

*Dženan Gušić • Edina Kadrić Durmiš*

# VIZUELNO

*Programiranje*

*kroz Eticques*



**UNIVERZITETSKI UDŽBENIK**

UNIVERZITET U SARAJEVU  
PRIRODNO-MATEMATIČKI FAKULTET



Dženan Gušić  
Edina Kadrić Durmiš

## VIZUELNO PROGRAMIRANJE (kroz Etics)

Univerzitetsko izdanje  
Sarajevo, 2025



**Autori:**  
Dženan Gušić  
Edina Kadrić Durmiš

**Naziv udžbenika:**  
*Vizuelno programiranje (kroz Eticsques)*

**Prvo izdanje**

**Izdavač:**  
Univerzitet u Sarajevu  
Prirodno-matematički fakultet

**Recenzenti:**  
Prof. dr. Elmedin Selmanović  
Prof. dr. Nevzudin Buzadija

**Naslovna stranica:**  
Džemal Poparić

**Lektor:**  
Biljana Stojanović

**Korekcija:** Autori

**Tehnička obrada:** Autori

**Godina izdanja:** 2025

-----  
CIP – Katalogizacija u publikaciji  
Nacionalna i univerzitetska biblioteka Bosne i Hercegovine, Sarajevo  
Autori: Dženan Gušić, Edina Kadrić Durmiš  
Naslov: Vizuelno programiranje (kroz Eticsques)  
Vrsta građe: udžbenik  
Jezik: bosanski  
Godina: 2025  
Izdavanje i proizvodnja: Sarajevo, Prirodno-matematički fakultet UNSA  
Karakteristike elektronskog izvora: El. knjiga  
ISBN: 978-9926-453-99-2  
Bibliografija: str. [343]-344.  
Predmetne odrednice: Programski jezik "Eticsques" -- Vizuelno programiranje -- Udžbenici za fakultete  
UDK: 004.432(075.8)  
Link: <https://ebooks.unsa.ba/index.php/ebooks/catalog/book/88>  
COBISS.BH-ID: 67417606  
-----

Odlukom Senata Univerziteta u Sarajevu broj: 01-14-165/25 od 26.11.2025. godine ova publikacija je dobila univerzitetsku saglasnost.

Zabranjeno je svako parcijalno ili integralno umnožavanje ove knjige bez odobrenja izdavača.

