Palatul Copiilor Drobeta Turnu Severin

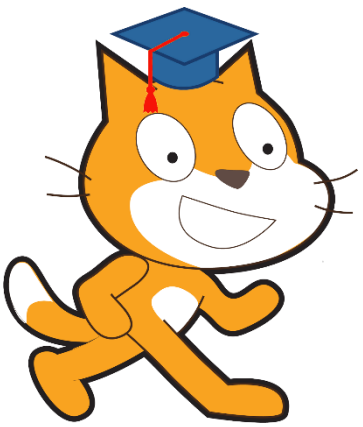
Filiala Orșova

Să învățăm programare jucându-ne în Scratch

Curs pentru începători

Auxiliar curricular

Prof. Mihai Agape



Orșova

2016

Cuprins

ARGUMENT **3**

1 SCURTĂ INTRODUCERE ÎN LIMBAJUL SCRATCH **6**

1.1 CE ESTE SCRATCH? 6
1.2 EDITORUL SCRATCH 2.0 - ONLINE SAU OFFLINE? 6
1.3 INTERFAȚA SCRATCH 2.0 6
1.4 SALUT! 8
1.5 REMIXEAZĂ 8

2 MIȘCARE ȘI SUNET **9**

2.1 INTRODUCERE 9
2.2 TUTORIAL 1: SĂ FACEM PUȚINĂ MIȘCARE 9
2.3 TUTORIAL 2: ADĂUGAREA UNUI SUNET 10
2.4 TUTORIAL 3: E TIMPUL PENTRU DANS 10
2.5 TRANSFER 11

3 EVENIMENTE ȘI CONTROL **12**

3.1 INTRODUCERE 12
3.2 TUTORIAL 1: IAR ȘI IAR 12
3.3 TUTORIAL 2: STEAGUL VERDE 13
3.4 TRANSFER 14

4 SĂ NE JUCĂM CU CULORILE **15**

4.1 INTRODUCERE 15
4.2 TUTORIAL 1: MODIFICAREA CULORII 15
4.3 TUTORIAL 2: CONTROLAREA ACȚIUNILOR DIN TASTE 16
4.4 TRANSFER 17

5 CREEAZĂ-ȚI PROPRIILE PERSONAJE **18**

5.1 INTRODUCERE 18
5.2 TUTORIAL: CREAREA UNUI PERSONAJ 18
5.3 TRANSFER 19

6 VORBIRE ȘI GÂNDIRE **20**

6.1 INTRODUCERE 20
6.2 TUTORIAL: VORBIREA 20
6.3 TRANSFER 21

8 SUNETE, VOCI ȘI MUZICĂ **22**

8.1 INTRODUCERE 22
8.2 TUTORIAL: SUNETE, VOCI, MUZICĂ 22
8.3 TRANSFER 23

9 CREAREA ANIMAȚIILOR **24**

9.1 INTRODUCERE 24
9.2 TUTORIAL: CREEAZĂ ANIMAȚII 24
9.3 TRANSFER 25

10 ROBOT CARE MERGE LA ȚINTĂ **26**

10.1 Inițializarea poziției și orientării personajului pe scenă ... 26
10.2 Deplasează-l pe Karel la țintă ... 26
10.3 Limitează viteza ... 27
10.4 Trei ținte ... 29
10.5 Drawbot ... 33
10.6 Țintă mobilă ... 39

11 ROBOT CARE EVITĂ OBSTACOLELE ... 40

11.1 Ce este un robot care evită obstacolele? ... 40
11.2 Senzori tactili ... 41
11.3 Evitarea obstacolelor ... 42
11.4 Mai multe teste, mai multă distracție ... 42
11.5 Doi roboți pe scenă ... 44

12 ROBOT CARE URMĂREȘTE PERETELE ... 45

12.1 Ce este un robot care urmărește peretele? ... 45
12.2 Senzorul de distanță ... 45
12.3 Robot care urmărește peretele cu ajutorul unui singur senzor ... 47
12.4 Robot care urmărește peretele cu ajutorul a doi senzori ... 48

13 ROBOT URMĂRITOR DE LINIE ... 50

13.1 Ce este un robot care urmărește linia? ... 50
13.2 Robot care urmărește linia cu ajutorul a doi senzori ... 51
13.3 Robot urmăritor de linie cu 12 senzori ... 52

14 ROBOT CARE SOLUȚIONEAZĂ LABIRINTUL DIN LINII ... 55

14.1 Introducere în labirint ... 55
14.2 Rezolvarea labirintului - provocare ... 56
14.3 Soluție pentru rezolvarea labirintului ... 56
14.4 Soluție îmbunătățită pentru rezolvarea labirintului ... 56

BIBLIOGRAFIE ... 58

LISTĂ DE FIGURI ... 59

ANEXA A - REGULAMENTUL CONCURSULUI INTERNAȚIONAL DE PROGRAMARE ÎN SCRATCH SCRIPT ... 61

Argument

Spunem despre copiii de astăzi că sunt nativi digitali. Atunci se pune întrebarea: de ce să insistăm atât de mult să-i învățăm să folosească o tehnologie cu care se simt deja în largul lor? Răspuns: cei mai mulți copii folosesc calculatorul pentru jocuri, muzică, filme sau socializare. Doar o parte dintre copii utilizează calculatorul în învățare și chiar și în acest caz, cei mai mulți sunt doar consumatori ai produselor informatice. În zilele noastre este tot mai evidentă necesitatea transformării copiilor din consumatori de programe în creatori de programe. Cred că în cel mult un deceniu, programarea calculatoarelor va avea aceeași importanță pe care o au astăzi scrisul, cititul și socotitul. Indiferent de profesia pe care și-o vor alege elevii de astăzi, ei vor trebui să aibă competențe în domeniul programării. Tendința este susținută de numărul mare de inițiative, atât pe plan european cât și mondial, care contribuie la promovarea programării în rândul elevilor: introducerea învățării programării încă din ciclul primar în toate școlile din Marea Britanie, începând cu anul școlar 2014 – 2015; evenimentul ”CodeWeek”, organizat de Comisia Europeană; inițiativa americană ”Hour of Code”. Toate acestea au scopul de a crește numărul persoanelor care programează.

În dorința de a stimula implicarea elevilor din ciclul primar și gimnazial în activități de programare, m-am orientat către limbaje de programare vizuale, precum Snap, BYOB și Scratch. Unul dintre programele cele mai îndrăgite de copii, este limbajul de programare Scratch (http://scratch.mit.edu), care este asemenea unui joc, un joc serios prin care copiii pot descoperi unele dintre conceptele fundamentale ale programării. Scratch este un limbaj de programare grafic dezvoltat de grupul Lifelong Kindergarten din cadrul MIT Media Lab.

Scratch sprijină dezvoltarea competențelor secolului XXI. În timp ce lucrează la proiecte Scratch, copiii învață să selecteze, creeze și să administreze diferite forme de media, printre care text, imagine, animație și sunet (informații). Pe măsură ce copiii câștigă experiență în crearea de media, ei devin mai receptivi și mai critici când trebuie să analizeze media din jurul lor. Comunicarea eficientă în lumea de astăzi necesită mai mult decât abilitatea de a citi și scrie text (comunicare în masă). Scratch angajează copiii în alegerea, manipularea și integrarea unei varietăți de media pentru a se exprima creativ și persuasiv (comunicare). Pentru a construi proiecte, copiii trebuie să coordoneze în timp interacțiunile dintre mai mulți actori (gândire sistemică). Crearea unui proiect Scratch constă în generarea unei idei, transformarea ei într-o serie de pași și implementarea cu blocurile de programare Scratch (rezolvare de probleme). Scratch implică copiii în căutarea de soluții noi la probleme neașteptate. Copiii vor învăța nu doar cum să rezolve o problemă predefinită, ci sunt pregătiți să găsească soluții noi atunci când apar noi provocări (creativitate). Programele Scratch sunt construite din blocuri grafice, codul programului fiind mai ușor de citit și împărtășit decât în cazul altor limbaje de programare (colaborare). Să pornești de la o idee și să te gândești cum să realizezi un program în Scratch necesită perseverență și practică (autoorganizare). Când copiii creează proiecte Scratch, au în vedere o audiență și trebuie să se gândească la modul în care alți oameni vor reacționa și răspunde la proiectele lor. Deoarece proiectele Scratch sunt ușor de modificat și corectat, copiii pot modifica propriile proiecte pe baza feedback-ului primit de la alții (responsabilitate și adaptabilitate). Deoarece

programele Scratch sunt ușor de împărtășit, copiii pot folosi Scratch pentru a porni discuții, despre probleme pe care le consideră importante, cu alți membri din propriul mediu de învățare sau din marea comunitate Scratch (responsabilitate socială). Desigur, cei mai mulți copii nu vor deveni programatori profesioniști atunci când vor crește, la fel cum nu toți copiii devin scriitori profesioniști. Totuși, a învăța să programezi este folositor pentru că te ajută să te exprimi mai creativ, să îți dezvolți gândirea logică și să înțelegi tehnologia cu care venim în contact zi de zi.

Proiectele Scratch sunt realizate din obiecte numite personaje. Poți modifica modul în care arată un personaj folosind diferite costume. Poți face un personaj să arate ca o persoană, un tren, un fluture sau orice altceva. Poți folosi orice imagine ca și costum: poți desena o imagine în Editorul Grafic, o poți importa de pe hard disc sau de pe un site web și chiar poți s-o capturezi cu camera de pe laptop. Poți să-i dai instrucțiuni personajului, spunându-i să se mute, să cânte, să vorbească sau să reacționeze la celelalte personaje. Pentru a-i spune personajului ce să facă, îmbini împreună blocuri grafice în stive numite scripturi. Când clichezi pe un script, Scratch rulează blocurile din script începând de la partea de sus către partea de jos.

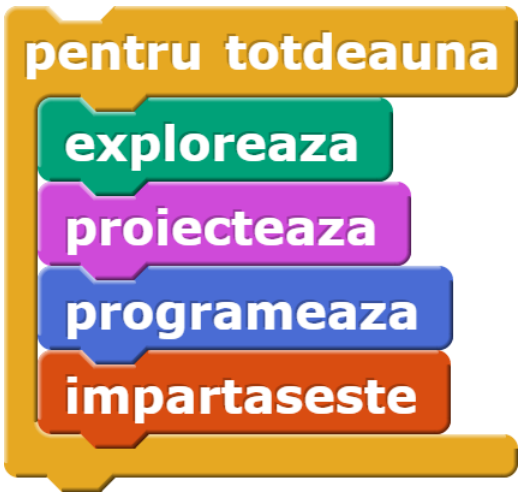
Din 2010, când am început primele activități de programare în Scratch cu elevii cercului de Electronică de la Filiala Orșova, am putut observa că aceștia sunt foarte bucuroși să lucreze în el. De asemenea, în cadrul atelierelor de lucru organizate cu cadre didactice din țară și din Europa, am văzut o bucurie la fel de mare pe chipul profesorilor. Pentru a sprijini activitatea independentă a acestora, am creat acest mic îndrumar disponibil în format electronic, care îi poate ajuta să facă primii pași în programare cu ajutorul Scratch.

Cu ajutorul acestui curs se pot învăța repede câteva din caracteristicile programului Scratch. Prezentul auxiliar curricular are la bază două lucrări ale autorului: auxiliarul curricular „Scratch – Curs pentru începători” 2011 [1] și cursul Robo Scratch [2], publicat în 2015 pe platforma Moodle a portalului Scientix. Cursul este tradus în toate limbile oficiale ale UE, versiunea în engleză aflându-se la http://moodle.scientix.eu/course/view.php?id=161. Datorită celor două surse diferite, se poate observa că lecțiile din prima parte a îndrumarului sunt structurate diferit de cele din a doua parte. Ambele surse au fost îmbunătățite. Auxiliarul, care a fost realizat pentru versiunea Scratch 1.4, a fost adaptat pentru versiunea Scratch 2.0. Am adăugat conținut, imagini și am creat tutoriale video. În cazul cursului Robo Scratch, am adaptat conținutul de la curs online la document. De asemenea am îmbunătățit traducerea, care a fost realizată în cadrul proiectului Scientix, dar probabil nu de niște traducători familiarizați cu limbajul de programare Scratch.

În prima parte se vor învăța lucrurile de bază precum: mișcarea personajelor, adăugarea sunetelor, schimbarea culorilor, controlarea acțiunilor cu mouse-ul și tastatura și crearea propriilor personaje. Aceste lecții sunt accesibile chiar și copiilor de 6 ani, care încă nu știu să citească, ei putând să folosească tutorialele video. În partea a doua vei învăța să programezi diferite comportamente pentru robotul Karel. În această parte există și lecții ușor de parcurs, dar există și unele cu grad ridicat de dificultate.

În anexa A este prezentat regulamentul Concursului Internațional de Programare în Scratch SCRIPT (SCRatch International Programming Trial), pe care l-am inițiat în 2012, pentru a sprijini dezvoltarea activităților de programare. Multe cadre didactice din România au început să astfel de activități după ce au aflat de programul Scratch prin intermediul concursului SCRIPT. La SCRIPT 2015 au participat elevi de pe patru continente (America de Nord, America de Sud, Asia și Europa). Mai multe puteți afla de pe pagina web a concursului, de la adresa http://nonformal.ro/ro/content/script.

Sloganul copiilor de la cercul Conexiuni al Filialei Orșova este:



Explorează proiectele altor utilizatori Scratch de pe site-ul oficial Scratch.

Proiectează – concepe proiecte Scratch scriind scenarii, creând personaje și decoruri.

Programează – scrie scripturile pentru personaje și decoruri.

Împărtășește proiectul pe site-ul oficial Scratch.

Bine ați venit la cursul „Să învățăm programare jucându-ne în Scratch”!

Mihai Agape

1 Scurtă introducere în limbajul Scratch

Obiectiv: să folosești corect denumirile principalelor elemente ale interfeței programului Scratch 2.0.

1.1 Ce este Scratch?

Scratch este un limbaj de programare elaborat de Lifelong Kindergarten Group din cadrul MIT Media Lab și oferit cu titlu gratuit. Adresa site-ului oficial Scratch este http://scratch.mit.edu/.

Cu ajutorul lui Scratch, programarea animațiilor interactive, poveștilor, jocurilor, graficii, simulărilor etc. devine o joacă de copil. Pentru a-ți face o idee asupra tipului de proiecte pe care le poți realiza cu Scratch, te invit să arunci o privire pe pagina cu proiecte pentru începători a site-ului oficial Scratch, la adresa http://scratch.mit.edu/starter_projects/. Pentru a rula un proiect, clichează pe steagul verde.

1.2 Editorul Scratch 2.0 - online sau offline?

În cadrul acestui curs, vei scrie programe folosind versiunea Scratch 2.0. Poți folosi editorul online sau offline. Îți recomand folosirea variantei online.

1.2.1 Editorul online

Scratch 2.0 poate rula în browser, nefiind necesară instalarea sa pe computerul tău. Pentru a folosi editorul online, accesează site-ul Scratch, http://scratch.mit.edu/, apoi clichează pe butonul **Creează**, sau accesează direct editorul online Scratch, la adresa http://scratch.mit.edu/projects/editor/?tip_bar=getStarted.

Poți începe să scrii programe Scratch cu ajutorul lecției **Primii pași cu Scratch** (**Getting Started with Scratch**), care poate fi accesată apăsând butonul **Sfaturi**.

Dacă dorești să-ți salvezi programele în cloud, trebuie să îți creezi un cont Scratch. Pentru aceasta, accesează site-ul Scratch, http://scratch.mit.edu/ și clichează pe butonul **Alătură-te Scratch**.

1.2.2 Editorul offline

Dacă nu vrei să folosești editorul online Scratch 2.0, poți instala editorul offline Scratch 2.0 pe calculatorul tău. În acest scop, urmează instrucțiunile de pe pagina corespunzătoare, de la adresa http://scratch.mit.edu/scratch2download/.

1.3 Interfața Scratch 2.0

În continuare, vei învăța principalele elemente ale interfeței Scratch 2.0 (Figura 1).

1.3.1 Paleta de blocuri

Pentru a programa un personaj, trebuie să tragi blocuri din paleta de blocuri în zona de scripturi. Blocurile sunt grupate în zece categorii: **Mișcare**, **Aspect**, **Sunet**, Creion, Date, **Evenimente**, **Control**, Detecție, Operatori și Mai Multe Blocuri.

Pentru a testa un bloc, clichează pe el. Dacă vrei să știi ce face respectivul bloc, fă clic dreapta pe el și selectează **Ajutor** din meniul contextual.

1.3.2 Zona de scripturi

Blocurile din paleta cu blocuri sunt plasate în zona de scripturi, fiind combinate în scripturi (programe) prin simpla îmbinare a blocurilor sub forma unei stive. Dacă clichezi oriunde pe stivă, scriptul va fi executat de sus în jos.

Când tragi un bloc în zona de scripturi, o zonă evidențiată cu alb îți indică locul unde poți plasa blocul pentru a forma o conexiune corectă cu un alt bloc. Pentru a deplasa o stivă, selecteaz-o cu ajutorul blocului din vârf. Dacă scoți un bloc din mijlocul unei stive, toate blocurile aflate sub acesta se vor deplasa odată cu el.

Pentru a elimina un bloc, trage-l în paleta de blocuri. O altă modalitate de a șterge un bloc, este de a face clic dreapta pe el și a selecta **Șterge** din meniul contextual.

Figura 1 Interfața Scratch 2.0

1.3.3 Scena

Scena este spațiul în care proiectele tale prind viață. Personajele se deplasează și interacționează unele cu altele pe scenă. Pentru a executa un proiect, clichează pe steagul verde. Pentru a-l întrerupe, clichează pe butonul roșu.

Scena (Figura 2) are o lățime de 480 unități și o înălțime de 360 de unități. Mijlocul scenei are coordonatele (x, y) = (0, 0).

1.3.4 Lista cu personaje

Lista cu personaje afișează miniaturi pentru toate personajele dintr-un proiect. Numele fiecărui personaj apare sub miniatura sa.

Pentru a vedea și modifica scripturile, costumele și sunetele unui personaj, clichează pe miniatura acestuia din lista cu personaje.

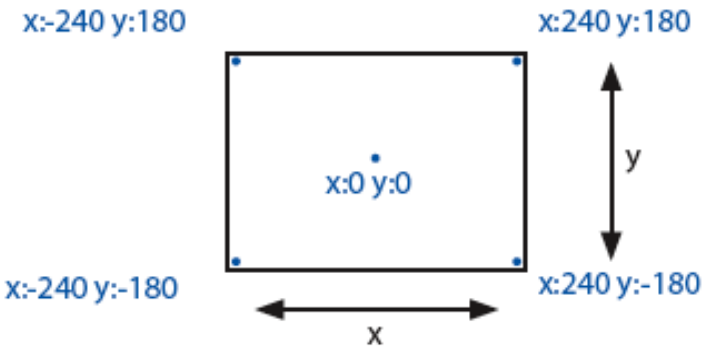


Figura 2 Dimensiunile scenei

1.3.5 Rucsacul

Rucsacul permite utilizatorilor să facă schimb de costume, personaje, sunete și scripturi între proiecte (să plaseze elemente dintr-un proiect în rucsac, apoi din rucsac în alt proiect).

Mai multe detalii găsești pe pagina wiki Interfața Utilizatorului Scratch (Scratch User Interface), la adresa http://wiki.scratch.mit.edu/wiki/Scratch_User_Interface.

1.4 Salut!

Bănuiesc că deja te-ai înscris pe site-ul Scratch. Dacă vrei să remixezi (i.e. să creezi un remix) proiectele altora, să le modifici și să le salvezi ca pe propriile proiecte, trebuie să fii înscris pe site-ul Scratch.

Remixează programul **Salut!**, https://scratch.mit.edu/projects/102736099/#editor. Dacă nu vrei să folosești editorul online, poți descărca fișierul programului și să lucrezi cu editorul offline.

