najti svojo mačko in princu najti svojega psa. To stori tako, da gre do princeske in ji z risanjem črte pokaže pot do mačke, na enak način pokaže tudi princu pot do psa. Na svoji poti se mora deklica izogniti srečanju med živalmi, zato se poti ne smejo križati. Naloge: V prvem koraku morajo učenci izbrati ustrezno ozadje (labirint), nato dodajo 5 likov: dekle, princesko, princa, mačko in psa. Nato napišejo kodo za premikanje s tipkami (z uporabo dogodkov), kjer pa morajo dekletu preprečiti, da stopi na travo. Sledi risanje s svinčnikom in spreminjanje barve z uporabo dogodkov. Prav tako morajo napisati kodo, ki na začetku izbriše vse poti, in napisati začetna navodila, ki jih poda dekle. **Cilj: Učenci bodo spoznali risanje s premikanjem lika. Poleg tega se bodo naučili, kako z uporabo pogojnega stavka preprečiti liku prosto gibanje po celotnem labirintu.**
Trajanje aktivnosti	30 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Frontalna učna oblika Individualna učna oblika
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Za lažji začetek lahko damo učencem predlogo, v kateri imajo na voljo: • Ozadje labirinta • Lik dekleta • Kodo za premikanje v eno smer Dekle se odloči, da bo pomagalo princesi najti svojo mačko in princu svojega psa. To bo storila tako, da jima bo pokazala (narisala) pot do njinuh živali. V

izogib srečanju živali mora poskrbeti, da se poti ne bodo križale, označila pa bo tudi vsako pod z drugo barvo.



[Korak 1]

Za realizacijo pogoja »če se dotika barve« mora biti ozadje (trava) enobarvna ali pa mora imeti pot enobarvni okvir, kot v našem primeru. Da se izognemo tem »težavam« pri iskanju ustreznega ozadja, ponudimo učencem zgornje ozadje.

[Korak 2]

Učenci imajo dan lik dekleta, najti morajo še ostale 4 in jih postaviti v labirint. Za vse like morajo na začetku nastaviti ustrezno velikost (ki je manjša od poti v labirintu). Za vsak lik uporabijo kodo:

Za dekle je priporočena velikost 8%, ostali liki so lahko večji.

[Korak 3]

V naslednjem koraku morajo napisati kodo za premikanje dekleta z uporabo tipk. Predvidevamo, da ti že znajo iz prejšnjih učnih ur. Za pomoč imajo podano kodo za premikanje v eno smer. Naredijo še ostale tri.

[Korak 4]

Sedaj morajo dekletu preprečiti gibanje po travi. To naredijo z uporabo ukaza *če se dotika rjave barve*. Če se dekle dotika rjave barve (konca poti), se pomakne za 10 korakov nazaj (Pozor! Rjava barva mora biti enakega odtenka kot tista v labirintu). Teh dveh korakov (premik za 10 naprej in nato za 10 nazaj) ne vidimo in zdi se, kot da dekle ostane na istem mestu. Spodnja koda prikazuje premikanje v desno, torej 10 korakov nazaj pomeni spremeni x za -10.

Kodo dodamo k prejšnji. To je sedaj koda za premik v desno:

Podobno kodo je potrebno narediti še za ostale 3 smeri.

[Korak 5]

Sledi programiranje risanja. Rišemo z uporabo ukazov *svinčnik spuščen* in *svinčnik dvignjen* ter z uporabo dogodka *ko prisitnemo na tipko*.

Ko pritisnemo na tipko »D« in se dekle premakne nariše pot. Ko pritisnemo na tipko »E«, konča z risanjem.

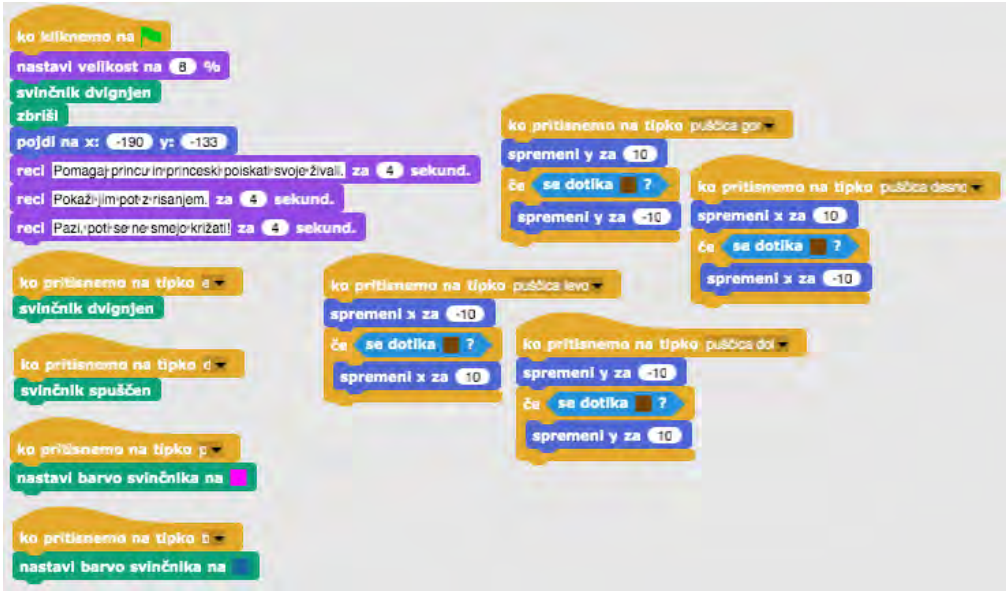
Podobno naredimo kodo za barvo poti.

[Korak 6]

V zadnjem koraku napišemo še kodo *ko kliknemo na zeleno zastavico*, kjer učenci dodajo nekaj navodil, ki jih dekle poda na začetku igre.

Z igranjem igre, risanjem poti ter ponovnim igranjem, bodo učenci videli, da je na začetku pametno uporabiti naslednje ukaze: *svinčnik dvignjen* (v primeru, da je ostal v prejšnji igri spuščen), *zbriši* (izbriše risanje iz prejšnje igre), *pojdi na x, y* (začetne koordinate za dekle).

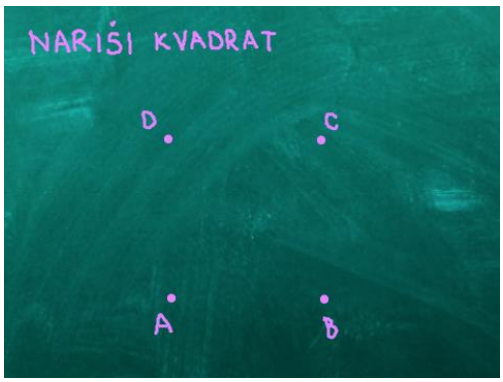
Za določanje začetnih koordinat za dekle, lik primemo in povlečemo na pozicijo, kjer želimo, da se pojavi na začetku igre. Nato kliknemo na skupino ukaznih blokov *Premikanje*, kjer najdemo *položaj x* in *položaj y*. S klikom na *položaj x* ugotovimo x koordinato lika, podobno s klikom na *položaj y*.

	[Končna koda] Dekle  Npr. Princeska  [Dodatne naloge] Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom: • Nastavi začetno pozicijo princa in princese ter napiši kode za njuno premikanje. Po potrebi ju pomanjšaj. Narišeta naj pot do svoje živali. • Dodaj še eno žival za deklico. • Vsak naj riše pot z drugo barvo. • Prilagodi začetna navodila. • Dodaj navodila za premikanje lika in risanje s klikom na lik.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G7_Pomagaj_princu_in_princeski%20-%20Cela • Primer dodatne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G7_Pomagaj_princu_in_princeski%20-%20Cela%20%2B%20Dodatek • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena.

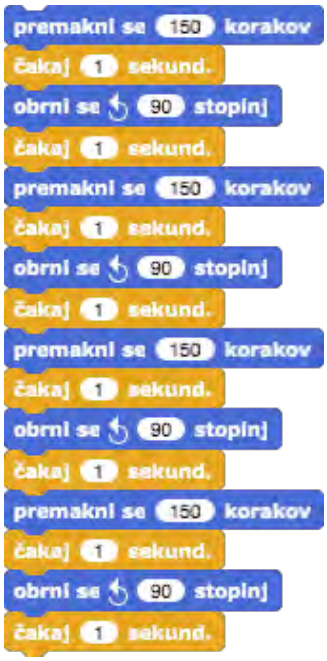
	• Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G7_Pomagaj_princu_in_princeski • Navodila za učenca (C4G7_NavodilaZaUcenca.docx)

Učni scenarij 8 – Risanje s kredo

Naslov učnega scenarija	Risanje s kredo
Pričakovano programersko predznanje	Dodajanje besedila za lik Risanje s svinčnikom (svinčnih spuščen, dvignjen, barva) Premikanje lika Uporaba zank Uporaba dogodkov
Učni cilji	Splošni učni cilji: • Zanka ponovi • Obrat za 90 stopinj • Obrni se v smeri • Menjava ozadja Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporabi zanko ponovi, ko se ista koda ponovi 2/4 krat • Učenec uporabi *obrat za 90 stopinj* pri risanju kvadrata, pravokotnika, črke »T« • Učenec razume ukaz *obrni se v smeri 90* • Učenec zna zamenjati ozadje v kombinaciji z uporabo dogodka *ko pritisnemo na tipko*
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Igralec mora s pritiskom na tipko povezati tičke v tri različne oblike – kvadrat, pravokotnik in črko »T«. Naloge: Učenci izberejo ozadje *tablaKvadrat* in napišejo kodo za risanje kvadrata. Začetna pozicija je točka A. Pri risanju kvadrata ponovijo določene

	korake 4 krat, zato namesto pisanja iste kode 4 krat, uporabijo zanko ponovi 4 krat. Nato napišejo kodo za risanje pravokotnika, kjer uporabijo zanko ponovi 2 krat. V zadnjem koraku povečejo točke v črko T. Ko je mogoče, uporabijo zanko. Zamenjati morajo tudi ozadje za pravokotnik in črko T. **Cilji: Učenci se bodo naučili risanja različnih oblik s kodo. Naučili se bodo uporabo zanke ponovi za krajšanje kofe in menjavo ozadja.**
Trajanje aktivnosti	60 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Frontalna učna oblika Individualna učna oblika / Delo v paru
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Učenci imajo podana: • 3 ozadja s točkami, ki jih morajo povezati • Lik krede Kreda bi rada narisala kvadrat, pravokotnik in lik oblike črke T, ampak se ne zna premikati in obračati. Pomagaj kredi in ji povej, kaj mora narediti! [Korak 1] 

Učenci začnejo z zgornjim ozadjem. Napisali bodo kodo za premikanje krede. Kreda začne v točko A, se premake X korakov do točke B, se obrne za 90 stopinj v levo, se premakne za X korakov do točke C, se obrne za 90 stopinj v levo, se premakne X korakov do točke D, se obrne za 90 stopinj v levo in se premakne za X korakov to točke A (in se obrne za 90 stopinj v levo).



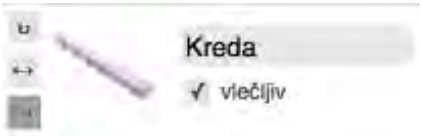
Uporabaili smo ukaz *obrni se za 90 stopinj*. Druga možnost je obračanje z ukazi *obrni se v smeri 0, 90, 180 in -90*, ampak je možnost 1 lažja, saj je ukaz vedno isti (obrni se za 90 stopinj), prav tako pa lahko uporabimo tudi zanko za ponovitev 4-ih enakih korakov, česar v možnosti 2 ne moremo narediti.

Ukaz *čakaj 1 sekund* je dodan, da se pri risanju vidijo vsi koraki (animacija). Brez tega ukaza se cela koda izvede v trenutku in koraki niso vidni. Učenci naj poskušajo najprej brez uporabe ukaza *počakaj*, da vidijo njegov pomen.

Učenci naj sami pomislijo, kako bi skrajšali kodo, če je to mogoče. Se kateri deli ponavljajo? Odgovor je da. Namesto, da pišemo isto kodo 4 krat, pri programiranju uporabimo zanko *ponovi*.

Če želimo, da kreda riše pot, moramo pred zanko dodati še ukaz *svinčnik spuščen*.

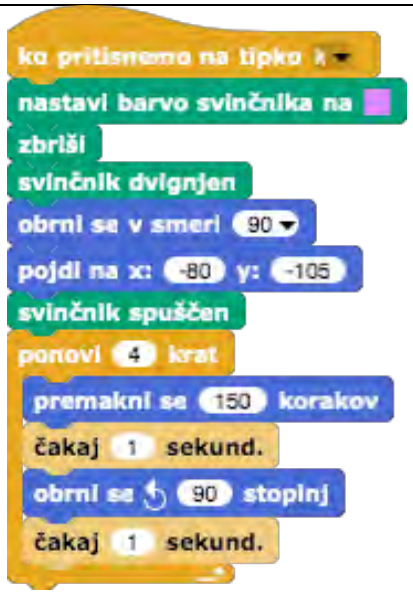
Če želimo, da se lik krede ne vrti med obračanjem, kliknemo na ukaz *ne vrti*.



[Korak 2]

Za začetek izvedbe kode lahko uporabimo ukaz *ko pritisnemo na tipko K*. Prav tako lahko nastavimo barvo krede, spustimo svičnik (v primeru, da je ostal v prejšnji igri spuščen), zbrišemo (izbriše risanje iz prejšnje igre), nastavimo začetne koordinate kredi (v točki A).

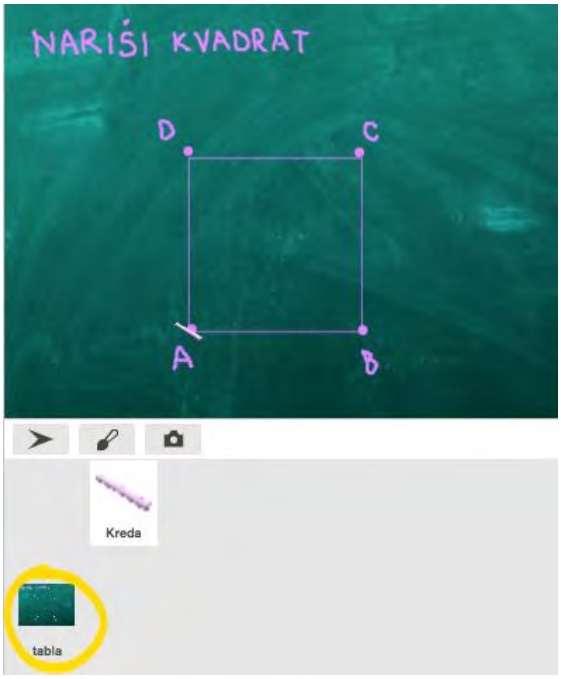
Včasih se zgodi, da program prekinemo med izvajanjem in ostane kreda obrnjena v napačno smer. Težava nato nastane, ko program zaženemo, in se kreda namesto v desno, premakne v levo, gor ali dol. Da se temu izognemo, na začetku nastavimo ukaz *obrni se v smeri 90*.



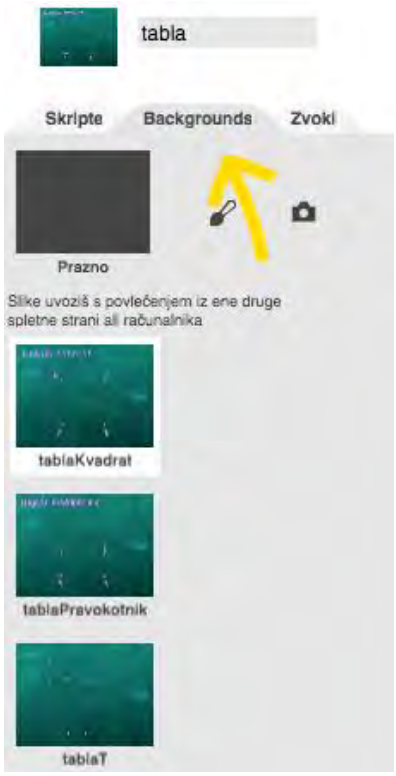
[Korak 3]

Sledi koda za pravokotnik. Najprej moramo zamenjati ozadje. To naredimo v dveh korakih:

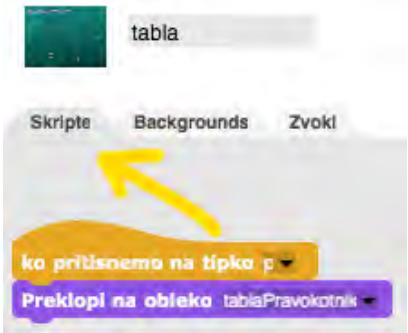
a) Kliknemo na ozadje (imenovano *tabla*)



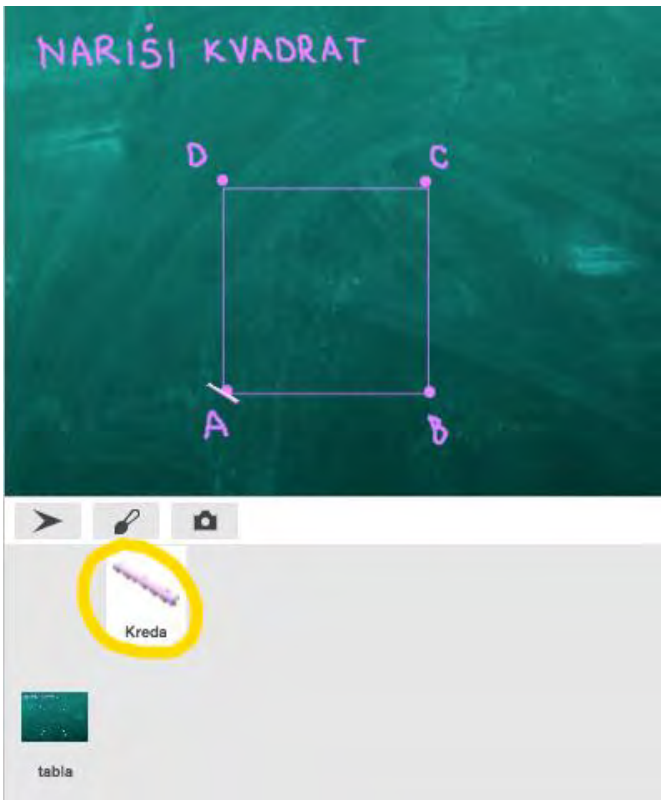
S klikom na Backgrounds vidimo vsa tri ozadja, ki jih potrebujemo za to aktivnost (tablaKvadrat, tablaPravokotnik, tablaT), ter še nekaj dodatnih ozadij.



Za pisanje kode kliknemo na *Skripte*. Za spreminjanje ozadja s klikom na tipko uporabimo ukaza *ko pritisnemo na tipko P* ter *Preklopi na obleko tablaPravokotnik*.



b) Kliknemo ponovno na kredo.

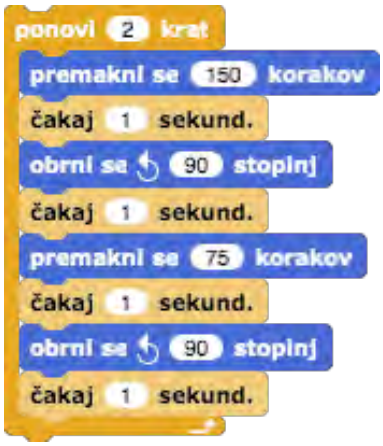


Pod kodo iz [Korak 2] dodamo kodo, ki bo igralcu povedala, naj pritisne tipko P za spremembo ozadja.

[Korak 4]

S klikom na tipko P se ozadje spremeni v zgornje ozadje. Podobno kot prej, napišemo kodo za premikanje krede, ki bo povezala točke in narisala

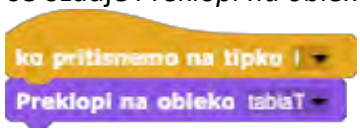
pravokotnik. Učenci lahko prejšnjo kodo podvojijo in ustrezno spremenijo. Spremeniti morajo zanko ponovi, ki se bo sedaj izvedla 2 krat.



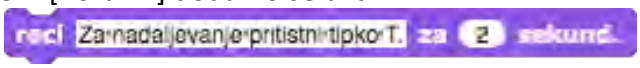
[Korak 5]

Po pravokotniku sledi še zadnja naloga in sicer bodo učenci povezali točke v obliki črke T. Spremeniti morajo torej ozadje, kar pomeni, da ponovijo [Korak 3] in spremenijo le črko P v T in ozadje *tablaPravokotnik* v *tablaT*.

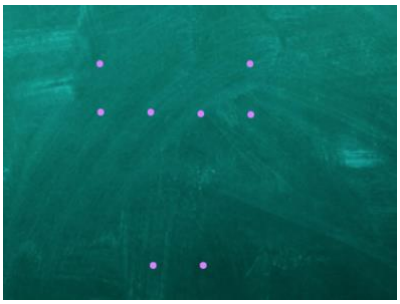
a) *Ko pritisnemo na tipko T* se ozadje *Preklopi na obleko tablaT*.



b) Po koncu kode iz [Korak 4] dodamo še ukaz:



[Korak 6]



S pritiskom na tipko T se ozadje spremeni v zgornje. Podobno kot prej moramo točke povezati v črko T. Učenci lahko kopirajo prejšnjo kodo in jo ustrezno spremenijo.

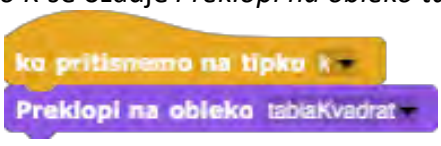
Spremeniti morajo začetne koordinate, ki niso enake prejšnjim. Kako določijo pravilne začetne koordinate že vedo.

Nato točke povežejo v črko T. Ugotoviti morajo ustrezno število korakov (v pomoč imajo navodila za učenca). Možna rešitev:

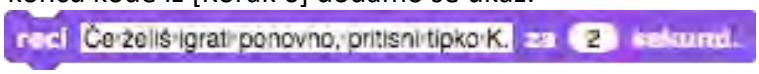
[Korak 7]

Na koncu dodamo še kodo za spremembo ozadja na začetno ozadje za risanje kvadrata. Ponovimo [Korak 3/5].

a) *Ko pritisnemo na tipko K* se ozadje *Preklopi na obleko tablaKvadrat*.



b) Po koncu kode iz [Korak 6] dodamo še ukaz:



[Končna koda]

[Dodatne naloge]

Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom:

- Dodaj novo ozadje in nariši točke / oglišča.
- Napiši kodo, ki poveže oglišča. Ozadje lahko narišeš sam ali pa uporabiš dano ozadje.

Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G8_Risanje_s_kredo • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G8_Risanje_s_kredo%20-%20Delno • Navodila za učenca (C4G8_NavodilaZaUcenca.docx)

Učni scenarij 9 – Pobiranje smeti in čiščenje parka

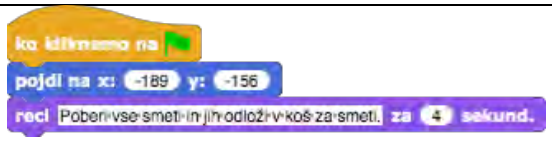
Naslov učnega scenarija	Pobiranje smeti in čiščenje parka
Pričakovano programersko predznanje	Določanje začetnih koordinat Nastavljanje velikosti lika Dodajanje besedila za lik Premikanje lika s tipkami in uporabo dogodkov Uporaba pogoja *če se lik dotika*
Učni cilji	Splošni učni cilji: • Spremenljivke • Prikaži in skrij lik • Podvoji lik • Podvoji del kode • Pogojni stavek Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporabi spremenljivko za štetje pobranih smeti • Učenec zna skriti lik, ko se ga dekle dotakne, in prikazati lik na začetku igre • Učenec zna podvojiti lik (iz 1 steklenice naredi npr. 4) • Učenec zna podvojiti del kode (npr. kodo steklenice podvoji in prenese na kodo papirja)

	• Učenec zna uporabiti pogojni stavek za preverjanje, ali je lik prikazan in ali so vse smeti pobrane
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Park je poln smeti in dekle se odloči, da ga bo očistilo. Ko pobere vse smeti, jih vrže v koš. Naloge: Učenci nastavijo dekletu začetne koordinate. Igra se konča, ko dekle pobere vse smeti in jih odvrže v koš. Za preverjanje, ali so vse smeti v košu, bodo učenci uporabili spremenljivko (1 pobrane smeti = 1 točka). Ko se dekle dotakne steklenice, se steklenica skrije in točke povečajo za 1. Ko pobere vse, gre do koša, koš pa ji sporoči, ali je pobrala vse smeti ali ne. **Cilji: Učenci bodo spoznali rabo spremenljivk in podvajanje kode ali celotnega lika.**
Trajanje aktivnosti	45 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Frontalna učna oblika Individualna učna oblika
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Učencem je dano: • Ozadje • Lik dekleta (s kodo premikanja), lik steklenice, lik papirja in lik koša Dekle na sprehodu po parku opazi, da je v parku veliko smeti in se odloči, da ga bo očistila. Ko pobere vse smeti, jih vrže v koš in nato lahko uživa v lepem sončnem vremenu v čistem parku.

[Korak 1]

Poleg ozadja je podana tudi koda za premikanje s tipkami in pogojem za dotikanje rjave barve:

Na začetku nastavimo začetne x in y koordinate dekleta. Učenci lahko poljubno nastavijo x in y, pomembno je le, da so znotraj poti. Dodajo tudi začetna navodila, npr:



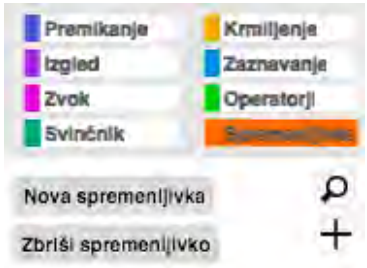
[Korak 2]

Za štetje pobranih smeti bomo uporabili spremenljivko.

