**O3 – Izobraževalne vsebine za učitelje**

**Zbirka učnih scenarijev za učenje programiranja na osnovi oblikovanja iger**

**Podatki o dokumentu**

**Rezultat**: O3/A1 – Zbirka učnih scenarijev za učenje programiranja na osnovi oblikovanja iger

**Intelektualni rezultat:** O3 – Izobraževalne vsebine za učitelje

**Odgovorni partner:** South-West University “Neofit Rilski” (Bolgarija)

**Sodelujoči partnerji:** Univerza v Ljubljani (Slovenija), Univerza na Reki (Hrvaška)

**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.

Coding4Girls, 2018-2020

https://licensebuttons.net/l/by-sa/3.0/88x31.png

## Creative Commons Attribution-ShareAlike 4.0

## International Public License ([CC BY-SA 4.0](https://creativecommons.org/licenses/by-sa/4.0/))

**KAZALO**

[1. UVOD 4](#_Toc46233838)

[2. UČNI SCENARIJI 5](#_Toc46233839)

[OSNOVNI UČNI SCENARIJI 6](#_Toc46233840)

[Učni scenarij 1 – Uvod v okolje Snap! 6](#_Toc46233841)

[Učni scenarij 2 – Lik oživi 11](#_Toc46233842)

[Učni scenarij 3 – Premikanje po odru 15](#_Toc46233843)

[Učni scenarij 4 – Menjava obleke in obrat 21](#_Toc46233844)

[Učni scenarij 5 – Zvoki na kmetiji 28](#_Toc46233845)

[Učni scenarij 6 – Kameleon na počitnicah 36](#_Toc46233846)

[Učni scenarij 7 – Pomagaj princu in princeski najti svoje živali 46](#_Toc46233847)

[Učni scenarij 8 – Risanje s kredo 53](#_Toc46233848)

[Učni scenarij 9 – Pobiranje smeti in čiščenje parka 63](#_Toc46233849)

[Učni scenarij 10 – Nahrani mačke 71](#_Toc46233850)

[Učni scenarij 11 – Mačje zavetišče 79](#_Toc46233851)

[NAPREDNEJŠI UČNI SCENARIJI 88](#_Toc46233852)

[Učni scenarij 12 – Lovljenje zdrave hrane 88](#_Toc46233853)

[Učni scenarij 13 – Sestavi zgodbo 96](#_Toc46233854)

[Učni scenarij 14 – Onesnažen zrak 106](#_Toc46233855)

[Učni scenarij 15 – Ulovi miš 117](#_Toc46233856)

[Učni scenarij 16 – Kupovanje hrane za piknik 125](#_Toc46233857)

[Učni scenarij 17 – Računanje 133](#_Toc46233858)

[Učni scenarij 18 – Recikliranje 141](#_Toc46233859)

[Učni scenarij 19.1 – Zaigraj na klavir 1 144](#_Toc46233860)

[Učni scenarij 19.2 – Zaigraj na klavir 2 151](#_Toc46233861)

[Učni scenarij 20 – Test 158](#_Toc46233862)

[Učni scenarij 21 – Enostavni PACMAN 161](#_Toc46233863)

[3. VIRI 168](#_Toc46233864)

1. UVOD

Vodilni psiholog prejšnjega stoletja je identificiral igro kot eno najpomembnejših aktivnosti za razvoj pomembnih življenjskih veščin, ne glede na starost ali stopnjo razvoja. Otrok se skozi igro hitro prilagaja in sprejema nove okoliščine. Med igro odkriva osnovne pojme iz resničnega sveta in na ta način se oblikujejo prvi odnosi med njima.