## PREDGOVOR

Spektar programskih okruženja u modernoj strukturi razvoja softvera i obrazovanja uopšte je ogroman. Na jednoj strani su visoko apstraktni, vizuelni alati dizajnirani za usvajanje računarskog načina razmišljanja putem grafičkog interfejsa, a na drugoj tekstualni, kompajlerski jezici koji nude jednostavan pristup hardveru i omogućavaju preciznu kontrolu nad izvršavanjem programa. Klasični predstavnici pomenutih klasa su Etics, kao vizuelna programska platforma zasnovana na generalizovanom Scratch 3 frameworku, te C++, kao jedan od najmoćnijih i široko korištenih kompajlerskih jezika u modernom računarstvu. Etics zauzima jedinstveno mjesto u ovom ekosistemu. Iako nasljeđuje vizuelnu, blokovsku strukturu okruženja kojeg ima u svojoj osnovi, ističe se i vlastitim naprednim mogućnostima: podrškom za JavaScript proširenja, integracijom sa NW.js za samostalno generisanje izvršnih verzija projekata, te djelimičnom mogućnošću interakcije sa naredbama operativnog sistema. To nije samo alat za edukaciju. To je platforma za izradu prototipova i izvođenje eksperimenata koja nudi dostupnost vizuelnog dizajna uz kontrolisane uvide i proboje u ponašanje softvera u stvarnome svijetu. S druge strane, C++, jezik zasnovan na performansama, preciznosti i direktnoj interakciji sa sistemskim resursima, podržava objektno orjentisane, proceduralne i generičke paradigme programiranja. Zahtijeva pedantnu pažnju posvećenu sintaksi, upravljanju memorijom, arhitektonskom dizajnu – ukratko kvalitetima koji istovremeno i osnažuju ali i izazivaju programera. Etics koristi grafički blok interfejs, čime se u potpunosti eliminišu sintaksičke greške. Vizuelni model omogućava programerima da konstruišu logiku povezivanjem modularnih blokova koji predstavljaju događaje, uslove, petlje i funkcije. Sve navedeno znatno smanjuje kognitivno opterećenje u smislu učenja sintakse, te omogućava korisnicima direktan fokus na strukturu i tok logike. Poređenja radi, C++ se oslanja na strogu tekstualnu sintaksu, što zahtijeva savladavanje jezičkih pravila, tipova podataka, interpukcije. Greške u postavljanju tačke – zareza, deklaraciji varijabli ili rukovanju memorijom često polučuju greškama kompajlera ili padovima sistema tokom izvršavanja programa. Jasno, strogost nameće kako disciplinu tako i preciznost – ponovo kvalitete neophodne u oblastima od kritične važnosti za performanse. Etics projekti se obično interpretiraju ili pakuju putem NW.js u samostalne aplikacije. Iako ovo omogućava brz razvoj i testiranje, rezultira manjom brzinom izvršavanja i većom veličinom aplikacije. Takođe, slojevi apstrakcije i oslanjanje na JavaScript inherentno ograničavaju performanse. Etics nudi visok nivo apstrakcije. Koncepti poput upravljanja memorijom i sistema fajlova su skriveni ili potpuno apstrahovani. Ovo osigurava pristupačnost i omogućava korisnicima da izgrade potpuno funkcionalne aplikacije bez potrebe za razumijevanjem osnovne arhitekture. Napredne funkcije poput prilagođenih JavaScript ekstenzija omogućavaju neke dublje interakcije, ali unutar strogo kontrolisanih granica. Etics je najprikladniji za: projektovanje vizuelnih igara i edukativnih alata, kreiranje interaktivnih priča i simulacija, podučavanje algoritamskog mišljenja i logike dizajna softvera, te izradu više – platformskih samostalnih exe aplikacija (putem NW.js). Etics projekti se mogu proširiti prilagođenim blokovima zasnovanim na JavaScript – u, a moguć je i određeni pristup Node.js API – jima. Ipak, iako ovo poboljšava funkcionalnost izvan tipičnih vizuelnih uređivača, ostaje ograničenost opsega u poređenju sa tradicionalnim programskim okruženjima. Prelazak npr. sa Etics – a na C++ ne bi bio samo prelazak sa



blokova na zgrade. Više bi to bila riječ o pitanju koncepta i filozofiji kojom se pristupa rješavanju konkretnih problema, performansama, preciznosti, itd. Ukratko, Eticques naglašava pristupačnost, modularno razmišljanje, i neposredne vizuelne povratne informacije. Smanjuju se barijere na ulazne podatke, potiče se eksperimentisanje, te promoviše razumijevanje kroz apstrakciju. Jednostavno, služi kao gnijezdo logike, kreativnosti i softverske arhitekture u nekim njenim ranim fazama. Odabrane jedinice koje u udžbeniku koristimo za prolaz kroz upravo opisanu tematiku su:

- Okruženje Scratch 3
- Događaji, uslovi, varijable
- Od Scratch-a do Eticques-a
- Petlje, skripte
- Liste objekata
- Vizuelno objektno orijentisano programiranje
- Osvjetljenje i zvuk
- Naprednije modifikacije objekata
- Povratne vrijednosti (u realnom vremenu)
- Izrada složenijih struktura
- NPC algoritmi
- Pametni anti algoritmi
- Ažuriranje sadržaja
- Suplementi
- TurboWarp Packager i exe fajlovi

Svaka od navedenih sekcija popraćena je odgovarajućom sekcijom „Riješeni zadaci“, koja sadrži određeni broj riješenih zadataka pogodnih za bolje i brže razumijevanje tematike, logike, koncepta i sintakse okruženja, te sekcijom „Zadaci za vježbu“ predviđenom za samostalan rad studenata. Inače, knjiga je koncipirana tako da prati nastavni plan i program (2025. – ) predmeta prve godine studija po kojem nosi ime, smjera „Kompjuterske nauke“ Odsjeka za matematičke i kompjuterske nauke PMF – a UNSA (gdje prvi autor izvodi nastavu), ali i nastavni plan i program istoimenog predmeta prve godine studija studijskog programa „Matematika i informatika“ Filozofskog fakulteta UNZE (oba autora izvode nastavu).