Kaj je spremenljivka?

Spremenljivko si lahko predstavljamo kot škatlo, v kateri shranjujemo podatke. Lahko si predstavljamo, da dekle pobere smeti in jih shrani v spremenljivki, poimenovani *tocke*. Spremenljivka *tocke* torej šteje, koliko smeti je dekle pobralo.

Kako ustvarimo spremenljivko?



Izberemo ukazne bloke z imenom *Spremenljivka*, nato kliknemo *Nova spremenljivka*, vpišemo *ime spremenljivke* in nato *V redu*.

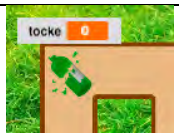
Ime spremenljivke naj:

- bo smiselno in naj poimenuje to, kar bo v njej shranjeno – npr. *tocke*, *stOdpadkov* ipd.,
- **ne** vsebuje črk **č, š, ž** ipd.,
- **ne** vsebuje **presledkov**. Če želimo spremenljivko poimenovati npr. število odpadkov, lahko torej uporabimo ime *stevilo_odpadkov*, *steviloOdpadkov*, ali krajše *st_odpadkov* ali *stOdpadkov* ipd.

Ko dodamo spremenljivko, se le ta pojavi med 

ukazi:

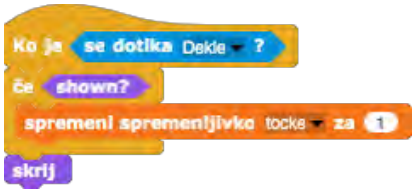
Kljukica pomeni, da bo spremenljivka vidna na zaslonu:



Na začetku igre mora biti vrednost spremenljivke 0, saj dekle še ni pobralo nobene smeti. Pod kodo iz [Korak 1] dodamo ukaz *nastavi __ na 0*. S klikom na puščico izberemo spremenljivko, ki jo želimo nastaviti, kar je v našem primeru spremenljivka *tocke*:

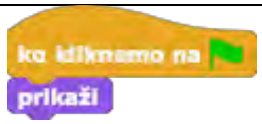
[Korak 3]

Učenci napišejo kodo za lik steklenice. Ideja je, da lik izgine (se skrije), ko se ga dekle dotakne. Koda se bo torej izvedla, ko se steklenica dotakne dekleta. Potrebno je razmisliti, v katerem primeru lahko dekle steklenico pobere. Rekli smo, da se smeti skrijejo, ko so pobrane, torej lahko steklenico pobere, če je le-ta še vedno na tleh (je prikazana – angl. *shown*). Če je steklenica na tleh, jo poberemo »in damo v škatlo (spremenljivko)«. Pred tem je bila vrednost spremenljivke *tocke* 0, sedaj je 1. Ko poberemo steklenico, torej spremenimo vrednost spremenljivke (tocke) za 1. Ko steklenico poberemo, jo skrijemo.



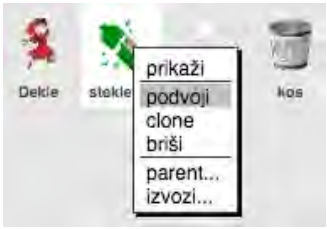
Igro zaženemo s klikom na zeleno zastavico in preverimo, ali koda deluje. Z dekletom se pomaknemo do steklenice, ki mora izginiti in spremenljivka se mora povečati za 1. Nato želimo igrati še enkrat in ponovno kliknemo na zeleno zastavico. Kaj se zgodi? Kje je steklenica?

Steklenica je skrita, saj smo jo v prejšnji igri pobrali (skrili). Zato moramo na začetku igre napisati kodo za prikaz steklenice:



[Korak 4]

Sedaj želimo imeti več steklenic. Lik steklenice lahko enostavno podvojimo z desnim klikom na lik in izberemo podvoji.

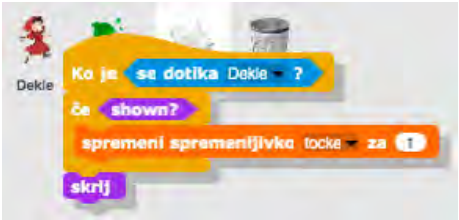


Podvojen lik se pojavi na ekranu. Kliknemo z miško nanj in ga povlečemo za željeno mesto v labirintu.

Postopek lahko ponovimo in podvojimo steklenico poljubno krat.

[Korak 5]

Podobno kodo kot za steklenico želimo imeti tudi za lik papirja. Kodo preprosto primemo, povlečemo na željen lik in spustimo.



Na enak način prenesemo tudi kodo *ko kliknemo na zeleno zastavico – skrij*.

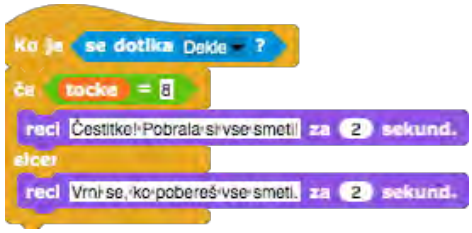
Sedaj lahko ponovimo [Korak 4] in podvajamo lik papirja, da imamo tudi več likov papirja po labirintu.

[Korak 6]

Zadnji korak je še koda za lik koša za smeti. Lik je podan in poljubno ga lahko prestavijo na drugo mesto v labirintu.

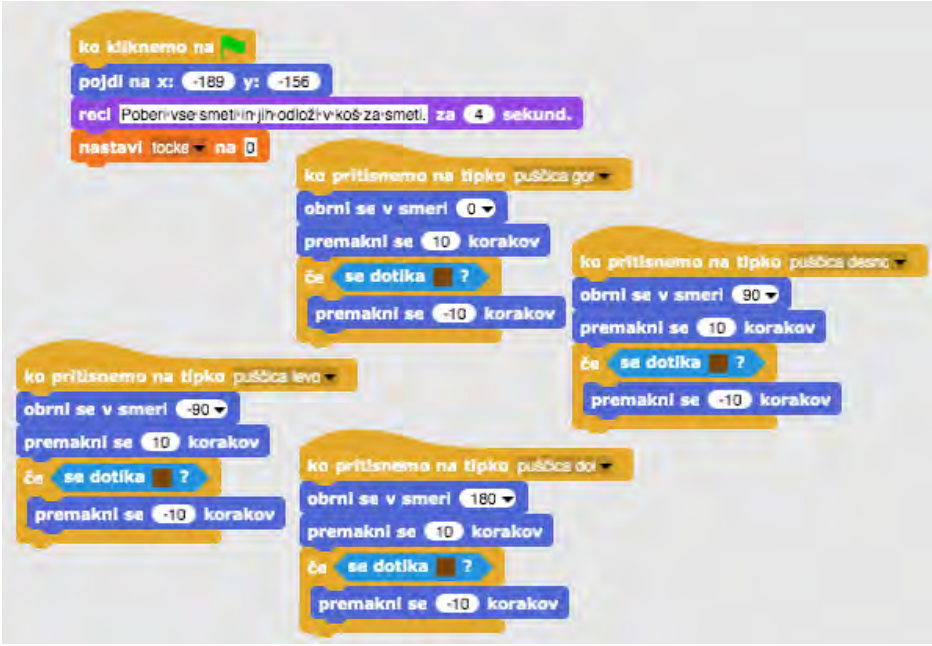
Tudi ta koda se bo začela izvajati, ko se dekle dotakne koša.

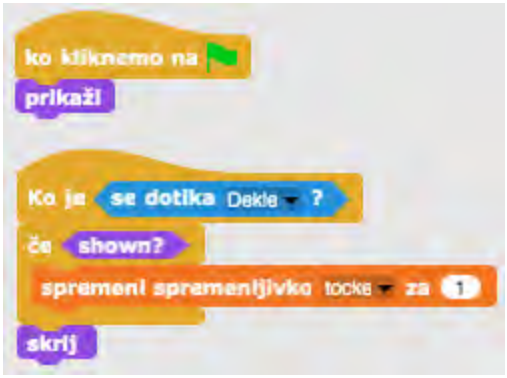
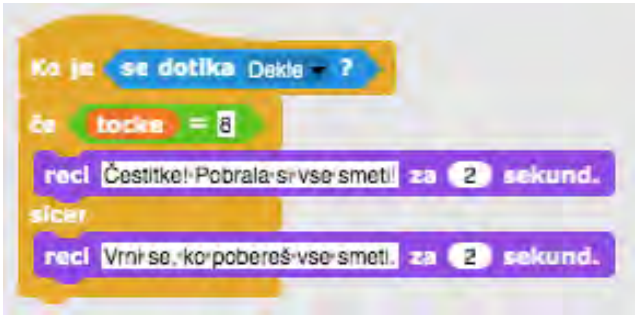
Koš bo moral preveriti, ali so pobrane vse smeti ali ne. S pomočjo spremenljivke *tocke* bo to zelo enostavno. Recimo, da imamo v igri 8 likov smeti, zato bo koš preveril, ali je vrednost spremenljivke enaka 8. Če je, to pomeni, da so pobrane vse smeti, drugače pa ne. To bomo preverili z pogojnim stavkom *če* in dodali odziv, ali je igralec pobral vse smeti ali ne.



[Končna koda]

Dekle



	Steklenica / Papir  Koš  [Dodatne naloge] Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom: • Dodaj še tretjo vrsto odpadkov (npr. bio). • Koš naj izpiše: »Pobral/a si X steklenih, Y papirnatih in Z bio odpadkov«. • Če igralec pobere vse smeti, koš reče »Čestitam, pobral/a si vse smeti!«. • Če igralec ni pobral vseh smeti, mu koš pove, katerih smeti ni pobral vseh. Npr. »Nisi pobral vseha papirja«, »Nisi pobral vseh bio odpadkov«. Nato reče še »Vrni se, ko pobereš vse smeti«.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G9_PobiranjeSmeti • Primer dodatne aktivnosti v Snap!-u:

	https://snap.berkeley.edu/project?user=mateja&project=C4G9_PobiranjeSmeti%20%2B%20Dodatek • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G9_PobiranjeSmeti%20-%20Delno • Navodila za učenca (C4G9_NavodilaZaUcenca.docx)

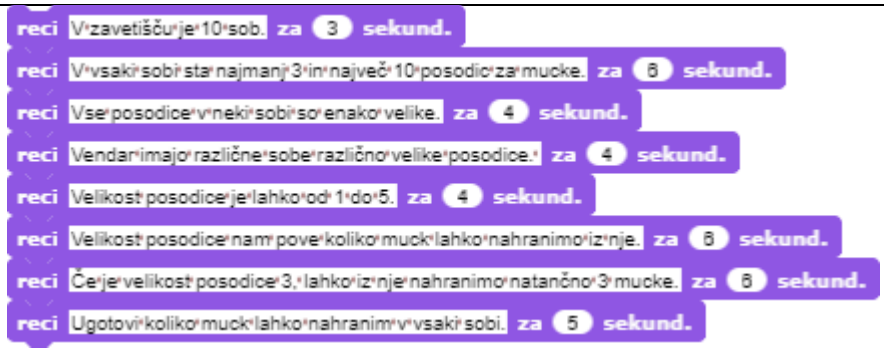
Učni scenarij 10 – Nahrani mačke

Naslov učnega scenarija	Nahrani mačke
Pričakovano programersko predznanje	• pogojni stavki (if, if-else blok) • izpis besedila (blok “reci”)
Učni cilji	Splošni učni cilji: • nastavljanje in povečevanje vrednosti spremenljivke, • nastavljanje vrednosti spremenljivki znotraj/zunaj zanke, • for zanka (oz. zanka ponovi n-krat), • naključna števila, • združevanje nizov, • operatorji: logični, aritmetični • uporabnikov vnos. Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • učenec prepozna situacijo, v kateri je smiselno uporabiti zanko, ki se ponovi natanko n-krat, • učenec loči med nastavljanjem vrednosti spremenljivki pred zanko in med vsako ponovitvijo znotraj zanke,

	• učenec zna uporabiti blok “vprašaj” in z njim dobi številko, ki jo je vpisal uporabnik, • učenec zna uporabiti aritmetične operatorje in z njimi izračunati vrednost pravilnega odgovora, • učenec zna uporabiti pogojni blok “če-sicer” in z njim preveri pravilnost odgovora, ki ga je podal uporabnik, • učenec zna uporabiti spremenljivko za štetje pravilnih odgovorov.
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Izdelajte igro v kateri bo moral igralec opraviti deset računov množenja in šteti pravilne odgovore. Naloga: Izdelajte igro, v kateri bo oskrbnica mačjega zavetišča Marta ponavljajoče spraševala igralca o številu mačk, ki jih lahko nagrani v eni od desetih sob v zavetišču. Odgovor je odvisen od števila in velikosti posod v vsaki od sob. Število posod in njihova velikost se morata za vsako sobo posebej določiti naključno. Poleg tega želimo imeti števec, ki bo štel pravilne odgovore. Igra se naj začne tako, da lik oskrbnice razloži nalogo v igri, nato pa se igra začne. Igra se zaključi, ko oskrbnica desetkrat vpraša po številu nahranjenih mačk. Vsakič ko vpraša in uporabnik odgovori mu mora podati povratno informacijo o tem ali je pravilno odgovoril. Ko s spraševanjem konča mora igralcu podati povzetek njegove uspešnosti, tako da mu pove kolikokrat je odgovoril pravilno in kolikokrat napačno. **Učenci se bodo preko aktivnosti seznanili s konceptom ponavljajočega prirejanja naključne vrednosti spremenljivki znotraj zanke in se naučili kako je ta situacija drugačna od prirejanja vrednosti spremenljivki izven zanke. Prav tako se bodo naučili kako lahko pridobijo uporabnikov odgovor, ga preverijo in štejejo pravilne.**
Trajanje aktivnosti	45 minut

Učne strategije in metode	aktivno učenje, kolaborativno učenje, reševanje problemov
Učne oblike	frontalni način podajanja snovi individualno delo/delo v parih/skupinsko delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Oskrbnica želi nahraniti mačke v desetih sobah mačjega zavetišča. V vsaki sobi je naključno število posod (od 2 do 10), ki imajo različne velikosti (od 1 do 5), posode, ki so v isti sobi pa se po velikosti ne razlikujejo. Velikost posode pove koliko mačk lahko nahranimo iz nje, npr. če je velikost posode 3, to pomeni, da bomo iz nje nahranili tri mačke. Pomagaj ugotoviti število mačk, ki jih lahko nahranimo v posamezni sobi. [1. korak] Učencem naj na začetku izdelajo neko zanimivo ozadje za igro. Če želimo prihraniti čas, jim ga posredujemo v predlogi.  [2. korak] Učenci naj privzetemu objektu (želvici) določijo nov izgled, ki bo predstavljal oskrbnico mačjega zavetišča. 

	[3. korak] Preko pogovora, učencem pomagamo ugotoviti, da si bomo morali v igri uporabljati naslednje tri vrednosti in jih bomo morali zaradi tega hraniti v spremenljivkah: 1) število pravilnih odgovorov, 2) naključno število posod v eni sobi (2-10) in 3) naključno število, ki bo predstavljajo velikost posode znotraj ene sobe (1-5). Na začetku bomo morali nastaviti spremenljivko, ki bo hranila pravilne odgovore na 0, drugi dve pa bomo nastavili znotraj zanke, saj se bo njuna vrednost morala nastaviti pri vsaki ponovitvi. Le na ta način bomo za vsako sobo dobili nova naključna števila. Opozorimo jih na dejstvo, da za štetje sob ne bomo potrebovali dodatne spremenljivke, saj bomo v ta namen uporabili spremenljivko, ki jo uporablja zanka for za štetje ponovitev. Njena vrednost se bo namreč na začetku nastavila na 1, nato pa povečevala za ena do končne vrednosti: 10. Spreminjanje vrednosti te spremenljivke v vsaki ponovitvi zanke pa nam ravno opisuje štetje sob. Na začetku, ko bomo v prvi sobi bo vrednost te spremenljivke 1, nato v naslednji, ko bomo v drugi sobi 2, in tako naprej do zadnje desete sobe.  [4. korak] Na začetku igre moramo igralcu podati navodila za igranje. To naredimo z uporabo bloka “reci [besedilo] in počakaj [n] sekund”, ki ga najdemo v skupini ukazov “Izgled”. Učence opozorimo, da naj upoštevajo količino besedila, ki se bo prikazalo in ustrezno nastavimo čas prikazovanja.



[5. korak]

Z učenci se pogovorimo o tem, katere aktivnosti se bodo ponovile v vsaki sobi in bodo zato enake. Ukaze teh aktivnosti moramo postaviti znotraj telesa zanke, da se bodo izvedli pri vsaki ponovitvi te zanke. To kar se bo ponavljalo je naslednje: najprej bomo morali naključno določiti vrednost spremenljivke v kateri bomo hranili število posod, nato velikost posod, prav tako pa bomo morali vsakič uporabnika pozvati, da vpiše svoj odgovor. Odgovor bo treba nato vedno znova preveriti in podati ustrezno povratno informacijo v primeru ko bo odgovor pravilen oz. napačen. Ob pravilnem odgovoru bomo morali vrednost spremenljivke, ki nam bo služila za štetje pravilnih odgovorov, povečati za ena. Na tem mestu jim razložimo razliko med blokoma “povečaj za” in “nastavi na”. Pri prvem se bo trenutna vrednost povečala za izbrano vrednost, v drugem pa nastavila na neko vrednost. Za štetje bomo morali uporabiti prvo možnost.

[6. korak]

Spremenljivki naključno določimo vrednost z uporabo blokov: “nastavi [ime spremenljivke] na” in “naključno število od [n] do [m]”.

[7. korak]

Uporabnika bomo za vsako ponovitev pozvali, da naj vpiše število mačk, ki jih lahko nahranimo v posamezni sobi. Želimo, da se mu vprašanje izpiše tako, da bo v enem stavku dobil vse informacije (število posode, velikost posode), saj bo drugače posamezen podatek izpisan določeno število sekund, nato pa bo izginil, kar je lahko pri igranju moteče. To bomo lahko dosegli z združevanjem nizov, združen stavek pa bo tako kombinacija besedila in referenc na spremenljivke z vrednostmi. Nize lahko združimo z uporabo bloka “poveži”, ki ga najdemo v skupini ukazov “Operatorji”. Blok lahko razširimo tako, da pritiskamo na puščice na desni strani bloka, v konkretnem primeru bomo potrebovali pet prostorčkov.

[8. korak]

Združen niz nato postavimo v blok “vprašaj [besedilo] in čakaj”, da na ta način pozovemo uporabnika za vnos in ga dobimo v program. Odgovor se bo shranil v blok “odgovor”, ki ga najdemo v skupini ukazov “Zaznavanje”.

[9. korak]

Potem, ko igralec odgovori, moramo preveriti, če je odgovor pravilen. Učencem pojasnimo, da gre v tem primeru za situacijo z dvema možnostma, odgovor je namreč lahko ali pravilen, ali pa napačen. V teh situacijah pa uporabimo blok “če-sicer”. Skupaj z njimi premislimo kako bomo ugotovili ali je odgovor pravilen. Pravilen odgovor se izračuna tako, da med seboj zmnožimo vrednosti: število

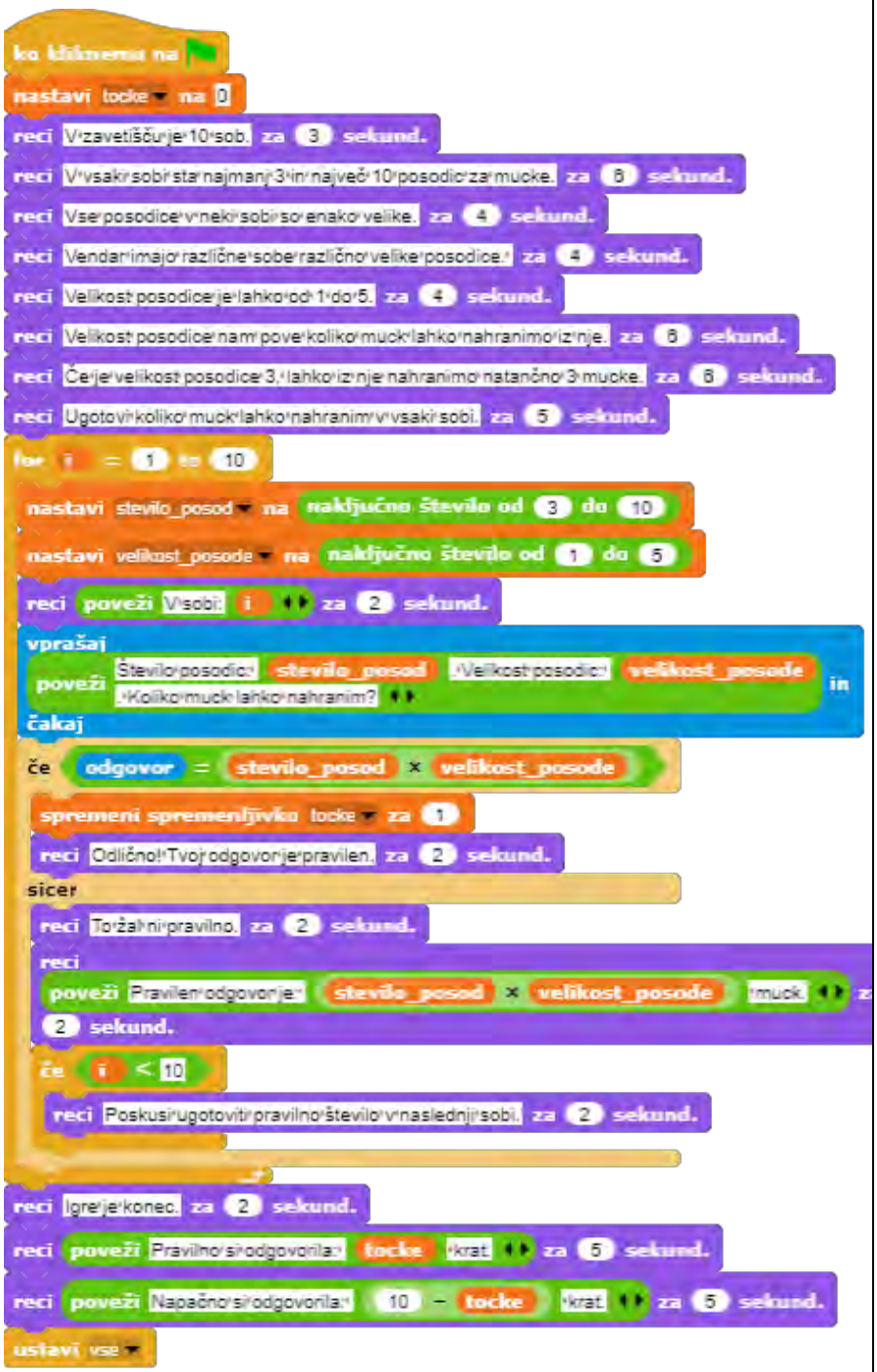
posod in velikost posode. Če je igralčev odgovor enak tej vrednosti, potem je pravilen, drugače pa ne. V primeru, da je pravilen, moramo vrednost spremenljivke, kjer hranimo pravilne odgovore, povečati za ena in podati ustrezno povratno informacijo v obliki besedila. V primeru nepravilnega odgovora pa damo samo povratno informacijo, saj takrat vrednosti števca pravilnih odgovorov ne spreminjamo.

[10. korak]

V tem koraku se odločimo za ustrezno zanko. Če želimo šteti sobe, je najboljša možnost uporaba for zanke, saj lahko vrednost spremenljivke “i”, ki šteje ponovitve zanke uporabimo pri štetju sob.

[11. korak]

Ko se izvajanje zanke zaključi je igre konec. Takrat posredujemo povzetek o uspešnosti igralca. Število pravilnih odgovorov imamo shranjeno v spremenljivki, število napačnih pa lahko izračunamo tako, da vsem poskusom, ki jih je deset, odštejemo število pravilnih odgovorov.