Aruncă o privire pe scripturi și modifică-le cum dorești.

1.5 Remixează

Explorează tipurile de proiecte găzduite pe site-ul oficial Scratch, care poate fi accesat la adresa https://scratch.mit.edu/ și rulează-le. O să-ți faci o idee despre ceea ce poți face cu Scratch.

2 Mișcare și sunet

Obiectiv: să programezi personajul să se miște la stânga și la dreapta și să adaugi sunete.

2.1 Introducere

În această lecție vei folosi elementele de programare de bază (Figura 3) pentru mișca un personaj înainte sau înapoi. Vei învăța cum să selectezi blocurile de programare din diferite meniuri (începând cu meniul albastru pentru **Mișcare**) și cum să execuți o acțiune cu un simplu clic. Apoi vei adăuga câte un sunet (de tobă) de fiecare dată când are loc o mișcare. Vei învăța cum să selectezi blocurile pentru meniul **Sunete** și cum să interconectezi 2 blocuri pentru a le executa unul după celălalt. Blocul **cântă la toba () pentru () timpi** îți permite să selectezi tipul de sunet și durata sunetului.

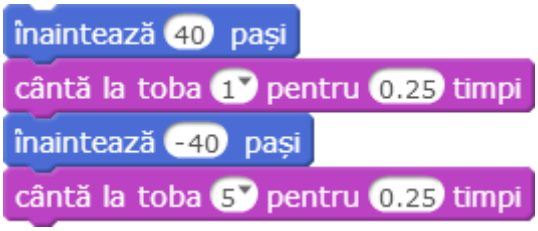


Figura 3 Mișcare și sunet

2.2 Tutorial 1: Să facem puțină mișcare

- Trage un bloc **înaintează (10) pași** din meniul **Mișcare** în zona de scripturi.
- Clichează pe blocul **înaintează (10) pași** pentru a mișca personajul la dreapta.
- Modifică numărul în unul negativ și personajul se va mișca la stânga.

Repetă acțiunile din tutorial. Experimentează cu mărimea și direcția pașilor.

*Figura 4 Blocul **înaintează () pași***

Tutorialul video poate fi urmărit la adresa https://youtu.be/yivonrobat8. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Să facem puțină mișcare”

1) În acest exercițiu vei învăța să deplasezi personajul folosind blocul **înaintează (10) pași** din meniul **Mișcare**.
2) Trage blocul **înaintează (10) pași** din meniul **Mișcare**, în zona de scripturi.
3) Dacă clichezi pe blocul **înaintează (10) pași**, observi că personajul se mișcă înainte.
4) De fiecare dată când clichezi pe blocul **înaintează (10) pași**, pisica înaintează 10 pași.
5) Poți modifica numărul de pași în unul negativ, ceea ce va face pisica să se deplaseze înapoi.
6) De fiecare dată când clichezi pe blocul **înaintează (10) pași**, pisica se mișcă 10 pași înapoi.

7) În continuare vei experimenta cu numărul pașilor:
   a) Modifică numărul de pași la 50 și clichează pe bloc. Personajul se deplasează înainte pe o distanță mai mare, respectiv 50 de pași.
   b) Modifică numărul de pași la -50 și clichează pe bloc. Personajul se deplasează înapoi pe o distanță de 50 de pași.
8) Acum știi să folosești blocul **înaintează (10) pași**.

2.3 Tutorial 2: Adăugarea unui sunet

- Trage un bloc **înaintează (10) pași** din meniul **Mișcare** în zona de scripturi.
- Trage un bloc **cântă la toba () pentru () timpi** din meniul **Sunete** în zona de scripturi.
- Conectează cele două blocuri.
- Clichează pe stivă: personajul se mișcă și se aude un sunet.
- Modifică numărul care urmează după tobă, pentru a selecta un instrument diferit.

Figura 5 Adăugarea unui sunet

Repetă acțiunile din tutorial. Experimentează cu alte sunete.

Tutorialul video poate fi urmărit la adresa https://youtu.be/hCZmflPkJBg. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Adăugarea unui sunet”

1. În acest exercițiu vei învăța să adaugi sunete de fiecare dată când personajul se mișcă.
2. Pentru asta ai nevoie de blocul **înaintează () pași** din meniul **Mișcare** și blocul **cântă la toba () pentru () timpi** din meniul **Sunete**.
3. Blocul **cântă la toba () pentru () timpi** redă sunetul unui instrument de percuție (pe care îl putem selecta) pentru o durată pe care o putem modifica.
4. Dacă apropii un bloc de celălalt, ele se conectează. Când clichezi pe unul dintre blocuri, cele 2 blocuri vor fi executate unul după celălalt.
5. Dacă apropii un bloc de celălalt, ele se conectează. Când clichezi pe unul dintre blocuri, cele 2 blocuri vor fi executate unul după celălalt. Pisica înaintează 10 pași și toba cântă pe durata selectată.
6. Dacă clichezi pe prima casetă a blocului **cântă la toba () pentru () timpi**, poți selecta un alt instrument.
7. Acum poți experimenta cu diverse sunete și diferite durate.

2.4 Tutorial 3: E timpul pentru dans

- Trage în zona de scripturi un bloc **înaintează () pași** din meniul **Mișcare**.
- Trage în zona de scripturi un bloc **cântă la toba () pentru () timpi** din meniul **Sunete**.
- Mai trage un bloc **înaintează () pași** din meniul **Mișcare**.

- Trage încă un bloc **cântă la toba () pentru () timpi** din meniul **Sunete**.
- Conectează cele 4 blocuri.
- Schimbă valoarea pentru cel de al doilea bloc **înaintează () pași** la -10 pași.
- Pentru cel de al doilea bloc **cântă la toba () pentru () timpi** modifică numărul pentru a selecta un alt instrument.
- Fă clic pe stivă și urmărește dansul personajului pe scenă.

Repetă acțiunile din tutorial. Experimentează mărimea pașilor și diferite sunete.

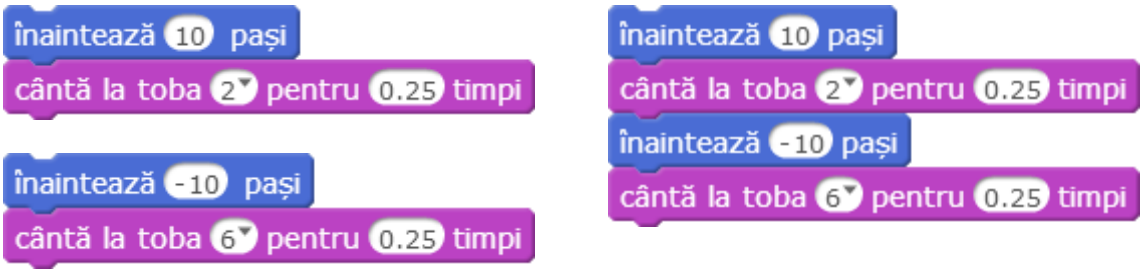


Figura 6 E timpul pentru dans

Tutorialul video poate fi urmărit la adresa https://youtu.be/GZcVofqZgXU. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „E timpul pentru dans”

1. În acest tutorial vei învăța cum să faci câțiva pași de dans. Vei face personajul să se miște mai întâi la dreapta și apoi la stânga. Fiecare mișcare este acompaniată de un sunet de tobă.
2. În acest exercițiu ai nevoie de 2 blocuri **înaintează () pași** din meniul **Mișcare** și de 2 blocuri **cântă la toba () pentru () timpi** din meniul **Sunet**.
3. Aranjează blocurile astfel încât personajul să se miște la dreapta, urmat de un sunet și apoi să se miște la stânga urmat de un alt sunet.
4. Conectează toate blocurile într-o singură stivă. După ce clichezi pe stivă, personajul începe să danseze.
5. Dacă inversezi locurile celor 2 blocuri **înaintează () pași**, personajul se deplasează mai întâi la stânga și apoi la dreapta.
6. Experimentează cu diverse valori ale pașilor și diferite sunete. Învață personajul și alți pași de dans.

2.5 Transfer

Revezi subiectele principale introduse în această lecție și discută temele propuse în secțiunea Extinderi.

2.5.1 Extinderi

1. Experimentează cu mărimea pașilor: foarte mici, mici, mari, foarte mari.
2. Experimentează cu sunete: selectează și combină diferite sunete.
3. Experimentează cu dansurile: 2, 3, mai mulți pași în dans. Selectează și combină pași scurți și pași lungi.

3 Evenimente și control

Obiectiv: să repeți acțiunile cu blocul **pentru totdeauna**, să pornești scriptul cu **steagul verde** și să-l oprești cu semnul **stop**.

3.1 Introducere

În această lecție vei folosi bloc de control **pentru totdeauna**, din meniul **Control**. Vei muta cele 4 blocuri din exemplul precedent—**înaintează () pași** și **cântă la toba () pentru () timpi**—în „gura” blocului de control **pentru totdeauna**. Când clichezi pe stivă, acțiunile sunt repetate pentru totdeauna, făcând personajul să danseze în continuu. Pentru a opri dansul trebuie să folosești iconița **stop**. În această lecție vei folosi de asemenea o nouă iconiță, **steagul verde**. Când clichezi pe **steagul verde**, blocurile conectate la el vor fi executate.

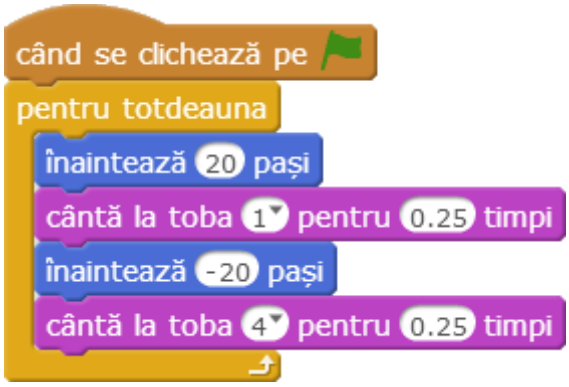


Figura 7 Evenimente și control

3.2 Tutorial 1: Iar și iar

- Începe cu cele 4 blocuri din lecția anterioară.
- Trage un bloc **pentru totdeauna** din meniul **Control** în zona de script.
- Trage stiva cu cele 4 blocuri conectate în interiorul blocului **pentru totdeauna**.
- Apasă pe stiva de blocuri și observă ce se întâmplă.
- Experimentează cu diferite valori pentru numărul pașilor și pentru durata sunetelor.

Exersează acțiunile din tutorial. Experimentează adăugând un al doilea bloc de repetare.

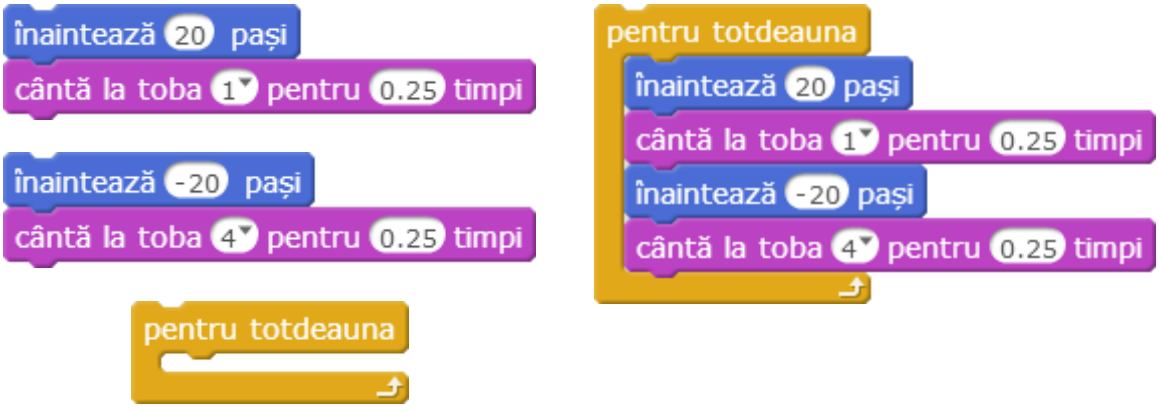


Figura 8 Buclă infinită

Tutorialul video poate fi urmărit la adresa https://youtu.be/LviKwzBENOU. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Iar și iar”

1. În acest exercițiu vei învăța cum să repeți o acțiune la nesfârșit. Există câteva blocuri care ne permit să repetăm acțiuni și aceste blocuri se află în meniul **Control**.
2. Vei folosi blocurile din tutorialul anterior „E timpul pentru dans” pe care le vei face să se repete prin utilizarea blocului **pentru totdeauna**.
3. Selectează blocul **pentru totdeauna**, care are forma literei C. Clichează pe primul bloc al stivei și gliseaz-o în gura blocului **pentru totdeauna**.
4. Remarcă linia albă din interiorul blocului **pentru totdeauna**, atunci când ajungi în poziția corectă. Apoi eliberează mouse-ul.
5. Dacă clichezi pe stiva de blocuri, personajul va repeta mișcările (fără să obosească).
6. Pentru a-l opri, clichează pe semnul roșu de stop aflat în dreapta sus.
7. Dacă vrei să schimbi puțin coregrafia, poți folosi alt bloc pentru repetarea acțiunilor și anume **repetă de (10) ori.** Trage un astfel de bloc din grupul **Control**.
8. Modifică numărul de repetiții la 2 și copiază blocul **repetă de (2) ori** printr-un clic dreapta pe bloc și selectarea opțiunii **duplică**.
9. Scoate stiva cu patru blocuri din interiorul buclei infinite **pentru totdeauna**, apucând cu mouse-ul de primul bloc al stivei. Separă stiva de 4 în două stive de 2, trăgând de cel de al treilea bloc.
10. Glisează în interiorul primului bloc **repetă de (2) ori** prima instrucțiune pentru deplasare și sunetul corespunzător. Dacă clichezi pe stivă vei observa că personajul face doi pași în aceeași direcție, în loc de unul.
11. Procedează similar pentru al doilea bloc **repetă de (2) ori** și cea de a doua instrucțiune de deplasare.
12. Asamblează blocurile astfel încât personajul să repete la nesfârșit următorul dans: de două ori la stânga și de două ori la dreapta.
13. Poți experimenta cu diferite valori ale pașilor și diferite sunete. Pentru a face dansul mai rapid sau mai lent, micșorează sau mărește durata sunetelor.

3.3 Tutorial 2: Steagul verde

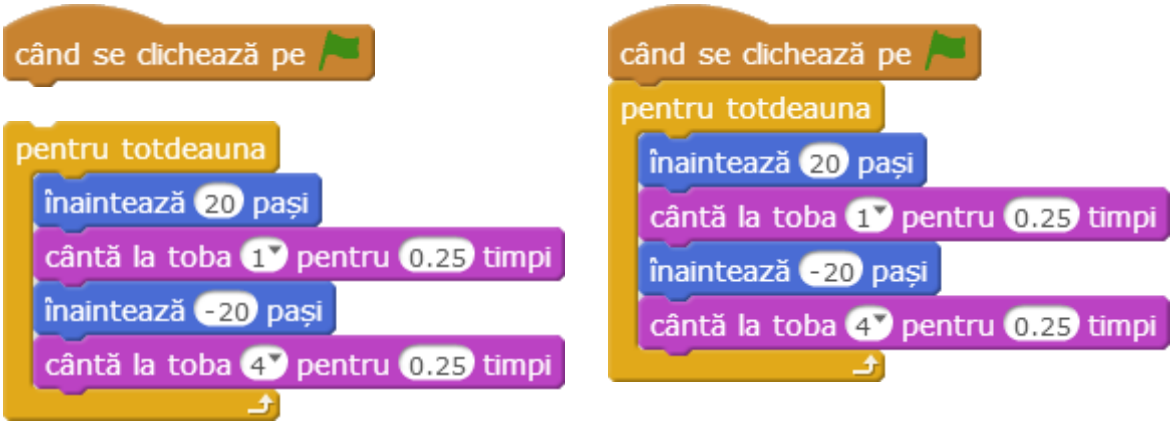


*Figura 9 Instrucțiunea **când se clichează pe steagul verde***

- Începe cu cele 5 blocuri din lecția anterioară și trage blocul **când se clichează pe steagul verde** din meniul **Evenimente** în zona de script.
- Atașează blocul **când se clichează pe steagul verde** deasupra blocului **pentru totdeauna**.

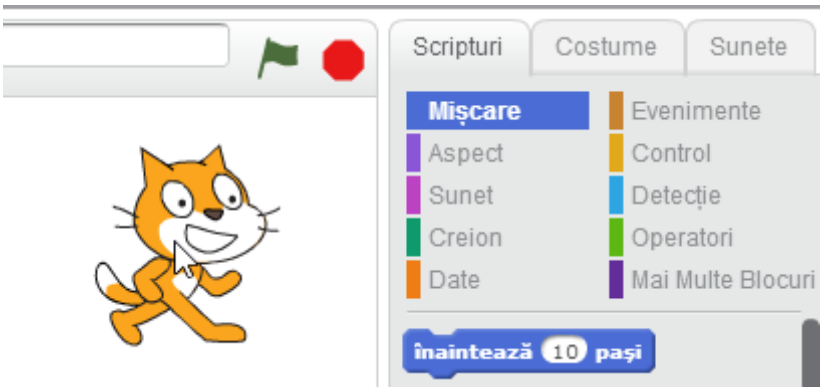


Figura 10 Steagul verde și iconița roșie stop se află deasupra scenei în colțul din dreapta

- Când clichezi pe iconița **steagul verde** (Figura 10), de deasupra scenei, personajul începe să danseze.
- Dacă clichezi pe iconița roșie **stop**, de deasupra scenei, personajul se oprește din dans. Repetă acțiunile din tutorial. Experimentează adăugând steagul verde la toate blocurile de repetiție.

Tutorialul video poate fi urmărit la adresa https://youtu.be/kYz2KgVsah8. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Steagul verde”

1. O modalitate de a porni executarea unui script, este folosirea steagului verde situat deasupra scenei.
2. Blocul **când se clichează pe steagul verde** se află în grupul **Evenimente**.
3. Atașează blocul **când se clichează pe steagul verde** deasupra blocului **pentru totdeauna** din exercițiul anterior.
4. Dacă clichezi pe steagul verde de deasupra scenei, vei activa scriptul.
5. Pentru a opri executarea scriptului apasă pe semnul **stop** (de culoare roșie) de deasupra scenei.
6. Acum știi cum să faci ca să vizualizezi proiectele tale într-o fereastră mai mare.

3.4 Transfer

Revezi temele principale introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

3.4.1 Extinderi

1. Experimentează cu un al doilea bloc de repetiție pentru un al doilea personaj.
2. Experimentează crearea a două sau trei blocuri de repetiție și controlează-le în același timp cu steagul verde și semnul stop.

4 Să ne jucăm cu culorile

Obiectiv: să modifici culorile personajului și să-i controlezi acțiunile cu tastatura.

4.1 Introducere

Din meniul **Aspect** vei folosi în acest tutorial blocul **modifică efectul [] cu ()**. Când clichezi pe acest bloc, personajul își schimbă culoarea cu o anumită cantitate specificată. Poți selecta, de asemenea, alte modificări grafice, cum ar fi vârtej, mozaic, etc. În această lecție vei folosi un alt bloc de control pentru a executa acțiunile. Blocul **când tasta [] este apăsată** va executa toate acțiunile blocurilor conectate la el.

Figura 11 Să ne jucăm cu culorile

4.2 Tutorial 1: Modificarea culorii

- Trage în zona de script un bloc **modifică efectul [] cu ()** din meniul **Aspect**.
- Clichează pe bloc de mai multe ori, pentru a modifica culoarea personajului.
- Setează efectul la strălucire și clichează pentru a face personajul mai luminos sau mai întunecat.
- Observă efectul de vârtej asupra discului colorat din proiectul de la adresa https://scratch.mit.edu/projects/102617366/.(Figura 13).