Dandanes se igre najpogosteje uporabljajo v zgodnjih fazah otrokovega razvoja doma in v vrtcu, v šoli pa učenje še vedno prepogosto temelji na tradicionalnem prenosu znanja z učiteljem v glavni vlogi in pasivnimi učenci. Po drugi strani pa teorije učenja, razvite v prejšnjem stoletju, spodbujajo k novim pristopom učenja in poučevanja, ki so usmerjeni na učenca, temeljijo na reševanju problemov, so usmerjeni k doseganju višjih taksonomskih ravni, predstavljajo motivacijo za učence in so pogosto podprti z informacijsko-komunikacijsko tehnologijo (IKT).

S pristopom CODING4GIRLS bomo učence spodbudil k sodelovanju v programerskih aktivnostih, kjer bodo najprej reševali lažje primere, kasneje pa bodo prešli na čedalje zahtevnejše izzive, namenjene predvsem uspešnejšim učencem (*low entry - high ceiling approach*). Učence bomo spodbudili k dokončevanju delnih rešitev ali k ustvarjanju popolnoma novih. Aktivnosti si sledijo po težavnosti: od osnovnih z enim programerskim konceptom do zahtevnejši z več programerskimi koncepti. Pripravljene aktivnosti v vizualnem programskem jeziku Snap! smo poskušali povezati z resničnimi problemi ter upoštevati preference deklet pri igranju iger.

Pripravljeni učni scenariji vsebujejo vse informacije, ki jih potrebujejo učitelji za uporabo pristopa učenja programiranja z izdelavo iger in s tem gradnje programerskih veščin za deklice in dečke. V učnih scenarijih so na voljo:

* pričakovano predznanje,
* splošni učni cilji,
* koncepti, ki jih aktivnost pokriva,
* specifični učni cilji,
* pričakovani rezultati,
* navodila po korakih za izvedbo Coding4Girls pristopa učenja z izdelavo iger,
* vprašanja za spodbujanje razprave med učenci v razredu.

Pripravili smo več kot 20 učnih scenarijev. Učitelji lahko uporabijo scenarije v predlaganem zaporedju ali pa jih prilagodijo svojim željam in potrebam. Scenariji opisujejo splošno funkcionalnost predlagane izobraževalne igre, načine interakcije z uporabniki, povratne informacije in opise učnih aktivnosti v predlagani igri.

Učni scenariji so na voljo v angleškem, bolgarskem, grškem, hrvaškem, italijanskem, portugalskem, turškem in slovenskem jeziku.

1. UČNI SCENARIJI

Pripravljeni učni scenariji so razvrščeni v dva dela: osnovni učni scenariji pokrivajo en programerski koncept, naprednejši pa več. Spodnja tabela prikazuje predlagan vrstni red aktivnosti.

|  |  |  |
| --- | --- | --- |
| **OSNOVNI UČNI SCENARIJI** | | |
| 1 | **Uvod v okolje Snap!** | UL |
| 2 | **Lik oživi** | UL |
| 3 | **Premikanje po odru** | UL |
| 4 | **Menjava obleke in obrat** | UL |
| 5 | **Zvoki na kmetiji** | UL |
| 6 | **Kameleon na počitnicah** | UL |
| 7 | **Pomagaj princu in princeski najti svoje živali** | UL |
| 8 | **Risanje s kredo** | UL |
| 9 | **Pobiranje smeti in čiščenje parka** | UL |
| 10 | **Nahrani mucke** | UL |
| 11 | **Mačje zavetišče** | UL |
| **NAPREDNEJŠI UČNI SCENARIJI** | | |
| 12 | **Lovljenje zdrave hrane** | UL |
| 13 | **Sestavi zgodbo** | SWU |
| 14 | **Onesnažen zrak** | UNIRI |
| 15 | **Ulovi miš** | UL |
| 16 | **Kupovanje hrane za piknik** | 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** | UL |