	[Končna koda] 
Orodja in viri za učitelja	• Celotna aktivnost v orodju Snap!: https://snap.berkeley.edu/project?user=zapusek&project=nahrani_mucke • Lažja rešitev brez for zanke:

	https://snap.berkeley.edu/project?user=zapusek&project=nahrani_mucke_spremenjeno • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Viri/gradiva za učence	• Predloga s slikovnim materialom: https://snap.berkeley.edu/project?user=zapusek&project=nahrani_mucke_predloga

Učni scenarij 11 – Mačje zavetišče

Naslov učnega scenarija	Mačje zavetišče
Pričakovano programersko predznanje	• pogojni stavki (if blok) • izpis besedila (blok reci)
Učni cilji	Splošni učni cilji: • naključna števila • določanje vrednosti spremenljivkam • uporabnikov vnos • zanka “ponavljaj dokler” • pogojni stavki • operatorji za primerjanje • števec Specifični učni cilji, ki so usmerjeni na algoritmično mišljenje: • učenec zna prirediti vrednost spremenljivki, • učenec zna uporabiti blok “vprašaj [besedilo] in čakaj”, da z njim pridobi uporabnikov vnos, • učenec zna uporabiti zanko “ponavljaj dokler”, s katero ponavljajoče sprašuje uporabnika za vnos in nato vrednost vnosa tudi preveri,

	• učenec zna uporabiti pogojni stavek v povezavi z operatorji za primerjanje, da preveri pravilnost vnosa in poda ustrezen odziv, • učenec zna nastaviti izhodni pogoj zanke “ponavljaj dokler”, da z njim preverja, če je igre konec, • učenec se zaveda, da ni potrebno posebej preverjati, če je igre konec, saj je preverjanje implicitno vključeno v pogoj zanke, • učenec zna implementirati števec, ki šteje ugibanja in uporabiti njegovo končno vrednost za razlikovanje med dvema različnima izidoma igre.
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Izdelajte igro, v kateri bo igralec uganjeval naključno določeno število od 1 do 100. Igralec bo uganjeval tako, da bo vpisoval vrednosti in dobil povratno informacijo o tem ali je trenutno vpisano število: večje, manjše ali enako številu, ki ga mora uganiti. Naloga: Izdelajte igro, v kateri si bo oskrbnica zavetišča Marta na začetku naključno določila število mačk v njenem zavetišču, vprašala igralca po imenu in mu razložila navodila igre. Nato naj igralca pozdravi z vpisanim imenom in začne s ponavljajočim spraševanjem o številu mačk. Ko igralec vpiše svoj poskus, mu naj poda ustrezno povratno informacijo: 1) če je vpisano število manjše, naj reče: “v zavetišču je več mačk”, 2) če je vpisano število večje, naj reče: “v zavetišču je manj mačk”, 3) če je vpisano število pravilno, pa: “čestitke, uganili ste pravilno število mačk v zavetišču”. V igro vključite tudi števec poskusov, ko bo igralec število ugotovil pa preverite njegovo vrednost. Če je vrednost števca manj ali enako 5, naj oskrbnica reče: “Odlično ti je šlo, za nagrado si lahko izbereš eno.”, v drugem primeru pa: “Poskusi še enkrat.” **Cilj: Učenci se bodo seznanili z zanko “ponavljaj dokler” in kako lahko z nastavljanjem izhodnega pogoja ugotovijo, da je igre konec.**

	Prav tako se bodo naučili uporabljati spremenljivko v dveh različnih situacijah: kot števec in za beleženje igralčevega vnosa.
Trajanje aktivnosti	45 minut
Učne strategije in metode	aktivno učenje, kolaborativno učenje, reševanje problemov
Učne oblike	frontalni način podajanja snovi individualno delo/delo v parih/skupinsko delo
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Oskbnica mačjega zavetišča Marta želi, da uganeš število mačk, ki jih ima v svojem zavetišču. V zavetišču ima vedno vsaj eno mačko in nikoli več kot sto. Ko igralec vpiše svoj poskus mu pove, ali je vpisano število: manjše, večje ali enako uganjevanemu številu. Če igralec ugane število mačk v petih ali manj poskusih dobi za nagrado mačko, drugače pa ga oskrbnica pozove, da igro odigra še enkrat. [1. korak] Prva naloga, ki jo damo učencem je, da izdelajo zanimivo ozadje za igro. Učenci ga lahko narišejo sami, ali pa uporabijo slike iz spleta, pri čemer jih opozorimo oz. naučimo kako najdejo takšne, ki imajo ustrezno licenco. Če želimo prihraniti na času, jim sliko ozadja pripravimo vnaprej v predlogi.

[2. korak]

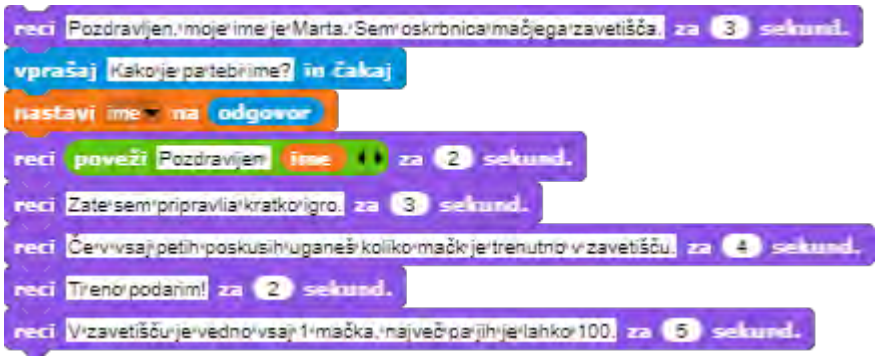
Učenci naj privzetemu objektu (želvici) določijo nov izgled, ki bo predstavljal oskrbnico mačjega zavetišča.

[3. korak]

Z učenci se pogovorimo, da bo igra bolj zanimiva za igranje, če se bo število mačk za vsako igro določilo naključno, saj bi drugače bila igra smiselna le enkrat. Naključno vrednost bomo uporabili pri primerjanju dejanske z vpisano vrednostjo, zato si jo moramo shraniti v spremenljivko. Spremenljivka je namreč (predpostavljamo, da še ne poznajo koncepta seznama) edini način, da si vrednost zapomnimo v programu Snap!. Določitev naključne vrednost se mora zgoditi preden začne igralec ugibati, skratka na začetku programa. Za ime spremenljivke izberemo nekaj mnemoničnega, da bomo že iz imena vedeli pomen vrednosti, ki je shranjena v njej.

[4. korak]

Skrbnica zavetišča vpraša igralca po imenu, zato da jo bo lahko pozdravila. To lahko naredimo z uporabo bloka *vprašaj [besedilo] in čakaj*, ki ga najdemo v skupini *Zaznavanje*. Igralčev odgovor se avtomatično shrani v spremenljivko *odgovor*, ki je ne ustvarimo sami, ampak je že privzeto v programu. S pomočjo odgovora, ki je shranjen v tej spremenljivki in bloka za povezovanje nizov združimo igralčevo ime s pozdravom. Za izpis besedila uporabimo blok *reci [besedilo] za [n] sekund*. Zaporedje teh blokov uporabimo tudi za podajanje navodil. Učence spomnimo, da je trajanje izpisa besedila povezano z njegovo dolžino in da je potrebno to smiselno nastaviti.



[5. korak]

Z učenci se pogovorimo, da ni mogoče predvideti kolikokrat bo igralec uganjeval, da bo ugotovil pravilno število. Lahko se mu posreči v prvem poskusu, lahko, da jih bo potreboval pet, tega ni mogoče predvideti. Pomembno je, da učenci ugotovijo, da gre pri tem problemu za situacijo, kjer potrebujejo ustrezno zanko. Tako, kjer bo njeno ponavljanje vezano na pogoj in ne na vnaprej določeno število

ponovitev, t.j. zanka *ponavljaj, dokler <pogoj>*. Pogoj lahko relativno očiten, saj se bo igra izvajala dokler ne bo igralec ugotovil pravilnega števila oz. dokler ne bo število, ki ga je vpisal enako številu mačk.

[6. korak]

Nato moramo ugotoviti kateri ukazi se bodo ponavljali in jih bomo postavili v telo zanke. Torej kaj se bo ponavljalo dokler igralec ne bo ugotovil števila? Najprej ga bomo morali pozvati k ugibanju, nato pa se bomo morali glede na vpisano vrednost ustrezno odzvati.

[7. korak]

Zadnja stvar, ki se jo moramo z učenci pogovoriti je, kdaj bo zanka zaključila s ponavljanjem in kaj lahko iz tega sklepamo. Pa si podrobneje poglejmo kaj se bo zgodilo, ko bo igralec vpisal pravilen odgovor. Takrat bosta oba pogoja v telesu nepravilna, zato bo zanka šla v ponovno preverjanje pogoja, da se odloči ali se bo ponovila še enkrat. Pogoj v glavi zanke bo takrat izpolnjen, kar pomeni, da se bo zaključila. Ukazi, ki sledijo zanki se bodo tako izvedli le takrat, ko bo igralec vpisal pravilno število oz. povedano drugače, ko bo igre konec. Na to pa se ustrezno odzovemo.

	[8. korak] Če želimo šteti ugibe, moramo ustvariti spremenljivko v kateri bomo hranili to vrednost in jo na začetku programa nastaviti na 0. Z učenci se pogovorimo o pomenu prirejanja začetne vrednosti spremenljivke (inicializaciji) in o razliki med nastavljanjem in spreminjanjem vrednosti spremenljivke. V prvem primeru se vrednost nastavi na neko vrednost, prejšnja se na ta način izgubi, v drugem pa gre za relativno spreminjanje vrednosti, saj se trenutna vrednost spremeni za tisto, ki smo jo določili. V tej situciji bi radi števcu ugibov prištevali 1, za vsakič, ko bo igralec ugibal. Tako bomo morali zunaj zanke najprej nastaviti vrednost na 0, nato pa ga znotraj zanke povečevati za 1, ko bo na novo ugibal. [9. korak] Igralec dobi nagrado, če je uganjeval manj ali enako petkrat. To moramo preveriti po tem, ko je končal z uganjevanjem oz. takrat, ko se zanka zaključi. Snap! nima možnosti uporabe operatorja manj ali enako, zato uporabimo operator manj in preverjamo, če je bilo poskusov strogo manj od 6. Primer odločanja o nagradi je dober primer uporabe pogojne stavka if-else, saj imamo natanko dve možni situaciji - nagrado je dobil ali pa je ni dobil.

	[Končna koda] 
Orodja in viri za učitelje	• Celotna aktivnost v programu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=macje_zavetisce_SLO • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.

Viri/gradiva za učence	• Predloga s slikami: https://snap.berkeley.edu/project?user=mateja&project=C4G11_MacjeZavetisce-predloga

Napredni učni scenariji

Učni scenarij 12 – Lovljenje zdrave hrane

Naslov učnega scenarija	Lovljenje zdrave hrane
Pričakovano programersko predznanje	Dodajanje besedila za lik Prikazovanje in skrivanje lika Uporaba ukaza obrni se v smeri Uporaba naključnega števila Uporaba spremenljivk za štetje točk Uporaba zanke ponovi Uporaba zanke za vedno Uporaba pogojnega stavka
Učni cilji	Splošni učni cilji: • Spremenljivke • Pogojni stavek • Zanka • Obrni se v smeri • Naključno število Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporabi spremenljivko za preprečevanje prezgodnjega začetka igre (opcijsko) • Učenec zna uporabiti pogojni stavek (v kombinaciji s spremenljivko) za začetek igre • Učenec uporabi *zanko ponovi dokler pogoj ni izpolnjen* za premikanje hrane • Učenec uporabi *obrni se v smeri 180 stopinj* (dol) za premikanje likov navzdol • Učenec uporabi naključno število za naključno hitrost premikanja likov hrane • Učenec uporabi naključno število za pomikanje likov hrane na naključno pozicijo

	• Učenec uporabi naključno število za pomikanje likov hrane na naključno x pozicijo in določeno y pozicijo (opcijsko)
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Dekle lovi hrano tako, da klika z miško nanjo. Pri tem mora biti pozorna, saj samo zdrava hrana prinaša točke! Naloge: Učenci morajo sprogramirati dva različna lika: dekle, ki poda navodila, in hrano, ki pada z vrha enkrana (vsa hrana ima skoraj enako kodo, le pri štetju točk se malenkost razlikujeta kodi za zdravo in nezdravo hrano). Dodatno lahko učenci dodajo spremenljivko in pogoj za nadziranje začetka igre. **Cilji: Učenci se bodo naučili premikanja za naključno število korakov, premikanja na naključno pozicijo ter uporabe spremenljivke in pogojnega stavka za preprečevanje dogodkov.**
Trajanje aktivnosti	45 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Individualna učna oblika / Delo v paru
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Dekle lovi hrano. Zdrava hrana prinese 1 točko, medtem ko nezdrava hrana odšteje 1 točko. Na začetku se pojavi dekle, ki poda navodila. Nato se dekle skrije in pojavi se hrana. Hrana pada iz vrha, dokler igralec ne doseže 5 točk. Na koncu se ponovno pojavi dekle in zaključi igro.

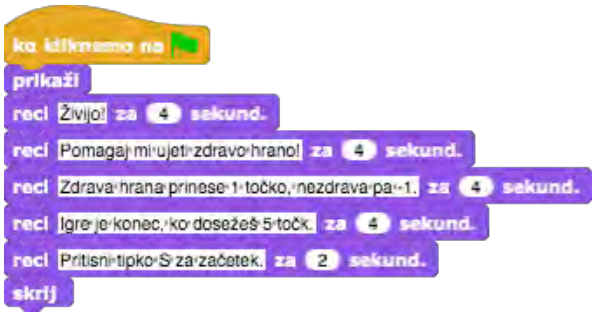
[Korak 1]

Aktivnost je namenjena individualnemu delu ali delu v paru. Učitelj pomaga z namigi, pojasni težje dele in po potrebi pomaga.

Na začetku je učencem na voljo:

- Ozadje
- Lik dekleta

Učenci lahko ozadje in lik dodajo / zamenjajo. Dekle na začetku poda navodila in nato izgine. Kot smo videli pri prejšnjih aktivnostih, moramo na začetku dekle najprej prikazati (če ostane skrita iz prejšnjih poskusov igranja). Primer kode:

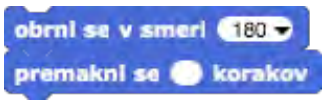


H kodi dekleta se bomo še vrnili kasneje.

[Korak 2]

Dodamo nov lik zdrave hrane, npr. jabolko.

Najprej napišemo kodo za premikanje navzdol, zato uporabimo naslednja ukaza:



Ker ne želimo, da je jabolko narobe obrnjeno, kliknemo na ukaz *ne vrti*. Prav tako odkljukamo ukaz *vlečljiv*, saj to pomeni, da lika ne moremo premikati v miško (kar bi se med igro lahko zgodilo).

Da bo igra bolj zanimiva, nastavimo število korakov na naključno vrednost, kar pomeni, da bo hitrost premikanja poljubna, npr:

Nato sledi razmislek, kaj se zgodi z jabolko, ko le-ta pride do dna enkrata (če je pred tem ne kliknemo)?

V tem primeru uporabimo ukaz 

v kombinaciji s pogojnim stavkom *če*. Če se jabolko dotakne roba, se bo pomaknilo na naključno pozicijo. Snap! Nam ponuja ukazni blok

Ta ukaz bi naključno izbral x in y in jabolko bi se lahko pokazalo kjerkoli na zaslonu (rdeče pikice na sliki desno).

Če pa želimo, da se jabolko vedno pokaže nekje pri vrhu ekrana, lahko y vrednost določimo sami, x vrednost pa naj bo izbrana naključno (slika desno).

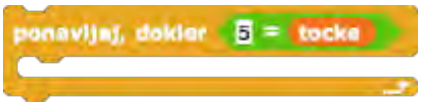
S spodnjo kodo se bo torej jabolko vedno pojavilo na vrhu ekrana:

[Korak 3]

Sedaj bomo ustvarili spremenljivko, *tocke*, s katero bomo šteli pobrano hrano. Spremenljivko na začetku postavimo na 0 (na liku dekleta).

[Korak 4]

Če želimo, da se jabolko nenehno premika, moramo uporabiti zanko. Uporabimo zanko *ponovi dokler* in izberemo pogoj. Če npr. želimo, da je igre konec, ko igralec doseže 5 točk, se bo zanka izvajala dokler pogoj ni izpolnjen. Ko igralec doseže 5 točk, se zanka ustavi.



[Korak 5]

Na začetku ne želimo, da je jabolko prikazano, ampak želimo, da se prikaže, ko dekle konča z navodili, torej s klikom na tipko S. Uporabimo torej ukaz ko pritisnemo na tipko, nato se jabolko prikaže, ko dosežemo 5 točk pa se skrije. Koda trenutno izgleda tako:

[Korak 6]

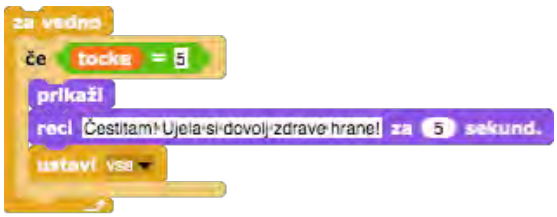
Kaj se pa zgodi, ko kliknemo na jabolko?

Jabolko se mora skriti, prišteti moramo točko, nato se jabolko premakne na naključno pozicijo in ponovno prikaže. Število točk se bo povečalo za 1, koda za pozicijo pa je enaka prejšnji.

[Korak 7]

Vrnimo se h kodi za dekle.

Dekle se mora na koncu spet prikazati in reči npr. »Čestitam Ujel/a si dovolj zdrave hrane!« Za izvedbo le-tega bomo potrebovali *zanko za vedno*, ki bo konstantno preverjala, ali je igralec dosegel 5 točk. V primeru, da je igralec dosegel 5 točk, se dekle najprej prikaže, nekaj pove, nato pa izvajanje končamo z ukazom *ustavi vse* (brez tega ukaza bo dekle ponavljalo »Čestitam! ...« za vedno).



[Korak 8]

S ponovnim igranjem bodo učenci že poznali navodila (iz [Korak 1]) in nedvomno jih bodo želeli preskočiti. S pritiskom na tipko S bo hrana že začela padati, dekle pa bo že vedno govorilo.

Da to preprečimo, si lahko pomagamo z drugo spremenljivko (imenujno jo *start*), ki jo na začetku nastavimo na 0, ko dekle poda navodila, pa jo nastavimo na 1.

Sedaj moramo nastaviti še, da se koda jabolka začne izvajati pod pogojem, da je vrednost spremenljivke *start* enaka 1.

S ponovnim igranjem igre se lahko zgodi tudi, da igro ustavimo, ko imamo 3 točke in jabolko ne bo izginilo. Ko bomo ponovno kliknili na zeleno zastavico bo jabolko še vedno vidno, dekle pa bo podajalo navodila. Zato dodamo še kodo, da se jabolko na začetku igre najprej skrije.

Koda jabolka sedaj izgleda tako:

[Korak 9]

Sedaj lik jabolka podvojimo poljubno krat in likom zamenjamo obleko. Koda bo enaka. Edina razlika bo pri nezdravi hrani, saj bo igralec s klikom nanjo 1 točko izgubil.

[Končna koda]

Dekle

Jabolko

	[Dodatne naloge] Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom: • Igro spremeni tako, da bo igralec hrano lovil s premikanjem sklede. • Dodaj nov lik (skledo). Lik lahko narišeš, najdeš na spletu (pazi na pravice) ali pa uporabiš priloženo sliko sklede. • Skledi nastavi ji začetno pozicijo (npr. na dnu ekrana) in naredi njeno premikanje levo in desno (po želji lahko tudi gor in dol). Živila morajo sedaj izginiti in se ponovno pojaviti na naključni lokaciji z dotikom sklede (in ne s klikom na živilo kot do sedaj). • Spremeni pravila, da se igra konča, ko igralec doseže 20 točk (zmaga) ali ko pobere 3 nezdrava živila (izgubi). • Dodaj več živil, da bo igra zanimivejša. • Spremeni obleko liku skleda, ko igralec doseže npr. 5, 10, 15 točk.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G12_LovljenjeZdraveHrane • Primer dodatne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G12_LovljenjeZdraveHrane%20%2B%20Dodatek • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G12_LovljenjeZdraveHrane - Delno • Navodila za učenca (C4G12_NavodilaZaUcenca.docx)

Učni scenarij 13 – Sestavi zgodbo

Naslov učnega scenarija	Sestavi zgodbo
Pričakovano programersko predznanje	Prikazovanje in skrivanje lika Uporaba pogojnega stavka Dodajanje besedila za lik Uporaba ukaza Menjava ozadja
Učni cilji	Splošni učni cilji: • Premikanje lika in spreminjanje njegove velikosti • Pošiljanje in prejemanje obvestil • Struktra zgodbe Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učencec načtruje dialoge in aktivnosti likov v zgodbi • Učencec pošilja obvestila za izvedbo dialogov med liki • Učenec uporabi premikanje in spreminjanje velikosti likov • Učenec uporabi skrivanje in prikazovanje lika
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Zajec pove zgodbo o Alici v čudežni deželi. Zgodba se začne v gozdu. Alica hodi in se sprašuje »Kje sem?« /Za realizacijo njenega premikaja stran, se zmanjšuje velikost lika/. Alica pride do razpotja in tam vidi mačko na drevesu. Začne se pogovor med Alico in mačko. Pogovor je predsavljen na sliki.  Naloge: Učenci eksperimentirajo s kratkim primerom iz zgodbe (SestaviZgodboAlice1) o srečanju med Alico in mačko, ki temelji na sinhronizaciji dialoga z uporabo ukaznega bloka

	čakaj _ sekund. Nato si pogledajo drugo verzijo (SestaviZgodboAlice2) z uporabo pošiljanja obvestila. Učenci dokončajo zgodbo. Za pomoč jim je zgodba / scenarij ter napisani dialogi (v dveh tabelah na koncu dokumenta). Po želji lahko dodajo / spreminjajo zgodbo. **Cilji: Učenci se bodo naučili načrtovanja zgodbe, uporabe obvestil za sinhronizacijo aktivnosti likov in menjane ozadja.**
Trajanje aktivnosti	90 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Individualna učna oblika / Delo v paru / Skupinska učna oblika
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Učitel/ica se z učenci pogovori o zgodbici Alice v čudežni deželi in jim pokaže sliko Alice, ki sreča mačko. Učencu si pozorno pogledajo kodo aktivnosti v Snap!-u - SestaviZgodboAlice1: https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice1. Diskusija: Kdo spregovori prvi? Kdaj začne Alica in kdaj mačka? Zakaj ni sinhronizacije v dialogu? Odgovor leži v napačnem izračunu, kdaj kateri lik *spregovori* in *kdaj naslednji nadaljuje* z govorom.

Skupaj si ogledamo kodo in tabelo:

Lik	Aktivnost	Kdaj začne	Kdaj konča	Trajanje
Zajec	Reče: SI že slišal/a za Alice v čudežni deželi? Poglejmo si eno od njenih zgodbic!	0	14	14
Alica	Reče: *"Mi lahko prosim poveš, kam moram iti?"*	9	21	12
Mačka	Reče: *"To je odvisno od tega, KAM želiš priti!"*	10	20	10

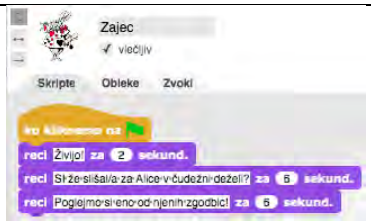
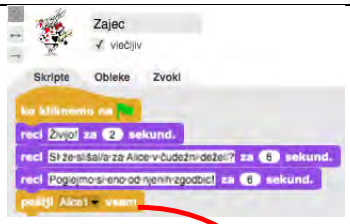
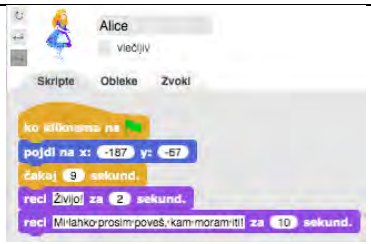
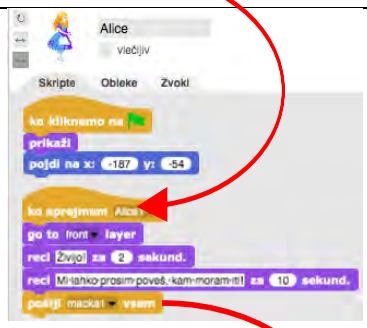
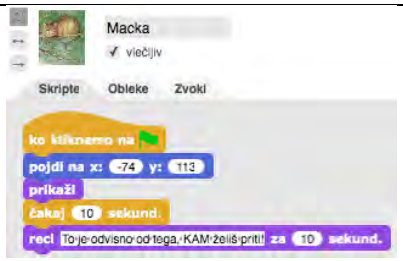
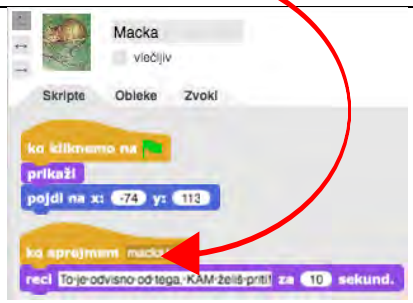
Kot lahko vidimo, je sinhronizacija dialogov z uporabo ukaza *čakaj __ sekund* zelo zakomplicirana in lahko vodi do številnih napak v zgodbi.

Učitelj/ica pokaže nekoliko drugačno kodo aktivnosti v Snap!-u – SestaviZgodboAlice2:

https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice2.

Kateri nepoznani ukazi se pojavijo?

Primerjajmo kodi Alice1 in Alice2:

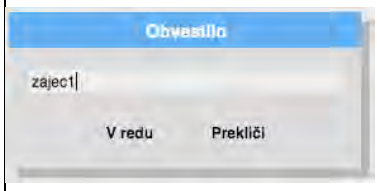
Alice1	Alice2
	
	
	

Predstavimo ukaze za pošiljanje obvestil:

Uporabljali bomo prvi ukaz, s katerim obvestilo pošljemo, in tretji ukaz, s katerim obvestilo prejmemo in izvajamo kodo naprej.

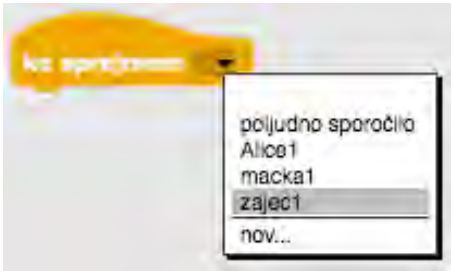
Učencem pokažemo, kako ustvarimo obvestilo.

Izberemo ukazni blok *pošlji* __ *vsem* in kliknemo na puščico. Izberemo željeno sporočilo ali kliknemo *nov...*



ter vpišemo *ime* obvestila.

Uporaba dogodka:



Izberemo ukazni blok *ko sprejmem* __, kliknemo na puščico in izberemo željeno obvestilo.

Kar zapišemo pod ta ukazni blok se bo izvedlo, ko bo sporočilo prejeto.

Učenci v parih dokončajo zgodbo z začetne slike. Obvestilo, ki ga mačka pošlje Alici, npr. omenujejo Alice2, obvestilo, ki ga Alica pošlje mački, pa macka1.

Učitelj/ica pripomne, da pripovedovanje zgodb navadno vsebuje tudi menjavo ozadij. *»Dopolnimo zgodobo tako, da zgodbo začne Zajec na ozadju start, nato se ozadje spremeni (gozd), pojavi se Alica, se sprašuje »Kje sem?« ter se oddaljuje in izginja v gozd, nato se ozadje zamenja (v ozadje srecanje), pojavi se Macka in začne se pogovor med Alico in Macko.«*

Učitelj/ica pokaže celotno zgodbo – SestaviZgodboAlice: https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice.