Figura 12 Modificarea culorii

Repetă acțiunile din tutorial. Experimentează cu diferite valori de modificare a efectelor.

Tutorialul video poate fi urmărit la adresa https://youtu.be/hGeedu_UzKg. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Modificarea culorii”

1. Există mai multe efecte pe care le poți aplica personajelor tale. Unul dintre ele este modificarea culorii.
2. Pentru aceasta selectează blocul **modifică efectul [] cu ()** din meniul **Aspect**.
3. Dacă clichezi pe acest bloc, culoarea personajului se modifică.
4. Dacă clichezi pe blocul **anulează efectele grafice** din grupul **Aspect**, personajul revine la aspectul de dinaintea aplicării efectului.

5. Poți modifica valoarea cu care se schimbă efectul editând numărul din caseta text a blocului.
6. Experimentează cu diferite valori pentru modificarea culorii.
7. Alege efectul strălucire și clichează pentru a face personajul mai luminos sau mai întunecat.
8. Experimentează modul în care este afectat personajul de diferitele efecte.

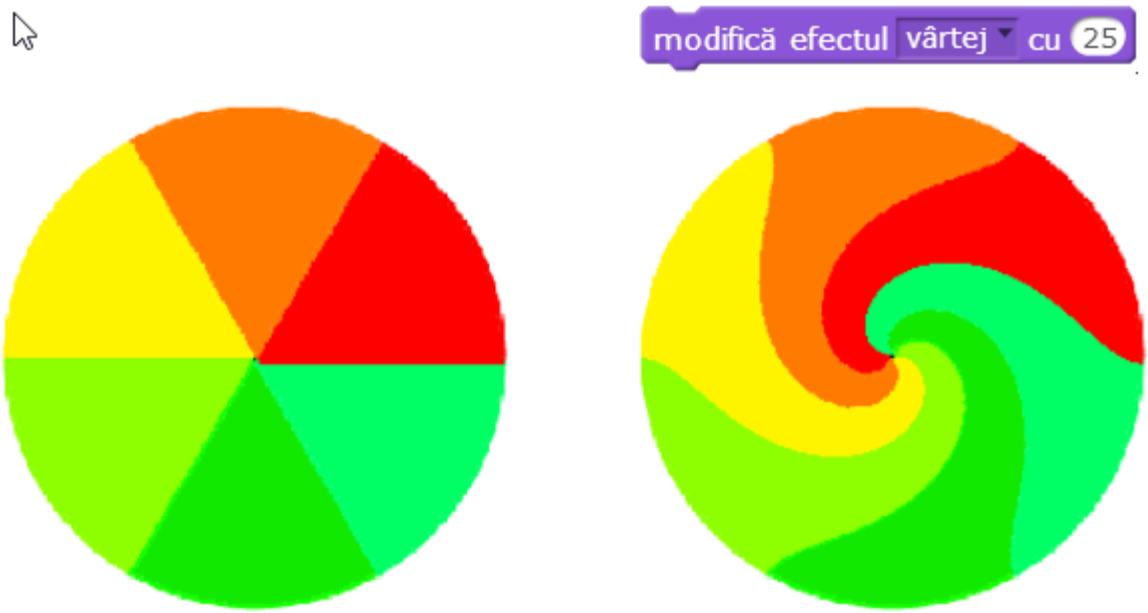


Figura 13 Crearea efectelor de imagine

4.3 Tutorial 2: Controlarea acțiunilor din taste

- Trage în zona de programare blocul **când tasta [] este apăsată** din meniul **Control**.
- Conecteaz-o deasupra blocului **modifică efectul [] cu ()** și alege efectul culoare prin clicarea săgeții și selectarea din meniul derulant.
- Apasă tasta **spațiu** și observă că de fiecare dată când o apeși, personajul își schimbă culoarea.
- Observă că poți folosi o altă tastă, dacă în cadrul blocului **când tasta [] este apăsată** clichezi pe săgeata orientată în jos și o selectezi din meniul derulant.

Repetă acțiunile din tutorial. Experimentează cu diferite taste.

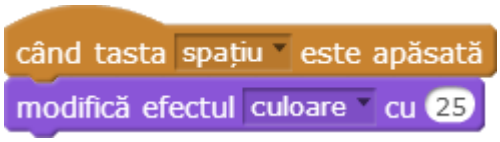


Figura 14 Controlarea acțiunilor din taste

Tutorialul video poate fi urmărit la adresa https://youtu.be/Q8kaxyi1KIQ. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Controlarea acțiunilor din taste”

1. În acest exercițiu vei învăța cum să controlezi acțiunile personajului apăsând pe diferite taste ale tastaturii.
2. Pentru asta vei folosi blocul **când tasta [spațiu] este apăsată** din grupul **Evenimente**.
3. Poți conecta blocul **când tasta [spațiu] este apăsată** cu blocul **modifică efectul [culoare] cu (25)**.
4. De fiecare dată când apeși tasta **spațiu**, personajul își schimbă culoarea.
5. Dacă rulezi programul prin apăsarea steagului verde, personajul va dansa și dacă apeși tasta **spațiu**, acesta își va schimba culoarea.
6. Tasta poate fi aleasă din meniul derulant care apare dacă se clichează pe micul triunghi din dreapta cuvântului spațiu. Experimentează cu alte taste ale tastaturii.

4.4 Transfer

Revezi principalele subiecte introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

4.4.1 Extinderi

1. Experimentează cu diferite viteze de modificare a culorii.
2. Experimentează cu două personaje pentru a vedea cum se modifică culorile.
3. Experimentează cu diferite taste pentru a controla modul în care se modifică culorile.
4. Experimentează adăugarea de sunete când schimbi culorile.
5. Experimentează cu toate efectele grafice: ochi de pește, vârtej, pixelare, mozaic, strălucire și fantomă.
6. Experimentează aplicarea a două efecte de imagine simultan la același personaj.
7. Experimentează controlarea valorilor efectelor aplicate imaginii.
8. Experimentează adăugarea de sunete la efectele de imagine.

5 Creează-ți propriile personaje

Obiectiv: să îți creezi propriile personaje sau să le imporți dintr-o bibliotecă de personaje.

5.1 Introducere

În această lecție vei învăța cum să imporți personaje în proiectele tale. Ca alternativă, poți folosi editorul grafic pentru a-ți crea propriile personaje.

Figura 15 Personaje

5.2 Tutorial: Crearea unui personaj

- Clichează pe butonul **Alege personaj din bibliotecă** și importă un personaj dintr-una din bibliotecile cu personaje.
- Clichează pe butonul **Desenează personaj nou** pentru a intra în editorul grafic și a desena propriul personaj. Poți încerca să desenezi o față veselă. După ce ai terminat de desenat apasă butonul OK. Editorul grafic se închide și personajul nou se află pe scenă și este pregătit să execute instrucțiunile.
- Clichează pe butonul **Încarcă personaj din fișier** pentru a încărca o imagine stocată într-un dosar.
- Clichează pe butonul **Personaj nou de la cameră** pentru a folosi o imagine realizată cu camera laptopului.

Repetă acțiunile din tutorial. Experimentează creându-ți propriile personaje.

Tutorialul video poate fi urmărit la adresa https://youtu.be/48vTcYlJa8I. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Crearea unui personaj”

1. Până acum ai lucrat cu același personaj, pisica. Poți folosi și alte personaje în programele tale. În acest tutorial vei învăța cum să adaugi un personaj nou.
2. Poți importa personaje deja create. Pentru asta clichează pe butonul Alege personaj din bibliotecă. Acum ai acces la bibliotecile de personaje din Scratch.
3. Dacă apeși butonul **Desenează personaj nou** se deschide editorul grafic și poți desena propriul tău personaj.
4. Poți începe prin a desena o față zâmbitoare. Selectează instrumentul pentru desenarea cercurilor.
5. Mai întâi desenează capul. Apoi, desenează un ochi. Folosește instrumentul ștampilă pentru a-l copia. Așază ochii la locul potrivit în interiorul capului.
6. Creează o copie a părții de jos a capului cu ajutorul instrumentului ștampilă, micșoreaz-o la dimensiunea gurii și deplaseaz-o la locul potrivit.
7. Poți încărca un personaj dintr-un dosar personal clicând pe butonul **Încarcă personaj din fișier** .
8. Dacă clichezi pe butonul **Personaj nou de la cameră** vei putea folosi imaginea pe care o vei captura cu camera video a laptopului.
9. Acum poți experimenta cu cele patru modalități de a adăuga un personaj programului tău.

5.3 Transfer

Revezi subiectele principale introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

5.3.1 Extinderi

1. Experimentează desenarea și colorarea propriilor personaje.
2. Experimentează dansul, modificarea culorilor și a sunetelor cu propriile personaje.
3. Experimentează și explorează bibliotecile de personaje din Scratch.
4. Experimentează crearea unor grupuri înrudite de personaje.

6 Vorbire și gândire

Obiectiv: să programezi personajele să-și exprime ideile sau gândurile.

6.1 Introducere

În această lecție vei învăța să folosești blocurile de programare care îți permit să afișezi o formă pentru vorbire sau gând și mesajul corespunzător. Mesajul este afișat permanent sau doar pentru o anumită durată de timp.

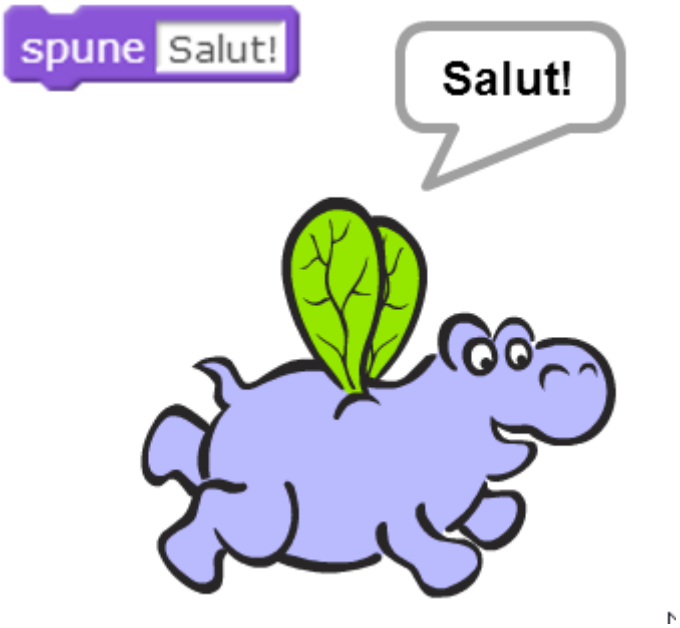


Figura 16 Vorbire

6.2 Tutorial: Vorbirea

- Selectează blocul **spune []** din meniul **Aspect**.
- Modifică textul **Hello!** cu textul **Bună! Ce mai faci?**
- Dacă clichezi pe bloc, personajul afișează mesajul.
- Poți afișa mesajul pentru o anumită perioadă de timp folosind blocul **spune [] pentru () secunde**.
- Selectează blocul **gândește []**.
- Selectează blocul **gândește [] pentru () secunde**.
- Testează executarea blocurilor pentru a observa diferențele dintre **spune []** și **gândește []** pentru o perioadă nelimitată și pentru o anumită perioadă de timp.
- Conectează patru blocuri:
  - **spune [Bună!] pentru (1) secunde**
  - **spune [Ce] pentru (1) secunde**
  - **spune [mai] pentru (1) secunde**
  - **spune [faci?] pentru (1) secunde**
- Testează scriptul pe care l-ai scris.

Tutorialul video poate fi urmărit la adresa https://youtu.be/a4B-Q_dcupU. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Vorbirea”

1. În acest tutorial vei asocia un text unui personaj.
2. Pentru aceasta se folosește blocul **spune []** din grupul **Aspect**.
3. Scrie în interiorul blocului mesajul pe care vrei să-l transmită personajul tău, de exemplu **„Bună! Ce mai faci?**”
4. Când clichezi bloc, personajul afișează mesajul.
5. Poți spune mesajul pentru un timp limitat folosind blocul **spune [] pentru () secunde**.
6. Cu ajutorul a patru astfel de blocuri poți transmite mesajul anterior:
   - **spune [Bună!] pentru (1) secunde**
   - **spune [Ce] pentru (1) secunde**
   - **spune [mai] pentru (1) secunde**
   - **spune [faci?] pentru (1) secunde**
7. Îmbină cele patru blocuri și observă care este efectul lor.
8. Experimentează cu cele două blocuri **gândește []**, care se află în același grup **Aspect**.

6.3 Transfer

Revezi subiectele principale introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

6.3.1 Extinderi

1. Experimentează scrierea ideilor și a gândurilor.
2. Experimentează cu două personaje care vorbesc unul cu celălalt.
3. Experimentează folosirea tastelor pentru a-i controla când vorbesc.
4. Experimentează adăugarea sunetelor când personajele vorbesc.

8 Sunete, voci și muzică

Obiectiv: să adaugi sunete, voci și muzică în programe.

8.1 Introducere

Folosind fila **Sunete** din zona blocurilor pentru scripturi, poți asocia personajelor fișiere audio înregistrate sau importate din altă parte.

Fiecărui personaj îi poți asocia mai multe fișiere audio.

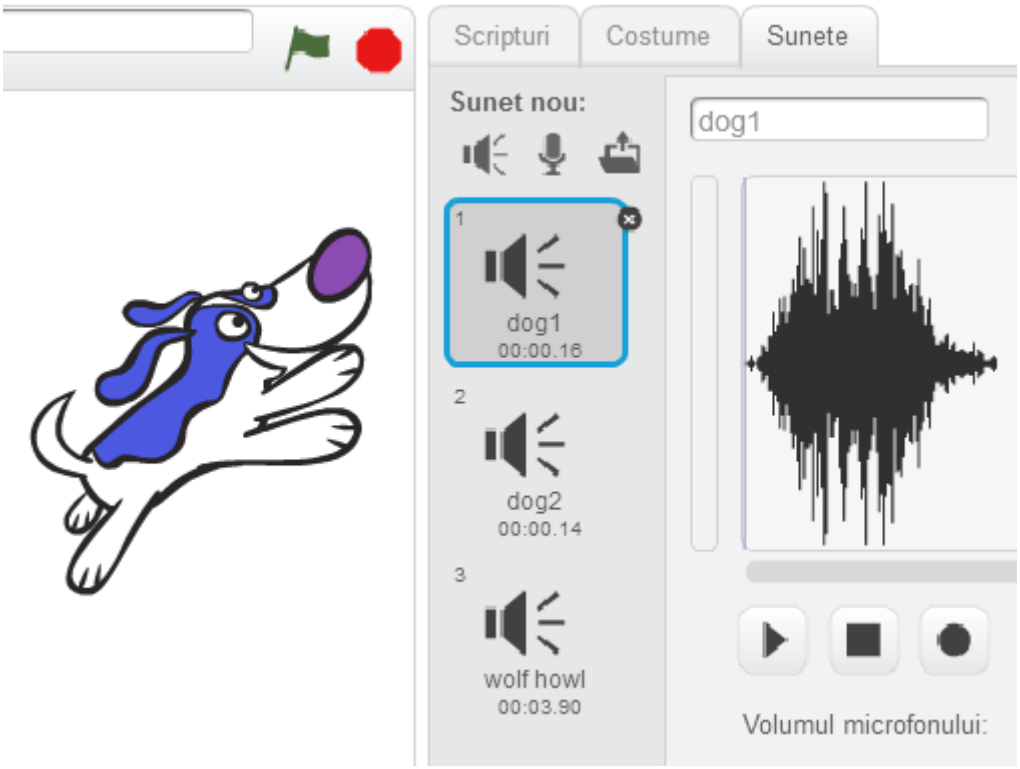


Figura 17 Sunete, voci, muzică

8.2 Tutorial: Sunete, voci, muzică

- Clichează pe fila **Sunete**.
- Importă câteva sunete.
- Clichează pe fila **Scripturi**.
- Trage în zona de programare blocul **cântă sunetul []** din meniul **Sunete**.
- Clic pe blocul **cântă sunetul []** pentru a reda unul din sunetele asociate personajului; schimbă sunetul și execută blocul.

Repetă acțiunile din tutorial. Experimentează cu diferite sunete.

Tutorialul video poate fi urmărit la adresa https://youtu.be/-Oiono-POSA. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Sunete, voci, muzică”

1. În acest exercițiu vei învăța să asociezi fișiere audio personajului.
2. Pentru aceasta clichează pe fila **Sunete** din zona de programare.

3. Poți alege sunete din bibliotecă, le poți înregistra sau le poți importa din dosarele tale.
4. Unui personaj îi poți asocia cât de multe fișiere audio dorești.
5. Selectează un fișier audio.
6. Pentru redarea sunetului în program, apasă pe fila **Scripturi** și din grupul **Sunet**, folosește blocul **cântă sunetul [].**
7. Experimentează cu sunetele: înregistrează sunete, importă sunete și fă personajul să vorbească sau să cânte.

8.3 Transfer

Revezi subiectele principale introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

8.3.1 Extinderi

1. Experimentează prin explorarea sunetelor din bibliotecile Scratch.
2. Experimentează cu două sau mai multe personaje care cântă împreună.
3. Experimentează îmbinarea de efecte de imagine cu muzică.
4. Experimentează controlarea secvenței de sunete de la tastatură.

9 Crearea animațiilor

Obiectiv: să creezi animații cu personaje existente în biblioteca programului sau cu cele proprii.

9.1 Introducere

Poți asocia imagini sau costume personajelor, folosind fila **Costume** din zona de blocurilor pentru scripturi. Poți importa costume existente sau poți folosi editorul grafic pentru a crea unele noi. Unui personaj îi poți asocia mai multe costume. După ce le imporți, poți modifica ordinea acestora.

Poți folosi schimbarea costumelor pentru crearea efectelor de animație. Când clichezi pe **costumul următor**, personajul își schimbă costumul, creând iluzia de mișcare.

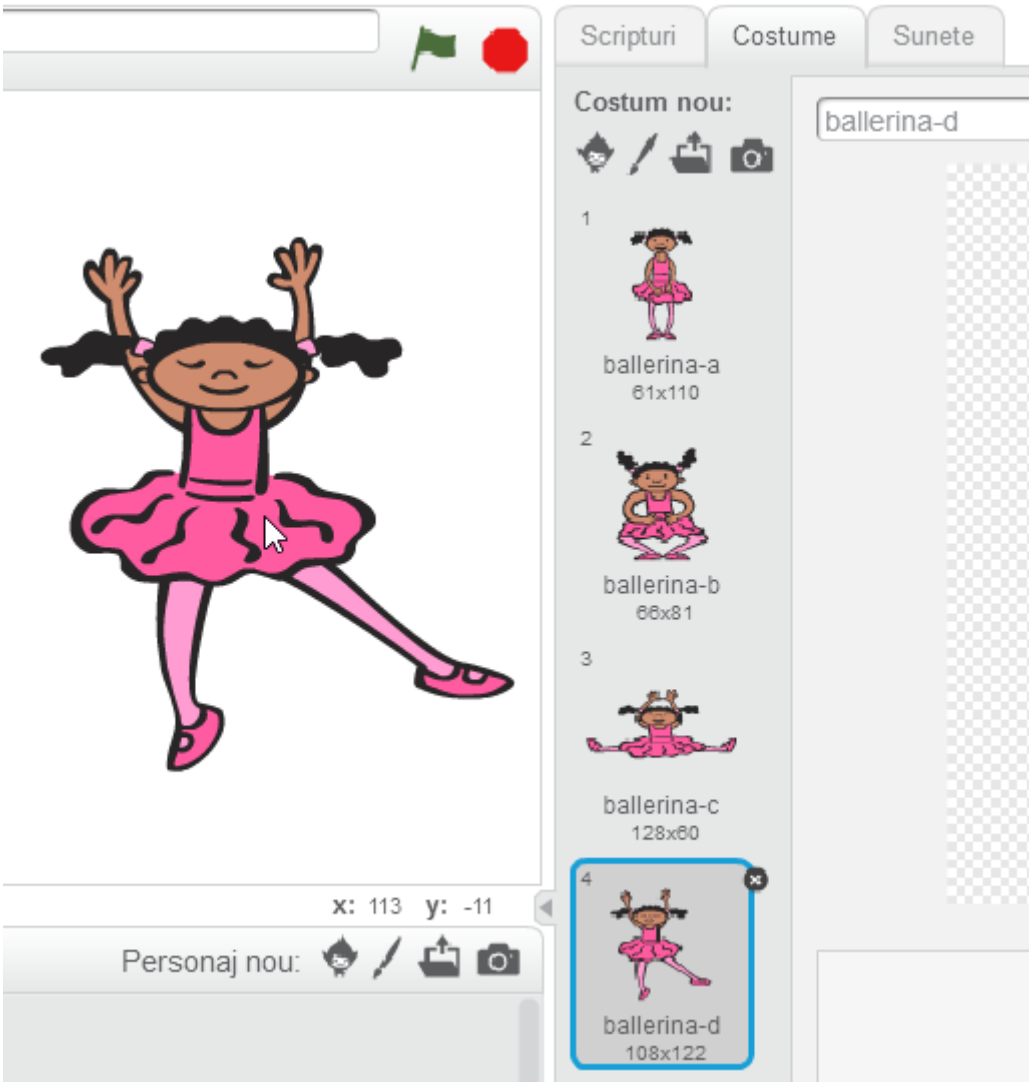


Figura 18 Crearea animațiilor

9.2 Tutorial: Creează animații

- Clichează pe fila **Costume**, aflată în dreapta filei **Scripturi**.
- Importă câteva costume.
- Trage din meniul **Aspect** blocul **costumul următor** în zona de scripturi.