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 C:\Users\Špela\Downloads\C4G_sprite_background script pic.png z levega dela okna na del, ki je namenjen sestavljanju skript, to je delčkov programa:    Če sedaj dvakrat kliknete na delček C:\Users\Špela\Downloads\C4G_sprite_background script pic.png, 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:  C:\Users\Špela\Downloads\C4G_sprite_background script pic (2).png  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 C:\Users\Špela\Downloads\Neimenovano script pic (2).png in C:\Users\Špela\Downloads\C4G_sprite_background script pic (3).png. O razliki se pogovori s sošolcem.  [6. korak]  Oba delčka reci ste našli v kategoriji Izgled. Glavna razlika med njima je, da pri delčku C:\Users\Špela\Downloads\Neimenovano script pic (2).png 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 C:\Users\Špela\Downloads\C4G_sprite_background script pic (5).png. 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](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 /Users/mateja.bevcic/Downloads/Neimenovan script pic.png in /Users/mateja.bevcic/Downloads/Neimenovan script pic (1).png.  Možna rešitev naloge:  [C:\Users\Špela\Downloads\C4G3_Ujemi_zogo_resitev script pic.png](https://snap.berkeley.edu/project?user=mateja&project=C4G3_Ujemi_zogo_resitev)  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 /Users/mateja.bevcic/Desktop/slike za navodila/C4G3_Plezanje_opice script pic.png in /Users/mateja.bevcic/Downloads/Neimenovan script pic (1).png . 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:  [C:\Users\Špela\Downloads\C4G3_Plezanje_opice_resitev script pic.png](https://snap.berkeley.edu/project?user=mateja&project=C4G3_Plezanje_opice_resitev)  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 C:\Users\Špela\Downloads\C4G3_Plezanje_opice_resitev script pic (2).png.  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 C:\Users\Špela\Downloads\C4G3_Plezanje_opice_resitev script pic (3).png.  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:  C:\Users\Špela\Downloads\C4G3_Ujemi_zogo_resitev script pic (1).png  Koda za opico: C:\Users\Špela\Downloads\C4G3_Plezanje_opice_resitev script pic (4).png  \*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](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](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](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](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.  /Users/mateja.bevcic/Desktop/slike za navodila/Screen Shot 2020-02-10 at 20.37.28.png  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 /Users/mateja.bevcic/Desktop/slike za navodila/Neimenovano script pic (1).png. 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:  C:\Users\Špela\Downloads\C4G4_Balerina script pic.png  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:  C:\Users\Špela\Downloads\C4G4_Balerina script pic (1).png  [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:  [C:\Users\Špela\Downloads\C4G4_Avery script pic.png](https://snap.berkeley.edu/project?user=mateja&project=C4G4_Avery)  [3. naloga]  /Users/mateja.bevcic/Desktop/slike za navodila/Screen Shot 2020-02-10 at 21.47.40.pngDoslej 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 /Users/mateja.bevcic/Desktop/slike za navodila/C4G4_Najdi_sir script pic (1).png ali obrni se v nasprotni smeri od urinega kazalca /Users/mateja.bevcic/Desktop/slike za navodila/C4G4_Najdi_sir script pic.png. 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](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:  C:\Users\Špela\Downloads\C4G4_Najdi sir resitev script pic (1).png  [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:  /Users/mateja.bevcic/Desktop/slike za navodila/Screen Shot 2020-02-10 at 22.29.13.png   1. Krožna puščica pomeni prosto vrtenje in lik se lahko obrača v vse smeri (kot tvoja miš) 2. Puščica levo-desno pomeni, da se lahko lik obrača le levo ali desno (to lahko uporabiš pri psu, da ne hodi »po glavi«) 3. 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](https://snap.berkeley.edu/project?user=mateja&project=C4G4_Balerina) * Avery:  [https://snap.berkeley.edu/project?user=mateja& project=C4G4\_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](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.  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2013.55.08.png  [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.  /Users/tadejanemanic/Desktop/Screen Shot 2019-12-11 at 12.24.23.png  [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.   /Users/tadejanemanic/Downloads/farmer.png  [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*.  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2014.20.53.png  Odpre se knjižnica zvokov, v kateri učenci poiščejo zvok psa, ga označijo in za uvoz v program kliknejo na gumb *Import*.  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2014.28.17.png  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*.  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2014.38.29.png  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].*  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2014.41.41.png  [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*.  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2014.49.21.png  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  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2015.01.34.png  Ozadje  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2015.01.25.png |