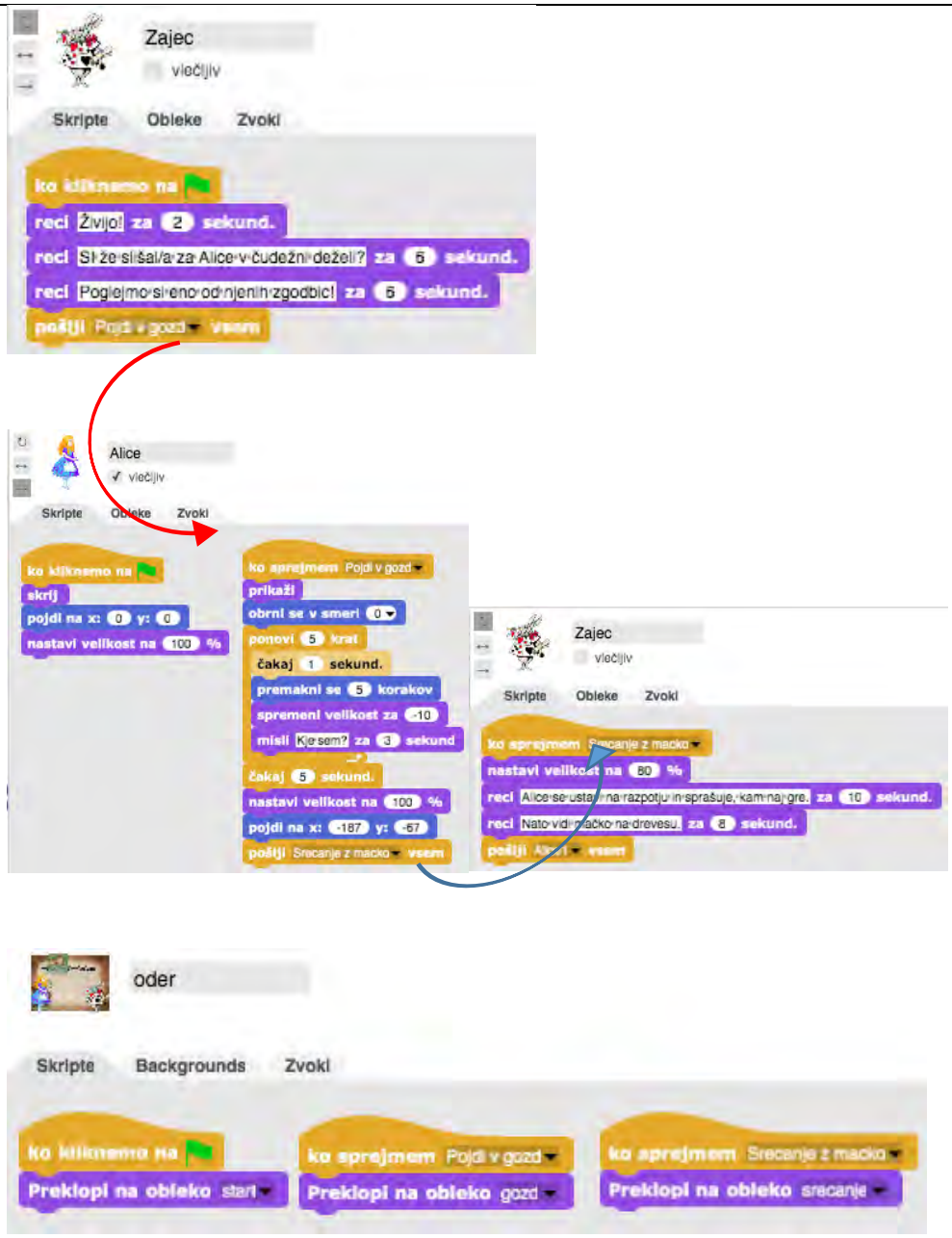
Skupaj komentiramo zgodbo.

»Kdaj se zamenja ozadje? Kdaj se Alica prikaže in kaj počne? Kdaj se pojavi Macka in kaj počne ona?«

Spet si ogledamo celotno aktivnost Alice. Imamo 3 ozadja, od tega je eno v uporabi. S katerim ozadjem začnemo? Kaj moramo narediti, da se Alica in Macka ne pojavita na začetku? Kako zamenjamo ozadje? Ozadje lahko zamenjamo s pošiljanjem obvestila in sicer ko Zajec konča z uvodom. Alica se pojavi, ko se ozadje zamenja z obvestilom *Pojdi v gozd*.

Opozorilo:

Končna aktivnost SestaviZgodboAlice (leva slika) se od delne aktivnosti SestaviZgodboAlice 2 (desna slika) na začetku razlikuje, zato je tudi prvo obvestilo, ki ga pošlje Zajec, nekoliko drugačno, saj je dodana scena z gozdom:



Ko je Alica na poti v gozd in se sprašuje, kje je, se njena velikost spreminja z uporabo ukaznega bloka *spremeni velikost za -10* ter *uporabo zanke*.

	Nato se spremeni ozadje z obvestilom *Srecanje z macko*. Obvestilo prejme tudi Zajec, ki se zmanjša na 80% in nadaljuje s pripovedovanjem. Macka tu ni prikazana, saj je del ozadja. Prikaže se z obvestilom *macka1*. Povemo lahko, da je bil lik mačke izrezan iz ozadja z drugim programom, kajti Snap! tega na žalost ne omogoča. Ko zajec konča in pošljemo obvestilo *Alice1*, se zgodba nadaljuje kot v aktivnosti SestaviZgodboAlice2. Učitelj/ica komentira, da moramo pri pripovedovanju in sestavljanju zgodb najprej izdelati scenarij. V nadaljevanju se nahaja tabela, ki bo pri tem v pomoč. Tabelo lahko po želji spremeni / doda scene učitelj ali pa učenci. Učenci v parih dokončajo scenarij in zgodbo v Snap!-u. Nadaljujejo z aktivnostjo SestaviZgodboAlice2.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Začetni primer v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice1 • Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice2 • Navodila za učenca, ki vsebuje tudi spodnji tabeli z zgodbo/scenarijem in liki (C4G13_NavodilaZaUcenca.docx)

Zgodba / scenariji

Ime	Ozadje	Akcija	Notes
1. start		Zgodba se začne s prizorom (ko kliknemo na zeleno zastavico).	Na tem ozadju Zajec predstavi zgodbo.
2. gozd		Ta scena se pojavi, ko Zajec pove uvodni del (ko program sprejme obvestilo *Pojdi v gozd*).	Alica se pojavi na sredini zaslona. Začne se premikati in se sprašuje »*Kje sem?*«. Lik postopoma zmanjšuje svojo velikost (5x za -10). Ko pride do konca poti (do razpotja), se scena spremeni na *srecanje*. Alica pošlje obvestilo *Srecanje z macko*.
3. srecanje		Ta scena se pojavi, ko program sprejme obvestilo *Srecanje z macko*.	Ko se prizor spreminja, Zajec nadaljuje s pripovedovanjem zgodbe. Kasneje poteka pogovor med Alico in Macko.

Liki

Lik	Akcija	Ozadje
	Na začetku: Reče: Živijo! (za 2 sek.) Reče: SI že slišal/a za Alico v čudežni deželi? (za 6 sek.) Reče: Poglejmo si eno od njenih zgodbic! (for 6 sec.) Pošlje obvestilo *Pojdi v gozd*.	 start
	Na začetku: Se skrije. Pripravimo jo na novo ozadje (*gozd*), tako da gre na x: 0, y: 0 in nastavi velikost na 100%.	 start
	Na začetku: Se skrije. Pripravimo jo na novo ozadje (*srecanje*), zato gre na x: -74, y: 113.	 start
	Prejme obvestilo *Pojdi v gozd*: Lik se prikaže. Obrne se v smeri 0. 5x se ponovi: čakaj 1 sekundo, premakni se 5 korakov, spremeni velikost za -10, misli si »*Kje sem?*«. Priprava na novo ozadje: čakaj 5 sekund, nastavi velikost na 100% in pojdi na x: -187, y: -67. Pošlje obvestilo *Srecanje z macko*.	 gozd
	Brez akcije. Ostane prikazan od prej.	 gozd

	Prejme obvestilo *Srecanje z macko*. Nastavi veliksot na 80% Reče: *"Alica se ustavi na razpotju in sprašuje, kam naj gre."* (za 10 sek.). Reče: *" Nato vidi mačko na drevesu."* (za 8 sek.) Pošlje obvestilo *Alice1*	 srecanje
	Prejme obvestilo *Alice1*. Postavi se v ospredje - blok *go to front layer* (drugače se lahko zgodi, da se lik Macke pojavi čez oblak in ni vidno, kaj Alica govori). Reče: *"Živijo!"* (za 2 sek.) Reče: *"Mi lahko prosim poveš, kam moram iti?"* (for 10 seconds). Pošlje Macki obvestilo *macka1*.	 srecanje
	Prejme obvestilo *macka1*. Lik Macke se pojavi na zaslonu. Reče: *" To je odvisno od tega, KAM želiš priti!"* (za 10 sekund). Pošlje obvestilo *Alice2*.	 srecanje
	Prejme obvestilo *Alice2*. Reče: …… Pošlje obvestilo *macka2*.	 srecanje
	Prejme obvestilo *macka2*. Reče: …… Pošlje obvestilo *zajec1*.	 srecanje
	Prejme obvestilo *zajec1*. Reče: “Kaj je morala zgodbe?” (za 8 sek.) Reče: “Če želiš vedeti, kam naprej, moraš najprej določiti svoj cilj.”	 srecanje

Učni scenarij 14 – Onesnažen zrak

Naslov učnega scenarija	Onesnažen zrak
Pričakovano programersko predznanje	Učenci poznajo in znajo uporabljati bloke za risanje, spreminjati izgled lika, dodajati nove spremenljivke, uporabljati zanke, uporabljati računske operatorje.
Učni cilji	Splošni učni cilji: ● Učenci se spoznajo s konceptom paralelizma v orodju Snap! Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● učenci uporabljajo bloke za risanje; ● učenci uporabljajo zanke za risanje likov (kvadrat, krog); ● učenci uporabijo spremenljivko za štetje točk; ● učenci uporabijo naključna števila za določanje pozicije svinčnika; ● učenci uporabijo sporočila za paralelno izvajanje dogodkov.
Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Učenci sestavijo igro, katere cilj je očistiti onesnažen zrak v okolici industrije. Igralec onesnažen zrak poskuša izboljšati s posaditvijo dreves (smrek in hrastov), pri čemer zbira točke. Ob narisani smreki prejme 2 točki, za narisan hrast pa 3 točke, saj je ta bolj učinkovit pri čiščenju zraka, kot smreka. Igralec ima možnost narisana drevesa izbrisati. V tem primeru se točke izničijo. Po zbranem določenem številu točk (npr. 10) se zrak v okolju industrije izboljša in posledično prične rast zelenje. Ko zrak več ni onesnažen, se igra zaključi. **Naloga:** Učenci naprej definirajo začetek igre, v kateri je v ozadju industrija in na sredini ekrana svinčnik. Nato napišejo kodo za risanje smreke, ki se izriše v primeru, ko igralec klikne na ikono smreke. Podobno sestavi kodo tudi za izris hrasta. V naslednjem koraku učenec sestavi kodo za možnost izbrisa vseh narisanih dreves. Nato doda še

	točkovanje, kjer igralec za vsako narisano smreko prejme 2 točki, za vsak hrast pa 3. Učenec mora nato določiti, koliko točk mora igralec doseči (npr. 10), da se igra zaključi. Učenec mora dodati kodo tako, da se ozadje spremeni, ko igralec doseže določeno število točk. **Cilji:** Učenci se spoznajo sm konceptom paralelizma. Učenci razumejo v skripte katerih likov je potrebno dodajati kodo. Za paralelno izvajanje dogodkov v skriptah različnih likov uporabljajo boke za pošiljanje in sprejemanje sporočil.
Trajanje aktivnosti	45 minut
Učne strategije in metode	učenje z igro, učenje z ustvarjanjem igre, problemsko učenje
Učne oblike	Frontalna oblika, delo v paru
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) **Motivacija-uvod** Učencem podamo v igranje že ustvarjeno igro, da postanejo motivirani za ustvarjanje svoje. Ob igri se pogovorimo o konceptih, ki jih bomo spoznali skozi sestavljanjem igre. Na primeru jim frontalno pokažemo uporabo blokov za pošiljanje in sprejemanj sporočil. Učenci lahko predlagajo nadgradnjo igre, ki jo lahko tudi realizirajo na koncu šolske ure. **Implementacija** 1. Učenci odprejo predlogo programa, v kateri sta dodani dve ozadji – industrija in travnik ter liki, kot so svinčnik za risanje dreves, križec z namenom brisanja narisanih dreves, ikoni smreke in hrasta.

2. Na začetku morajo učenci najprej dodati kodo tako, da bo na začetku igre v ozadju industrija in svinčnik postavljen na sredini ekrana. Pri tem je potrebno paziti, da je koda dodana v primerno skripto.

Skripta ozadja:

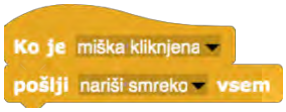
Skripta svinčnika:

Tu gre za osnovno znanje programiranja v orodju Snap!, zato ta korak učenci opravijo samostojno.

3. V nadaljevanju morajo učenci dodati kodo za risanje smreke. Pri tem morajo upoštevati, da se smreka izriše po tem, ko igralec klikne na ikono smreke.

Torej, ko igralec klikne s miško na ikono smreke, mora svinčnik prejet sporočilo naj nariše smreko.

Skripta smreke:



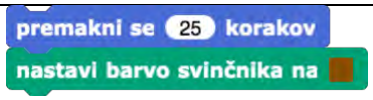
Skripta svinčnika:

Po tem, ko svinčnik prejme sporočilo, mora narisati smreko. Učenec mora pri tem matematično razmisliti, kako narisati smreko, kot je prikazana na sliki.

Pri risanju uporabljamo bloke svinčnika. Da svinčnik spušča za sabo sled, je potrebno najprej svinščnik spustit. Nato dodamo kodo za risanje krošnje, kjer je potrebno najprej nastaviti barvo svinčnika in smer svinčnika v desno stran.

Sledi risanje krošnje v obliki trikotnika. S pomočjo zanke narišemo tri stranice trikotnika tako, da se svinčnik najprej pomakne za 40 korakov (lahko več ali manj, odvisno kako velike krošnje želimo) in nato obrne v levo za 120° (prileżni kot notranjega kota enakostraničnega trikotnika). To ponovi 3-krat.

Sledi risanje debla v obliki kvadrata. Ker je svinčnik končal z risanjem v levem kotu, ga je potrebno premakniti za 25 korakov (da bo deblo na sredini - stranica je dolga 40 korakov, 10 korakov pa bo široko deblo) v desno, kjer bo začel z risanjem debla. Svinčniku nastavimo rjavo barvo.



Potem z zanko narišemo tri stranice kvadrata (4. stranico ni potrebno risati po krošnji) tako, da svinčnik najprej obrnemo za 90° v desno in nato premaknemo za 10 korakov, kar se ponovi 3-krat.

Sedaj, ko je smreka narisana, je potrebno svinčnik dvignit, da ne spušča več sledi.

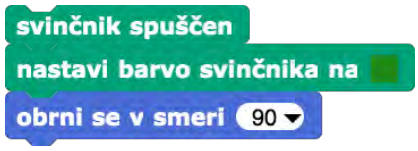
Po vsaki narisani smreki, naj se svinčnik premakne na naključno pozicijo, z namenom, da ne riše eno smreko na drugo.

4. Na podoben način morajo učenci sestaviti kodo za risanje hrasta. Ko igralec klikne na ikono hrasta, prejme svinčnik sporočilo naj nariše hrast.

Skripta hrasta: *Skripta svinčnika:*

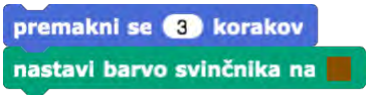
Po tem, ko svinčnik prejme sporočilo, mora narisati hrast. Učenec mora razmisliti, kako pri tem matematično prikazan na narisati hrast, kot je sliki.

Pri risanju uporabljamo bloke svinčnika. Da svinčnik spušča za sabo sled, je potrebno najprej svinščnik spustit. Nato dodamo kodo za risanje krošnje, kjer je potrebno najprej nastaviti barvo svinčnika in smer svinčnika v desno stran.



Sledi risanje krošnje v obliki kroga. Krog narišemo s pomočjo zanke. V tem primeru se zanka ponovi 120-krat, pri čemer se svničnik pomakne za 1 korak in se obrne v levo za 3° (podatki so lahko tudi drugačni, pri tem je potrebno le upoštevati, da se izriše polni krog 360°).

Sledi risanje debla v obliki kvadrata, po podobnem postopku kot pri risanju smreke. V kodi določimo, kje naj začne z risanjem debla (premik za npr. 3 korake) in svinčniku nastavimo rjavo barvo.



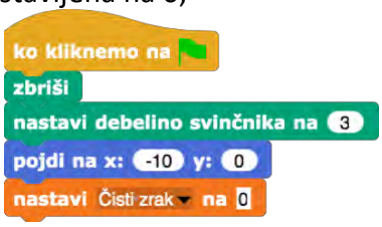
Nato dodamo isto kodo kot pri risanju smreke. Narišemo deblo v obliki kvadrata, prekinemo risanje z dvigom svinčnika in naključno izberemo novo lokacijo svinčnika.

5. Naslednji korak je dodajanje možnosti izbrisa vseh narisanih dreves. Ko je z miško kliknjen križec (lik *Počisti*), ta svinčniku pošlje sporočilo, naj počisti vsa narisana drevesa.

Skripta lika Počisti:

Skripta lika Svinčnik:

6. V naslednjem koraku je potrebno v program dodati kodo, ki bo štela pridobljene točke igralca. Najprej je potrebno dodati novo spremenljivko (→ Spremenljivke → Nova spremenljivka), v kateri se bo shranjevala vrednost doseženih točk. V našem primeru se spremenljivka imenuje *Čisti zrak*. Ko je spremenljivka ustvarjena, jo lahko z bloki za nastavitev spremenljivke dodamo v kodo (v skripti lika *Svinčnik*). Pri tem moramo upoštevati naslednje:
   - na začetku igre mora biti vrednost spremenljivke nastavljena na 0;

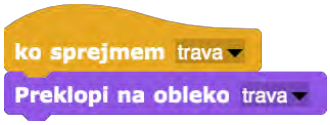


   - vsakič, ko igralec izbere risanje smreke, se vrednost spremenljivke poveča za 2 točki;

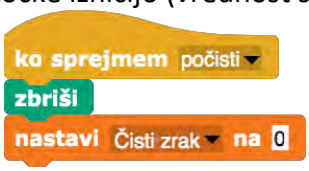
- vsakič, ko igralec izbere risanje hrasta, se vrednost spremenljive poveča za 3 točke;

- ko je vrednost spremenljivke večja od npr. 10, se igra zaključi (spremeni se ozadje);

Skripta lika Svinčnik: *Skripta ozadja:*



- ko igralec izbere možnost brisanja dreves (rdeč križec), se točke izničijo (vrednost spremenljivke je 0).



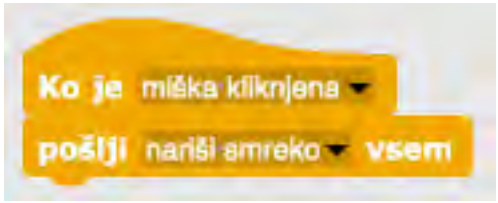
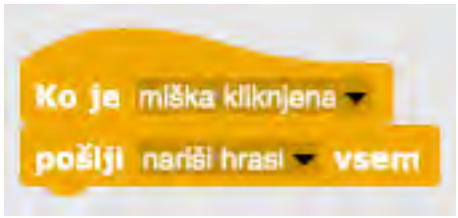
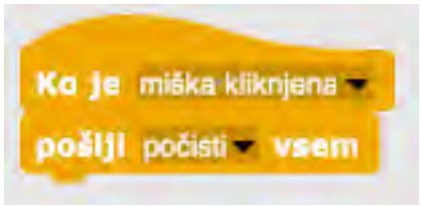
Refleksija in vrednotenje

Učenci svojo igro, ki so jo ustvarili v paru, nadgradijo individualno s svojimi idejami, pri čemer morajo vključiti koncept paralelizma oz. uporabiti bloke za pošiljanje in sprejemanje sporočil. Pri tem upoštevamo, koliko učiteljeve pomoči so potrebovali.

[Končna koda]

Ozadje

Svinčnik

	Smreka  *Hrast*  *Počisti* 
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Onesnaževanje zraka • Lajovic, S. (2011). Scratch. Nauči se programirati in postani računalniški maček. Ljubljana: Pasadena. • Vorderman, C. (2017). Računalniško programiranje za otroke. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Zrak_za%C4%8Detno • Navodila za učenca (C4G14_NavodilaZaUcenca.docx)

Učni scenarij 15 – Ulovi miš

Naslov učnega scenarija	Ulovi miš
Pričakovano programersko predznanje	• dodajanje ozadja; • dodajanje novega lika; • dodajanje zvoka; • govorjenje lika; • spreminjanje obleke lika kot animacija; • premikanje lika s smernimi tipkami; • pogojni stavki.
Učni cilji	Splošni učni cilji: • spoznavanje neskončne zanke; • spoznavanje naključnih števil; • spoznavanje števnika; • spoznavanje časovnika; Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • učenci uporabijo neskončno zanko za premikanje lika; • učenci uporabijo naključna števila za določanje naključne pozicije lika, premikanje lika za naključno število korakov in obračanje lika za naključno število stopinj; • učenci ustvarijo števnik z dodajanjem nove spremenljivke in končni rezultat uporabijo za povzemanje uspešnosti igralca pri lovljenju miši; • učenci uporabijo časovnik za določitev konca igre.
Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Učenci sestavijo igro, v kateri igralec s premikanjem mačke lovi miš. **Naloga:** Naloga učenca je, da sestavi igro lovljenje miši. Mačko premika igralec s smernimi tipkami, miš pa se po sobi premika naključno. Miš vedno mački uide, zato vsakič, ko se mačka dotakne miši, se ta prikaže na naključni lokaciji v sobi. Potrebno je dodati števnik, ki šteje kolikokrat se mačka dotakne miši. Igra se zaključi po določenem

	času (npr. 30 s), zato potrebujemo še časovnik. Na koncu igre, dekle pove kolikokrat je mačka ujela miš. **Cilj:** Učenec se spozna z dodeljevanjem naključnih vrednosti in pri tem se nauči kako uporabiti operator *naključno_število_od_[x]_do_[y].*
Trajanje aktivnosti	45 min
Učne strategije in metode	Aktivni pouk, sodelovalno učenje, problemsko učenje, učenje z ustvarjanjem iger
Učne oblike	Frontalna oblika Delo v parih ali v manjših skupinah
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) **Motivacija-uvod** Učence motiviramo s prikazom igre, ki jo bodo morali tudi sami sestaviti. Ob igri se pogovorimo, kako začeti z ustvarjanjem igre. Skupaj z učenci oblikujemo korake ustvarjanja igre, kot na primer: 1. izberi ozadje in dodaj like; 2. mački sestavi program tako, da se bo premikala s smernimi tipkami; 3. miši sestavi program tako, da se bo premikala naključno; 4. sestavi program tako, da se bo miš ob vsakem dotiku mačke skrila in prikazala na naključni lokaciji v sobi; 5. dodaj števnik, ki bo štel kolikokrat se mačka dodakne miš; 6. dodaj časovnik in določi konec igre; 7. dodaj dekle, ki na koncu igre pove, kolikorat je mačka ujela miš; 8. dekletu sestavi program tako, da bo skočila vsakič, ko se dotakne miši; 9. dodaj zvok miši ali mačke; 10. itd.

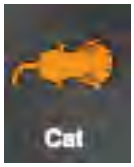
V naslednjem koraku učencem predstavimo blok, ki omogoča določanje naključne vrednosti:

Učenci nato pričnejo z ustvarjanjem igre v parih/manjših skupinah. Učitelj jim je pri tem v podporo in pomoč.

Implementacija

[1. korak]

Prvi korak je dodajanje ozadja. Učenci lahko na spletu poiščejo sliko, ki je prosto dostopna in ima dovoljenje za uporabo. V naslednjem koraku dodajo nova lika – mačko in miš.



[2. korak]

V drugem koraku učenci sestavijo program tako, da se mačka premika s smernimi tipkami. Določiti morajo tudi, kaj se zgodi, če mačka pride na rob ekrana.

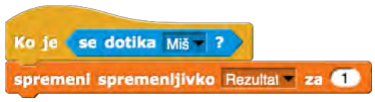
[3. korak]

V naslednjem koraku je potrebno sestaviti program tako, da se miš po sobi premika naključno. V tem primeru je koda sestavljena tako, da miš neskončnokrat (zanka za vedno) premakne za naključno število korakov in se obrne za naključno število stopinj. V program je tako potrebno dodati bloka *Premikanje/premakni_se_[x]_korakov* in *Premikanje/obrni_se_[x]_stopinj*, v katera vstavimo operator *naključno_število_od_[x]_do_[y].*

[4. korak]

Sedaj je potrebno sestaviti program tako, da se miš ob dotiku mačke skrije in prikaže na naključni lokaciji v sobi. V tem primeru miš

	pobegne mački. Učenci lahko tukaj dodajo tudi svoje pravilo igre. V vsakem primeru pa morajo v programu uporabiti operator *naključno_število_od_[x]_do_[y].*  [5. korak] V tem koraku dodamo igri števec, ki bo štel kolikorat mačka ulovi miš oz. kolikokrat se mačka dotakne miši. Učenci morajo najprej ustvariti novo spremeljivko, v našem primeru poimenovano *Rezultat*, in jo dodati v kodo mačke. Pri tem je potrebno paziti, da je spremenljivka *Rezultat* na začetku igre nastavljena na vrednost 0. To naredimo s pomočjo bloka *Spremenljivke/nastavi_[spremenljivka]_na_[x].* V primeru prikazovanja vrednosti spremenljivke *Rezultat* na igralnem ekranu dodamo v kodo blok *Spremenljivke/prikaži_spremenljivko_[spremenljivka].* Vsakič, ko mačka ulovi miš, se rezultat poveča za 1. Za to je potrebno v program dodati še kontrolni blok *Control/when* in blok *Spremenljivke/spremeni_spremenljivko_[spremenljivka]_za_[x]*.