- De fiecare dată când clichezi pe blocul **costumul următor**, personajul își schimbă poziția.
- Remarcă faptul că schimbarea costumului nu înseamnă schimbarea hainelor, ci doar schimbarea posturii personajului.

Repetă acțiunile din tutorial. Experimentează cu diferite mărimi și direcții ale pașilor.

Tutorialul video poate fi urmărit la adresa https://youtu.be/0CD_mAqTcTs. Mai jos este dat transcriptul filmului.

Transcriptul tutorialului video „Creează animații”

1. În acest tutorial înveți să creezi animații folosind blocul **costumul următor** din meniul **Aspect**.
2. Folosind fila **Costume** din zona de programare, poți asocia imagini sau costume unui personajului. Poți importa costume existente sau poți folosi editorul grafic pentru a crea propriile noastre costume.
3. Poți asocia mai multe costume fiecărui personaj.
4. Importă mai multe costume și eventual schimbă-le ordinea.
5. Când clichezi pe **costumul următor**, personajul își schimbă costumul, creând efectul de mișcare.
6. Dacă personajul se mișcă și își schimbă costumele, poți face ca mersul personajului să pară mai real.
7. Observă care este efectul scriptului următor: pisica se deplasează de la stânga la dreapta scenei.
8. Dacă introducem în script instrucțiunea **costumul următor**, pisica își va schimba cele două costume și mersul va părea mai natural.
9. Poți experimenta cu alte seturi de costume și alte animații.

9.3 Transfer

Revezi subiectele principale introduse în această lecție și discută cu colegii temele propuse în secțiunea Extinderi.

9.3.1 Extinderi

1. Experimentează realizarea de animații cu personaje din bibliotecile programului.
2. Experimentează crearea costumelor personajelor și realizarea animațiilor proprii.
3. Experimentează animarea a două sau mai multe personaje în același timp.
4. Experimentează adăugarea de sunete redate în timp ce personajele se mișcă.

10 Robot care merge la țintă

Obiectiv: să folosești câteva blocuri Scratch pentru a programa robotul Karel să îndeplinească o serie de sarcini ușoare.

Îți poți face o idee despre ceea ce vei învăța în această unitate, dacă vei urmări filmul de la adresa https://youtu.be/498vHVw_ICs.

10.1 Inițializarea poziției și orientării personajului pe scenă

Vrem ca personajele noastre să se afle la începutul programului în anumite poziții și cu anumite orientări pe scenă.

10.1.1 Inițializarea poziției și orientării – provocare

Remixează proiectul de la http://scratch.mit.edu/projects/41637186/#editor. În acest proiect există două personaje: robotul Karel și o țintă. Scrie un program de inițializare a pozițiilor celor două personaje:

- Karel trebuie amplasat în punctul de coordonate (-100, -100) și orientat spre est.

- Ținta trebuie amplasată în punctul de coordonate (100, -100).

Folosește următoarele blocuri:

Testează-ți programul (clichează pe steagul verde).

10.1.2 Inițializarea poziției și orientării - soluție

Testează proiectul de la adresa http://scratch.mit.edu/projects/41688062/#editor.

Dacă ai nevoie de ajutor, urmărește tutorialul video de la https://youtu.be/vVl1WOP0mKg.

10.2 Deplasează-l pe Karel la țintă

În această secțiune vei scrie un script pentru a-l deplasa pe Karel la țintă.

10.2.1 Direct la țintă – provocare

Folosește proiectul pe care l-ai creat anterior sau remixează proiectul de la adresa http://scratch.mit.edu/projects/41688204/#editor.

Cele două personaje ale proiectului sunt:

- Karel, care inițial se află în punctul (-100, -100) și este îndreptat spre est.

- Ținta, care este amplasată în punctul (100, -100).

Scrie un program pentru a-l deplasa pe Karel la țintă când apeși tasta **t**.

Poți folosi blocurile următoare:

În locul blocului **înaintează () pași**, poți folosi oricare dintre blocurile următoare:

Pentru a-ți testa programul, clichează pe steagul verde sau apasă tasta **t**.

10.2.2 Direct la țintă - soluții

Clichează pe steagul verde și apasă tasta **t** pentru a rula una dintre soluțiile posibile.

Compară următoarele patru soluții, disponibile la adresele următoare:

- http://scratch.mit.edu/projects/41688302/#editor
- http://scratch.mit.edu/projects/41688928/#editor
- http://scratch.mit.edu/projects/41689048/#editor
- http://scratch.mit.edu/projects/41689082/#editor

După cum poți observa vedea, în cazul primelor trei soluții, Karel se deplasează spre țintă aproape instantaneu. În cazul ultimei soluții, care folosește blocul **alunecă în () secunde la x: () y: ()**, Karel se deplasează spre țintă cu o viteză constantă. Acesta este comportamentul dorit al lui Karel.

În următoarea secțiune, vei învăța cum îl poți deplasa pe Karel cu o viteză limitată, folosind blocul **înaintează () pași**.

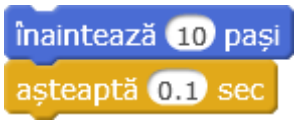
10.3 Limitează viteza

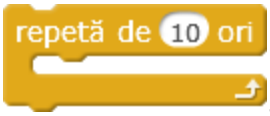
În ultima secțiune, ai învățat că poți folosi blocul **alunecă în () secunde la x: () y: ()** pentru a-l deplasa pe Karel spre țintă cu o viteză constantă. Dezavantajul folosirii blocului **alunecă în () secunde la x: () y: ()** este că acesta întrerupe derularea scriptului cât timp personajul se deplasează, împiedicând astfel scriptul să execute și alte comenzi (facă și alte lucruri) în timp ce personajul alunecă.

Din acest motiv, vei încerca să-l programezi pe Karel să se deplaseze cu o viteză constantă, folosind blocul **înaintează () pași.** În acest scop, vor fi repetate următoarele două acțiuni:

- Karel se va deplasa un număr de pași;
- Pauză de câteva fracțiuni de secundă.

Să presupunem că vrem să-l deplasăm pe Karel 100 de pași timp de 1 secundă. În cazul în care Karel face câte 10 pași o dată, trebuie să repetăm următorul script de 10 de ori:



Pentru a repeta scriptul anterior de 10 de ori, putem folosi blocul 

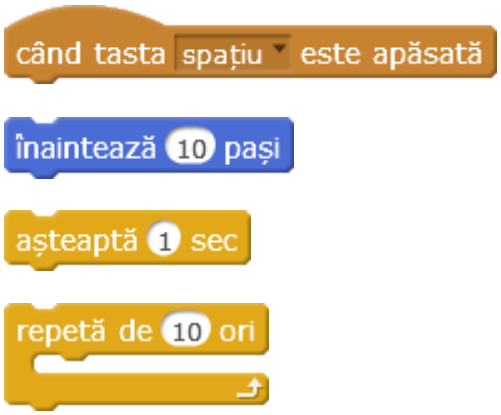
.

10.3.1 Limitează viteza - provocare

Remixează proiectul de la adresa http://scratch.mit.edu/projects/41690448/#editor.

Scrie un program care să-l deplaseze pe Karel la țintă când apeși tasta **t**. Karel trebuie să se deplaseze pe o distanță de 200 de pași în două secunde.

Trebuie să folosești următoarele blocuri:



Testează-ți programul (apasă steagul verde și apoi tasta **t**).

10.3.2 Limitează viteza - soluție

În prima soluție, http://scratch.mit.edu/projects/41690702/#editor, robotul se deplasează de 200 ori, câte un pas o dată (200 de repetiții temporizate cu câte 0,01 secunde). Dacă vei rula programul, vei vedea că robotul se deplasează mult mai încet decât am anticipat (aproape șapte secunde în loc de două).

În cea de a doua soluție, http://scratch.mit.edu/projects/41693402/#editor, Karel se deplasează doar de 20 ori, câte zece pași o dată (temporizare de 0,1 secunde). Karel se deplasează mai rapid ca în proiectul anterior, însă tot prea încet.

Comportamentul acesta se datorează faptului că folosim blocul de repetare (bucla), iar acesta este programat cu o mică întârziere între repetiții, prin urmare acțiunile sunt efectuate încet.

În cea de a treia soluția, http://scratch.mit.edu/projects/41690902/#editor, măsurăm timpul de execuție. Pentru aceasta, folosim o variabilă numită **moving time** (timp de mișcare). Variabila este inițializată la începutul deplasării. Următorul bloc, **reset timer** (resetare timer), resetează cronometrul. La finalul deplasării, variabila **moving time** este actualizată la valoarea cronometrului.

Valorile variabilei **moving time** diferă în funcție de setări. Modificați variabilele blocurilor din interiorul buclei de repetare conform valorilor din următorul tabel. Rulați programul și verificați dacă valorile variabilei **moving time** (timp de deplasare) sunt similare cu valorile din Tabelul 1.

Număr repetiții	Număr pași	Temporizare (secunde)	Timp de deplasare anticipat (secunde)	Timp de deplasare real (secunde)
200	1	0,01	2,0	6,667
100	2	0,02	2,0	3,333
20	10	0,1	2,0	2,133

Tabelul 1 – Introducerea de întârzieri în interiorul buclelor

După cum se observă, la o rezoluție mai mare, timpul de deplasare este mai mare decât cel anticipat.

10.4 Trei ţinte

Remixează proiectul de la adresa http://scratch.mit.edu/projects/42090906/#editor. Sunt patru personaje: Karel, Target1, Target2 și Target3. Am creat blocul **initialisation** (inițializare), necesar la inițializarea poziției (și orientării) personajului. Urmărește tutorialul de la adresa https://youtu.be/cfax_wg4id4 pentru a vedea cum se creează și folosește un bloc nou în Scratch.

Dacă vei consulta scripturile celor patru personaje, vei vedea că pornesc din următoarele poziții: Karel -> (-100, -100); Target1 -> (100, -100); Target2 -> (100, 100); Target3 -> (-100, 100). Aceasta înseamnă că personajele sunt amplasate în cele patru colțuri ale unui pătrat (Figura 19) cu latura de 200 unități (pași).

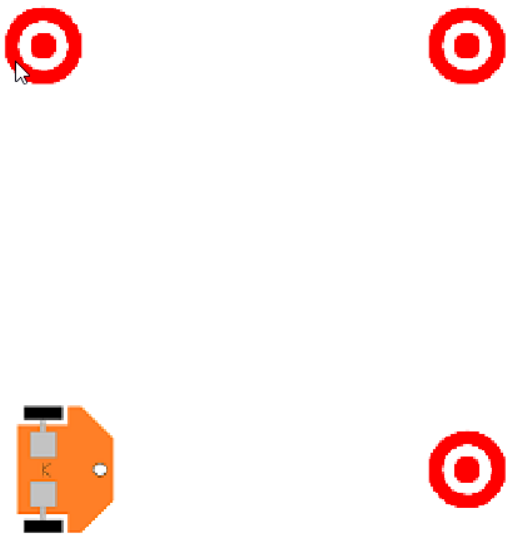


Figura 19 Karel și trei ținte

10.4.1 Trei ținte – provocarea 1

Scrie un script care să-l programeze pe Karel să viziteze fiecare țintă, apoi să revină la poziția și orientarea inițiale. Va trebui să folosești blocul **înaintează () pași**, iar Karel trebuie să se deplaseze cu o viteză limitată.

10.4.2 Trei ținte – soluție pentru provocarea 1

În Figura 20 prezint o posibilă soluție.

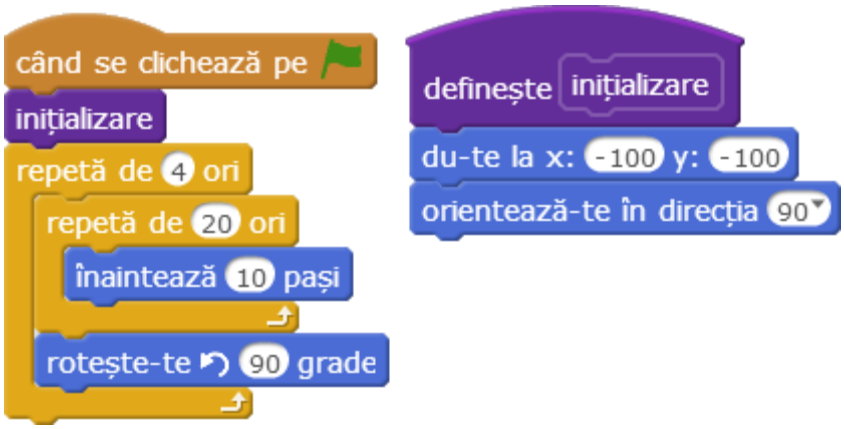


Figura 20 Trei ținte - scripturi pentru provocarea 1

10.4.3 Trei ținte – provocarea 2

Ar fi frumos să-l programezi pe Karel să-și exprima bucuria reușitei (atingerea țintei) cu un cântecel (câteva note). Continuă cu programul anterior sau remixează proiectul https://scratch.mit.edu/projects/102920310/#editor.

Creează un bloc **cântă** format din câteva instrucțiuni **cântă nota () pentru () timpi** și programează-l pe Karel să execute blocul **cântă** de fiecare dată când atinge ținta. Karel nu trebuie să se oprească în timp ce cântă.

10.4.4 Trei ținte – provocarea 2 – soluții

Prima soluție care mi-a venit în minte are scripturile din Figura 21.

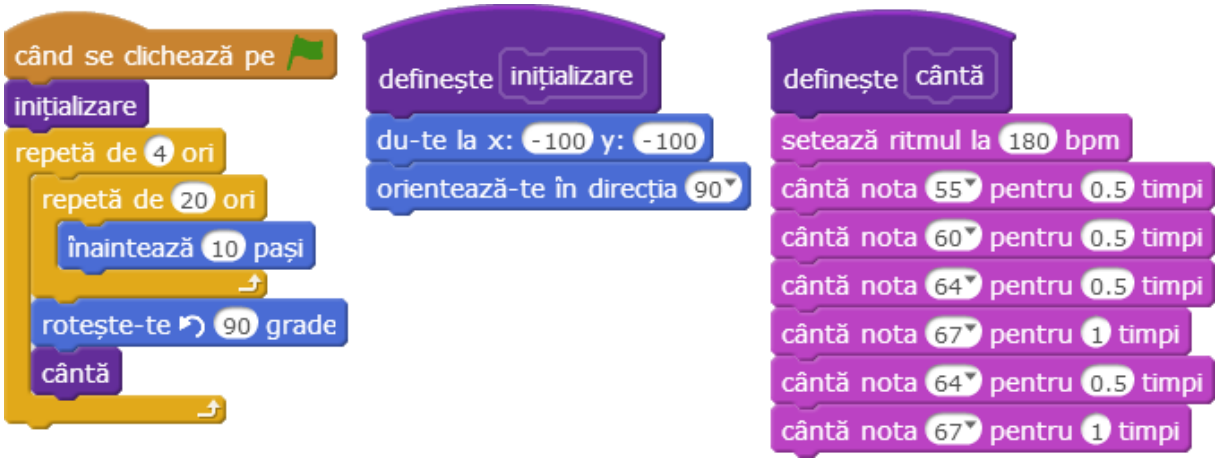


Figura 21 Trei ținte – provocarea 2 – scripturi soluție greșită

Poți crea un proiect cu aceste scripturi sau poți rula proiectul aflat la adresa https://scratch.mit.edu/projects/102920577/#editor.

Din păcate, soluția dată în acest proiect nu este bună din două motive:

1. Karel cântă melodia nu numai când atinge ținta, ci și când ajunge în poziția inițială.

2. Karel nu se deplasează cât timp cântă.

Putem soluționa aceste probleme folosind difuzarea: Karel va difuza mesajul **cântă** de fiecare dată când ajunge la o țintă. Acest eveniment va declanșa interpretarea cântecului. Astfel, Karel se va deplasa și va cânta în paralel (nu secvențial).

În soluția de mai jos, poți observa că am folosit blocul **dacă () atunci** pentru a stabili dacă personajul Karel atinge o țintă. Condiția blocului **dacă () atunci** este blocul boolean **atinge culoarea [] ?**, care raportează valoarea **adevărat** când Karel atinge o țintă. Bineînțeles, parametrul blocului boolean este culoarea roșie a țintelor.

Figura 22 Trei ținte – provocarea 2 – scripturi soluție bună

Se pot observa și alte modificări. Deoarece durata cântecului era mai mare decât timpul în care Karel se deplasa între două ținte, am micșorat cântecul, eliminând note și stabilind tempoul la 360 bătăi pe minut. În același timp, am redus viteza lui Karel, crescând numărul de repetiții ale buclei interne **repetă()** și micșorând ca atare numărul de pași ai blocului **înaintează () pași**.

10.4.5 Trei ținte – provocarea 3

Ar fi interesant dacă țintele și-ar schimba aspectul când sunt atinse de Karel.

Remixează programul https://scratch.mit.edu/projects/102923466/#editor.

După cum observi, fiecare țintă are două costume: unul roșu și unul albastru.

La început, toate țintele vor trebui să fie roșii. În acest scop, trebuie să faci modificări la blocul **inițializare** al fiecărei ținte.

Programează fiecare țintă să-și schimbe costumul din roșu în albastru când este atinsă de Karel.