Autori se zahvaljuju svojim najbližim na svakoj vrsti podrške tokom izrade knjige. Hvala dragim kolegama recenzentima, prof. dr. Elmedinu Selmanoviću, profesoru Odsjeka za matematičke i kompjuterske nauke PMF – a UNSA, i prof. dr. Nevzudinu Buzadžiji, profesoru Politehničkog fakulteta UNZE, na nesebičnom i savjesnom pregledu rukopisa. Zahvalnost dugujemo i kolegici Biljani Stojanović, dipl. bibliotekarki i komparativistici, te kolegi Džemalu Popariću, studentu prve godine studija matematike PMF – a u Sarajevu, na izradi naslovne stranice. Svi oni koji su na bilo koji način pomogli pri izradi knjige, takođe imaju našu zahvalnost.

Sarajevo, 2025

Autori

KNJIGA JE POSVEĆENA STUDENTIMA





## SADRŽAJ

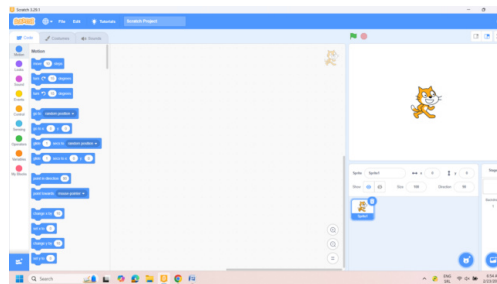
|                |                                                       |
|----------------|-------------------------------------------------------|
| PREDGOVOR..... | iii                                                   |
| 1              | Okruženje Scratch 3 ..... 1                           |
| 1.1            | Riješeni zadaci ..... 17                              |
| 1.2            | Zadaci za vježbu ..... 21                             |
| 2              | Događaji, uslovi, varijable..... 23                   |
| 2.1            | Riješeni zadaci ..... 38                              |
| 2.2            | Zadaci za vježbu ..... 40                             |
| 3              | Od Scratch-a do Eticques-a ..... 43                   |
| 3.1            | Riješeni zadaci ..... 60                              |
| 3.2            | Zadaci za vježbu ..... 62                             |
| 4              | Petlje, skripte ..... 65                              |
| 4.1            | Riješeni zadaci ..... 76                              |
| 4.2            | Zadaci za vježbu ..... 81                             |
| 5              | Liste objekata..... 83                                |
| 5.1            | Riješeni zadaci ..... 97                              |
| 5.2            | Zadaci za vježbu ..... 101                            |
| 6              | Vizuelno objektno orjentisano programiranje ..... 103 |
| 6.1            | Riješeni zadaci ..... 116                             |
| 6.2            | Zadaci za vježbu ..... 121                            |
| 7              | Osvjetljenje i zvuk ..... 123                         |
| 7.1            | Riješeni zadaci ..... 134                             |
| 7.2            | Zadaci za vježbu ..... 138                            |
| 8              | Naprednije modifikacije objekata ..... 139            |
| 8.1            | Riješeni zadaci ..... 155                             |
| 8.2            | Zadaci za vježbu ..... 159                            |



|                     |                                                |     |
|---------------------|------------------------------------------------|-----|
| 9                   | Povratne vrijednosti (u realnom vremenu) ..... | 161 |
| 9.1                 | Riješeni zadaci .....                          | 182 |
| 9.2                 | Zadaci za vježbu .....                         | 184 |
| 10                  | Izrada složenijih struktura .....              | 187 |
| 10.1                | Riješeni zadaci .....                          | 206 |
| 10.2                | Zadaci za vježbu .....                         | 208 |
| 11                  | NPC algoritmi .....                            | 209 |
| 11.1                | Riješeni zadaci .....                          | 223 |
| 12                  | Pametni anti algoritmi .....                   | 229 |
| 12.1                | Riješeni zadaci .....                          | 245 |
| 12.2                | Zadaci za vježbu .....                         | 249 |
| 13                  | Ažuriranje sadržaja .....                      | 251 |
| 13.1                | Riješeni zadaci .....                          | 267 |
| 13.2                | Zadaci za vježbu .....                         | 274 |
| 14                  | Suplementi .....                               | 275 |
| 14.1                | Riješeni zadaci .....                          | 296 |
| 14.2                | Zadaci za vježbu .....                         | 303 |
| 15                  | TurboWarp Packager i exe fajlovi .....         | 305 |
| 15.1                | Riješeni zadaci .....                          | 335 |
| 15.2                | Zadaci za vježbu .....                         | 341 |
| BIBLIOGRAFIJA ..... |                                                | 343 |