[6. korak]

Učenec določi kdaj bo igre konec. V tem primeru dodamo časovnik oz. štoparico, ki jo najdemo pod zavihkom *Zaznavanje*. Po določenem času (npr. 30 s) se mačka, miš in spremenljivka *Rezultat* skrijejo in igre je konec. Upoštevati je potrebno tudi to, da je štoparica ne začetku igre nastavljena na 0 (*reset štoparice*).

Kodo je potrebno dodati tako v skripto mačke kot miši, saj se po 30 s skrijeta obe.

[7. korak]

Na koncu igre dekle igralcu pove, kako uspešen je bil pri lovljenju miši. V primeru, da igralec ne ulovi nobene miši, mu ta sporoči: »Nisi

ujel/a nobene miši!« V nasprotnem primeru pa mu sporoči: »Čestitamo! Ujel/a si *X* miši.«

[Dodatne naloge]

Učenci lahko igro dopolnijo po svojih idejah. Na primer, lahko dodajo dekle na pojstli, ki skočne vsakič, ko se dotakne miši.

Igri lahko učenci dodajo tudi zvok. Na primer, vsakič ko mačka ujame miš, ta zamjavka.

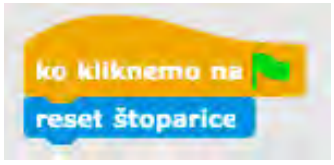
Refleksija in vrednotenje

Učenci prilagodijo svojo kodo:

- mačka se za vedno premika naključno od 20 do 60 korakov;
- ko se miš dotakne mačke, gre na naključno lokacijo, pri čemer je x = 100;
- miš se vedno obrne za 90 stopnij;
- ipd.

[Končna koda]

Mačka

	Miš  *Dekle*  *Ozadje* 
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Ulovi%20mi%C5%A1_1 • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Ulovi%20mi%C5%A1

	• Navodila za učenca (C4G15_NavodilaZaUcenca.docx)

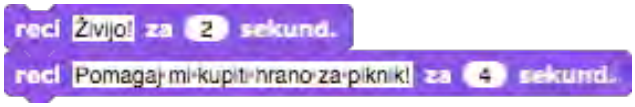
Učni scenarij 16 – Kupovanje hrane za piknik

Naslov učnega scenarija	Kupovanje hrane za piknik
Pričakovano programersko predznanje	Dodajanje besedila za lik Prikazovanje in skrivanje lika Uporaba operatorjev Uporaba spremenljivk Uporaba združevanja nizov Uporaba pogojnega stavka
Učni cilji	Splošni učni cilji: • Spremenljivke • Pogojni stavek • Operatorji Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporablja spremenljivke za določanje cene izdelka • Učenec prireja vrednosti spremenljivke glede na porabo denarja • Učenec uporablja pogojni stavek za preverjanje količine denarja, ki je na voljo • Učenec uporablja operatorje za združevanje nizov (besedilo – vrednost spremenljivke – besedilo) • Učenec uporablja operatorje za primerjavo cene izdelka in denarja, ki je na voljo • Učenec uporablja operatorje za spreminjanje (odštevanje) vrednosti spremenljivke
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Dekle potebuje pomoč pri kupovanju hrane za piknik. Na voljo ima 15 EUR. Ko kupi določen izdelek, se vrednost njene »denarnice« (denarja, ki ga ima na voljo) spremeni. Če je v denarnici premalo denarja, izdelka ne more kupiti. Naloge: Učenci morajo napisati tri različne kode: za dekle, hrano (lik podvojijo z majhno spremembo v kodi) ter gumbom za konec igre. Dekle pove navodila in

	koliko denarja je na voljo, s klikom na gumb konec pa pove še, koliko zdravih in nezdravih izdelkov je igralec kupil. Ko gremo z miško čez določen izdelek, se izpiše njegova cena. Če ima igralec na voljo dovolj denarja, lahko izdelek kupi, v nasprotnem primeru pa ga ne more kupiti. **Cilji: Učenci se bodo naučili delati s spremenljivkami: nastavljanje različne začetne vrednosti, uporaba pogojnega stavka za primerjanje vrednosti spremenljivk, spreminjanje vrednosti spremenljivk z odštevanjem, uporaba spremenljivk za štetje (ne)zdrave hrane. Dodatno, ponovili bodo dodajanje besedila, združevanje besedila in pogojni stavek.**
Trajanje aktivnosti	45 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Individualna učna oblika / Delo v paru
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Dekle v trgovini nakupuje hrano za piknik. Na voljo ima 15 EUR. Ceno izdelka lahko vidi, ko se z miško pomakne čez izdelek, kupi pa ga s klikom na izdelek. Kupi ga lahko le, če ima na voljo dovolj denarja. S klikom na gumb Konec pove, koliko zdravih in nezdravih izdelkov je bilo kupljenih.

[Korak 1]

Aktivnost je namenjena samostojnemu delu učencev ali delu v paru. Učitelj/ica pomaga z namigi, pojasni težje dele in po potrebi pomaga. Učenci izberejo ozadje in dodajo glavni lik, npr. dekle. Dekle na začetku poda navodila:

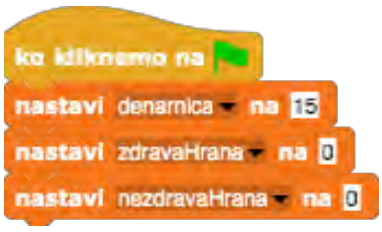


[Korak 2]

V igri bomo potrebovali nekaj spremenljivk:

- *denarnica* bo povedala, koliko denarja je še na voljo,
- *zdravaHrana* bo štela, koliko zdravih izdelkov smo kupili,
- *nezdravaHrana* bo štela, koliko nezdravih izdelkov smo kupili,
- spremenljivko za vsak izdelek, npr. *cenaLubenice*, s katero bomo nastavili ceno vsakega izdelka posebej.

Na začetku nastavimo vrednost spremenljivke *denarnica* na 15 (EUR). Ostali dve spremenljivki nastavimo na 0. Kodo lahko dodamo pred kodo s [Korak 1].



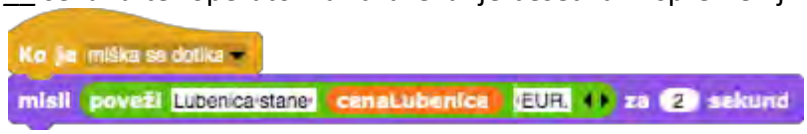
[Korak 3]

Dodamo lik izdelka in izberemo njegovo okleko, npr. lubenico.

Koda lubenice potrebuje 3 dogodke:

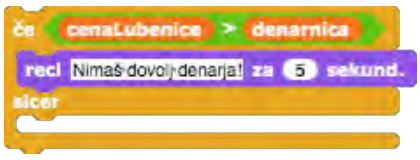
1) *Ko kliknemo na zeleno zastavico*: lik se mora prikazati, nastavimo tudi njegovo ceno, npr.:

2) *Ko je miška se dotika*: igralcu povemo, koliko izdelek stane. Uporabimo blok *misli za __ sekund* ter operator za združevanje besedila in spremenljivke:



3) *Ko je miška kliknjena*: tu je potreben krajši razmislek.
   a. V katerem primeru lahko igralec kupi izdelek in v katerem ne?
   b. Kaj se zgodi z razpoložljivim denarjem (spremenljivko *denarnica*), če je izdelek kupljen?
   c. Kako štejemo kupljene izdelke?
   d. Kaj se zgodi z izdelkom na polici?

a. Igralec lahko izdelek kupi, če ima na voljo dovolj denarja. To pomeni, da moramo primerjati dve spremenljivki: *denarnica* in *cenaLubenice*. Če lubenica stane več, kot je denarja na voljo, igralec izdelka ne more kupiti. Dodamo še besedilo, ki igralcu pove, da izdelka ne more kupiti.



b. Če ima igralec 15 EUR in kupi lubenico za 4 EUR, ima nato na voljo 15 – 4 = 11 EUR. Vrednost spremenljivke bo torej prejšnja *vrednost denarnice* – *cenaLubenice*. Dodamo še besedilo.

c. Število kupljenih izdelkov štejemo s pomočjo spremenljivke zdravaHrana, ki jo povečamo za 1.

d. Ko je izdelek kupljen, se *skrije*.

Možna koda:

[Korak 4]

Na polici želimo imeti več izdelkov, zato lik lubenice podvojimo. Drugi lik bo npr. torta. Koda iz [Korak 3] potrebuje nekaj sprememb in sicer moramo:

- zamenjati obleko,
- ustvariti novo spremenljivko: *cenaTorte,*
- nastaviti ceno torte = nastaviti vrednost spremenljivke *cenaTorte,*
- v kodi zamenjati spremenljivko *cenaLubenice* v *cenaTorte,*
- zamenjati spremenljivko *zdravaHrana* v *nezdravaHrana.*

Koda za torto, ko je miška kliknjena, bo npr.:

Korak večkrat ponovimo, da imamo na polici več izdelkov.

[Korak 5]

Ko igralec konča z nakupovanjem, klikne na lik Konec. Lik konec pošlje obvestilo o koncu nakupovanja.

[Korak 6]

Vrnimo se h kodi dekleta.

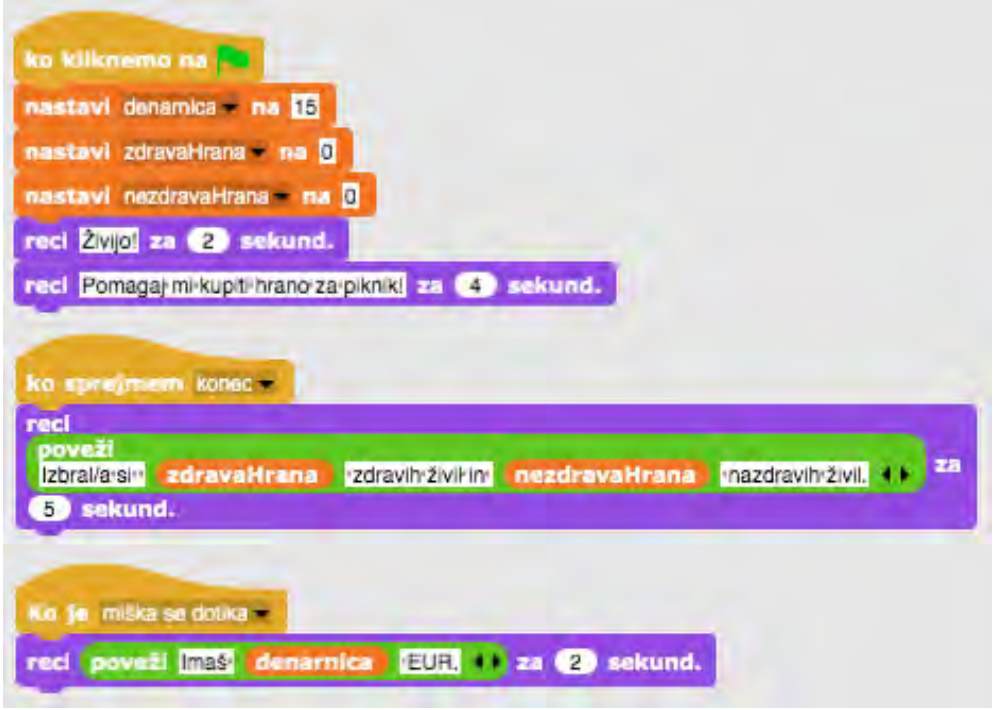
Sedaj želimo, da dekle pove, koliko zdravih in nezdravih izdelkov je igralec kupil. Uporabimo dogodek *ko sprejmem* obvestilo *konec*:

[Korak 7]

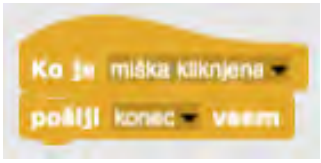
Kadarkoli med igro lahko igralec preveri, koliko denarja ima še na voljo. To stori tako, da se z miško pomakne na dekle. Primer kode:

[Končna koda]

Dekle



Hrana

	Konec  [Dodatne naloge] Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom: • Spremeni igro tako, da lahko vsako živilo kupiš 3x. • Igralec naj ima na voljo več denarja. • Na koncu dekle pove še, koliko katerih izdelkov si kupil. Npr. »Kupil/a si 2x lubenico, 1x grozdje, 2x krompirček«.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G16_KupovanjeHraneZaPiknik • Primer dodatne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G16_KupovanjeHraneZaPiknik%20%2B%20Dodatek • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Navodila za učenca (C4G16_NavodilaZaUcenca.docx)

Učni scenarij 17 – Računanje

Naslov učnega scenarija	Računanje
Pričakovano programersko predznanje	Uporaba spremenljivk za štetje točk in izbiranje videza ozadja in lika. Uporaba naključnih števil za izbiro naključnega ozadja in videza lika. Uporaba ponavljajoče zanke. Uporaba pogojnega stavka. Uporaba primerjalnih operatorjev. Uporaba bloka za zaznavanje dialoga (vprašaj in čakaj). Uporaba bloka za pošiljanje dogodkov (pošlji vsem).
Učni cilji	Splošni učni cilji: ● spremenljivke ● pogojni stavki ● zanke ● zaznavni bloki ● pošiljanje dogodkov Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenci uporabijo spremenljivke za štetje točk. ● Učenci uporabijo spremenljivke za videz ozadja in lika. ● Učenci uporabijo pogoje in logične operacije. ● Učenci uporabijo pošiljanje dogodkov za spremembo lika in izračun končnega rezultata.
Cilji, naloge in kratek opis aktivnosti	**Kratek opis:** Med igro preverimo, ali igralec obvlada računske operacije v Snap!-u. Pravila so naslednja: igralec bo zaporedoma dobil za izračunati 10 naključnih računov, pri čemer bo prvo število vedno 6, naključno pa bo vsakič izbiralo med računskimi operacijami (seštevanje, odštevanje, množenje, deljenje) in drugim številom, ki pa je lahko 1, 2 ali 3. Igralec mora vnesti izračunani rezultat, pri čemer se mu pravilni in

	napačni odgovorijo štejejo. Po desetem izračunu, se izpiše koliko točk je igralec dosegel. **Naloge:** Učenci morajo določiti ozadje in videz lika, načrtovati potrebne spremenljivke in določiti katere bloke bodo potrebovali. Nato morajo sestaviti kodo ozadja in lika. Dodatna naloga: Učenci lahko na koncu igre dodajo povratno informacijo, kot je: »Odlično!« ali »Računskih operacij ne poznate še dobro!«. **Cilji:** Učenci bodo izboljšali svoje predhodno pridobljeno znanje z poznavanjem spremenljivk, naključnih števil, zank, pošiljanja dogodkov.
Trajanje aktivnosti	45 minut
Učne strategije in metode	Aktivni pouk (diskusija, eksperimentiranje z že pripravljeno igro), učenje z ustvarjanjem iger, problemsko učenje
Učne oblike	Individualno delo / delo v paru / frontalna oblika
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) **MOTIVACIJA - UVOD** 1. Učitelj za uvodno motivacijo učencem poda v reševanje že ustvarjeno igro.  2. Učitelj se z učenci pogovori o pogojih naloge. V igri je desetkrat naključno izbrana računska operacija s prvim številom 6, drugo število pa je izbrano naključno, od 1 do 3. Igralec vnese

rezultat. Pravilni in napačni rezultati se štejejo. Na koncu igre igralec dobi povratno informacijo kako uspešen je bil pri računanju.

3. Učitelj se z učenci pogovori o spremenljivkah, na kakšen način so definirane, nastavljene in kako se spreminjajo.
4. Pogovorijo se tudi o kodi v skriptih ozadja in lika. V tem primeru je glavna koda zapisana v skriptih ozadja, v skripti lika pa je zapisana koda za spreminjanje videza lika in za izračun končnega izračuna.

IMPLEMENTACIJA

Koda v skripti ozadja vsebuje nastavitev spremenljivk napačnih in pravilnih odgovorov, ki so na začetku igre nastavljeni na 0.

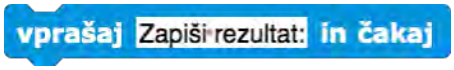
Nato v zanki, ki se ponovi 10-krat (10 različnih računov), najprej naključno izberemo eno izmed 4 računskih operacij (seštevanje, odštevanje, množenje, deljenje), ki so prikazana kot različni videzi ozadja. Kar pomeni, da naključno izberemo enega izmed 4 videzov ozadja.

V naslednjem koraku naključno izberemo še drugo število v računu. To naredimo tako, da pošljemo sporočilo »spremeni število« vsem skriptam v orodju Snap!. Ko lik *Število* sprejme sporočilo, da spremeni število, naključno izbere eno izmed števil od 1 do 3, ki so prikazana kot videzi lika *Število*.

Skripta ozadja:

Skripta lika *Število*:

Sedaj, ko je račun naključno sestavljen, je potrebno omogočiti, da igralec vnese rezultat. To naredimo z naslednjim blokom:



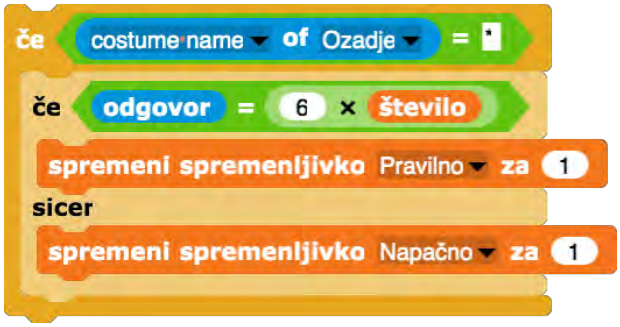
Vneseni rezultat je potrebno nato preveriti, če je pravilen. Najprej je potrebno preveriti za katero računsko operacijo gre oz. kateri videz

ozadja je trenutno nastavljen (videzi ozadja so poimenovana +, -, *, /). Nato pa za posamezno računsko operacijo preverimo ali je vnesen rezultat igralca enak pravilnemu rezultatu. V primeru, da je enak, se spremenljivka *Pravilno* poveča za 1, v nasprotnem primeru se za 1 poveča spremenljivka *Napačno*.

Seštevanje:

Odštevanje:

Množenje:



Deljenje:

Za podajanje povratne informacije igralcu, kako je bil uspešen pri računanju, je potrebno najprej poslati sporočilo »seštej dosežene točke« vsem skriptam v orodju Snap!.

Ko lik *Število* sprejme sporočilo, sporoči igralcu koliko točje je dosegel. Dosežene točke izračuna tako, da število napačnih odgovorov odšteje od števila pravilnih odgovorov.

Skripta lika *Število*:

OPOMBA: V primeru, da imajo učenci primero predznanje o poznavanju računskih operacij, lahko nalogo nadgradimo z dodajanjem različnih naključnih iger, pri čemer je potrebno pri odštevanju in deljenju upoštevati dodatne pogoje (npr. zmanjševanec mora biti večje od odštevanca, števili morata biti deljivi). V nalogo lahko vključimo tudi operacijo modulo (mod).

[Končna koda]

Ozadje

	Lik Število 
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Racunanje • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=tadeja&project=Racunanje_0 • Navodila za učenca (C4G17_NavodilaZaUcenca.docx)

Učni scenarij 18 – Recikliranje

Naslov učnega scenarija	Recikliranje
Pričakovano programersko predznanje	Postavljanje lika na določeno mesto na odru Prikazovanje in skrivanje lika Uporaba pogojnega stavka Uporaba preverjanja dotikanja Uporaba spremenljivk Uporaba zank Pošiljanje in prejemanje sporočil
Učni cilji	Splošni učni cilji: ● Razdeli igro na manjše dele in z njimi sestavi celoto ● Podvoji in ustrezno prilagodi del kode Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenec uporabi spremenljivko za štetje pravilno razvrščenih odpadkov in za preverjanje, ali je igralec že pospravil vse odpadke ● Učenec za vsak odpadek preveri, če se lik dotika drugega lika ● Učenec ustrezno skrije in pokaže lik ● Učenec nastavi ali je lik vlečljiv ali ne ● Učenec uporabi pošiljanje sporočil za interakcijo med liki ● Učenec uporabi zanko ponavljaj, dokler (ni odpadek v ustreznem košu) ● Učenec uporabi čakaj, dokler (niso pobrani vsi odpadki)
Cilji, naloge in kratek opis aktivnosti	**Cilj: Načrtovanje izdelave igrice** **Naloge:** Za vsak lik napiši, kaj je njegova naloga. Označi, katere naloge že znaš sestaviti. Sestavi igro in jo testiraj.

	Kratek opis: Učencem predstavimo igrico, ki jo bodo izdelovali v tej uri in jim naročimo, naj razmislijo, kaj mora narediti posamezen lik. Skupaj pregledamo, kaj je potrebno narediti. Učenci označijo kaj že znajo in prosijo za pomoč, če se jim kje zatakne. Sestavijo igrico po svoji predlogi in preverijo pravilnost njenega delovanja.
Trajanje aktivnosti	45 minut
Učne strategije in metode	Možganska nevihta Razgovor Metoda praktičnega dela
Učne oblike	Frontalno delo Individualno delo Delo v paru
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) [predstavitev igre] Robot Smetko je opazil, da na otroškem igrišču ležijo papirnati in stekleni odpadki in se zato otroci tam ne morejo brezskrbno igrati. Ker želi otrokom in njihovim staršem pomagati pri čiščenju igrišča, je prinesel dve posodi, v kateri je potrebno razvrstiti odpadke z igrišča - v zeleni koš sodijo odpadki iz stekla, v modrega pa odpadni papir. Sestavi igrico, s katero se bodo otroci naučili, v kateri zabojnik morajo odvreči posamezen odpadek, da bo igrišče spet čisto, odpadki pa pospravljeni v ustrezno posodo. [Načrtovanje igre] Za izdelavo igre boš uporabil_a naslednje like: robota Smetka, dva koša za odpadke in dve vrsti različnih odpadkov. Razmisli, kaj je naloga posameznih likov (kaj morajo narediti, kdaj morajo biti vidni, kdaj ali jih smemo vleči…) in to zapiši na učni list. Skupaj pogledamo, kaj so napisali in po potrebi dopolnimo, popravimo

	Primer rešitve: Robot Smetko - Pove zgodbo in navodila za igro - Sporoči, da se igra začenja - Šteje pospravljene smeti - Počaka na konec igre - Pove rezultat - Prikazan na začetku in koncu, skrit med igro Koša za smeti - Se prikažeta na določenem mestu - Ne moremo jih vleči Smeti - Pokaže se na določenem mestu - Lahko jo vlečemo - Preverja, če je v napačnem košu in opozori, če je - Preverja, če je v pravem košu, in ko je se skrije in šteje za pospravljeno Med naštetim nalogami likov označite tiste, za katere menite, da jih znate narediti sami. O ostalih razmislite, kaj vas zanima in si na učni list zapišite vprašanja, ki jih imate. O teh vprašanjih se pogovorite s sošolcem. Če tudi on_a ne zna odgovoriti, se posvetujeta z učiteljem. [Izdelava igre] V Snap!-u sestavite igro, kot ste jo načrtovali. Pri izdelavi si lahko pomagate s predlogo (https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=recikliranje_predloga), v kateri lahko najdete vse like. Posamezne dele igre sproti testirajte, da vidite, če delujejo pravilno. [Refleksija] Vprašamo učence, kaj jim je bilo pri takšni izdelavi igre všeč, kaj so pogrešali, če so imeli kakšne težave, kako so jih rešili. Kako bi igro nadgradili?

Učni pripomočki, sredstva za učitelja	• Primer končne rešitve osnovnega primera v Snap!-u: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_18_recikliranje
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=recikliranje_predloga • Učni list za učenca (C4G18_UcniListZaUcenca.docx) • Navodila za učenca, ki je malo manj vešč programiranja iger (C4G18_NavodilaZaUcenca.docx)

Učni scenarij 19.1 – Zaigraj na klavir 1

Naslov učnega scenarija	Zaigraj na klavir 1
Pričakovano programersko predznanje	Uporaba spremenljivk za štetje točk Uporaba dogodka *ko je gumb miška pritisnjen* Uporaba zanke Uporaba pogojnega stavka Uporaba pošiljanja obvestil za menjavo ozadij in upravljanje likov
Učni cilji	Splošni učni cilji: • Spremenljivke • Pogojni stavek • Zanke • Pošiljanje obvestil • Zvok • Sestavljanje kode za melodijo Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec uporabi spremenljivko za štetje točk. • Učenec uporabi ukaz za pošiljanje obvestil za menjavo ozaij in upravljanje likov. • Učenec uporabi ukaze za zvok za ustvarjanje melodije. • Učenec uporabi zanko za zmanjšanje števila ukazov v kodi. • Učenec razširi funkcionalnost igre.