| **Učni pripomočki, sredstva za učitelja** | * Primer celotne aktivnosti v Snap!-u: [https://snap.berkeley.edu/project?user=tadeja& project=Kmetija\_SLO](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](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.  /Users/mateja.bevcic/Desktop/Learning Scenario/igre/1_risanje_park/Screen Shot 2019-12-11 at 12.48.37.png  [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:  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (1).png  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.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 10.34.10.png  [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.  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (2).png  Kodo dodamo k prejšnji. To je sedaj koda za premik v desno:  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (3).png  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*.  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (4).png /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (5).png  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.  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (6).png /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (7).png  [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*.  /Users/mateja.bevcic/Downloads/C4G7_Pomagaj_princu_in_princeski - Cela script pic (8).png  [Končna koda]  Dekle  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 10.58.41.png  Npr. Princeska  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 10.59.49.png  [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](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](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](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]  /Users/mateja.bevcic/Downloads/tablaKvadrat.png  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).  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (1).png  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*.  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (2).png  Če želimo, da kreda riše pot, moramo pred zanko dodati še ukaz *svinčnik spuščen*.  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (3).png  Če želimo, da se lik krede ne vrti med obračanjem, kliknemo na ukaz *ne vrti*.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.08.55.png  [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*.  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (4).png  [Korak 3]  Sledi koda za pravokotnik. Najprej moramo zamenjati ozadje. To naredimo v dveh korakih:   1. Kliknemo na ozadje (imenovano *tabla*)   /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.21.55 copy.png  S klikom na Backgrounds vidimo vsa tri ozadja, ki jih potrebujemo za to aktivnost (tablaKvadrat, tablaPravokotnik, tablaT), ter še nekaj dodatnih ozadij.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.24.44 copy.png  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*.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.30.23 copy.png   1. Kliknemo ponovno na kredo.   /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.21.55 copy2.png  Pod kodo iz [Korak 2] dodamo kodo, ki bo igralcu povedala, naj pritisne tipko P za spremembo ozadja.  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (5).png  [Korak 4]  /Users/mateja.bevcic/Downloads/tablaPravokotnik.png  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.  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (6).png  [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*.   1. *Ko pritisnemo na tipko T* se ozadje *Preklopi na obleko tablaT*.   /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (7).png   1. Po koncu kode iz [Korak 4] dodamo še ukaz:   /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (8).png  [Korak 6]  /Users/mateja.bevcic/Downloads/tablaT.png  /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (9).png  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].   1. *Ko pritisnemo na tipko K* se ozadje *Preklopi na obleko tablaKvadrat*.   /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (12).png   1. Po koncu kode iz [Korak 6] dodamo še ukaz:   /Users/mateja.bevcic/Downloads/C4G8_Risanje_s_kredo script pic (11).png  [Končna koda]  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 12.58.14.png  [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](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](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.  /Users/mateja.bevcic/Desktop/Learning Scenario/igre/3_spremenljivke_smeti/ozadje.png  [Korak 1]  Poleg ozadja je podana tudi koda za premikanje s tipkami in pogojem za dotikanje rjave barve:  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 15.16.44.png  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:  /Users/mateja.bevcic/Downloads/C4G9_PobiranjeSmeti script pic.png  [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?  /Users/mateja.bevcic/Desktop/slike za navodila/9/spr1.png/Users/mateja.bevcic/Desktop/slike za navodila/9/spr2.png  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.   /Users/mateja.bevcic/Desktop/slike za navodila/9/Screen Shot 2020-02-14 at 13.12.32.png  Ko dodamo spremenljivko, se le ta pojavi med ukazi:  Kljukica pomeni, da bo spremenljivka vidna na zaslonu:  /Users/mateja.bevcic/Desktop/slike za navodila/9/Screen Shot 2020-02-14 at 13.12.37.png  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*:  /Users/mateja.bevcic/Downloads/C4G9_PobiranjeSmeti script pic (1).png  [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.  /Users/mateja.bevcic/Downloads/C4G9_PobiranjeSmeti script pic (2).png  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:  /Users/mateja.bevcic/Downloads/C4G9_PobiranjeSmeti script pic (3).png  [Korak 4]  Sedaj želimo imeti več steklenic. Lik steklenice lahko enostavno podvojimo z desnim klikom na lik in izberemo podvoji.  /Users/mateja.bevcic/Desktop/slike za navodila/9/Screen Shot 2020-02-14 at 13.46.03.png  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.  /Users/mateja.bevcic/Desktop/slike za navodila/9/Screen Shot 2020-02-14 at 13.55.35.png  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.  /Users/mateja.bevcic/Downloads/C4G9_PobiranjeSmeti script pic (4).png  [Končna koda]  Dekle  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 16.19.33.png  Steklenica / Papir  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 16.19.51.png  Koš  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-20 at 16.20.05.png  [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](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](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](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]  ../../../../Downloads/macje_zavetisce_SLO%20script%20pic.png |
| **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> |