# 1 Okruženje Scratch 3

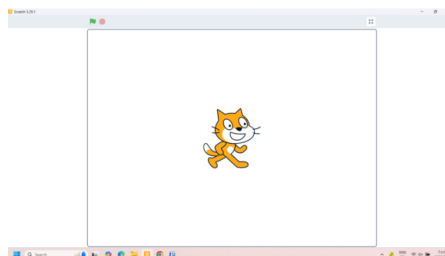
Pokrenemo Scratch 3 (verzija 3.29.1). Dobijamo prozor



Prozor desno, gdje stoji mačak je naš ekran, tj. to je ekran koji vidimo kada pokrenemo igricu (odnosno program). Dimenzija tog ekrana je  $480 \times 360$ , dakle nije nešto prevelika, čak možemo reći da je jako mala (ta rezolucija je mala za današnji pojam rezolucija ekrana, televizora, fotografija, itd). Ekran gdje stoji mačak je ekran u malom. Ako odemo na ikonu



koja se nalazi gore desno, i kliknemo je, dobit ćemo ekran u punom formatu, tj. dobit ćemo prikaz

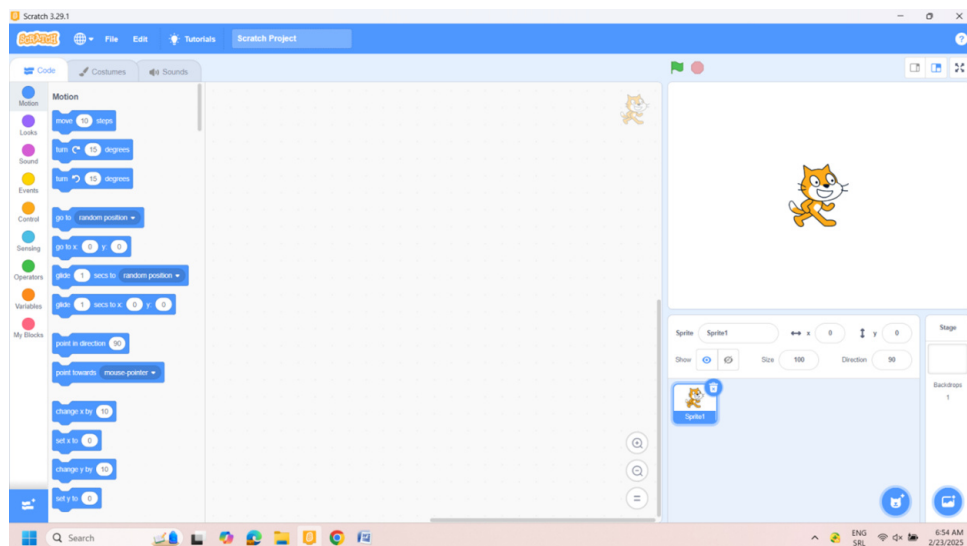


Na ovoj slici treba uočiti zelenu zastavicu i crveni krug (gore lijevo).