Cilji, naloge in kratek opis aktivnosti	Kratek opis: Pojdimo v čudoviti svet kraljice Mary. Igralca povabi v grad, kjer bosta poslušala glasbo. V plesni dvorani njen prijatelj Dino igra na klavir. Dino zaigra nekaj tonov, igralec pa mora ugotoviti, kateri ton je Dino zaigral. Pravilen odgovor prinese 1 točko. Nato Dino zaigra pesem, igralec pa mora ugotoviti njen naslov. Pravilen odgovor prinese 5 točk. Naloge: Učenci imajo na voljo predlogo z ozadji in liki (z več oblekami). Ustvariti morajo spremenljivko za štetje točk, ugotoviti, katere ukazne bloke potrebujejo, se spoznati z ukaznimi bloki za zvok ter napisati kodo za predvajanje melodij/e. **Cilji: Učenci se bodo naučili napisati kodo za ustvarjanje melodije ter izboljšali svoje znanje o spremenljivkah, zankaj, pogojnih stavkih in dogodkih.**
Trajanje aktivnosti	90 minut
Učne strategije in metode	Aktivno učenje (diskusija, eksperimentiranje z vnaprej pripravljeno igro), učenje z izdelavo iger, reševanje problemov
Učne oblike	Individualna učna oblika / Delo v paru / Skupinska učna oblika Frontalna učna oblika s sodelovanjem celotnega razreda
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) [Korak 1]  Podamo uvodna navodila: Naredili bomo igro, kjer se kraljica Mary pojavi pred gradom in igralca povabi v grad. Po prihodu v plesno dvorano, dinozaver Dino na klavir zaigra nekaj tonov, ki jih mora igralec uganiti. Pravilen odgovor prinese 1 točko. Nato igralec klikne na klavir in Dino zaigra pesem, ki jo mora igralec prav tako uganiti. Sedaj

pravilen odgovor prinese 5 točk. Kraljica na koncu pove, koliko točk je igralec dosegel. Če igralec ugane pesem, Dino zapleše.

[Korak 2]

Predvajamo delen program:

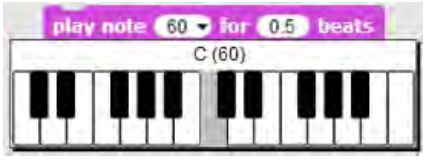
https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir_1

Pogovorimo se, kaj v igri še manjka, glede na začetna navodila (odhod v grad, Dinov ples, Mary pove število doseženih točk). Kako bomo napisali kodo za predvajanje glasbe? Ker tega še ne znamo, si bomo skupaj pogledali v naslednjem koraku.

[Korak 3]

Učencem razložimo, kako delujejo ukazi za zvok. Uporabimo ukaz

Številka 60 pomeni, kateri ton bo predvajan, 0.5 beats pa pomeni, koliko časa bo ton predvajan. Ni potrebno, da poznamo številke (60) tonov na pamet, saj se nam s klikom na puščico poleg številke pokaže klavir, kjer izberemo ustrezen ton.



V desnem polju določamo dolžino tona. Vpišemo lahko (decimalno) število, ali pa število v obliki ulomka.

ali

Učenci lahko sami poskušajo in določajo trajanje tona.

[Korak 4]

Učenci bodo napisali kodo za ugibanje zaigranega tona. Ton se bo najprej zaigral, nato se pojavi vprašanje. Če je odgovor pravilen, prištejemo točko. Dodamo tudi povratno informacijo o pravilnem / nepravilnem odgovoru. Koda lahko izgleda tako:

Korak lahko večkrat ponovimo, pri tem zamenjamo zaigran ton in odgovor.

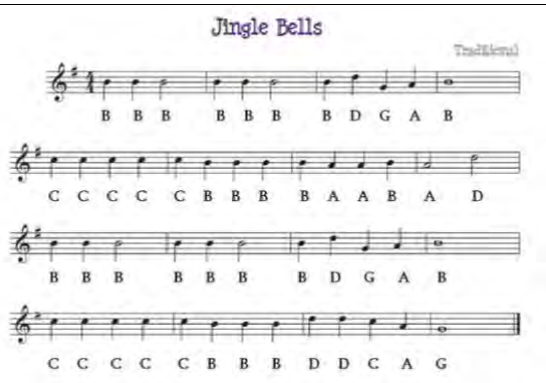
Celotna koda se lahko izvede s prejemom obvestila, npr.

,

zaključi pa tako, da igralcu povemo, kako naj z igro nadaljuje. Npr.

[Korak 5]

S klikom na klavir se bo zaigrala pesem. Lahko napišejo kodo za dano pesem ali pa najdejo na spletu note za poljubno pesem (koda za pesem je na koncu priprave).



Pomebno je, da uporabijo zanko, ko je to mogoče. Na koncu klavir vpraša za naslov pesmi, šteje točke in pošlje obvestilo za konec igre. Ob pravilnem odgovoru Dino zapleše.

[Korak 6]

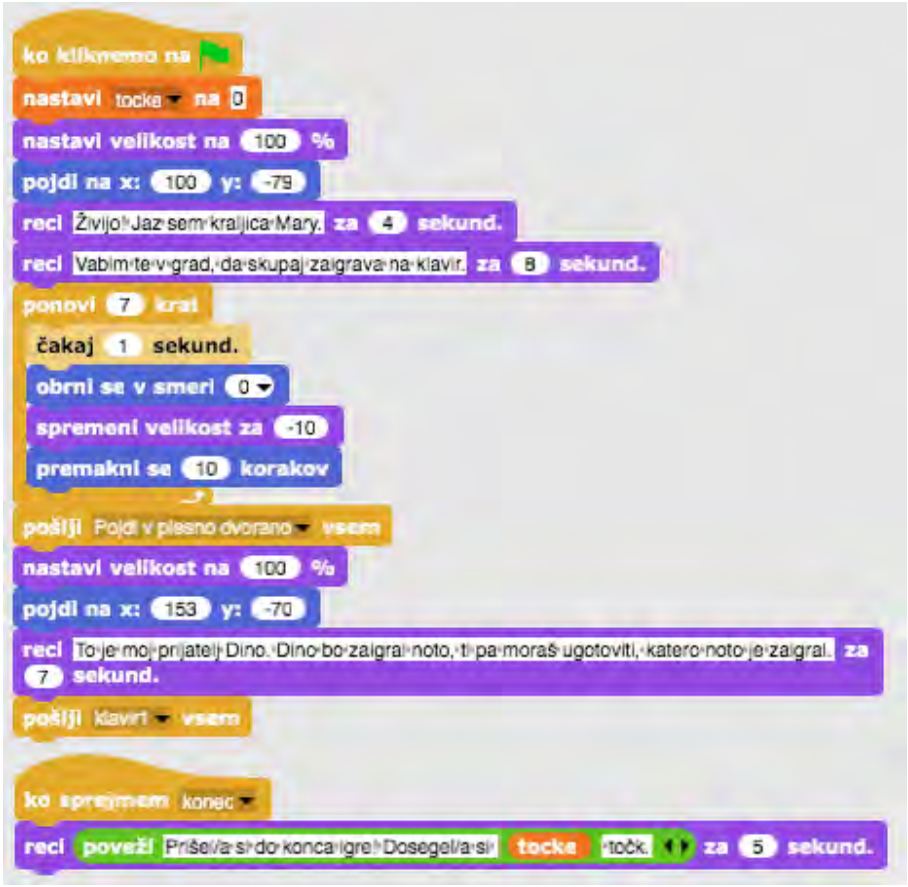
Učencem lahko pokažemo primer celotne kode: https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir,

oni pa začnejo ustvarjati svojo igro z dano predlogo: https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir_Delno.

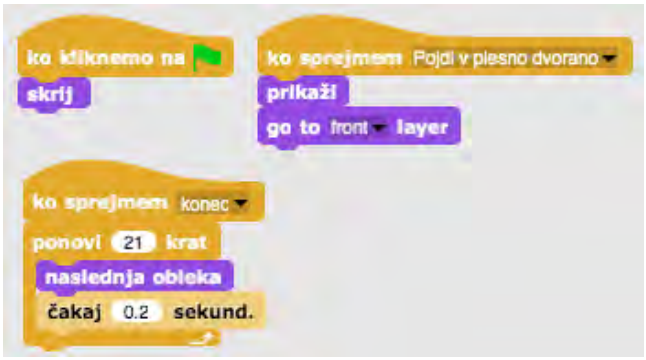
Razdelimo jim učne liste za delo v paru (ali skupinah, presodite sami), kjer igro najprej načrtujejo, potem pa jo izvedejo v okolju Snap!. Učni list je na voljo kot priloga.

[Primer končne kode]

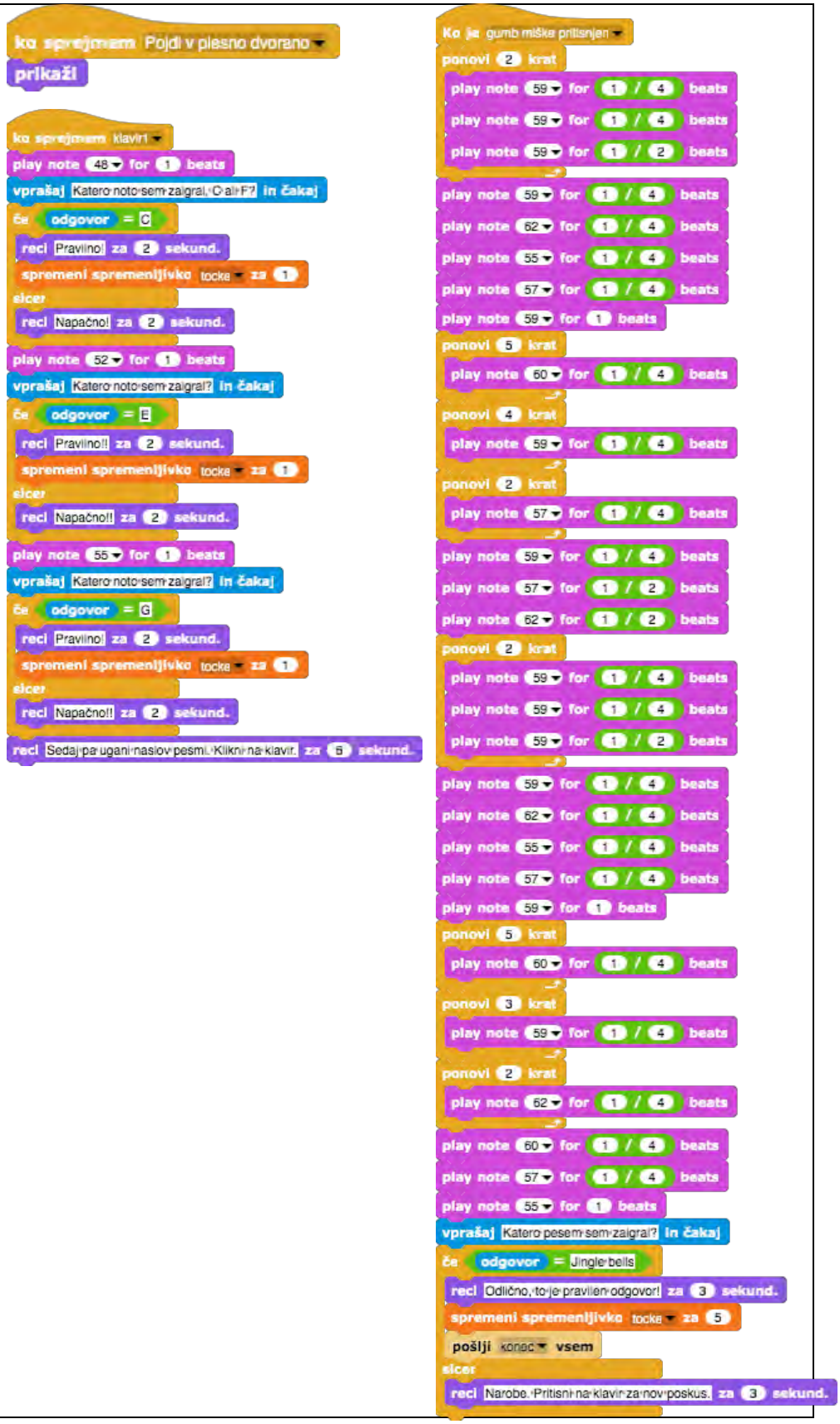
Kraljica

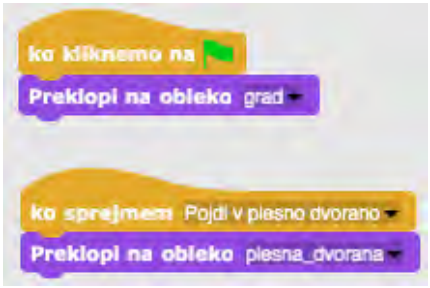


Dinozaver



Klavir

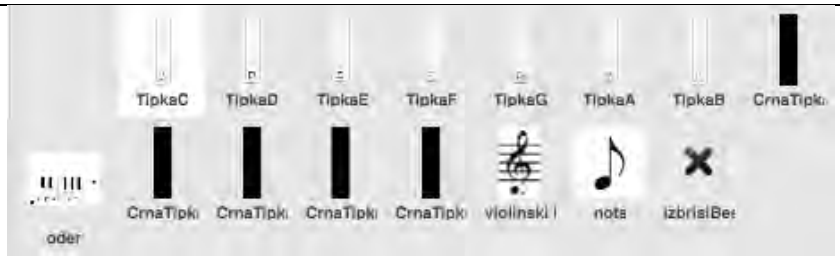

ko sprejmem Pojdi v plesno dvorano
prikaži
ko sprejmem klavir
play note 48 for 1 beats
vprašaj Katero noto sem zaigral, C ali F? in čakaj
če odgovor = C
reci Pravilno! za 2 sekund.
spremeni spremenljivko točke za 1
sicer
reci Napačno! za 2 sekund.
play note 52 for 1 beats
vprašaj Katero noto sem zaigral? in čakaj
če odgovor = E
reci Pravilno!! za 2 sekund.
spremeni spremenljivko točke za 1
sicer
reci Napačno!! za 2 sekund.
play note 55 for 1 beats
vprašaj Katero noto sem zaigral? in čakaj
če odgovor = G
reci Pravilno! za 2 sekund.
spremeni spremenljivko točke za 1
sicer
reci Napačno!! za 2 sekund.
reci Sedaj pa ugani naslov pesmi. Klikni na klavir. za 5 sekund.
Ko je gumb miške pritisnjen
ponovi 2 krat
play note 59 for 1 / 4 beats
play note 59 for 1 / 4 beats
play note 59 for 1 / 2 beats
play note 59 for 1 / 4 beats
play note 62 for 1 / 4 beats
play note 55 for 1 / 4 beats
play note 57 for 1 / 4 beats
play note 59 for 1 beats
ponovi 5 krat
play note 60 for 1 / 4 beats
ponovi 4 krat
play note 59 for 1 / 4 beats
ponovi 2 krat
play note 57 for 1 / 4 beats
play note 59 for 1 / 4 beats
play note 57 for 1 / 2 beats
play note 62 for 1 / 2 beats
ponovi 2 krat
play note 59 for 1 / 4 beats
play note 59 for 1 / 4 beats
play note 59 for 1 / 2 beats
play note 59 for 1 / 4 beats
play note 62 for 1 / 4 beats
play note 55 for 1 / 4 beats
play note 57 for 1 / 4 beats
play note 59 for 1 beats
ponovi 5 krat
play note 60 for 1 / 4 beats
ponovi 3 krat
play note 59 for 1 / 4 beats
ponovi 2 krat
play note 62 for 1 / 4 beats
play note 60 for 1 / 4 beats
play note 57 for 1 / 4 beats
play note 55 for 1 beats
vprašaj Katero pesem sem zaigral? in čakaj
če odgovor = Jingle bells
reci Odlično, to je pravilen odgovor! za 3 sekund.
spremeni spremenljivko točke za 5
pošlji konec vsem
sicer
reci Narobe. Pritisni na klavir za nov poskus. za 3 sekund.

	Ozadje 
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir • Delna aktivnost v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir_1 • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G19.1_ZaigrajNaKlavir_Delno • Učni list za učenca (C4G19.1_UcniListZaUcenca.docx) • Navodila za učenca (C4G19.1_NavodilaZaUcenca.docx)

Učni scenarij 19.2 – Zaigraj na klavir 2

Naslov učnega scenarija	Zaigraj na klavir 2
Pričakovano programersko predznanje	Uporaba zanke Uporaba spremenljivk Uporaba pogojnega stavka
Učni cilji	Splošni učni cilji: • Pogojni stavek • zanke Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • Učenec zna uporabiti zanko za igranje glasbe • Učenec zna napisati kodo, da ob pritisku tipke program predvaja zvok • Učenec doda zvok liku • Učenec zna zamenjati obleko lika s pritiskom na tipko
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Učenec po danih notah zaigra pesem na klavir. Naloge: Učenec napiše kodo za klavirske tipke – vsaka tipka (lik) mora zaigrati določen ton. Na zaslonu so še trije liki: violinski ključ, ki pokaže besedilo z notami; nota, ki zaigra pesem; X, ki izbriše besedilo pesmi. **Cilji: Učenci se bodo naučili, kako se predvaja glasba ter kako se zamenja obleka liku s klikom nanj.**
Trajanje aktivnosti	45 min
Učne strategije in metode	Aktivno učenje, učenje z izdelavo iger, reševanje problemov
Učne oblike	Individualna učna oblika / Delo v paru

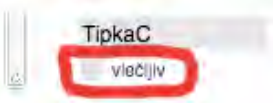
Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Učencem na začetku pokažemo, kako bo izgledala končna igra. Ko odpremo projekt, imamo na ekranu klavir, violinski ključ, noto ter gumb X (slika levo). S klikom na violinski ključ se pokaže besedilo z notami (slika desno), s klikom na noto se zaigra celotna pesem, s klikom na gumb X pa se ozadje z besedilo zamenja z belim ozadjem, tako da imamo ponovno sliko na levi. S klikom na posamezno belo tipko se zaigra določen ton in tipki se za delček sekunde spremeni obleka.  Učencem je na začetku dano: • Tipka C, ki je na svojem mestu in ima obleke za vse manjkajoče tipke • Črna tipka • Belo ozadje in ozadje z akordi • Vsi trije liki: violinski ključ, nota ter gumb X [Korak 1] Odpremo predlogo in sestavimo klavir. Tipko C podvojimo in povlečemo na svoje mesto. Zamenjamo ji obleko. Nato podvojimo še črno tipko in jo postavimo na svoje mesto. Postopek ponavljamo, dokler nimamo vrhnje (leve) slike. Za lažje delo v nadaljevanju, like preimenujemo.



Če se črne tipke skrivajo za belimi, uporabimo ukaz 

, ki bo lik postavil v ospredje. Na ukaz lahko kliknemo kar na levi strani, kjer se nahajajo vsi ukazi.

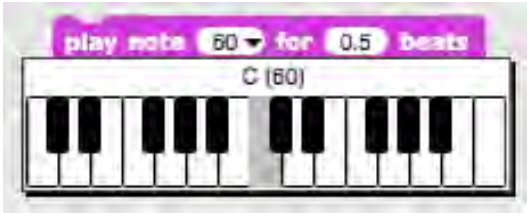
Pri vseh tipkah še odkljukamo ukaz vlečljiv, da ne bomo med igranjem tipk nehote premikali:



[Korak 2]

Sedaj bomo napisali kodo za predvajanje tonov. Uporabili bomo pošiljanje obvestil. Ko bo program sprejel obvestilo, bo zaigral določen ton. Ustvarimo torej obvestilo *igraj c* in izberimo ustrezen ton.

Kateri ton se bo zaigral določimo z ukazom *play note __ for 0.5 beats* in s klikom na puščico izberemo željen ton.



Številka 60 torej določa ton C, 0.5 beats pa določa dolžino predvajanja tona.

Za tipko C imamo sedaj takšno kodo:

Da se bo koda izvedla, moramo obvestilo *igraj c* tudi poslati.

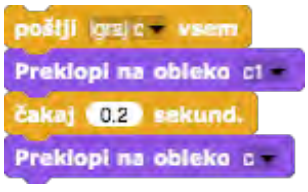
[Korak 3]

Program bo predvajal zvok, ko kliknemo z miško na tipko C ali ko pritisnemo tipko C na tipkovnici. Uporabili bomo torej 2 dogodka:

Kaj se bo zgodilo, ko kliknemo na tipko C oz. pritisnemo tipko C na tipkovnici?

- Predvajal se bo zvok,
- spremenila se bo obleka tipki (da bo videti, kot da smo v resnici prisnili na tipko) na obleko *c1*,
- spet bomo nastavili obleko na obleko *c*.

Za predvajanje zvoka uporabimo pošiljanje obvestila *igraj c*, ki smo ga ustvarili v [Korak 2], med spreminjanje obleke v c1 in nazaj v c pa damo še ukaz počakaj 0.2 sekund, da bo animacija sploh vidna. Koda za dogodek *ko je miška kliknjena* je torej naslednja:

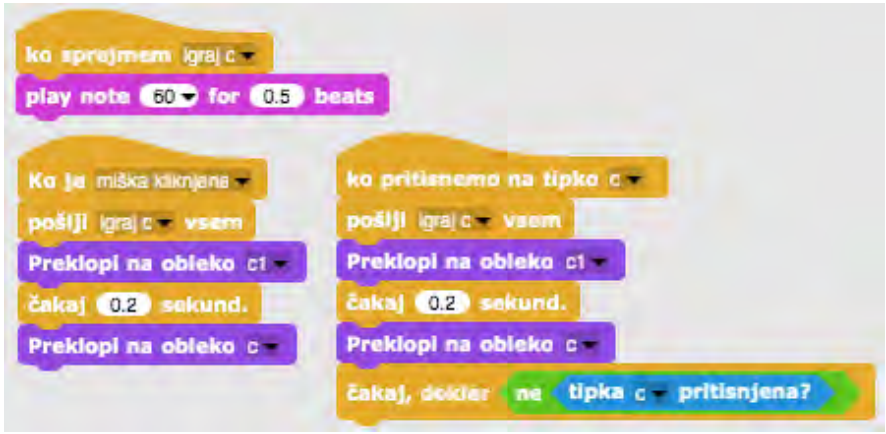


Razmislimo: Ali bo koda za dogodek *ko pritisnemo na tipko c* enaka? Odgovor je: skoraj.

Podvojimo zgornjo kodo in jo združimo z dogodkom *ko pritisnemo na tipko c.* Testirajmo. Pritisnemo na tipko C, ton se predvaja. Kaj pa, če tipko C

držimo? Če tipko C držimo, se ton konstantko predvaja, saj ukaz konstantko pošilja obvestilo *igraj c*. Da to preprečimo, bomo na konec kode dodali še kombinacijo ukazov *čakaj, dokler* + *operator ne* + *tipka c pritisnjena,* ki bo preprečila nenehno pošiljanje obvestila *igraj c*.

Celotna koda za tipko C je torej:



[Korak 4]

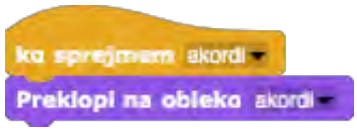
[Korak 3] ponovimo in celotno kodo za tipko C podvojimo za vse ostale like (tipke). Za tipko D tako zamenjamo:

- *ko sprejmem obvestilo igraj d*,
- play note *62*,
- *pošlji igraj d*,
- preklopi na obleko *d1* ter *d*,
- čakaj, dokler ni tipka *d* pritisnjena.

[Korak 5]

S klikom na lik violinski ključ bomo spremenili ozadje. To lahko storimo s pošiljanjem obvestil. Liku violinski ključ torej napišemo kodo:

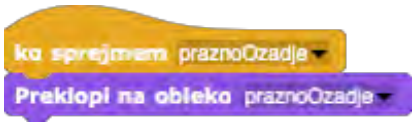
Na ozadju pa dodamo kodo



Postopek ponovimo za lik izbrisiBesedilo in preklop na prazno ozadje.

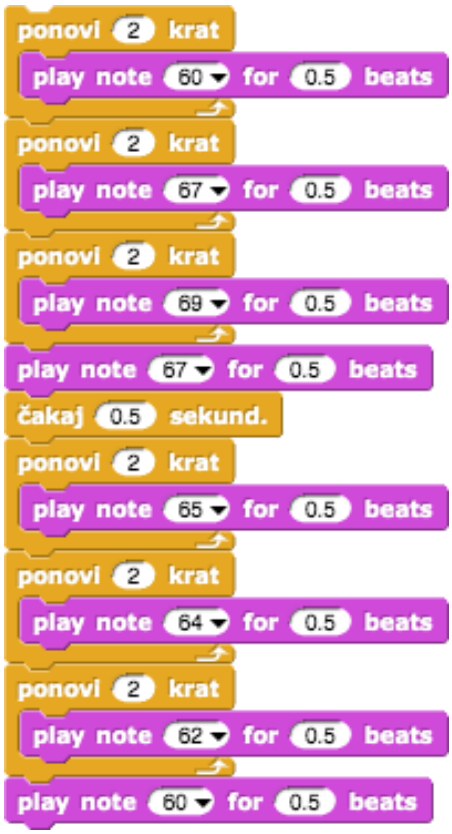
Lik izbrisiBesedilo:

Ozadje:



[Korak 6]

S klikom na noto se bo zaigrala celotna pesem. Napišemo kodo za celotno pesem in kjer je mogoče, uporabimo zanko ponovi.



Koda se nadaljuje na desni strani.