NAPREDNEJŠI 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:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic.png  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:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (2).png  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).  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 13.25.52.png  Da bo igra bolj zanimiva, nastavimo število korakov na naključno vrednost, kar pomeni, da bo hitrost premikanja poljubna, npr:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (3).png  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 /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (4).png v kombinaciji s pogojnim stavkom *če*. Če se jabolko dotakne roba, se bo pomaknilo na naključno pozicijo. Snap! Nam ponuja ukazni blok  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (5).png  /Users/mateja.bevcic/Desktop/Learning Scenario/igre/4_lovljenje_hrane/ozadje1.png  /Users/mateja.bevcic/Desktop/Learning Scenario/igre/4_lovljenje_hrane/ozadje2.pngTa 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:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (6).png  [Korak 3]  Sedaj bomo ustvarili spremenljivko, *tocke*, s katero bomo šteli pobrano hrano. Spremenljivko na začetku postavimo na 0 (na liku dekleta).  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (7).png  [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.  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (9).png  [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:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (12).png  [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.  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (10).png  [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).  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (11).png  [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.  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (13).png  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:  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (14).png  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (15).png  [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.  /Users/mateja.bevcic/Downloads/C4G12_LovljenjeZdraveHrane script pic (16).png  [Končna koda]  Dekle  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 14.04.06.png  Jabolko  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 14.03.58.png  [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](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](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](https://snap.berkeley.edu/project?user=mateja&project=C4G12_LovljenjeZdraveHrane%20-%20Delno)   * 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** | C:\Users\leo1.Portege\Pictures\coding4girls\Alice\4a298ba60c5345b8208fb7e2fbe8191a.jpgKratek 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](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.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.22.15.png/Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.22.00.pngSkupaj 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](https://snap.berkeley.edu/project?user=mateja&project=C4G13_SestaviZgodboAlice2).  Kateri nepoznani ukazi se pojavijo?  Primerjajmo kodi Alice1 in Alice2:   |  |  | | --- | --- | | Alice1 | Alice2 | | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.36.13.png | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.37.04.png | | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.39.46.png | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.37.55.png | | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.40.19.png | /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 15.41.12.png |   Predstavimo ukaze za pošiljanje obvestil:  /Users/mateja.bevcic/Desktop/slike za navodila/13/C4G13_SestaviZgodboAlice2 script pic.png /Users/mateja.bevcic/Desktop/slike za navodila/13/C4G13_SestaviZgodboAlice2 script pic (1).png /Users/mateja.bevcic/Desktop/slike za navodila/13/C4G13_SestaviZgodboAlice2 script pic (2).png  Uporabljali bomo prvi ukaz, s katerim obvestilo pošljemo, in tretji ukaz, s katerim obvestilo prejmemo in izvajamo kodo naprej.  /Users/mateja.bevcic/Desktop/slike za navodila/13/Screen Shot 2020-02-15 at 17.38.17.pngUčencem pokažemo, kako ustvarimo obvestilo.  Izberemo ukazni blok *pošlji \_\_ vsem* in kliknemo na puščico. Izberemo željeno sporočilo ali kliknemo *nov...*  /Users/mateja.bevcic/Desktop/slike za navodila/13/Screen Shot 2020-02-15 at 17.45.39.png  ter vpišemo *ime* obvestila.  Uporaba dogodka:  /Users/mateja.bevcic/Desktop/slike za navodila/13/Screen Shot 2020-02-15 at 17.45.51.pngIzberemo 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](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:  /Users/mateja.bevcic/Downloads/C4G13_SestaviZgodboAlice2 script pic copy.png  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 16.47.23.png  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 16.48.38.png/Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 16.49.14.png  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-21 at 16.50.09.png  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*.  /Users/mateja.bevcic/Downloads/C4G13_SestaviZgodboAlice script pic (1).png  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](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](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](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. |