10.4.6 Trei ținte – provocarea 3 – soluție

O posibilă soluție pentru scripturile Target1 este reprezentată în Figura 23

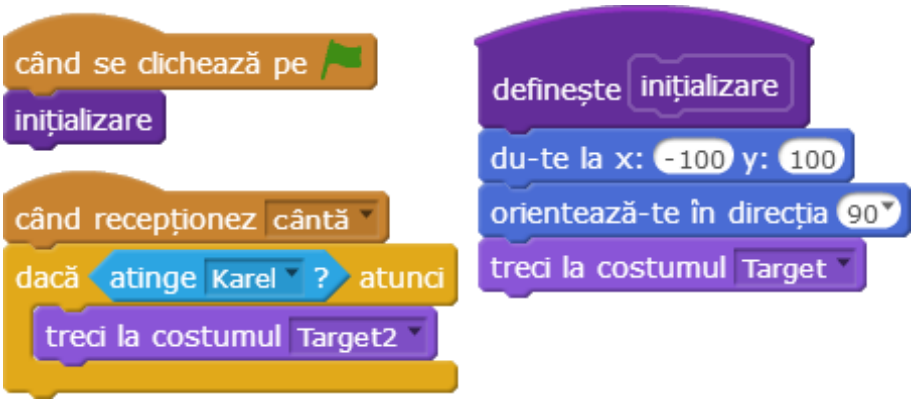


Figura 23 Trei ținte – provocarea 3 – soluție scripturi

10.4.7 Trei ținte – provocarea 4

Remixează proiectul de la https://scratch.mit.edu/projects/102925015/#editor http://scratch.mit.edu/projects/42119332/ - editorși efectuează modificările necesare la blocul **inițializare** al lui Karel, pentru a obține un comportament identic cu cel cu cel sugerat în Figura 24.

Dacă nu intuiești comportamentul cerut, poți rula proiectul (nu uita să clichezi pe steagul verde!) de la adresa https://scratch.mit.edu/projects/102926515/#editor, fără a te uita la scripturi.

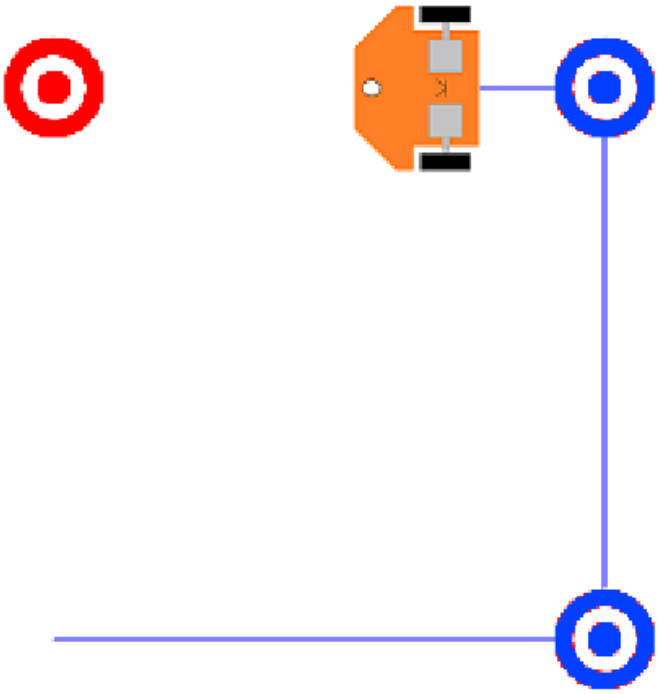


Figura 24 Evidențierea drumului parcurs și schimbarea culorii țintelor

10.4.8 Trei ținte – provocarea 4 – soluție scripturi

Modificarea comportamentului lui Karel se poate face prin modificarea scriptului care definește blocul **inițializare**, ca în Figura 25.

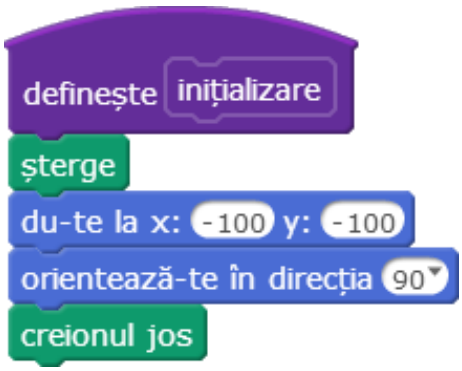


Figura 25 Trei ținte – provocarea 4 – modificare scripturi

10.5 Drawbot

Am văzut în ultima secțiune că putem crea desene în Scratch. În cadrul acestei secțiuni, vei învăța să creezi diverse tipuri de desene. În acest scop, îl vei folosi pe **Drawbot** (acronim care provine de la denumirea în engleză, **Draw**ing Ro**bot**—Robot Desenator în limba română) pentru a-l desena pe Karel.

10.5.1 Desenarea unui dreptunghi – provocare

Remixează proiectul de la adresa http://scratch.mit.edu/projects/42136036/#editor.
Creează blocul **rectangle(x, y, width, height)**, care desenează un dreptunghi cu următorii parametri:

(x, y) = coordonatele colțului stânga jos al dreptunghiului

width = lățimea dreptunghiului

height = înălțimea dreptunghiului

10.5.2 Desenarea unui dreptunghi – soluție

O posibilă soluție pentru definiția blocului **rectangle(x, y, width, height)** este dată în Figura 26. După cum se observă din definiția blocului, desenarea dreptunghiului se face astfel:

- Se începe din colțul stânga jos al dreptunghiului;
- Se desenează latura orizontală de jos;
- Se schimbă direcția cu 90 de grade în sens antiorar;
- Se desenează latura verticală din dreapta;
- Se schimbă direcția cu 90 de grade în sens antiorar;
- Se desenează latura orizontală de sus;
- Se schimbă direcția cu 90 de grade în sens antiorar;
- Se desenează latura verticală din stânga;
- Se schimbă direcția cu 90 de grade în sens antiorar, ajungându-se la orientarea inițială.

*Figura 26 Soluția pentru definiția blocului **rectangle (x, y, width, height)***

În proiectul de la adresa http://scratch.mit.edu/projects/40758538/#editor poți observa efectul optic al introducerii unui efect de nuanță (Figura 27).

Figura 27 Dreptunghiuri cu nuanțe diferite

10.5.3 Desenarea unui poligon – provocare

Remixează proiectul http://scratch.mit.edu/projects/42136036/#editor. Creează blocul **polygon(n, l, x, y)**, care desenează un poligon regulat cu **n** laturi cu lungimea **l**. Centrul poligonului este punctul de coordonate **(x, y)**.

10.5.4 Desenarea unui poligon – soluție

O posibilă soluție pentru definiția blocului **polygon(n, l, x, y)** este dată în Figura 28. Pentru a înțelege definiția acestui bloc, trebuie să-ți amintești câteva noțiuni elementare de geometrie.

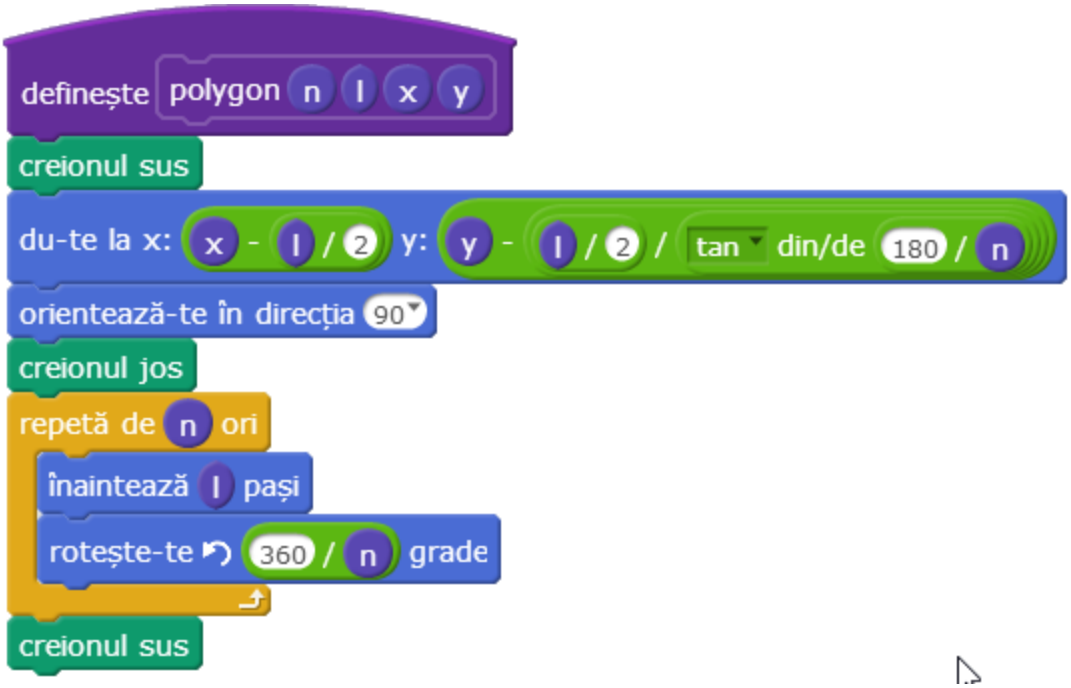


*Figura 28 Definiția blocului **polygon(n, l, x, y)***

Rulează proiectul de la adresa http://scratch.mit.edu/projects/42137406/#editor pentru a vedea cum **Drawbot** desenează poligoane cu diferite culori și număr de laturi.

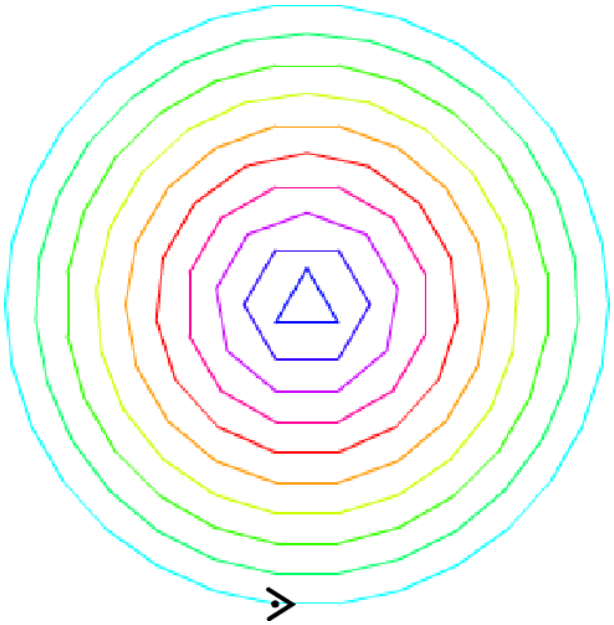


Figura 29 Poligoane cu număr de laturi diferite

După cum poți observa, un poligon cu multe laturi seamănă cu un cerc. În continuare, vei folosi aceste cunoștințe pentru a desena cercuri.

10.5.5 Desenarea un cerc – provocare

Remixează proiectul http://scratch.mit.edu/projects/42136036/#editor.

Creează blocul **circle(x, y, r)** care desenează un cerc. Centrul cercului are coordonatele **x** și **y**. Raza cercului este **r**. Vei aproxima cercul cu un poligon cu 90 de laturi. Folosește variabila **l** pentru a stoca valoarea lungimii laturii poligonului.

Folosește blocul **circle** pentru a desena o țintă.

10.5.6 Desenarea unui cerc – soluție

Poți folosi programul Scratch Paint Editor pentru a crea desene, însă cred că este mai simplu să-l programezi pe **Drawbot** să facă acest lucru în locul tău.

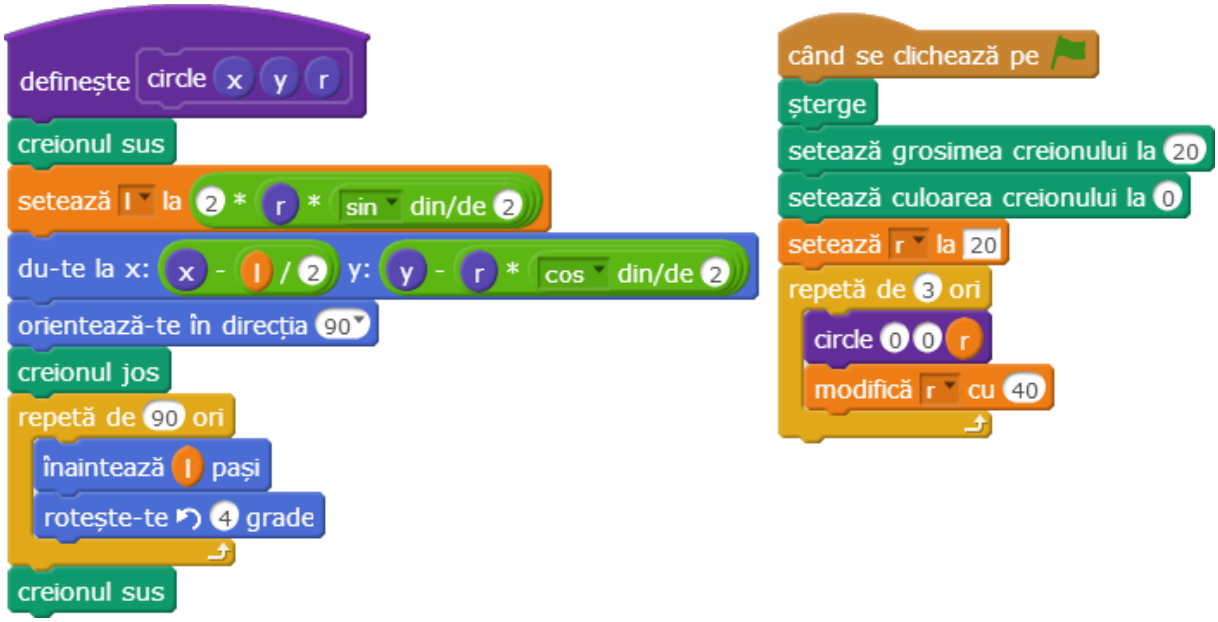


Figura 30 Scripturi pentru desenarea unui cerc și a unei ținte

În Figura 30 sunt prezentate scripturile pentru desenarea unui cerc și a unei ținte. Rulează proiectul de la adresa http://scratch.mit.edu/projects/42188032/#editor și vei vedea cum desenează **Drawbot** o țintă.

10.5.7 Desenarea lui Karel – provocare

Îl vei programa pe **Drawbot** să-l deseneze pe Karel.

Remixează proiectul de la adresa http://scratch.mit.edu/projects/42193186/#editor, care cuprinde următoarele blocuri noi:

- **draw rectangle(x, y, width, height)**
- **draw circle(x, y, r)**
- **set pen(color, shade, size)**

Creează blocul **draw chassis** pentru a desena șasiul lui Karel conform imaginii din Figura 31. Folosește blocul **du-te la x: () y: ()**.

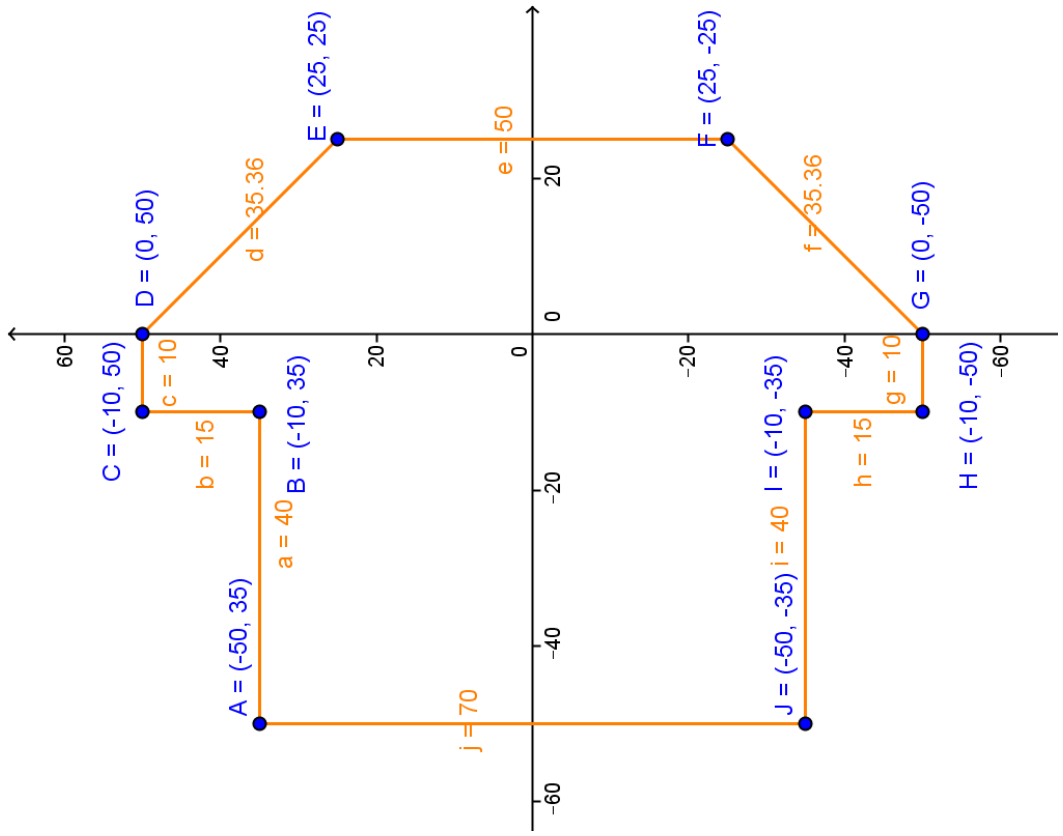


Figura 31 Șasiul robotului Karel

Folosește blocurile **draw rectangle (x, y, width, height)**, **draw circle (x, y, r)** și **draw chassis** pentru a-l desena pe Karel ca în Figura 32.

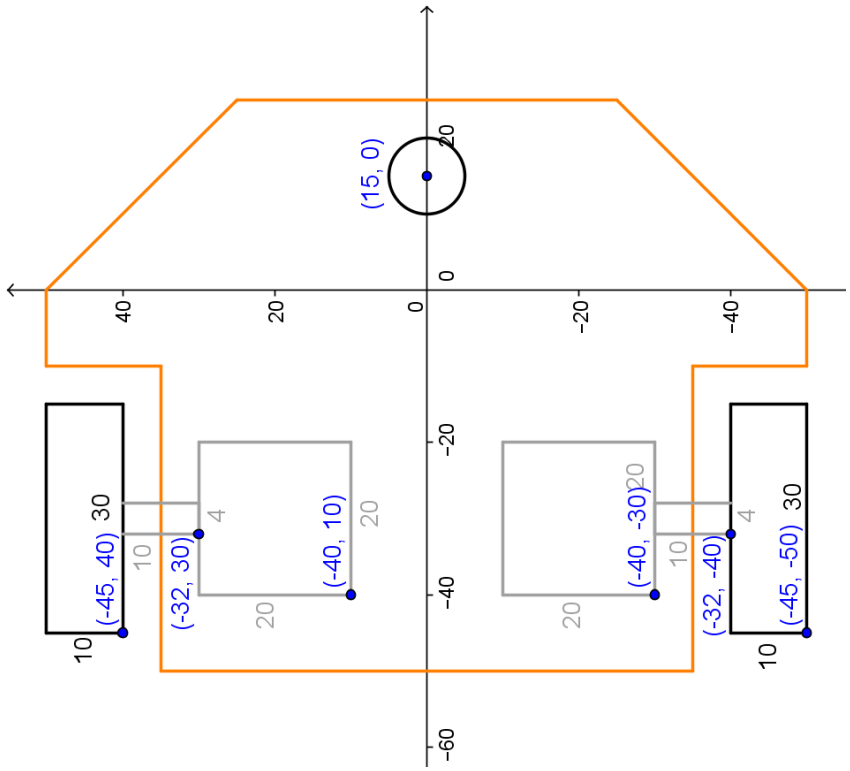


Figura 32 Desenul robotului Karel

10.5.8 Desenarea lui Karel – soluție

În Figura 33 este prezentată definiția blocului **draw chassis**.