	[Dodatna naloga] Učenec doda dodatne naloge po svojih željah ali sledi spodnjim navodilom: • Podvoji lik Nota (ter zamenjaj položaj lika na ozadju) ter napiši program, ki bo zaigral drugo pesem. • Dodaj ozadje z besedilom in akordi za novo pesem.
Učni pripomočki, sredstva za učitelja	• Primer celotne aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G19.2_ZaigrajNaKlavir%20-%20Cela • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.
Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=mateja&project=C4G19.2_ZaigrajNaKlavir%20-%20Delna • Navodila za učenca (C4G19.2_NavodilaZaUcenca.docx)

Učni scenarij 20 – Test

Naslov učnega scenarija	Test
Pričakovano programersko predznanje	Postavljanje lika na določeno mesto na odru Prikazovanje in skrivanje lika Uporaba pogojnega stavka Uporaba spremenljivk Pošiljanje in prejemanje sporočil
Učni cilji	Splošni učni cilji: ● Razdeli igro na manjše dele in z njimi sestavi celoto ● Zamenjaj ozadje ● Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: ● Učenec uporabi delček vprašaj, da dobi odgovor na vprašanje ● Učenec uporabi pogojni stavek za preverjanje pravilnost odgovora ● Učenec uporabi spremenljivki za štetje pravilnih in napačnih odgovorov ● Učenec uporabi pošiljanje sporočil za menjavanje ozadja ● Učenec uporabi delček za računanje razlike
Cilji, naloge in kratek opis aktivnosti	**Cilj: Sestaviti preverjanje znanja za sošolca** **Naloge:** Sestavi test, s katerim boš preveril sošolčevo znanje o Snap!-u **Kratek opis:** Učencem predstavimo nalogo: sestavite test z vprašanji, ki so prikazana kot ozadja na odru.
Trajanje aktivnosti	45 minut
Učne strategije in metode	Metoda praktičnega dela
Učne oblike	Frontalno delo Individualno delo

Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) [predstavitev naloge] Abby bi rada preverila sošolčevo znanje o Snap!-u. Sestavila je test z vprašanji, koda za zastavljanje vprašanj in preverjanje pravilnosti odgovorov pa se ji je izbrisala. Vprašanja je shranila kot različna ozadja. Pomagaj ji sestaviti program, v katerem se bo za vsakim vprašanjem pokazalo naslednje, lik Abby pa bo sošolcu povedal, s katerimi besedami lahko odgovori. Na koncu naj program sošolcu pove, kolikokrat je odgovoril pravilno in kolikokrat napačno, ter število točk, ki jih je dosegel (vsak pravilen odgovor prinese 1 točko, vsak napačen pa -1). Abbyjin program najdeš tukaj: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_tmp [Načrtovanje] Iz opisa si skicirajte, kako mora izgledati program. Lahko si naloge zapišete po alinejah ali na način, ki je vam bližji. Razmislite, kako bo oder izvedel, kdaj naj zamenja ozadje. Primer načrta: Oder: - Na začetku pokaže začetno ozadje - Ko prejme sporočilo, zamenjavo ozadja na naslednje Gumb: - Se na začetku prikaže - Ob kliku nanj pošlje sporočilo o začetku in se skrije Abby: - Nastavi vse točke na 0 - Pove navodila - Ob kliku na gumb začne s postavljanjem vprašanj: - Sporoči odru, naj zamenja ozadje za naslednje vprašanje - Postavi vprašanje z možnimi odgovori

	- Preveri ali je odgovor pravilen
	  o Če je, prišteje točko k pravilnim odgovorom
	  o Če ne, prištej točko k napačnim odgovorom
	- Pred novim vprašanjem zamenja ozadje, pove vprašanje, preveri odgovor – to ponovi večkrat
	- Na koncu pove število pravilnih odgovorov, število napačnih odgovorov, število točk
	
	[Izdelava]
	
	V Snap!-u sestavite preverjanje znanja kot ste ga načrtovali. Pri izdelavi si lahko pomagate s predlogo (https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_tmp), v kateri lahko najdete vse like in začetna Abbyjina navodila. Posamezne dele programa sproti testirajte, da vidite, če delujejo pravilno.
	
	[Dodatne naloge – če je čas]
	
	Dodatna naloga 1: Med igro Abby večkrat spremeni svoj izgled.
	
	Dodatna naloga 2: Sami sestavite podobno preverjanje znanja. Ker v Snap!-u ni možnosti za pisanje besedila, lahko slike za ozadje pripravite v poljubnem programu za izdelavo rastrske (npr. Slikar, Paint.net) ali vektorske grafike (npr. Inkscape) in jih uvozite v svoj kviz Snap!-u. Če slike niso potrebne, lahko vprašanje v celoti zastavi kar Abby.
	
	[Refleksija]
	
	Vprašamo učence, kaj jim je bilo pri takšni izdelavi igre všeč, kaj so pogrešali, če so imeli kakšne težave, kako so jih rešili. Kako bi igro nadgradili?

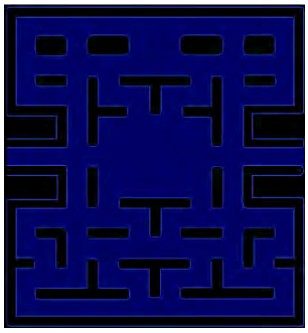
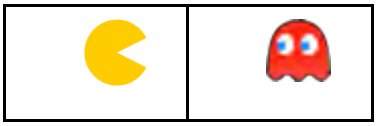
Učni pripomočki, sredstva za učitelja	● Primer končne rešitve osnovnega primera v Snap!-u: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test

Učni pripomočki za učenca	• Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/snap/snap.html#present:Username=spelac&ProjectName=C4G_20_test_tmp • Učni list za učenca (C4G20_UcniListZaUcenca.docx) • Navodila za učenca, ki je malo manj vešč programiranja iger (C4G20_NavodilaZaUcenca.docx)

Učni scenarij 21 – Enostavni PACMAN

Naslov učnega scenarija	**Enostavni PACMAN**
Pričakovano programersko predznanje	• pogojni stavki, • programiranje več objektov, • zaznavanje barve, • zanke (neskončna, ponavljaj dokler) • premikanje z uporabo dogodkov, • naključna števila
Učni cilji	Splošni učni cilji • kloniranje objektov, • definiranje obnašanja klona, • pošiljanje sporočil med objekti, • boolean vrednosti v logičnih izrazih, • definiranje, razlikovanje, dinamično preverjanje in odziv na dve različni stanji igre. Specifični učni cilji, ki so osredotočeni na algoritmično mišljenje: • učenec implementira premikanje objekta z uporabo smernih tipk in dogodkov, pri čemer upošteva omejitve, • učenec uporabi kloniranje, da naredi klone nekega objekta, • učenec zna določiti obnašanje klonov, • učenec pozna pomen pošiljanja sporočil med objekti, • učenec zna povezati pošiljanje sporočil iz klona za to, da poveča števec, ki je v nekem drugem objektu,

	● učenec zna ugotoviti, da je objekt prejel sporočilo in se takrat ustrezno odzvati.
Cilji, naloge in kratek opis aktivnosti	Kratek opis: Sprogramirajte igro, v kateri bo igralec pobiral zvezde, ki se bodo naključno pojavljale v labirintu in se pri tem izogibal duhcu. Naloge: Učenci morajo implementirati premikanje glavnega karakterja v igri, tako da se bo premikal zgolj znotraj labirinta in ne bo mogel iti čez stene. Nato morajo ustvariti objekt zvezdo, na začetku igre ustvariti njen klon na naključni lokaciji, pri čemer morajo paziti, da se ustvari na dovoljeni lokaciji, nato pa to ponoviti vsakič, ko igralec zvezdo pobere. Prav tako morajo beležiti število pobranih zvezd in končati z igro, ko jih igralec pobere dvajset. Igra bo bolj zanimiva, če vanjo dodajo še duhca, ki se bo po labirintu naključno premikal. V primeru, da se bo glavni karakter tega duhca dotaknil, pa bo igre takoj konec. **S to aktivnostjo bodo učenci ponovili znanje o premikanju objekta znotraj labirinta, kar so se naučili v eni od prejšnjih aktivnosti. Spoznali bodo koncept kloniranja objekta z upoštevanjem omejitev pri naključnem pozicioniranju na ekranu. Naučili se bodo narediti računalniško voden lik, ki se bo naključno premikal.**
Trajanje aktivnosti	90 minut
Učne strategije in metode	active learning, collaborative learning, problem solving
Učne oblike	frontalni način podajanja snovi individualno delo/delo v parih/skupinsko delo

Povzetek učnega procesa	(Motivacija-uvod, Implementacija, Refleksija in vrednotenje) Igralec zbira zvezdice, ki se naključno pojavljajo na ekranu, ob tem pa ga lovi rdeč duhec. Če glavni lik - Pacman trči z duhcem je igre konec (neuspešno), drugače pa je igre konec, ko zbere 20 točk (uspešno). [1. korak] Učenci izdelajo labirint v katerem je območje, po katerem se lahko premika glavni lik, v celoti neke barve (npr. modre), zidovi, ki ga ustavijo, pa so neke druge barve (npr. črne). Če želimo prihranit s časom, jim slikovni material pripravimo vnaprej.  [2. korak] Nato morajo narisati glavni lik - Pacman-a in pa rdečega duhca. Zvezdo lahko predstavlja majhen krogec, ki ga lahko narišejo znotraj programa Snap!.  [3. korak] Za programiranje premikanja glavnega lika lahko uporabimo več različnih možnosti. Spodnji primer prikazuje eno od njih. V njem uporabimo dogodke za ugotavljanje ali je igralec pritisnil določeno tipko, nato preverimo, kje se nahaja glavni lik. Če se dotika ustrezne barve potem ga obrnemo v smer premika in opravimo korak. Če se po koraku dotakne nedovoljene barve, ki označuje zid, pa ga premaknemo za enako dolg korak v nasprotno smer.

[4. korak]

Zdaj pa se lotimo programiranja zvezd. Vsaka od zvezd bo imela enak izgled in obnašanje. To je tipičen primer v katerem uporabimo klone, saj bi bilo kopiranje istega objekta neučinkovito in nerodno. V našem primeru bi morali to narediti dvajsetkrat, če bi želeli igro podaljšati na sto pobranih zvezdic, pa bi bilo to na ta način že skoraj nemogoče. Na začetku igre se bo ustvaril nov klon, ki se bo pojavil naključno na neki dovoljeni lokaciji v labirintu. Ko ga bo igralec pobral, bo izginil, nato pa se bo pojavil nov klon na novi lokaciji. Prvi klon na začetku igre bomo naredili z uporabo spodnje kode, ki jo bomo dali na ozadje:

	Ker ne želimo, da bi se originalni objekt zvezde prikazal, ampak želimo to le za klone, ga na začetku igre skrijemo. Nato se lotimo iskanja ustrezne lokacije na katero lahko postavimo zvezdo v labirintu. Če bi namreč zvezdo npr. postavili na zid, igralec do nje sploh ne bi mogel. Strategija, da dosežemo to je naslednja: 1. Pozicija objekta na zaslonu je izražena preko x in y koordinate, ki lahko imata katerokoli vrednost na intervalu [-140, 140], če dimenzij zaslona ne spreminjamo oz. ustrezno spremenjenem, če se za to odločimo. 2. Nato preverimo, če se klon dotika barve zidu. 3. Če se ne dotika barve zidu ga prikažemo (spomnimo, da smo original na začetku skrili, kar pomeni, da so tudi kloni privzeto skriti.) in v neskončni zanki preverjamo, če se je dotaknil glavnega lika. 4. Če se dotika barve zidu, ta lokacija ni dobra, zato naredimo nov klon (v upanju, da bo tokrat na boljši lokaciji), tega pa izbrišemo. To se bo ponavljalo dokler si ne bo izbral dovoljene lokacije. 5. Ko igralec zvezdo pobere, moramo števec zvezd povečati za ena. Ta števec mora biti v nekem objektu, ki ni klon, saj ga ne moremo ob uničenju klona izbrisati, saj bi tako izgubili njegovo vrednost. To lahko naredimo z uporabo sporočil in sicer tako, da pošljemo sporočilo takrat, ko se glavni lik dotakne klona. Nato ustvarimo novega, tega pa izbrišemo. 

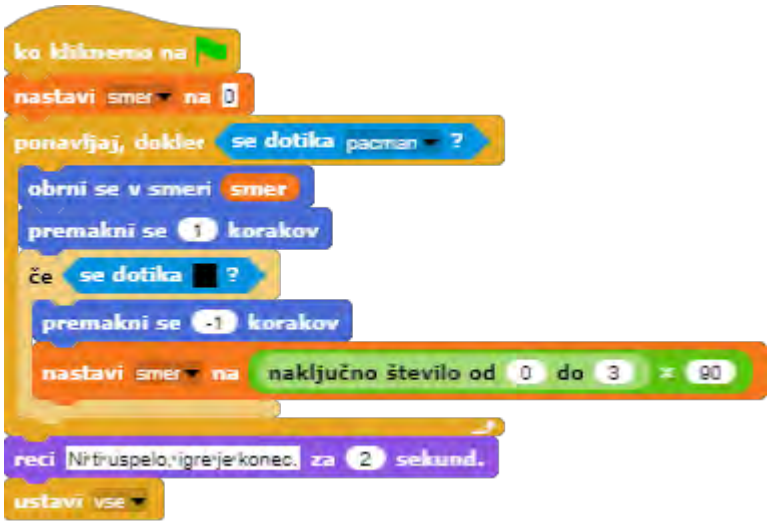
[5. korak]

Nato se lotimo programiranja duhca, ki se naključno premika po labirintu. Ko se dotakne zidu spremeni smer in nadaljuje premikanje v tej smeri. Če želimo, da je njegovo premikanje naključno lahko to naredimo tako, da si ob trku v zid naključno izbere novo smer premikanja. V programu Snap! so smeri izražene preko stopinj:

1. 0 degrees - GOR
2. 180 degrees - DOL
3. 90 degrees - DESNO
4. 270 degrees - LEVO

Opazimo, da so ta števila večkratniki števila 90, zato lahko naključnega med njimi dobimo tako, da si najprej izmislimo naključno število od 0 do 3, nato pa ga pomnožimo z 90. Vrednost, ki smo jo na ta način dobili predstavlja novo naključno smer premikanja.

Duhec se premika dokler ne trči ob glavni lik. To lahko uporabimo kot pogoj v zanki - *ponavljaj dokler*. Ko se to zgodi, pa je igre konec.



[6. korak]

Na koncu še poskrbimo za prištevanje točk in ugotavljanje kdaj je igralec uspešno končal igro. Števec smo implementirali znotraj kode glavnega lika. Na začetku smo ga postavili na 0, nato pa mu za vsakič, ko je klon oddal sporočilo, da se je dotaknil glavnega lika, povečali za 1. Ko se števec poveča samo preverimo, če je že dosegel vrednost 20. Takrat je igre konec.

Orodja in viri za učitelje	• Celotna aktivnost v programu Snap!: https://snap.berkeley.edu/project?user=zapusek&project=enostavni_pacman • Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena. • Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.

Viri/gradiva za učence	● Predloga aktivnosti v Snap!-u: https://snap.berkeley.edu/project?user=zapusek&project=pacman_template

PRILOGA 2. KODE UČNIH SCENARIJEV

Osnovni učni scenariji

Angleški scenariji

N.	TITLE	CODE
1	**Introduction to Snap! Interface** Students add a new sprite, add a costume to the sprite, edits the costume, and deletes one of them. Student creates a new background to the stage, edits it, and deletes unwanted ones.	**starting**
2	**Discover Snap! : move a spite** Help the students to discover the Snap! interface and code their first sprite so that it moves and speaks.	**dispubeng**
3	**Moving around the stage** Students learn how to move the sprite in x and y direction on the stage, code an easy program to solve the tasks given, learn how to turn the sprite in a different direction.	**monkey**
4	**Changing costumes and turning** Students learn how to change the sprite's costume to make an animation by changing between different types of rotation.	**Dancer**
5	**Sounds of the farm** Students learn how to program a simple game in which player can recognize the sounds of animals by pressing certain keys.	**farm**
6	**Chameleon's summer vacation** Program a simple game in which an object will change its costume based on the color of the background	**chameleoneng**
7	**Helping Prince and Princess to find their animals** A girl has to help the Princess find her cat and the Prince find his dog by avoiding that the animals can meet each other during their path.	**finding**
8	**Drawing with a chalk** Students will be introduced into drawing different shapes with a code. They will learn to use loop repeat for shorten the code and to change a background.	**chalk**
9	**Picking up trash and cleaning the park** Students will learn how to use variables and how to duplicate a block of code or even a whole sprite.	**cleaning**

10	**Feeding the cats** Students will be introduced to the concept of multiple variable random value assignment inside a loop and how it is different from when we do it outside a loop. They will also learn about how to get, test and count correct player inputs.	**feeding**
11	**Guessing the number of cats in a shelter** Students will be introduced to "repeat until" loop and how to set the condition to implicitly track the condition that stops the game. They will also learn how to use variable in a different situations: to set a random value, as a counter or to get the players input.	**shelter**

Slovenski scenariji

N.	NASLOV	KODE
1	**Uvod v Snap!** Učenec doda nov lik, doda obleko, uredi obleko in izbriše obleko/lik. Učenec ustvari novo ozadje, ga uredi, zbriše odvečna ozadja.	**uvod_slo**
2	**Razišči Snap!: premikaj lik** To uro boš spoznal/a, kako z bloki likom naročimo naj se premikajo po odru in govorijo.	**razisci_slo**
3	**Premikanje po odru** Učenec se nauči, kako premikati lik v x in y smeri, kako napisati preprost program ter kako obrniti lik v drugo smer	**premikanje_slo**
4	**Menjava obleke in obrat** Učenec se nauči, kako liku zamenjati obleko in kako narediti animacijo z obračanji lika	**obleka_slo**
5	**Zvoki na kmetiji** Učenec se nauči programiranja preproste igre, v kateri lahko igralec s pritiskom na določene tipke prepozna zvoke živali	**kmetija_slo**
6	**Kameleon na počitnicah** Izdelaj preprosto igro, v kateri bo glavni objekt spreminjal svojo obleko glede na barvo ozadja na njegovi trenutni poziciji.	**kameleon_slo**
7	**Pomagaj princu in princeski najti svoje živali** Dekle mora pomagati princesi poiskati svojo mačko in princu svojega psa. Pri tem se mora izogniti srečanju med živalmi.	**iskanje_zivali_slo**
8	**Risanje s kredo** Učenec se seznani z risanjem različnih oblik s kodo. Nauči se uporabljati zanko ponovi za krajšanje kode ter kako zamenjati ozadje.	**kreda_slo**

9	**Pobiranje smeti in čiščenje parka** Učenec se nauči uporabljati spremenljivke in kako podvojiti blok kode ali celotno kodo lika.	**ciscenje_parka_slo**
10	**Nahrani mucke** Učenec se seznani s konceptom dodeljevanja več spremenljivk z naključno vrednostjo znotraj zanke in kakšna je razlika, če vrednost dodelimo zunaj zanke. Spoznali bodo tudi, kako pridobiti in preveriti igralčev odgovor ter kako šteti pravilne odgovore.	**mucke_slo**
11	**Mačje zavetišče** Učenec se seznani z zanko "ponavljaj dokler" in kako ustvariti pogoj za implicitno sledeje pogoju, ki konča z igro. Nauči se tudi, kako uporabljati spremenljivko v različnih situacijah: nastaviti naključno vrednost, kot števec ali kot igralčev vnos.	**zavetisce_slo**

Italijanski scenariji

N.	TITLE	CODE
1	**Snap!: introduzione** Lo studente impara ad aggiungere un nuovo sprite, a modifica il costume e ad eliminarne uno di essi. Impara a crea un nuovo sfondo per il palco, lo modifica e cancella quelli indesiderati.	**Primipassi**
2	**Scoprire Snap!** Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	**Snap!**
3	**Muoversi sullo "stage"** Il discente impara a spostare il suo Sprite in direzione x e y sullo "stage"; a preparare un semplice programma per risolvere i compiti assegnati; a far girare il suo Sprite in diverse direzioni.	**stage**
4	**Cambiare il costume e creare rotazioni** Lo studente impara a cambiare il costume dello sprite per realizzare un'animazione.	**ballerina**
5	**I suoni di una fattoria** Gli studenti imparano come programmare un gioco semplice in cui il giocatore può riconoscere i suoni degli animali premendo determinati tasti.	**fattoria**
6	**Vacanze estive di un Camaleonte** Imparare a programmare un gioco semplice dove l'oggetto cambia il proprio colore secondo il colore dello sfondo	**camaleonte**

7	**Alla ricerca degli animali del Principe e della Principessa!** La ragazza deve aiutare la principessa a trovare il suo gatto e il principe a trovare il suo cane evitando che gli animali possano incontrarsi durante il loro.	**labirinto**
8	**Disegnare con un gesso** Gli studenti impareranno a disegnare forme diverse con un codice e ad usare la ripetizione ciclica per abbreviare il codice e cambiare lo sfondo.	**gesso**
9	**Raccogliere la spazzatura e pulire il parco** Gli studenti impareranno ad usare le variabili e a duplicare un blocco di codice o anche un intero Sprite.	**spazzatura**
10	**Dare da mangiare ai gatti** Gli studenti impareranno il concetto di assegnazione di più valori casuali, variabili all'interno e al di fuori di un ciclo. Impareranno anche come ottenere, testare e contare gli input corretti immessi dal giocatore.	**gatto**
11	**Indovinare il numero di gatti in un rifugio** Gli studenti impareranno ad utilizzare il ciclo "ripeti fino a" e ad impostare la condizione per interrompere il gioco. Impareranno anche ad usare la variabile in situazioni diverse.	**rifugio**

Hrvaški scenariji

N.	**TITLE**	**CODE**
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** Učenici će se upoznati kako koristiti varijable i kako kopirati blok kôda ili čak cijeli objekt.	**park**
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**

1 1	**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**

Bolgarski scenariji

N.	TITLE	CODE
1	**Увод в Snap!** Ученикът добавя нов спрайт, добавя костюм към спрайта, редактира костюма и изтрива едино от двете. Ученикът създава нов фон на сцената, редактира го и го изтрива.	**start_bg**
2	**Да разучим Snap!: Движещ се спрайт** Помогнете на учениците да разучат интерфейса на Snap! и да съставят код за първия им спрайт, така че той да се движи и говори.	**dissnap_bg**
3	**Движейки се по сцената** ченикът се научава как да движи спрайта в x и y посока на сцената, създава лесна програма за решаване на задачите и се научава как да завърти спрайт в различна посока.	**monkey_bg**
4	**Смяна на костюми и завъртане** Ученикът се учи как да промени костюма на спрайт при завъртане, за да направи анимация.	**dancer_bg**
5	**Звуци от фермата** Учениците научават как да програмират игра, в която играчът може да разпознава звуците на животните чрез натискане на определени клавиши.	**sounds_bg**
6	**Лятната ваканция на хамелеона** Програмирайте проста игра, в която обект да променя костюма си в зависимост от цвета на фона.	**cham_bg**
7	**Помогнете на принца и принцесата да намерят своите домашни любимци** Момиче трябва да помогне на принцесата да намери котката си, а на принцът да намери кучето си, като трябва да избегне възможността животните да се срещнат по пътя.	**find_bg**
8	**Рисуване с тебешир** Учениците ще бъдат запознати с рисуването на различни фигури(форми) използвайки код. Те ще се научат да използват цикъл с повторение за намаляне обема на кода и за промяна на фона.	**draw_bg**

9	**Събиране на боклук и почистване на парка** Учениците ще научат как да използват променливи и как да копират блок с код или дори цял спрайт.	**clean_bg**
10	**Нахранете котките** Учениците ще бъдат въведени в концепцията за присвояване на случайно число на няколко променливи в цикъл, както и каква е разликата при използването им вътре и извън цикъл. Те също така ще научат как за да зададат, тестват и изброят правилно въведеното от играча.	**cats_bg**
11	**Познайте броя на котките в приюта** Учениците ще бъдат запознати с цикъл "repeat until" и как да зададат условието, при което спира играта. Те също така ще се научат как да използват променливи в различни ситуации: за задаване на произволна стойност, като брояч или като стойност въведена от играча.	**shelter_bg**

Grški scenariji

N.	TITLE	CODE
1	**Εισαγωγή στο Snap!** Ο μαθητής προσθέτει ένα νέο στοιχείο, προσθέτει ένα κοστούμι στο στοιχείο, επεξεργάζεται το κοστούμι και διαγράφει ένα από αυτά. Ο μαθητής δημιουργεί ένα νέο φόντο στη σκηνή, το επεξεργάζεται και διαγράφει τα ανεπιθύμητα.	**start_gr**
2	**Ανακαλύψτε το Snap!: μετακινήσε μία εικόνα** Βοήθησε τους μαθητές να ανακαλύψουν την πλατφόρμα προγραμματισμού Snap! και να κωδικοποιήσουν την πρώτη τους εικόνα ώστε να κινείται και να μιλάει.	**dissnap_gr**
3	**Ας μετακινηθούμε γύρω από τη σκηνή** Ο μαθητής μαθαίνει πώς να μετακινεί ένα στοιχείο σε κατεύθυνση x και y στη σκηνή, δημιουργεί ένα εύκολο πρόγραμμα για την επίλυση των καθηκόντων του, μαθαίνει πώς να γυρίζει το στοιχείο σε διαφορετική κατεύθυνση.	**monkey_gr**
4	**Αλλαγή κοστουμιών και στροφή** Ο μαθητής μαθαίνει πώς να αλλάζει το κοστούμι του στοιχείου για να κάνει ένα κινούμενο σχέδιο αλλάζοντας μεταξύ διαφορετικών τύπων περιστροφής.	**dancer_gr**
5	**Ήχοι φάρμας** Οι μαθητές μαθαίνουν πώς να προγραμματίζουν ένα απλό παιχνίδι στο οποίο ο παίκτης μπορεί να αναγνωρίζει τους ήχους των ζώων πατώντας συγκεκριμένα κουμπιά.	**sounds_gr**