| 1. 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*. |
| 1. 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.      1. 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.   1. 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 120o (priležni 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 90o 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.     1. 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:*    */Users/tadejanemanic/Desktop/Screen Shot 2020-02-19 at 09.34.33.png*Po tem, ko svinčnik prejme sporočilo, mora narisati hrast. Učenec mora pri tem matematično razmisliti, kako narisati hrast, kot je prikazan 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 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 3o (podatki so lahko tudi drugačni, pri tem je potrebno le upoštevati, da se izriše polni krog 360o).    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.     1. 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:*     1. 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](https://snap.berkeley.edu/project?user=tadeja&project=Onesna%C5%BEevanje%20zraka) * 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.   /Users/tadejanemanic/Desktop/Screen Shot 2019-12-17 at 09.22.42.png  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.  /Users/tadejanemanic/Desktop/Screen Shot 2019-12-17 at 09.43.14.png /Users/tadejanemanic/Desktop/Screen Shot 2019-12-17 at 09.43.25.png  [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*  *../../../../Desktop/Screen%20Shot%202020-04-29%20at%2012.18.44.png*  *Miš*  *../../../../Desktop/Screen%20Shot%202020-04-29%20at%2012.18.29.png*  *Dekle*  *../../../../Desktop/Screen%20Shot%202020-04-29%20at%2012.19.04.png*  *Ozadje*  ../../../../Desktop/Screen%20Shot%202020-04-29%20at%2012.19.15.png |
| **Učni pripomočki, sredstva za učitelja** | * Primer celotne aktivnosti v Snap!-u: [https://snap.berkeley.edu/project?user=tadeja&project= Ulovi%20mi%C5%A1\_1](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](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.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 13.11.06.png  [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:  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (1).png  [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].  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (2).png  [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.:   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (3).png   1. *Ko je miška se dotika*: igralcu povemo, koliko izdelek stane. Uporabimo blok *misli za \_\_ sekund* ter operator za združevanje besedila in spremenljivke:   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (4).png   1. *Ko je miška kliknjena*: tu je potreben krajši razmislek.    1. V katerem primeru lahko igralec kupi izdelek in v katerem ne?    2. Kaj se zgodi z razpoložljivim denarjem (spremenljivko *denarnica*), če je izdelek kupljen?    3. Kako štejemo kupljene izdelke?    4. Kaj se zgodi z izdelkom na polici? 2. 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.   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (5).png   1. 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.   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (6).png   1. Število kupljenih izdelkov štejemo s pomočjo spremenljivke zdravaHrana, ki jo povečamo za 1.   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (7).png   1. Ko je izdelek kupljen, se *skrije*.   /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (8).png  Možna koda:  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (9).png  [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.:  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (10).png  Korak večkrat ponovimo, da imamo na polici več izdelkov.  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (11).png[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*:  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (12).png  [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:  /Users/mateja.bevcic/Downloads/C4G16_KupovanjeHraneZaPiknik script pic (13).png  [Končna koda]  Dekle  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 14.39.52.png  Hrana  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 14.40.29.png  Konec  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 14.40.42.png  [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](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](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.      1. 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. 