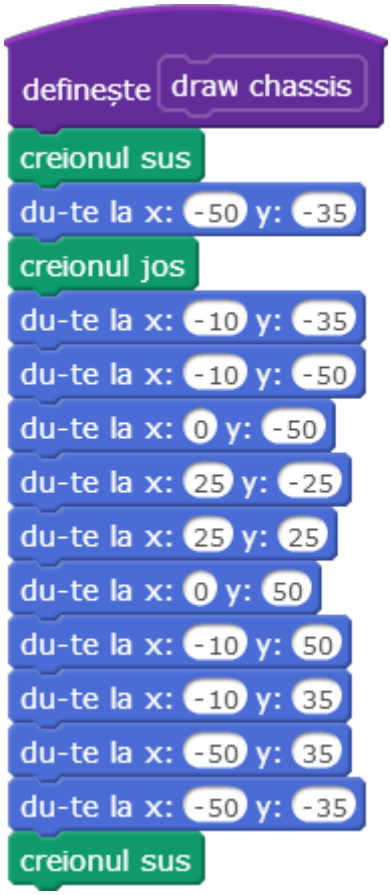


*Figura 33 Definiția blocului **draw chassis***

În proiectul de la adresa https://scratch.mit.edu/projects/42193628/#editor, Drawbot îl desenează pe Karel la o scară între 1 și 3, aleasă de utilizator. Se folosește funcția recursivă **check answer**, pentru a verifica dacă răspunsul se încadrează în domeniul 1 - 3. Vezi ce se întâmplă dacă nu introduci un număr între 1 și 3.

10.6 Țintă mobilă

Unul din comportamentele unui robot constă în urmărirea altui robot sau al unei persoane, asemănător unei escorte. În această secțiune îl vei programa pe Karel să urmărească o țintă mobilă.

10.6.1 Țintă mobilă – provocare

Creează un proiect în care Karel se deplasează către o țintă mobilă ca în filmul postat la adresa https://youtu.be/498vHVw_ICs. Programul va fi rulat după clicarea pe steagul verde. Vei deplasa ținta pe scenă cu ajutorul mouse-ului. Traseul parcurs de robot va fi evidențiat cu ajutorul creionului. Prin apăsarea tastei **c** se va șterge traseul desenat de creion.

10.6.2 Țintă mobilă – soluție

Proiectul de la pagina http://scratch.mit.edu/projects/41846786/#editor reprezintă o posibilă soluție care respectă constrângerile impuse în provocare.

11 Robot care evită obstacolele

Obiectiv: să programezi robotul Karel să evite obstacolele.

11.1 Ce este un robot care evită obstacolele?

Poți observa cum se comportă roboții care evită obstacolele, dacă urmărești filmul de la adresa https://youtu.be/4PzV8L0Ro2E.

Scopul unui robot care evită obstacolele (Figura 34) este să evite coliziunea cu obiectele înconjurătoare dintr-un mediu necunoscut.

Robotul are senzori care detectează obstacolele. De obicei, robotul se deplasează înainte. Când senzorii detectează un obstacol, robotul evită obstacolul și își continuă deplasarea înainte.

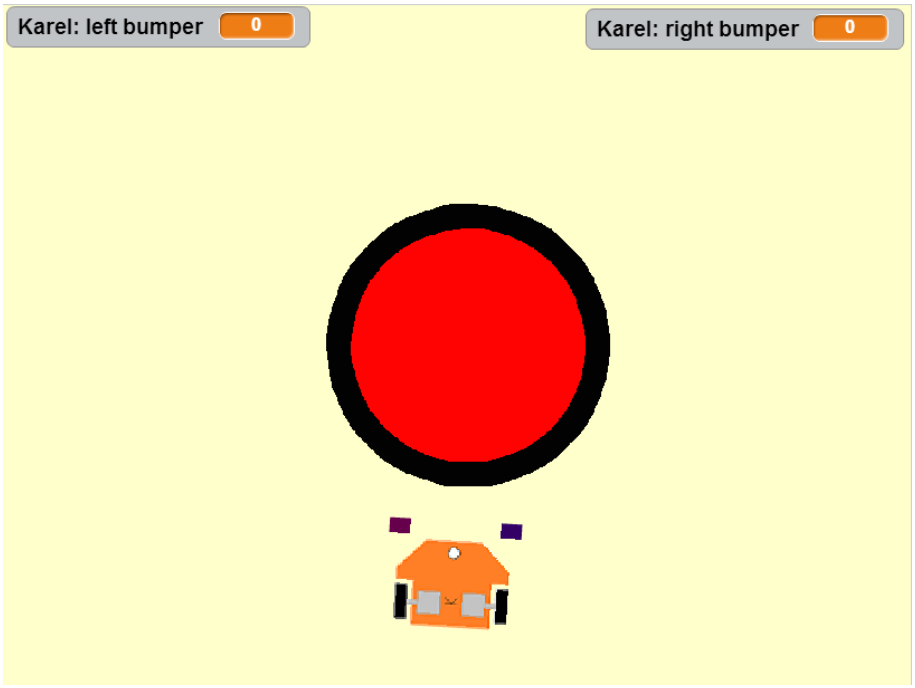


Figura 34 Robot care evită obstacolele

11.1.1 Senzori pentru un robot care evită obstacolele

Roboții adevărați folosesc senzori mecanici, optici sau cu ultrasunete pentru a detecta obstacolele. Cea mai accesibilă variantă de senzor tactil pentru un robot real poate fi realizată cu un comutator monostabil, acționat de un fir metalic rigid prin intermediul căruia este acționat comutatorul. Acesta comută în starea ON când lovește obstacolul.

Senzorii sunt ușor de simulat în Scratch. Putem concepe tamponul ca fiind un dreptunghi plin, amplasat în fața robotului. Pentru a detecta dacă senzorul-tampon atinge un obstacol, folosim blocul de detecție boolean **<culoarea [] atinge []?>** (Figura 35).

*Figura 35 Blocul **<culoarea [] atinge []?>***

11.1.2 Controlul unui robot care evită obstacolele

Un robot cu doi senzori frontali (drept și stâng) poate fi controlat astfel:

- Dacă niciun senzor nu atinge obstacolul, robotul se deplasează înainte.

- Dacă senzorul stâng atinge obstacolul, robotul se rotește spre dreapta (în sens orar).

- Dacă senzorul drept atinge obstacolul, robotul se rotește spre stânga (în sens antiorar).

- Dacă ambii senzori ating obstacolul, robotul se rotește cu 90 de grade.

Pentru a menține robotul pe scenă în cazul în care ajunge la marginea scenei, putem folosi blocul **dacă atingi marginea, ricoșează** (Figura 36).

Figura 36 Blocul care nu lasă personajul să părăsească scena

11.2 Senzori tactili

În această secțiune, îi vei adăuga doi senzori tactili lui Karel.

11.2.1 Senzori tactili – provocare

Remixează proiectul de la http://scratch.mit.edu/projects/41723100/#editor. După cum observi, obstacolul este un cerc roșu cu margine neagră, amplasat în mijlocul scenei. Senzorii trebuie să detecteze conturul negru al cercului.

1. Adaugă doi senzori în fața lui Karel (două dreptunghiuri mici de culori diferite).

2. Creează două variabile care stochează starea senzorilor: **left bumper** (senzor stânga) și **right bumper** (senzor dreapta).

3. Creează blocul **read sensors** (citește senzorii), care citește starea senzorilor și actualizează valorile celor două variabile care stochează stările senzorilor: **left bumper** și **right bumper.** Senzorii iau valoarea 1 dacă ating obstacolul (adică ating culoarea neagră) și 0 în rest. Poți folosi blocul **<culoarea [] atinge []?>** pentru a verifica dacă senzorii ating marginea obiectului (culoarea neagră).

4. Testează programul deplasându-l pe Karel în diverse poziții, cu orientări diferite, și verificând valorile variabilelor **left bumper** și **right bumper** afișate pe scenă.

11.2.2 Senzori tactili – soluții

În Figura 37 sunt prezentate două posibile soluții pentru definirea blocului **read bumpers**.

Proiectele sunt disponibile la http://scratch.mit.edu/projects/41722776/#editor și http://scratch.mit.edu/projects/41722656/#editor. Clichează pe steagul verde pentru a rula soluția aleasă. După cum observi, robotul urmărește cursorul mouse-ului. Deplasează senzorii peste culoarea neagră și verifică stările celor doi senzori afișate pe scenă. Asigură-te că în ambele soluții blocurile pentru citirea senzorilor au un comportament identic.

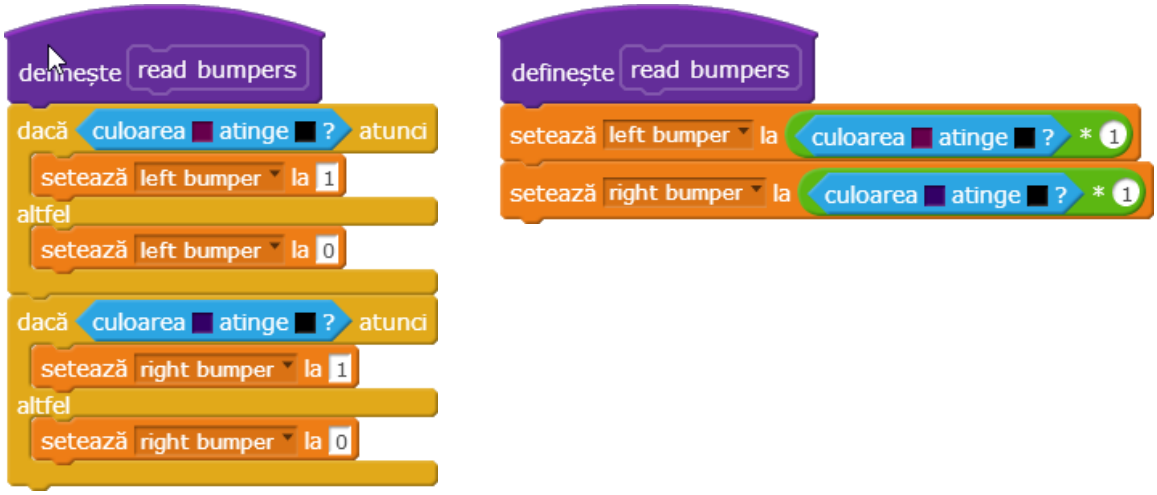


*Figura 37 Două soluții posibile pentru definirea blocului **read bumpers***

11.3 Evitarea obstacolelor

În această secțiune, vei scrie un program pentru un robot care evită obstacolele.

11.3.1 Evitarea obstacolelor - provocare

Remixează proiectul de la http://scratch.mit.edu/projects/41723854/#editor.

Controlează robotul folosind algoritmul din secțiunea 11.1.2.

Testează programul cu valori diferite ale vitezei (variabila **speed**).

11.3.2 Evitarea obstacolelor - soluții

Există 2 soluții disponibile la http://scratch.mit.edu/projects/41722446/#editor și http://scratch.mit.edu/projects/41720918/#editor.

Clichează pe steagul verde pentru a rula oricare dintre soluții. Modifică viteza și observă ce se întâmplă.

Compară cele două implementări ale aceluiași algoritm în cele două soluții.

11.4 Mai multe teste, mai multă distracție

Comportamentul robotului depinde de forma și poziția obstacolelor de pe scenă. Este timpul să creezi mai multe teste pentru robot (decoruri diferite).

11.4.1 Decoruri pentru testarea evitării obstacolelor – provocare

Remixează proiectul de la adresa https://scratch.mit.edu/projects/41725006/#editor.

Creează decoruri cu dispuneri diferite ale obstacolelor (e.g. Figura 38).

Testează comportamentul lui Karel pentru decoruri diferite.

Găsește soluții pentru modificarea scriptului în cazul în care Karel se blochează într-o anumită poziție.

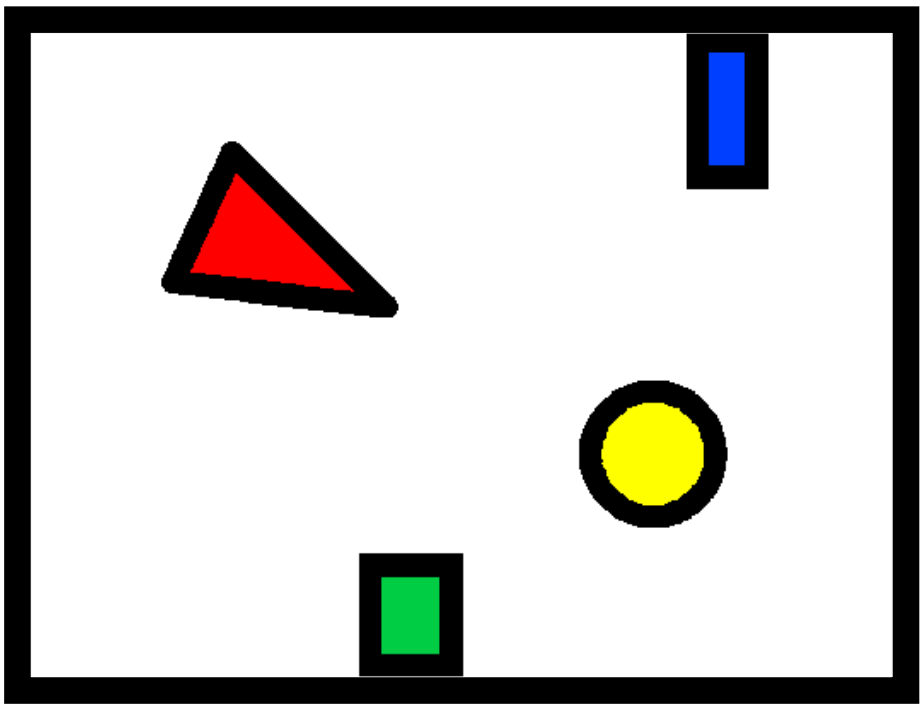


Figura 38 Decor pentru testarea evitării obstacolelor

11.4.2 Decoruri pentru testarea evitării obstacolelor – soluție

Proiectul de la adresa http://scratch.mit.edu/projects/41725104/#editor are decoruri noi, cu ajutorul cărora se poate testa abilitatea robotului de a se descurca în diferite medii.

În cazul unuia dintre decoruri, Karel se împotmolește, după cum poți observa dacă în proiectul anterior selectezi decorul din Figura 39 și rulezi programul prin clicarea steagului verde. Remixează proiectul și caută o soluție la problema lui Karel.

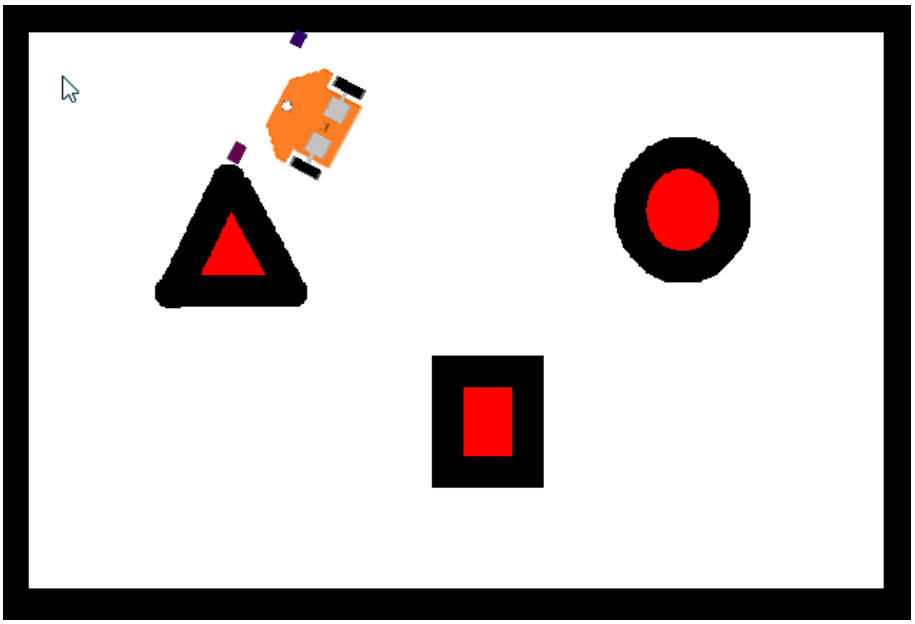


Figura 39 Decor care îi creează probleme lui Karel

11.5 Doi roboți pe scenă

E timpul pentru o sarcină mai dificilă. Vom avea doi roboți pe scenă simultan (Figura 40). Roboții trebuie să evite obstacolele și să se evite unul pe celălalt. O soluție posibilă ar fi să întoarceți robotul când distanța față de celălalt robot scade sub o anumită valoare. Este o acțiune ușor de implementat în Scratch.

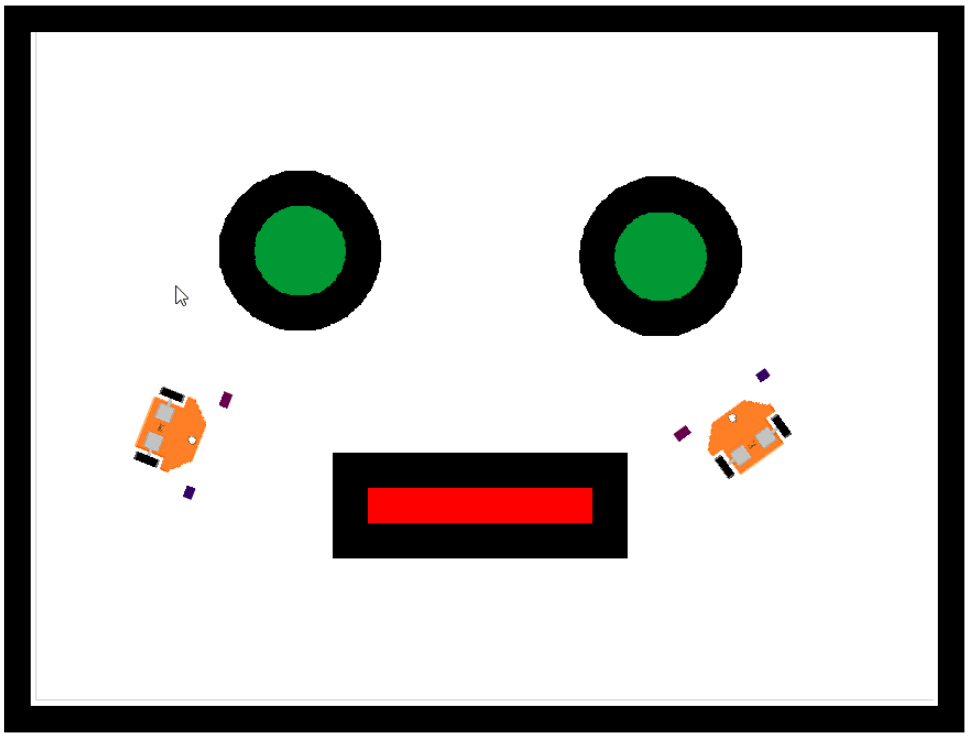


Figura 40 Doi roboți pe scenă

11.5.1 Doi roboți pe scenă – provocare

Remixează proiectul de la adresa http://scratch.mit.edu/projects/41726068/#editor.

După cum observi, există doi roboți pe scenă: Karel1 și Karel2.

1. Rulează programul pentru a verifica dacă ambii roboți evită obstacolele.

2. Creează blocul **avoid robot** (evită robot) pentru ca robotul **Karel1** să-l evite pe **Karel2**.

3. Testează programul.

4. Scrie un bloc similar, cu ajutorul căruia **Karel2** să-l evite pe **Karel1** și repetă testul.

11.5.2 Doi roboți pe scenă – soluție

La adresa http://scratch.mit.edu/projects/41727484/#editor este prezentată o soluție pentru doi roboți care evită atât obstacolele cât și unul pe celălalt. Poți remixa și îmbunătăți programul.

12 Robot care urmărește peretele

Obiectiv: să îl programezi pe Karel să urmărească un perete.

12.1 Ce este un robot care urmărește peretele?

Urmărește filmul postat la adresa https://youtu.be/p6MQ_LDUx3w, pentru a observa cum urmărește Karel peretele.

Un robot urmărește un perete atunci când se deplasează pe lângă perete păstrând o anumită distanță față de acesta, fără să-l lovească sau să se îndepărteze prea mult de el.