6	**Καλοκαιρινές διακοπές του Χαμαιλέοντα** Προγραμματίστε ένα απλό παιχνίδι στο οποίο ένα αντικείμενο θα αλλάξει το κοστούμι του με βάση το χρώμα του φόντου.	**cham_gr**
7	**Βοηθώντας τον Πρίγκιπα και την Πριγκίπισσα να βρουν τα ζώα τους** Το κορίτσι πρέπει να βοηθήσει την Πριγκίπισσα να βρει τη γάτα της και τον Πρίγκιπα να βρει τον σκύλο του, αποφεύγοντας τα υπόλοιπα ζώα που μπορεί να συναντήσουν κατά τη διάρκεια της πορείας τους.	**find_gr**
8	**Ζωγραφίζοντας με κιμωλία** Οι μαθητές θα μάθουν να ζωγραφίζουν διαφορετικά σχήματα με κώδικα. Θα μάθουν να χρησιμοποιούν βρόχους επανάληψης για να συντομεύσουν τον κώδικα και να αλλάξουν φόντο.	**draw_gr**
9	**Μαζεύοντας σκουπίδια και καθαρίζοντας το πάρκο** Οι μαθητές θα μάθουν πώς να χρησιμοποιούν μεταβλητές και πώς να αντιγράφουν ένα κομμάτι κώδικα ή ακόμα και ένα ολόκληρο στοιχείο.	**clean_gr**
10	**Ταΐζοντας τις γάτες** Οι μαθητές θα μάθουν την έννοια της πολλαπλής ανάθεσης τυχαίας τιμής σε μεταβλητές μέσα σε ένα βρόχο και πώς αυτό είναι διαφορετικό αν το κάνουμε εκτός βρόχου. Θα μάθουν επίσης για τον τρόπο λήψης, δοκιμής και μέτρησης των σωστών εισόδων από τον παίκτη.	**cats_gr**
11	**Μαντέψτε τον αριθμό των γατιών στο καταφύγιο** Οι μαθητές θα μάθουν το βρόχο «επανάληψη έως» και πώς να ρυθμίσουν τη συνθήκη ώστε να παρακολουθούν τη συνθήκη που σταματά το παιχνίδι. Θα μάθουν επίσης πώς να χρησιμοποιούν τη μεταβλητή σε διαφορετικές καταστάσεις: για να ορίσουν μια τυχαία τιμή, ως μετρητή ή για να πάρουν είσοδο από τον παίκτη.	**shelter_gr**

Portugalski scenariji

N.	TITLE	CODE
1	**Introdução à interface Snap!** Os alunos adicionam um novo sprite, adicionam um traje ao sprite, editam o traje e excluem um deles. O aluno cria um novo plano de fundo para o palco, edita-o e exclui os indesejados.	**starting_PT**
2	**Descubra o Snap! : mover um Sprite** Os alunos descobrem a interface do Snap! e codificam o seu primeiro sprite para que ele se mova e fale.	**dispubeng_PT**

3	**Movendo-se pelo palco** Os alunos aprendem como mover o sprite nas direções x e y do palco, codificam um programa fácil para resolver as tarefas dadas e aprendem a virar o sprite em uma direção diferente.	**monkey_PT**
4	**Mudando de traje e girando** Os alunos aprendem a mudar o traje do sprite para fazer uma animação e alternam entre os diferentes tipos de rotação.	**dancer_PT**
5	**Sons da quinta** Os alunos aprendem a programar um jogo simples no qual o jogador pode ouvir os sons dos animais pressionando certas teclas.	**farm_PT**
6	**Férias de verão do camaleão** Os alunos programam um jogo simples em que um objeto muda de traje de acordo com a cor do fundo	**chameleoneng_PT**
7	**Ajudando o príncipe e a princesa a encontrar seus animais** Uma menina tem que ajudar a Princesa a encontrar seu gato e o Príncipe a encontrar seu cachorro, evitando que os animais se cruzem durante o caminho.	**finding_PT**
8	**Desenho com giz** Os alunos aprendem a desenhar formas diferentes com um código. Aprendem a usar ciclos para encurtar o código e alterar um plano de fundo.	**chalk_PT**
9	**Recolher o lixo e limpar o parque** Os alunos aprendem a usar variáveis, a duplicar um bloco de código e um sprite.	**cleaning_PT**
10	**Alimentando os gatos** Os alunos aprendem o conceito de atribuição de valores aleatórios a múltiplas variáveis dentro de um loop e como é diferente de quando o fazemos fora de um loop. Eles também aprendem a obter, testar e contar as entradas corretas do jogador.	**feeding_PT**
11	**Adivinhando o número de gatos em um abrigo** Os alunos aprendem o ciclo "repetir até" e como definir a condição para rastrear a condição que interrompe o jogo. Aprendem ainda a usar variáveis em diferentes situações: definir um valor aleatório, como um contador ou para obter dados dos jogadores.	**shelter_PT**

Turški scenariji

N.	TITLE	CODE
1	**Snap! ara yüzünü tanıyalım** Öğrenciler yeni bir karakter (sprite/kukla) ekler, karaktere bir kostüm ekler, kostümü düzenler ve bunlardan birini siler. Öğrenciler sahne için yeni bir arka plan oluşturur, düzenler ve istenmeyenleri siler.	**başla**
2	**Snap!'i keşfet - Karakteri hareket ettirme** Öğrencilerin Snap! arayüzünü keşfetmelerine ve ilk kodlarını oluşturarak hareket eden ve konuşan bir karakter oluşturmalarına yardımcı olun.	**keşfet**
3	**Sahnede hareket etme** Öğrenciler sahnede karakteri x ve y yönünde hareket ettirmeyi öğrenir, verilen görevleri çözmek için basit bir program programlar, karakteri farklı yöne çevirmeyi öğrenir.	**hareket**
4	**Kostüm değiştirme ve dönüşler** Öğrenci, farklı rotasyon türleri arasında geçiş yaparak bir animasyon yapmak için karakerin kostümünü nasıl değiştireceğini öğrenir.	**dans**
5	**Çiftlikteki sesler** Öğrenciler, oyuncunun belirli tuşlara basarak hayvanların seslerini tanıyabileceği basit bir oyun programlamayı öğrenirler.	**çiftlik**
6	**Bukalemenun yaz tatili** Bir nesnenin kostümünü arka planın rengine göre değiştireceği basit bir oyun programlayın	**bukalemun**
7	**Prens ve Prensese evcil hayvanlarını bulmaları için yardım et** Kız, seyahatleri sırasında hayvanların birbirleriyle karşılaşmasını engelleyerek Prenses'in kedisini bulmasına ve Prens'in köpeğini bulmasına yardım etmelidir.	**prenses**
8	**Tebeşir ile çizim yapmak** Öğrencilere bir kodla farklı şekiller çizme tanıtılacaktır. Kodu kısaltmak ve arka planı değiştirmek için döngü tekrarını kullanmayı öğrenecekler.	**çizim**
9	**Çöpleri toplamak ve parkı temizlemek** Öğrenciler değişkenleri nasıl kullanacaklarını ve bir kod bloğunu veya hatta bütün bir hareketli grafiği nasıl kopyalayacaklarını öğrenecekler.	**çöpler**

10	**Kedileri besleme** Öğrencilere, bir döngü içinde çok değişkenli rastgele değer atama kavramı ve bir döngü dışında yaptığımızdan nasıl farklı olduğu anlatılacaktır. Ayrıca doğru oyuncu girdilerinin nasıl alınacağını, test edileceğini ve sayılacağını da öğrenecekler.	**kediler**
11	**Barınaktaki Kedi Sayısını Tahmin Etmek** Öğrencilere "----e kadar tekrar et" döngüsü ve oyunu durduran koşulu örtük olarak izlemek için koşulun nasıl ayarlanacağı tanıtılacaktır. Ayrıca değişkeni; rastgele bir değer belirlemek, sayaç olarak veya oyuncuların girdisini almak gibi farklı durumlarda nasıl kullnacaklarını da öğrencekler.	**barınak**

Napredni učni scenariji

Angleški scenariji

N.	TITLE	CODE
12	**Catching healthy food** Students will learn how to randomly move for X steps and choose a position and also how to use variables and conditionals for preventing other event.	**food**
13	**Storytelling** Students will learn how to plan storytelling, how to use broadcast messages for synchronisation of the activities of sprites and stage changes.	**storytelling**
14	**Drawing** Student will learn to draw in Snap!, to change the colour and the pen thickness, and how to use variables and conditionals that cause a new event.	**drawing**
15	**Catch the mouse** Student will be introduced to the concept of multiple variable random value assignment. They will learn how to use the Operators/pick random[x]to[y] block.	**mouse**
16	**Buying food for a picnic** Students will learn how to work with variables: setting different starting values, using conditionals to compare variables' value, changing variables' value, using variables for counting (un)healthy food. In addition, they will repeat adding text, joining texts and if statement.	**picnic**

17	**Operations** Students will improve their knowledge on variables, random numbers, loops, broadcast.	**operation**
18	**Recycling** Students will improve their knowledge and will extend the game scenario with new objects, code and changing code with respect to new sprites. They will be trained in code refactoring.	**recycling**
19.1	**Play a piano_1** Students will learn about melodies coding and playing and will improve their knowledge about variables, loops, conditional, broadcast and other events.	**music**
19.2	**Play a piano_2** Students will learn how to play music and change costume by clicking on a sprite.	**piano**
20	**Test** Students will improve their knowledge and will extend the game scenario with new background, code and changing code with respect to new stages.	**test**
21	**Simplified PACMAN game** Students will review their knowledge about movement inside a labyrinth with the use of sense color block. They will learn the concept of cloning the object with position restrictions and how to create a very simple non player character with its own random movement.	**pacman**

Slovenski scenariji

N.	NASLOV	KODE
12	**Lovljenje zdrave hrane** Učenec se nauči naključnega premikanja za X korakov in naključne izbire pozicije ter uporabo spremenljivk in pogojnih stavkov za preprečevanje drugih dogodkov.	**lovljenje_hrane_slo**
13	**Sestavi zgodbo** Učenec se nauči načrtovanja pripovedovanja zgodb, uporabe pošiljanja obvestil za sinhronizacijo dejavnosti likov ter zamenjave ozadij.	**zgodba_slo**
14	**Onesnažen zrak** Učenec se nauči risanja, spreminjanja barve, debeline svinčnika ter uporabe spremenljivke in pogojev za izvedbo drugega dogodka.	**zrak_slo**

15	**Ulovi miš** Učenec se seznani s konceptom dodeljevanja naključnih vrednosti in se nauči, kako uporabiti operator naključno število od_[x]_do_[y].	**mis_slo**
16	**Kupovanje hrane za piknik**	**piknik_slo**
17	**Računanje** Učenec izboljša svoje znanje o spremenljivkah, naključnih številkah, zankah ter pošiljanju obvestil.	**racunanje_slo**
18	**Recikliranje**	**recikliranje_slo**
19.1	**Zaigraj na klavir 1** Učenec se nauči sestavljanja melodij ter izboljša svoje znanje o spremenljivkah, zankah, pogojih, pošiljanju obvestil ter drugih dogodkov.	**ples_slo**
19.2	**Zaigraj na klavir 2** Učenec se nauči predvajati glasbo in liku spremeniti obleko s klikom nanj.	**klavir_slo**
20	**Test** Učenec izboljša svoje znanje in igro razširi z novimi ozadji, doda in spremeni kodo.	**test_slo**
21	**Enostavni Pacman** Učenec ponovi s pomočjo zaznavanja barve liku omejiti gibanje, spozna koncept kloniranja z omejitvami gibanja in se nauči ustvarjanja lika, ki se samostojno naključno premika po labirintu.	**pacman_slo**

Italijanski scenariji

N.	TITLE	CODE
12	**Raccogliere i cibi salutari** Gli studenti impareranno a muoversi casualmente per X passi, a scegliere una posizione, ad usare variabili e condizioni per prevenire altri eventi.	**cibo**
13	**Alice nel Paese delle Meraviglie** Gli studenti impareranno a pianificare una storia, ad utilizzare i messaggi inviati e ricevuti per la sincronizzazione delle attività degli Sprite e per i cambiamenti di scena.	**alice**
14	**Disegno** Lo studente imparerà a disegnare con Snap!, a cambiare il colore e lo spessore della penna e ad usare le variabili e le condizioni che determinano nuovi eventi.	**disegno**

15	**Alla caccia di un topolino!** Gli studenti comprenderanno il concetto di assegnazione di più valori casuali variabili. Impareranno come usare gli Operatori / scegliere il blocco casuale da [x] a [y].	**topolino**
16	**Acquistare cibo per un picnic** Gli studenti impareranno a lavorare con le variabili: impostare diversi valori iniziali, usare i condizionali per confrontare il valore delle variabili, cambiare il valore delle variabili, usare le variabili per contare alimenti sani (e non). Inoltre, aggiungeranno il testo, useranno l'unione di testi e la condizione "if".	**picnic_1**
17	**Operazioni** **Gli studenti miglioreranno le loro conoscenze su variabili, numeri casuali, cicli e broadcast.**	**operazioni**
18	**Raccolta differenziata** Gli studenti miglioreranno le loro conoscenze per estendere lo scenario di gioco con nuovi oggetti, codice e cambiando codice rispetto ai nuovi Sprite.	**riciclo**
19.1	**Suonare il piano_1** Gli studenti impareranno a codificare le melodie e miglioreranno le loro conoscenze su variabili, loop, condizionale, trasmissione e altri eventi.	**music_1**
19.2	**Suonare il piano_2** Gli studenti impareranno a suonare e a cambiare costume facendo clic su uno Sprite	**piano_2**
20	**Test** Gli studenti miglioreranno le loro conoscenze ed estenderanno lo scenario del gioco con nuovi background, codici e sue modifiche.	**test_1**
21	**PACMAN** Gli studenti rivedranno le loro conoscenze sul movimento all'interno di un labirinto con l'uso dei blocchi dei sensori del colore. Impareranno il concetto di clonazione dell'oggetto con restrizioni di posizione e a creare un personaggio "non giocatore".	**pacman_1**

Hrvaški scenariji

N.	TITLE	CODE
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** Studenti će upoznati kodiranje i sviranje melodija te će poboljšati svoje prethodno stečeno znanje o varijablama, petlji, uvjetnim, emitiranim i ostalim događajima.	**glazba**
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**

Bolgarski scenariji

N.	TITLE	CODE
12	**Подбор на здравословна храна** Учениците ще се научат как да се придвижат на случаен принцип с X стъпки и да изберат позиция, както и как да използват променливи и условности за предотвратяване на друго събитие.	**food_bg**
13	**Разказвач** Учениците ще научат как да планират разказването на истории, как да използват излъчвани съобщения за синхронизиране на дейностите на спрайтове и промени на сцената.	**story_bg**
14	**Рисуване** Ученикът ще се научи да рисува в Snap !, да променя цвета и дебелината на писалката и да използва променливи и условия, които причиняват ново събитие.	**drawing_bg**
15	**Хванете мишката** Учениците ще бъде запознати с концепцията за присвояване на случвайни стойности на променливи. Ще се научат как да използват блока Operators/pick random[x]to[y] block.	**mouse_bg**
16	**Купуване на храна за пикник** Учениците ще се научат как да работят с променливи: задаване на различни начални стойности, използване на условия за сравняване на стойностите на променливите, промяна на стойностите на променливите, използване на променливи за броене на (не) здравословна храна. В допълнение, те ще разучат добавяне на текст, съединяване на текстове и if оператор.	**picnic_bg**
17	**Операции** Учениците ще подобрят знанията си за променливи, случайни числа, цикли, Broadcast.	**oper_bg**

18	**Рециклиране** Учениците ще подобрят знанията си и ще разширят сценария на играта с нови обекти, код и променлив код по отношение на новите спрайтове. Те ще бъдат обучени как да редактират кодове.	**recycling_bg**
19.1	**Посвири на пиано Версия 1** Учениците ще се запознаят с кодирането и свиренето на мелодии и ще подобрят знанията си за променливи, цикли, условия, излъчване и други събития.	**music_bg**
19.2	**Посвири на пиано Версия 2** Учениците ще се научат как да свирят музика и да сменят костюма, като кликнат върху спрайт.	**piano_bg**
20	**Тест** Учениците ще подобрят знанията си и ще разширят сценария на играта с нов фон, код и променлив код по отношение на новите сцени.	**test_bg**
21	**Опростена игра PACMAN** Учениците ще приложат своите знания за движение в лабиринт с помощта на сензорен цветен блок. Те ще научат концепцията за клониране на обект с ограничения за позицията и как да създадат много прост персонаж, който не е играч, с опция за произволно движение.	**pacman_bg**

Grški scenariji

N.	TITLE	CODE
12	**Πιάνοντας υγιεινά τρόφιμα** Οι μαθητές θα μάθουν πώς να κινούνται για Χ τυχαία βήματα και να επιλέγουν μια θέση και επίσης πώς να χρησιμοποιούν μεταβλητές και συνθήκες για την πρόληψη άλλου συμβάντος.	**food_gr**
13	**Διήγηση ιστορίας** Οι μαθητές θα μάθουν πώς να σχεδιάζουν την αφήγηση μιας ιστορίας, πώς να χρησιμοποιούν τα μηνύματα μετάδοσης για συγχρονισμό των δραστηριοτήτων των στοιχείων με τις αλλαγές στη σκηνή.	**story_gr**
14	**Ζωγραφική** Ο μαθητής θα μάθει να ζωγραφίζει με το Snap!, να αλλάζει το χρώμα και το πάχος του στυλό και πώς να χρησιμοποιεί μεταβλητές και συνθήκες για ένα νέο συμβάν.	**drawing_gr**

15	**Πιάσε το ποντίκι** Ο μαθητής θα μάθει την έννοια της πολλαπλής ανάθεσης τυχαίας τιμής. Θα μάθει πώς να χρησιμοποιεί τους τελεστές / επιλέγει τυχαία [x] έως [y].	**mouse_gr**
16	**Αγοράζοντας φαγητό για πικνίκ** Οι μαθητές θα μάθουν πώς να δουλεύουν με μεταβλητές: ανάθεση διαφορετικής αρχικής τιμής, χρήση συνθήκης για σύγκριση της τιμής των μεταβλητών, αλλαγή της τιμής των μεταβλητών, χρήση μεταβλητών για την καταμέτρηση (μη) υγιεινών τροφίμων. Επιπλέον, θα επαναλάβουν την προσθήκη κειμένου, την ένωση κειμένων και τη δήλωση εάν.	**picnic_gr**
17	**Λειτουργίες** Οι μαθητές θα βελτιώσουν τις γνώσεις τους σχετικά με μεταβλητές, τυχαίους αριθμούς, βρόχους και μετάδοση.	**oper_gr**
18	**Ανακύκλωση** Οι μαθητές θα βελτιώσουν τις γνώσεις τους και θα επεκτείνουν το σενάριο του παιχνιδιού με νέα αντικείμενα, κώδικα και ανακατασκευή κώδικα σε σχέση με τα νέα στοιχεία. Θα εκπαιδευτούν στην ανακατασκευή κώδικα.	**recycling_gr**
19.1	**Παίξτε πιάνο_1** Οι μαθητές θα μάθουν να γράφουν κώδικα και να παίζουν μελωδίες και θα βελτιώσουν τις γνώσεις τους σχετικά με μεταβλητές, βρόχους, συνθήκες, μετάδοση γεγονότων και άλλα γεγονότα.	**music_gr**
19.2	**Παίξτε πιάνο_2** Οι μαθητές θα μάθουν πώς να παίζουν μουσική και να αλλάζουν κοστούμι κάνοντας κλικ σε ένα στοιχείο.	**piano_gr**
20	**Δοκιμαστικό μάθημα - Αλλαγή εμφάνισης και στροφή** Μάθετε πώς να αλλάζετε την εμφάνιση του στοιχείου για να κάνετε ένα κινούμενο σχέδιο, αλλάζοντας μεταξύ διαφορετικών τύπων περιστροφών.	**test_gr**
21	**Απλοποιημένο παιχνίδι PACMAN** Οι μαθητές θα επανεξετάσουν τις γνώσεις τους σχετικά με την κίνηση μέσα σε ένα λαβύρινθο. Θα μάθουν την έννοια της κλωνοποίησης του αντικειμένου με περιορισμό θέσης και πώς να δημιουργήσουν έναν πολύ απλό χαρακτήρα με τη δική του τυχαία κίνηση.	**pacman_gr**

Portugalski scenariji

N.	TITLE	CODE
12	**Comendo comida saudável** Os alunos aprendem a mover-se aleatoriamente X passos e a escolher uma posição. Aprendem a usar variáveis e instruções condicionais para prevenir um evento.	**food_PT**
13	**Narrativa** Os alunos aprendem a planejar a narração de histórias e como usar mensagens para sincronização das atividades dos sprites e mudanças de palco.	**storytelling_PT**
14	**Desenhando** Os alunos aprendem a desenhar no Snap!, a alterar a cor e a espessura da caneta e a usar variáveis e condicionais que geram um novo evento.	**drawing_PT**
15	**Apanhar o rato** Os alunos aprendem o conceito de atribuição de valores aleatórios de múltiplas variáveis. Aprendem ainda como usar o bloco de operadores de escolha aleatória [x] a [y].	**mouse_PT**
16	**Comprando comida para um piquenique** Os alunos aprendem a trabalhar com variáveis: valores iniciais, operadores condicionais para comparar o valor das variáveis, alterar o valor das variáveis, usar variáveis para contar alimentos saudáveis. Além disso, eles vão adicionar e juntar textos.	**picnic_PT**
17	**Operações** Os alunos vão melhorar os seus conhecimentos sobre variáveis, números aleatórios e ciclos.	**operation_PT**
18	**Reciclagem** Os alunos vão aumentar um cenário do jogo com novos objetos, código e código em relação a novos sprites. Aprendem ainda sobre refatoração de código.	**recycling_PT**
19.1	**Tocar piano_1** Os alunos aprendem sobre codificação e reprodução de melodias e melhoram os seus conhecimentos sobre variáveis, ciclos, instruções condicionais e outros eventos.	**music_PT**
19.2	**Tocar piano_2** Os alunos aprendem a tocar música e a mudar de traje clicando num sprite.	**piano_PT**
20	**Teste** Os alunos irão melhorar os seus conhecimentos alargando o cenário do jogo com um novo fundo, código e código mutável em relação a novos palcos.	**test_PT**

21	**Jogo PACMAN simplificado** Os alunos irão rever os seus conhecimentos sobre o movimento dentro de um labirinto com o uso do bloco de reconhecimento de cores. Aprendem o conceito de clonar o objeto com restrições de posição e como criar um personagem não-jogador muito simples com o seu próprio movimento aleatório.	**pacman_PT**

Turški scenariji

N.	TITLE	CODE
12	**Sağlıklı yiyeceği yakalamak** Öğrenciler, belirli bir adım atmak için rastgele hareket etmeyi, bir konum seçmeyi ve ayrıca diğer olayları önlemek için değişkenleri ve koşulluları nasıl kullanacaklarını öğreneceklerdir.	**yemek**
13	**Hikaye Anlatma** Öğrenciler hikaye anlatımını nasıl planlayacaklarını, karakterin aktiviteleri ile sahne değişikliklerinin senkronizasyonu için yayın mesajlarının nasıl kullanılacağını öğrenecekler	**hikaye**
14	**İklimi İyileştirmek-Çizim Yapma** Öğrenci, Snap!'te çizim yapmayı, rengi ve kalem kalınlığını değiştirmeyi ve yeni bir olaya neden olan değişkenler ile koşullu ifadeleri nasıl kullanacağını öğrenecek.	**iklim**
15	**Fareyi yakalamak** Öğrencilere çok değişkenli rastgele değer atama kavramı tanıtılacaktır. Operatörleri nasıl kullanacaklarını ve rastgele [x] ila [y] bloğu seçmeyi öğrenecekler.	**fare**
16	**Piknik için yemek almak** Öğrenciler değişkenlerle nasıl çalışılacağını öğrenecekler: farklı başlangıç değerleri belirleme, değişkenlerin değerini karşılaştırmak için koşullu ifadeler kullanma, değişkenlerin değerini değiştirme, sağlıklı yiyecekleri saymak (olmayan) için değişkenler kullanma. Ek olarak, metin eklemeyi, metinleri birleştirmeyi ve Eğer (if) ifadesini tekrarlayacaklar.	**piknik**
17	**İşlemler** Öğrenciler değişkenler, rastgele sayılar, döngüler, yayın hakkındaki bilgilerini geliştireceklerdir.	**işlem**

18	**Geri Dönüşüm** Öğrenciler bilgilerini geliştirecek ve oyun senaryosunu yeni karakterlere göre yeni nesneler, kodlar ve değişen kodlarla genişletecekler. Kod yeniden düzenleme konusunda eğitilecekler	**dönüşüm**
19.1	**Piyano Çalmak** Öğrenciler melodileri kodlama ve çalma hakkında bilgi edinecek ve değişkenler, döngüler, koşullu, yayın ve diğer olaylar hakkındaki bilgilerini geliştireceklerdir.	**Piyano1**
19.2	**Piyano Çalmak** Öğrenciler bir karaktere tıklayarak müzik çalmayı ve kostümü değiştirmeyi öğrenecekler.	**piyano2**
20	**Test- Kostümleri değiştirmek ve Dönüşler** Farklı döndürme türleri arasında geçiş yaparak bir animasyon yapmak için karakterin kostümünü nasıl değiştireceğinizi öğrenin	**test**
21	**Basitleştirilmiş PACMAN oyunu** Öğrenciler duyu renk bloğu kullanarak bir labirent içindeki hareketin nasıl kodlanması gerektiği hakkındaki bilgilerini gözden geçireceklerdir. Nesneyi konum kısıtlamaları ile rastgele hareket edecek şekilde basitçe kodlayacakar ve karakterleri klonlamayı öğrenecekler.	**pacman**

VIRI

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

Alice - Tell Stories. Build Games. Learn to Program. Pridobljeno iz http://www.alice.org/

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).

CodeCombat - Learn how to code by playing a game. Pridobljeno iz https://codecombat.com/

Coding for Kids | Tynker. Pridobljeno iz https://www.tynker.com/

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.* Pridobljeno iz 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. Pridobljeno iz 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. Pridobljeno iz 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.

LightBot. Pridobljeno iz http://lightbot.com/

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.

Scratch - Imagine, Program, Share. Pridobljeno iz https://scratch.mit.edu/

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.

Snap! (Build Your Own Blocks) 4.2. Pridobljeno iz https://snap.berkeley.edu/

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. Pridobljeno iz 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.