2. Učitelj se z učenci pogovori o spremenljivkah, na kakšen način so definirane, nastavljene in kako se spreminjajo. 3. 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](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](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](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](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)  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-24 at 14.01.08.png[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  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir_1 script pic.png  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.  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir_1 script pic.png ali /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir_1 script pic (2).png  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:  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic.png  Korak lahko večkrat ponovimo, pri tem zamenjamo zaigran ton in odgovor.  Celotna koda se lahko izvede s prejemom obvestila, npr.  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (1).png,  zaključi pa tako, da igralcu povemo, kako naj z igro nadaljuje. Npr.  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (2).png  [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  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-24 at 15.25.32.png  Dinozaver  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-24 at 15.27.50.png  Klavir  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (6).png/Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (3).png  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (4).png  /Users/mateja.bevcic/Downloads/C4G19.1_ZaigrajNaKlavir script pic (5).png  Ozadje  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-24 at 15.28.00.png |
| **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.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 15.29.46.png /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 15.29.36.png  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.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 15.42.21.png  Če se črne tipke skrivajo za belimi, uporabimo ukaz /Users/mateja.bevcic/Desktop/slike za navodila/19.2/C4G19.2_ZaigrajNaKlavir - Delna3 script pic.png, 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:  /Users/mateja.bevcic/Desktop/slike za navodila/19.2/kljukca.png  [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.  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (2).png  Kateri ton se bo zaigral določimo z ukazom *play note \_\_ for 0.5 beats* in s klikom na puščico izberemo željen ton.  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 15.55.17.png  Številka 60 torej določa ton C, 0.5 beats pa določa dolžino predvajanja tona.  Za tipko C imamo sedaj takšno kodo:  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (3).png  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:  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (4).png /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (5).png  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:  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (6).png  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*.  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (7).png  Celotna koda za tipko C je torej:  /Users/mateja.bevcic/Desktop/Screen Shot 2020-04-22 at 16.27.26.png  [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:  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic.png  Na ozadju pa dodamo kodo  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (1).png  Postopek ponovimo za lik izbrisiBesedilo in preklop na prazno ozadje.  Lik izbrisiBesedilo: Ozadje:  /Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (11).png/Users/mateja.bevcic/Downloads/C4G19.2_ZaigrajNaKlavir - Cela script pic (10).png  [Korak 6]  S klikom na noto se bo zaigrala celotna pesem. Napišemo kodo za celotno pesem in kjer je mogoče, uporabimo zanko ponovi.  ../../../../Downloads/C4G19.2_ZaigrajNaKlavir%20-%20Cela%20script%20pic%20(13).pn../../../../Downloads/C4G19.2_ZaigrajNaKlavir%20-%20Cela%20script%20pic%20(12).pn  *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](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](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](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   + Če je, prišteje točko k pravilnim odgovorom   + 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](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](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](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](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](https://snap.berkeley.edu/project?user=zapusek&project=pacman_template) |

1. VIRI

Lajovic, S. (2011). Scratch. *Nauči se programirati in postani računalniški maček*. Ljubljana: Pasadena.

Rugelj, J. (2019). Game design based learning of programming.

Vorderman, C. (2017). *Računalniško programiranje za otroke*. Ljubljana: MK.