Karel va urmări peretele cu ajutorul regulii mâinii stângii, care spune că peretele va fi întotdeauna pe partea stângă a robotului.

12.1.1 Senzori pentru un robot care urmărește peretele

Ca și în cazul robotului care evită obstacole, robotul care urmărește peretele va folosi senzori tactili digitali. Acești senzori au două stări: activat (senzorul atinge peretele) și dezactivat (senzorul nu atinge peretele).

Roboții adevărați folosesc și senzori analogici, care raportează distanța față de perete. În cadrul acestei unități, vom încerca să simulăm un senzor care raportează o distanță cu valori din intervalul 1 - 6.

12.1.2 Controlul unui robot care urmărește peretele

Putem controla robotul care urmărește peretele după cum urmează:

- Dacă senzorul raportează distanța corectă față de perete, robotul se deplasează înainte.

- Dacă robotul este prea aproape de perete, se îndepărtează de acesta.

- Dacă robotul este prea departe de perete, se va apropia de acesta.

Acțiunile robotului pot fi proporționale cu valoarea erorii reale.

12.2 Senzorul de distanță

În această secțiune, vei proiecta un senzor de distanță.

12.2.1 Introducere

În mod normal, roboții folosesc senzori optici (IR) sau cu ultrasunete pentru a măsura distanța față de un obiect.

În Scratch, este ușor să calculăm distanța dintre spiriduși, definită drept distanța dintre centrele acestora. Putem calcula distanța dintre spiriduși dacă ambii sunt rotunzi, micșorând distanța dintre centre cu suma razelor celor doi spiriduși. Metoda nu este însă bună pentru măsurarea distanței față de un zid.

Știm că putem folosi blocul **<color [] is touching []?>** pentru a crea un senzor tactil. Putem folosi același bloc pentru a crea un senzor de distanță. Senzorul nostru va fi un băț obținut

din dreptunghiuri de diferite culori. Cu ajutorul blocului **<color [] is touching []?>**, putem calcula distanța față de marginea unui obiect.

12.2.2 Senzor de distanță – provocare

Remixează proiectul de la adresa http://scratch.mit.edu/projects/41760210/#editor.

Adăugă un senzor de distanță în partea stângă a lui Karel. Senzorul de distanță este alcătuit din șase dreptunghiuri de diferite culori (Figura 41). Senzorul nu este orientat perpendicular pe axa longitudinală a robotului, ci la un unghi de 45 de grade față de axa longitudinală a robotului.

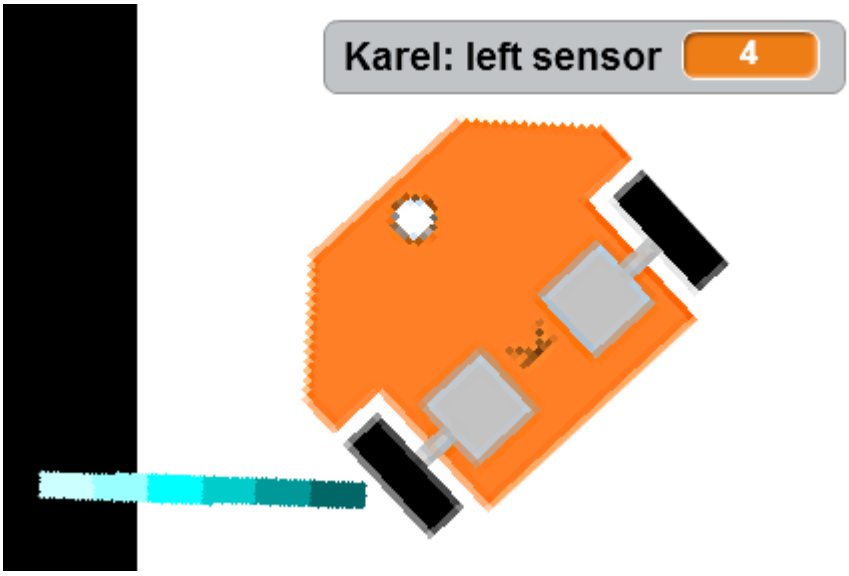


Figura 41 Senzor de distanță cu valori în intervalul 1 – 6

Creează un bloc denumit **read left sensor** (citește senzorul stâng), care va actualiza variabila **left sensor** (senzorul stâng) cu o valoare variind între 0 și 6, în funcție de distanța față de perete. Valoarea 6 înseamnă că senzorul nu atinge peretele.

Creează un program care să citească valoarea senzorului stâng, să o stocheze în variabila **left sensor** și să o afișeze pe scenă.

Testează programul modificând distanța dintre senzor și perete și verificând valoarea afișată a variabilei **left sensor**.

12.2.3 Senzor de distanță – soluție

O soluție este la adresa http://scratch.mit.edu/projects/41758864/#editor. Clichează pe steagul verde pentru a rula programul. Apasă tasta **săgeată stânga** pentru a mișca robotul spre stânga. Apasă tasta **săgeată dreapta** pentru a mișca robotul spre dreapta. Verifică valoarea distanței afișate pe scenă.

Remixează și îmbunătățește soluția anterioară.

12.3 Robot care urmărește peretele cu ajutorul unui singur senzor

În această secțiune, îl vei programa pe Karel să urmărească un perete folosind doar senzorul de distanță proiectat în secțiunea anterioară.

12.3.1 Urmăritor de perete cu un singur senzor – introducere

Vrem ca robotul Karel să urmărească peretele la o distanță dată (de exemplu, 3). Pentru a-l programa pe Karel să urmeze peretele, trebuie să citim valoarea senzorului, care poate varia între 0 și 6. Trebuie să calculăm eroarea de distanță, definită drept diferența dintre valoarea reală a senzorului de distanță și valoarea de referință (3).

Eroarea are o valoare pozitivă dacă distanța robot – perete este mai mare decât 3. În acest caz, Karel se va roti spre stânga (către perete). Eroarea are o valoare negativă dacă distanța robot – perete este mai mică decât 3. În acest caz, Karel se va roti spre dreapta (se va îndepărta de perete). Unghiul de rotire va fi proporțional cu valoarea erorii.

12.3.2 Urmăritor de perete cu un singur senzor – provocare

Remixează proiectul http://scratch.mit.edu/projects/41757022/#editor.

După cum observi, Karel are un senzor de distanță care poate fi citit cu ajutorul blocului **read left sensor** (citește senzorul stâng). Distanța reală (un număr între 0 și 6) este stocată în variabila **left sensor** (senzorul stâng).

Creează o variabilă denumită **lateral error** (eroarea laterală) pentru a stoca eroarea definită ca diferența dintre distanța reală stocată în variabila **left sensor** și distanța de referință (de ex. 3).

Controlează comportamentul lui Karel astfel:

- Dacă distanța față de perete este mai mică decât valoarea de referință (3), îndepărtează-te de perete.

- Dacă distanța față de perete este mai mare decât valoarea de referință (3), apropie-te de perete. Acțiunea poate fi realizată foarte ușor, rotind robotul în sens antiorar cu un unghi direct proporțional cu valoarea variabilei **lateral error**.

- Karel se deplasează înainte cu un număr de pași egal cu valoarea variabilei **speed**.

Testează programul cu decoruri și diferite valori ale parametrilor lui Karel (de ex. viteză, constantă de proporționalitate).

12.3.3 Urmăritor de perete cu un singur senzor – soluție

O posibilă soluție este la adresa http://scratch.mit.edu/projects/41756484/#editor. Clichează pe steagul verde pentru a rula această soluție. Observă ce se întâmplă dacă mărești viteza.

Se observă că robotul Karel întâmpină probleme cu urmărirea peretelui atunci când mărim viteza. Vom încerca să soluționăm problema în următoarea secțiune, în care vom proiecta un robot care urmărește peretele cu ajutorul a doi senzori de distanță.

Karel urmărește peretele doar dacă este poziționat bine la început, el nefiind programat să găsească peretele dacă nu se află în poziția inițială corectă.

Remixează și optimizează soluția prezentată.

12.4 Robot care urmărește peretele cu ajutorul a doi senzori

În această secțiune vei îmbunătăți comportamentul de urmărire a peretelui al lui Karel cu ajutorul a doi senzori de distanță.

12.4.1 Urmăritor de perete cu doi senzori – introducere

După cum am văzut în secțiunea anterioară, robotul care urmărește peretele cu un singur senzor poate întâmpina dificultăți când mărim viteza. Situația se datorează vitezei mici de reacție a robotului în cazul în care există un perete în fața lui.

Problema se poate soluționa folosind un senzor în fața robotului. Când acest senzor detectează un perete în fața sa, robotul se va roti spre dreapta și va continua să urmărească peretele cu ajutorul senzorului lateral.

12.4.2 Urmăritor de perete cu doi senzori – provocare

Remixează proiectul http://scratch.mit.edu/projects/41753442/#editor.

După cum observi, Karel are un senzor lateral care este citit folosind blocul **read left sensor** (citește senzorul stâng). Variabila **speed** stochează viteza lui Karel.

1. Adăugă un senzor de distanță în fața lui Karel.

2. Creează un bloc **read front sensor** (citește senzorul frontal) pentru a citi informațiile de la senzorul frontal și a le stoca în variabila **front sensor** (senzor frontal).

3. Definește blocul **read sensors** (citești senzorii) care citește ambii senzori (stânga și frontal).

4. Calculează erorile ambilor senzori și stochează-le în variabilele **frontal error** (eroarea frontală) și **lateral error** (eroarea laterală).

5. Controlează-l pe Karel astfel:

- Dacă senzorul frontal detectează un perete, virează la dreapta.

- În oricare altă situație, urmărește peretele folosind informații de la senzorul stâng.

6. Proiectează diverse decoruri pentru a testa comportamentul de urmărire a peretelui.

Testează robotul la viteze și valori diferite ale parametrilor.

12.4.3 Urmăritor de perete cu doi senzori – soluție

O soluție posibilă este la adresa http://scratch.mit.edu/projects/41750262/#editor.
Clichează pe steagul verde pentru a rula această soluție.

Remixează și optimizează soluția prezentată.

13 Robot urmăritor de linie

Obiectiv: să îl programezi pe Karel să urmărească o linie neagră pe o suprafață alb.

13.1 Ce este un robot care urmărește linia?

Urmărește filmul postat la adresa https://youtu.be/HjdDp7oMXTA, pentru a observa modul în care Karel urmărește linia.

Un robot urmăritor de linie este capabil să se deplaseze de-a lungul unui traseu (Figura 42). De obicei, traseul este o linie neagră pe o suprafață albă sau o linie albă pe o suprafață neagră. Concursurile de urmărirea liniei sunt foarte populare și au regulamente similare cu cel din [3].

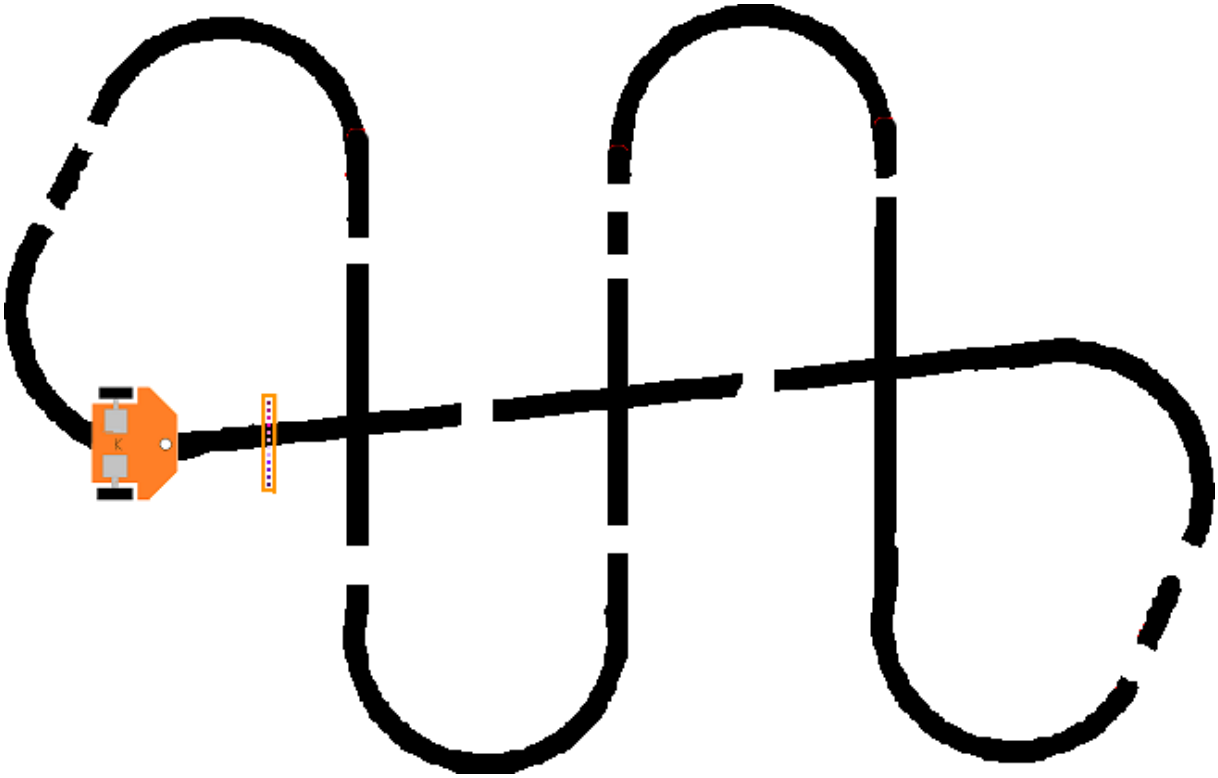


Figura 42 Robot urmăritor de linie

13.1.1 Urmăritor de linie – senzori

În mod normal, roboții care urmăresc linia folosesc senzori optici (IR), plasați frontal, pentru a detecta linia. Robotul care urmărește linia poate fi prevăzut doar cu un senzor digital, care poate fi analogic sau digital.

În cazul simulării în Scratch, vom folosi senzori digitali, care semnalează doar dacă senzorul este deasupra liniei sau nu. În cadrul acestei lecții, vom proiecta două variante de roboți: una cu doi senzori și alta cu 12 senzori.

13.1.2 Urmăritor de linie - control

Putem controla robotul care urmărește linia după cum urmează:

- Dacă robotul este deasupra liniei, atunci se va deplasa înainte.

- Dacă robotul este în stânga liniei, se va roti spre dreapta (în sens orar).

- Dacă robotul este în dreapta liniei, se va roti spre stânga (în sens anti orar).

Acțiunile robotului pot fi direct proporționale cu valoarea erorii reale.

13.2 Robot care urmărește linia cu ajutorul a doi senzori

Putem construi un robot care urmărește corect linia doar cu ajutorul a doi senzori. În această secțiune, îl vom transforma pe Karel într-un robot care urmărește linia cu ajutorul a doi senzori.

În continuare voi prezenta caracteristicile senzorilor și modul de control al robotului.

13.2.1 Senzorii unui urmăritor de linie cu doi senzori

Senzorii de linie sunt două mici dreptunghiuri (sau cercuri) de diferite culori, amplasate în fața robotului, unul pe partea stângă, altul pe dreapta (Figura 43). Distanța dintre marginile senzorilor trebuie să fie egală cu lățimea liniei.

Figura 43 Urmăritor de linie cu doi senzori

Vom folosi blocul **<culoarea [] atinge []?>** pentru a verifica dacă senzorul se află deasupra liniei (de culoare neagră). Valoarea senzorului este 1 dacă acesta se află deasupra liniei și 0 în caz contrar.

13.2.2 Urmăritor de linie cu doi senzori – control

Este foarte ușor să-l controlăm pe Karel:

1. Dacă senzorii au aceeași valoarea (0 sau 1), Karel se deplasează înainte.

2. Dacă senzorii au valori diferite, Karel se rotește spre partea senzorului cu valoarea mai mare (de ex., dacă senzorul stâng = 1 și cel drept = 0, atunci Karel se rotește spre stânga, adică în sens antiorar).

13.2.3 Provocare: Robot urmăritor de linie cu doi senzori

Remixează proiectul de la adresa http://scratch.mit.edu/projects/41779016/#editor.

Pentru a-l face pe Karel să urmărească linia poți urma următorii pași:

1. Amplasează doi senzori de linie (dreptunghiuri de diferite culori) în fața lui Karel.

2. Creează blocul **read sensors** (citește senzorii), care actualizează valorile variabilelor **left sensor** (senzor stânga) și **right sensor** (senzor dreapta). Valoarea este 1 dacă senzorul se află deasupra liniei și 0 în caz contrar.

3. Creează variabila **error** (eroare), care stochează diferența dintre variabilele **left sensor** și **right sensor**.

4. Creează variabilele **speed** (viteza) și **Kp** (constanta de proporționalitate), care stochează doi parametri ai robotului: viteza și constanta de proporționalitate.

5. **Control**ează robotul astfel:

- Robotul se rotește spre stânga cu un unghi egal cu produsul dintre **Kp** și **error** (de reținut că robotul se va roti spre dreapta dacă unghiul este negativ).

- Robotul se deplasează înainte cu un număr de pași egal cu variabila **speed**.

Testează comportamentul lui Karel pe trasee diferite (pentru schimbarea traseului se alege alt decor), precum și cu valori diferite ale parametrilor **Kp** și **speed**.

Ce se întâmplă dacă mărești viteza?

13.2.4 Soluție: Robot urmăritor de linie cu doi senzori

Clichează pe steagul verde pentru a rula una dintre soluțiile posibile, care poate fi accesată la http://scratch.mit.edu/projects/41770786/#editor. Ce se întâmplă dacă mărești viteza?

Remixează soluția prezentată și îmbunătățește-o pentru a mări viteza maximă.

13.3 Robot urmăritor de linie cu 12 senzori

În cadrul acestei secțiuni, îi vei adăuga 12 senzori de linie lui Karel, pentru a obține un robot care urmărește eficient linia.

13.3.1 Particularitățile unui robot urmăritor de linie cu 12 senzori

În secțiunea anterioară, am văzut că un robot care urmărește linia cu ajutorul a doi senzori pierde linia dacă raza minimă a curbelor este prea mică sau dacă viteza robotului este prea mare. Putem construi un robot care urmărește mai bine linia, cu ajutorul mai multor senzori, în cazul nostru, 12.

Cei 12 senzori de linie sunt digitali (1 deasupra liniei, 0 în caz contrar), prin urmare putem avea 2^12 = 4096 combinații. Chiar dacă multe dintre combinații nu se regăsesc în realitate, pare destul de dificil de interpretat informația oferită de senzori.

De fapt, este ușor să calculăm eroarea cu ajutorul informațiilor primite de la cei 12 senzori de linie: folosim media ponderată. Să presupunem că pozițiile celor 12 senzori, de la stânga la dreapta (imaginați-vă că sunteți șoferul) sunt -11, -9, …, -1, 1, …, 9, 11. Notăm valorile

senzorilor de linie, de la stânga la dreapta, cu S1, S2… S12. Dacă cel puțin un senzor se află deasupra liniei, eroarea poate fi calculată cu ajutorul formulei din Figura 44.

$$error = \frac{\sum_{n=1}^{12}(2 * n - 13) * S_n}{\sum_{n=1}^{12} S_n}$$

Figura 44 Formula pentru calculul erorii

Robotul este controlat la fel ca în varianta cu doi senzori.

13.3.2 Urmăritor de linie cu 12 senzori – provocare

Remixează proiectul http://scratch.mit.edu/projects/41807096/#editor.

După cum observi, Karel are deja 12 senzori de linie în față. Dacă dorești, poți reproiecta senzorii pentru a obține performanțe mai bune, modificând forma și dimensiunea senzorilor și / sau distanța dintre ei.