Zelena zastavica služi za pokretanje programa. Crveni krug služi za zaustavljanje programa. Ako trenutno kliknemo zelenu zastavicu, neće se dogoditi bilo šta, jer nismo unijeli nikakav kod, nismo ništa isprogramirali. Isto važi i za crveni krug. Kliknemo ponovo opciju



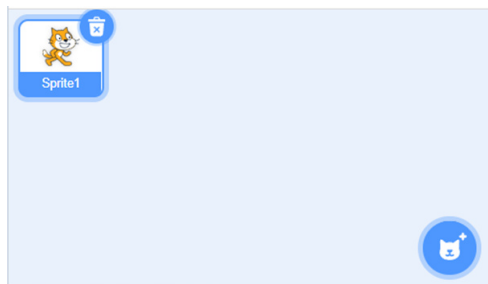
i vratimo se na početni prozor, tj. na prethodnu sliku



Mačak se inače zove Sprite. Nije da mu je to ime, nego je on određena ikona koja u našem programu treba da obavlja određeni zadatak, da trči, pravi krugove, ispisuje neki tekst, ukratko da radi ono što mu mi naredimo, tj. na način kako ga iskodiramo. Takve ikone se nazivaju Sprite-ovi.

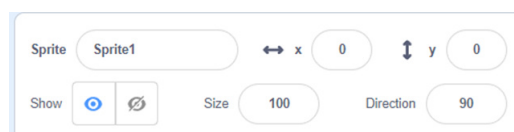
Dakle, mačak je jedan primjer Sprite-a.

Primijetiti sada dio desno, gdje imamo ikonu



To je prozor Sprite-ova. Dakle trenutno, naš jedini Sprite je mačak. Ispod njegove ikone piše Sprite1, što znači da se taj Sprite zaista tako i zove, tj. zove se Sprite1.

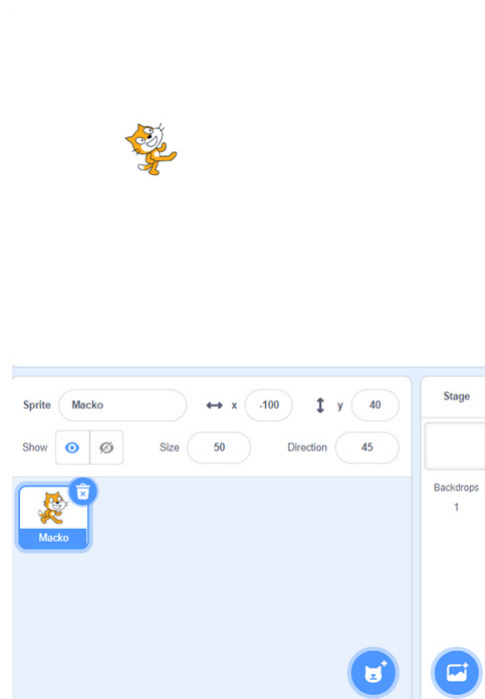
U vrhu ikone stoji korpa za smeće što znači da klikom na tu korpu uklanjamo, tj. brišemo Sprite o kojem je riječ. Iznad prozora Sprite-ova kojeg smo upravo pokazali, nalazi se prozor



U dijelu "Sprite" piše Sprite1, što smo rekli da je ime aktuelnog Sprite-a, tj. našeg mačka. To ime možemo promijeniti u tom otvoru. Dovoljno je da kliknemo iza riječi Sprite1, pobrišemo je, i upišemo ime koje nama treba. Zatim, tu su prozori za  $x$  i  $y$ , što su ustvari koordinate gdje se trenutno nalazi naš Sprite (mačak). Trenutne koordinate su  $x = 0$  i  $y = 0$ , što znači da se mačak nalazi na sredini našeg ekrana. Zatim, imamo opciju "size" koja je po defaultu uvijek postavljena da bude jednaka 100, i nju možemo mijenjati, dakle možemo povećati ili smanjiti veličinu u kojoj će se naš Sprite pojavljivati na našem ekranu. Obično se Sprite-ovi pojavljuju u dosta velikom obliku pa je njihovu veličinu često potrebno smanjiti. Napokon, imamo opciju "Direction", gdje trenutno piše broj 90, koja nam govori o kojem smjeru je riječ, tj. u kojem smjeru je okrenut naš sprite. Ovdje je riječ o uglu 90 stepeni, što znači da je Sprite tj. mačak okrenut u pravcu broja 3 na satu. Inače, uglovi se mjere odozgo prema desno. Naime, 0 stepeni je smjer prema broju 12 na satu, 90 stepeni je smjer prema broju 3 na satu, 180 stepeni je smer prema broju 6 na satu.