Creează blocul **read sensors** (citește senzorii), care actualizează valorile variabilelor S1 – S12, variabile ce stochează valorile senzorilor de linie.

Creează blocul **calculate error** (calculează eroarea), care calculează eroarea cu ajutorul mediei ponderate. Poți folosi unele variabile auxiliare precum **numerator** (numărător) și **denominator** (numitor). Eroarea este stocată în variabila **actual error** (eroarea actuală). Este utilă crearea unei alte variabile, **last error**, care să stocheze eroarea anterioară.

Creează variabilele **speed** și **Kp**, care stochează doi parametri ai robotului: viteza și constanta de proporționalitate.

Controlează robotul după cum urmează:

- Rotește robotul spre stânga cu un unghi egal cu produsul dintre variabilele **Kp** și **actual error** (de reținut că robotul se va roti spre dreapta dacă unghiul este negativ).

- Deplasează robotul înainte cu un număr de pași egal cu valoarea variabilei **speed**.

Testează robotul cu trasee diferite, precum și cu valori diferite ale parametrilor **Kp** și **speed**.

13.3.3 Urmăritor de linie cu 12 senzori – soluție

Există două soluții la adresele http://scratch.mit.edu/projects/41779714/#editor și http://scratch.mit.edu/projects/41800832/#editor. Clichează pe steagul verde pentru a vedea robotul în acțiune. Poți modifica valorile variabilelor **speed** și **Kp** (factorul de proporționalitate) pentru a vedea cum schimbă comportamentul robotului.

Dacă apeși tasta **săgeată jos**, creionul este lăsat jos și poți vedea traiectoria urmată de centrul robotului. Dacă apeși tasta **săgeată sus**, creionul este ridicat și nu mai desenează. Apasă tasta **c** pentru a șterge traseele desenate de creion pe scena.

Există o singură diferență între cele două implementări, legată de modul de calcul al variabilei **last error**. Dacă testezi cele două programe cu trasee diferite, vei vedea care implementare corespunde mai bine anumitor trasee.

Ambele proiecte conțin un script care te ajută să verifici senzorii. Robotul se va mișca de la stânga la dreapta și înapoi. Dacă există o linie neagră în poziție mediană, poți verifica dacă informațiile provenite de la senzori sunt corecte. De asemenea, poți verifica valoarea variabilei **actual error**. Scriptul este executat la apăsarea tastei **s.**

Remixează cele două soluții și încearcă să le îmbunătățești.

14 Robot care soluționează labirintul din linii

Obiectiv: să programezi robotul Karel să rezolve un labirint din linii.

14.1 Introducere în labirint

Urmărește filmul de la adresa https://youtu.be/_Ei2oibdSTc pentru a observa cum găsește Karel ieșirea din labirint.

Un robot care soluționează un labirint din linii se deplasează de la start la sosire, urmărind o linie, în cel mai scurt timp posibil (Figura 45). Un labirint din linii este, de obicei, desenat cu linii negre pe o suprafață albă. Prima dată, robotul explorează labirintul, astfel încât să găsească un drum între start și sosire. A doua oară îl parcurge pe cel mai scurt drum.

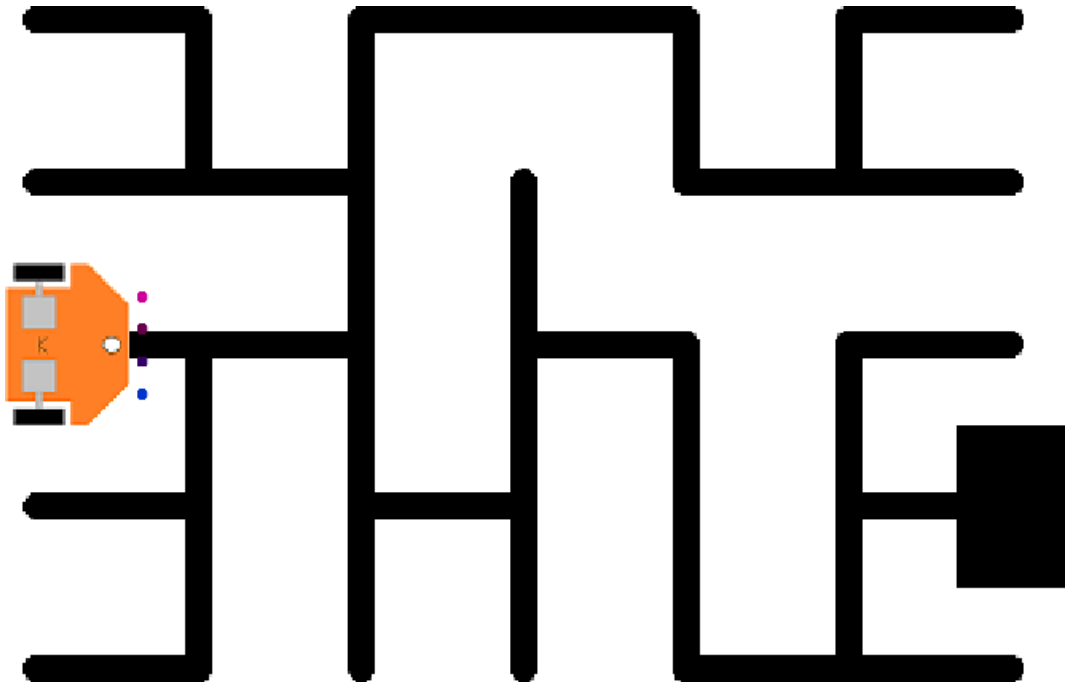


Figura 45 Labirint cu linii

Concursurile de rezolvare a labirinturilor sunt foarte populare. Mai multe despre un astfel de concurs poți afla din regulamentul [4].

14.1.1 Senzori pentru labirint cu linii

Un robot pentru parcurgerea unui labirint folosește senzori optici amplasați în fața robotului pentru a detecta linia. Este posibil să folosești doar patru senzori: doi urmăresc linia, iar ceilalți doi ajută la identificarea intersecțiilor (inclusiv fundăturile și sosirea).

14.1.2 Controlarea robotului care soluționează labirinturi

Un robot poate găsi ieșirea dintr-un labirint urmărind peretele. Vom folosi regula mâinii stângi, ceea ce înseamnă instrucțiunile pe care robotul le urmează sunt:

- Rotește-te spre stânga în loc să înaintezi sau să te rotești la dreapta.

- Înaintează în loc să te rotești la dreapta.

14.1.3 Optimizarea traseului robotului

De obicei, un labirint conține foarte multe fundături (linia se termină, iar robotul trebuie să se rotească la 180 de grade).

La prima rulare, robotul ajunge în multe fundături, ceea ce duce la creșterea timpului de parcurgere a labirintului. Iată de ce roboții inteligenți reține, cu ocazia primei rulări, tipurile de intersecții vizitate (inclusiv fundăturile).

Înainte de a efectua a doua cursă, robotul optimizează traseul, eliminând toate fundăturile. Cea de-a doua rulare este mai rapidă decât prima pentru că traiectoria nu mai conține fundături.

14.2 Rezolvarea labirintului - provocare

Iată o probă cu adevărat dificilă! Va trebui să-l programezi pe Karel să parcurgă un labirint din linii.

Remixează proiectul de la adresa http://scratch.mit.edu/projects/41881278/#editor.

După cum poți observa, Karel are patru senzori optici în față. Decorurile diferite îți permit să testezi comportamentul robotului în diferite cazuri particulare.

1. Programează-l pe Karel să parcurgă labirintul de la start la sosire.

2. Programează-l pe Karel să parcurgă labirintul pe calea cea mai scurtă.

Mult succes!

14.3 Soluție pentru rezolvarea labirintului

O soluție posibilă se găsește la http://scratch.mit.edu/projects/41812402/#editor.

Clichează pe steagul verde pentru a rula programul.

Apasă tasta **s** pentru a parcurge labirintul prima dată (**s** vine de la slow = încet).

Apasă tasta **q** pentru a parcurge labirintul a doua oară, după terminarea cântecului (**q** vine de la quick = rapid).

Poți remixa și optimiza soluția prezentată.

14.4 Soluție îmbunătățită pentru rezolvarea labirintului

Soluția îmbunătățită prezentată la https://scratch.mit.edu/projects/45753042/#editor se bazează pe cea anterioară, însă conține noi caracteristici. În această versiune, utilizatorul poate alege punctul de start (o fundătură a labirintului) și poate stabili vitezele robotului în cele două încercări.

Pentru a folosi programul, utilizatorul trebui să facă următoarele:

1. Clichează pe steagul verde pentru a rula programul.

2. Alege punctul de pornire clicând pe una dintre fundăturile labirintului.

3. Selectează viteza cursei lente (prima).

4. Selectează viteza cursei rapide (a doua).

5. Apasă tasta **s** pentru a rula prima cursă.

6. Apasă tasta **q** pentru a rula a doua cursă.

Poți testa comportamentul robotului cu puncte de pornire și viteze diferite. Ce se întâmplă dacă viteza este prea mare? Ce se întâmplă dacă punctul de pornire ales nu este o fundătură?

Remixează și optimizează soluția propusă.

Bibliografie

[1] M. Agape, Scratch - Curs pentru începători, Orșova, 2011.

[2] M. Agape, „Robo Scratch,” Scientix, 15 November 2015. [Interactiv]. Available: http://moodle.scientix.eu/course/view.php?id=593. [Accesat 15 March 2016].

[3] M. Agape, „Regulamentul concursurilor de urmărire a liniei,” 15 November 2015. [Interactiv]. Available: http://1drv.ms/1KJ2qh6. [Accesat 15 February 2016].

[4] M. Agape, „Regulamentul concursurilor de ieșire din labirint,” 15 November 2015. [Interactiv]. Available: http://1drv.ms/1XLGLaa. [Accesat 15 February 2016].

Listă de figuri

Figura 1 Interfața Scratch 2.0 7
Figura 2 Dimensiunile scenei 8
Figura 3 Mișcare și sunet 9
Figura 4 Blocul **înaintează () pași** 9
Figura 5 Adăugarea unui sunet 10
Figura 6 E timpul pentru dans 11
Figura 7 Evenimente și control 12
Figura 8 Buclă infinită 12
Figura 9 Instrucțiunea **când se clichează pe steagul verde** 13
Figura 10 Steagul verde și iconița roșie stop se află deasupra scenei în colțul din dreapta 14
Figura 11 Să ne jucăm cu culorile 15
Figura 12 Modificarea culorii 15
Figura 13 Crearea efectelor de imagine 16
Figura 14 Controlarea acțiunilor din taste 16
Figura 15 Personaje 18
Figura 16 Vorbire 20
Figura 17 Sunete, voci, muzică 22
Figura 18 Crearea animațiilor 24
Figura 19 Karel și trei ținte 30
Figura 20 Trei ținte - scripturi pentru provocarea 1 30
Figura 21 Trei ținte – provocarea 2 – scripturi soluție greșită 31
Figura 22 Trei ținte – provocarea 2 – scripturi soluție bună 31
Figura 23 Trei ținte – provocarea 3 – soluție scripturi 32
Figura 24 Evidențierea drumului parcurs și schimbarea culorii țintelor 33
Figura 25 Trei ținte – provocarea 4 – modificare scripturi 33
Figura 26 Soluția pentru definiția blocului **rectangle (x, y, width, height)** 34
Figura 27 Dreptunghiuri cu nuanțe diferite 35
Figura 28 Definiția blocului **polygon(n, l, x, y)** 36
Figura 29 Poligoane cu număr de laturi diferite 36
Figura 30 Scripturi pentru desenarea unui cerc și a unei ținte 37
Figura 31 Șasiul robotului Karel 38
Figura 32 Desenul robotului Karel 38
Figura 33 Definiția blocului **draw chassis** 39
Figura 34 Robot care evită obstacolele 40
Figura 35 Blocul **<culoarea [] atinge []?>** 40
Figura 36 Blocul care nu lasă personajul să părăsească scena 41
Figura 37 Două soluții posibile pentru definirea blocului **read bumpers** 42
Figura 38 Decor pentru testarea evitării obstacolelor 43
Figura 39 Decor care îi creează probleme lui Karel 43
Figura 40 Doi roboți pe scenă 44
Figura 41 Senzor de distanță cu valori în intervalul 1 – 6 46

Figura 42 Robot urmăritor de linie 50
Figura 43 Urmăritor de linie cu doi senzori 51
Figura 44 Formula pentru calculul erorii 53
Figura 45 Labirint cu linii 55

Anexa A - Regulamentul Concursului Internațional de Programare în Scratch SCRIPT

Concursul se adresează elevilor din ciclul primar și gimnazial și constă în realizarea, de către echipe formate din 2 – 4 elevi, a unui proiect în limbajul de programare Scratch.

Scratch este un limbaj de programare dezvoltat de grupul Lifelong Kindergarten din cadrul MIT Media Lab. Cu Scratch, copiii pot învăța de la vârste mici (6 ani), într-o manieră ludică, principalele concepte ale programării. Programul Scratch poate fi descărcat gratuit de la adresa http://scratch.mit.edu.

Prezentul regulament va fi tradus în limba engleză pentru participanții din afara României.

Regulamentul a fost elaborat de Mihai Agape, inițiatorul concursului SCRIPT.

Scop și obiective

Scopul concursului SCRIPT 2016 este promovarea programării în rândul elevilor din ciclul gimnazial și primar prin folosirea limbajului de programare Scratch.

Obiectivele specifice ale concursului sunt:

O1. Stimularea preocupărilor elevilor în domeniul programării.

O2. Dezvoltarea competențelor de exprimare în limba engleză a elevilor și cadrelor didactice.

O3. Promovarea elevilor valoroși din rândul elevilor participanți.

O4. Popularizarea celor mai interesante proiecte.

Secțiuni și categorii de vârstă

Concursul se desfășoară pe șase secțiuni:

- ✓ ***Felicitări***
- ✓ ***Jocuri***
- ✓ ***Muzică și dans***
- ✓ ***Povești***
- ✓ ***Simulări***
- ✓ ***Software educațional***

Fiecare dintre cele șase secțiuni se desfășoară pe nouă categorii de vârstă, câte o categorie pentru fiecare clasă de la clasa pregătitoare până la clasa a VIII-a.

Cine poate participa

La concurs pot participa elevi din ***cluburi***, ***palate ale copiilor*** și ***școli ,*** organizați în echipe formate din 2, 3 sau 4 membri. Elevii unei echipe trebuie să fie din aceeași unitate de învățământ și să se încadreze în categoria de vârstă corespunzătoare (nu se admite participarea la o categorie superioară). Fiecare echipă trebuie să aibă un coordonator care este profesor în aceeași unitate de învățământ cu elevii.

Un elev poate face parte din mai multe echipe, cu condiția ca echipele să fie înscrise la secțiuni diferite. Prin urmare, un elev nu poate face parte din două echipe care sunt înscrise la aceeași secțiune.

Desfășurarea concursului

Înscrierea în competiție

O unitate de învățământ poate participa la concurs doar dacă este înregistrată în competiție de către un cadru didactic al respectivei instituții, folosind formularul de înregistrare on line pus la dispoziție de organizatori. În formular se solicită atât datele de contact cât și informații referitoare la numărul de echipe îndrumate la fiecare secțiune a concursului.

Numărul de unități care se pot înscrie dintr-o țară, alta decât România, este de trei. Înscrierea unității se face înainte de termenul limită precizat în secțiunea 0.

Realizarea proiectelor și îndrumarea echipelor

Profesorul coordonator îndrumă elevii în dezvoltarea proiectului, dar acesta trebuie să fie rodul muncii elevilor din echipă. Proiectele trebuie să fie originale și trebuie să fie rezultatul muncii echipei care trimite proiectul.

Proiectele vor fi realizate în limba engleză. Proiectele care folosesc o altă limbă vor fi eliminate din concurs.

Autorii proiectului trebuie să dețină drepturile de copyright pentru muzica, sunetele, textul și imaginile folosite în proiect. Proiectele care încalcă aceste reguli vor fi descalificate.

Proiectele nu trebuie să conțină informații referitoare la autori, instituția de proveniență, coordonator sau țară. Proiectele care conțin astfel de informații vor fi eliminate din concurs.

Selectarea proiectelor pentru faza finală

O unitate de învățământ poate transmite la faza finală 1 proiect la fiecare secțiune și categorie de vârstă (i.e. un maximum posibil de 6 x 9 = 54 proiecte). Dacă la nivelul unei instituții există mai multe proiecte la aceeași secțiune și la aceeași categorie de vârstă, cadrul didactic coordonator sau cadrele didactice coordonatoare vor organiza o competiție la nivelul instituției de învățământ în urma căreia vor selecta cel mai bun proiect pentru fiecare secțiune și categorie de vârstă.

Transmiterea proiectelor

Proiectele calificate pentru faza finală vor fi transmise la adresa de email a concursului scriptrial@gmail.com înainte de termenul limită specificat la secțiunea 0.

Proiectele transmise vor fi încadrate (de către creatorii lor) în una din cele șase secțiuni ale concursului.

Organizatorii concursului își rezervă dreptul de a descalifica orice proiect al cărui conținut este considerat inadecvat sau care nu respectă regulile concursului.

Evaluarea proiectelor

Evaluarea proiectelor se va face de către un juriu format din cadre didactice. Ierarhizarea lucrărilor se va face în funcție de nota acordată de juriu.

Criterii de evaluare a lucrărilor:

- ✓ Originalitate
- ✓ Creativitate
- ✓ Folosirea caracteristicilor Scratch
- ✓ Calitatea globală a proiectului

Fiecare proiect va primi o notă de la 1 la 10. Pentru fiecare secțiune și categorie de vârstă se va întocmi un clasament al proiectelor în ordinea descrescătoare a notelor.

Premiile I, II, III și mențiunile se vor acorda proiectelor pe baza următoarelor 3 reguli:

- Primul proiect din clasamentul pentru o secțiune și categorie de vârstă poate primi premiul I, cel de al doilea premiul II, cel de al treilea premiul III și cel de al patrulea mențiune.
- Nota lucrării trebuie să fie între 9 și 10 pentru premiul I, mai mare sau egală cu 8 pentru premiul II, mai mare sau egală cu 7 pentru premiul III și mai mare sau egală cu 6 pentru mențiune.
- Numărul total de diplome (premiul I, II, II și mențiuni) acordat la o anumită categorie de vârstă și secțiune nu trebuie să depășească cuantumul de 25 % din numărul total al proiectelor primite la respectiva categorie de vârstă și secțiune. Excepție se poate face în cazul în care pentru o anumită secțiune și categorie de vârstă numărul de lucrări înscrise în concurs este mai mic de 16.

Popularizarea proiectelor

Pentru orice proiect înscris în concurs, autorii proiectului acordă organizatorilor concursului permisiunea de a folosi fără restricții proiectul în scop educațional sau publicitar. Organizatorii sunt obligați să facă cunoscuți autorii proiectului precum și unitatea de învățământ de la care provin aceștia.

Organizatorii vor face publice proiectele premiate prin postarea pe site-ul web http://scratch.mit.edu/.

Etape

Principalele momente cronologice în desfășurarea concursului sunt:

- 01 februarie – Lansarea proiectului.
- 25 februarie – Termenul limită pentru înscrierea unităților.
- 11 mai – Termenul limită pentru trimiterea proiectelor.
- 01 iunie – Anunțarea rezultatelor.
- 14 iunie – Transmiterea diplomelor prin poștă.
- 30 iunie – Realizarea unei galerii cu proiectele câștigătoare.

Nu se va organiza o fază finală cu participare directă. Participarea la concurs se face prin transmiterea electronică a proiectelor.

Dispoziții finale

Decizia juriului este finală și nu poate fi contestată.

Regulamentul poate fi îmbunătățit pe baza propunerilor participanților la competiție. Propunerile de modificare pot fi trimise la adresa scriptrial@gmail.com.

Modificările la prezentul regulament se vor transmite în timp util tuturor celor interesați.