Idemo promijeniti neke od opcija na posmatranom ekranu. Upišimo brojeve kao

na slici



Dakle, našem Sprite-u smo dali ime Macko, postavili smo ga da stoji na koordinatama  $x = -100$ ,  $y = 40$ , smanjili smo mu veličinu na 50, i usmjerili smo ga u pravcu 45 stepeni.

Vidimo na gornjem dijelu, tj. na prozoru koji označava naš ekran, kakve promjene je to izazvalo.

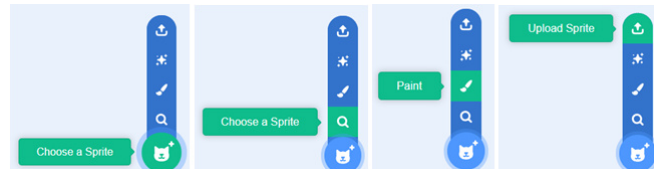
Mačak je pomjeren ulijevo ( $x = -100$ ) i prema gore ( $y = 40$ ), smanjen je i usmjeren je u pravcu 45 stepeni.

U dnu posljednje slike imamo ikonu



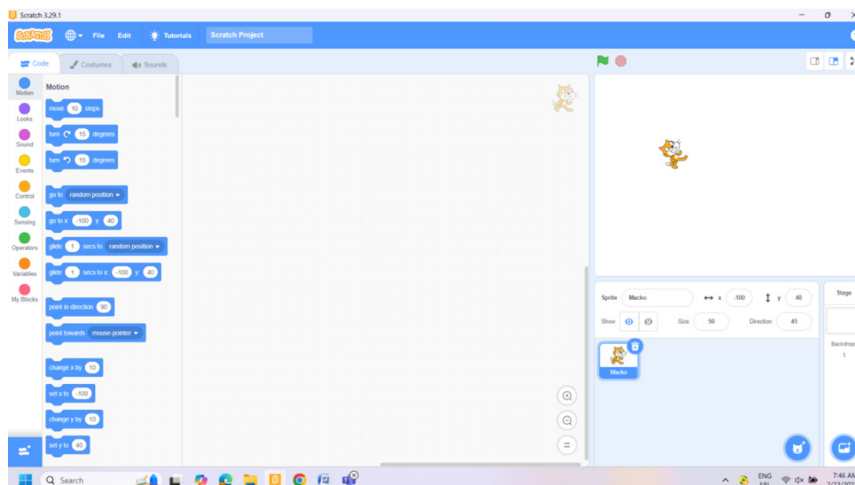
koja označava mjesto gdje bismo nove Sprite-ove, dodajemo ih, koristimo gotove pa

ih uključujemo, crtamo ih, tj. pravimo ih sami, ili ih upload-ujemo. Sve zavisi od naših potreba. Znači, kada kliknemo na pomenutu sličicu, dobijemo izbornik



Ovdje vidimo upravo pomenute opcije: Choose a Sprite (biramo neki gotov Sprite), Paint (sami sebi nacrtamo neki Sprite), Upload Sprite (upload-ujemo neki Sprite).

Primijetimo da dok se nalazimo na centralnoj slici



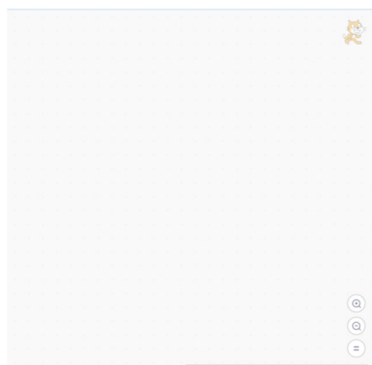
tj. dok nismo kliknuli na ikonu za puni ekran (Full Screen Control)



zelena zastavica i crveni krug su i dalje prisutni. Dakle, na njih možemo kliknuti i

iz ove pozicije, tj. ne moramo se nalaziti u punom ekranu igrice da bi pokrenuli i zaustavili igricu. To možemo znači uraditi i kada se ekran igrice nalazi u malom (kao na velikoj prethodnoj slici).

Ako nastavimo razmatranje sa aspekta pomenute slike, uočavamo dio



tj. tzv. srednji prozor. To je prozor za kod. Uočiti djelimično providnu sliku mačka u gornjem uglu ovog ekrana. To znači da se kod koji se unosi u ovaj prozor (to je prazna bijela površina na ovoj slici), odnosi na mačka. Uvijek se kod koji unosimo u ovaj prozor odnosi na Sprite koji se nalazi u gornjem desnom uglu prozora. U nas je trenutno aktivan Sprite mačak (Macko), jer smo pozicionirani na njemu, tj. jer je aktivna sličica



U dnu ekrana kojeg razmatramo, tj. srednjeg ekrana, imamo ikone

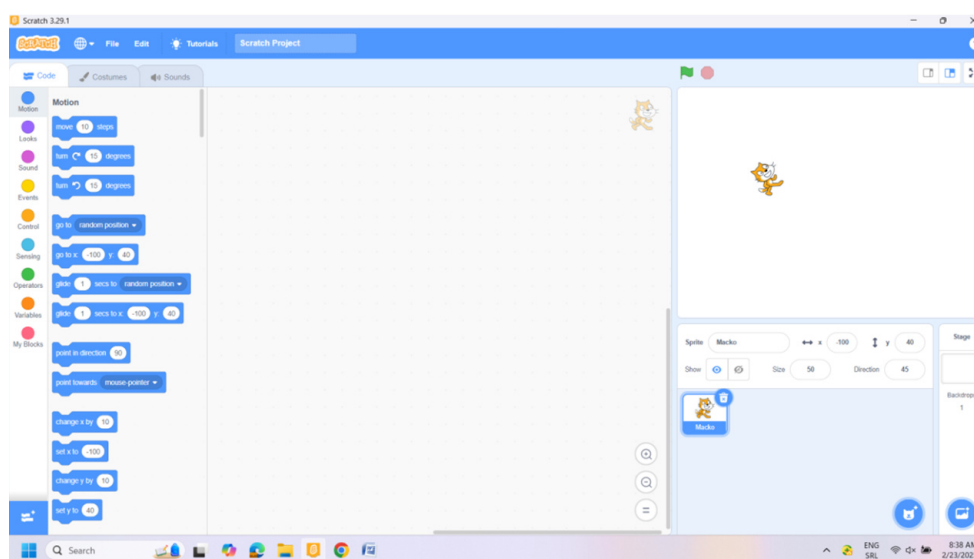


koje nam služe za povećavanje i smanjivanje veličine fonta koda (moći ćemo ih koris-



titi kada neki kod bude aktivan, tj. dostupan). Napomenimo da srednji ekran služi i za druge svrhe (u nastavku).

Na lijevoj strani okruženja



imamo kratice: Code, Costumes, Sounds.

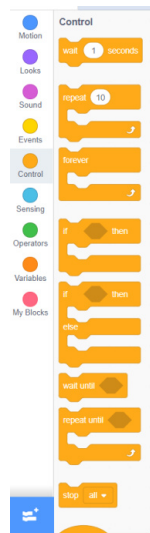
Po defaultu smo pozicionirani na kraticu Code, što znači da je centralni ekran podešen za unos koda Sprite-a koji je trenutno aktivan (to je kod nas Sprite Macko).

Osim toga, dok je aktivna kratica Code, na lijevoj strani ekrana imamo dostupne naredbe koje su razvrstane unutar kratica: Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, My Blocks. Navedene kratice su prikazane različitim bojama, i sadrže naredbe pomoću kojih pravimo (generišemo) naš kod, i koje unosimo na srednji prozor (prozor koda).

Tako, kratica Motion sadrži razne naredbe za kretanje: idi na tu i tu poziciju, idi na proizvoljnu poziciju, klizi do te i te tačke za toliko i toliko sekundi, itd. Dakle, riječ je o naredbama kretanja.

Za pristup naredbama koje manipulišu dostupnim zvukovima koristimo kraticu Sounds.

Nije teško pristupiti bilo kojem od izbornika i pogledati sadržaj naredbi koje on sadrži. Bitnije od izgleda emotikona su sigurno naredbe za razne događaje, tj. kratica Control. Klikom na nju dobijamo spisak naredbi sadržanih pod ovom kraticom



Tu su sadržane naredbe koje sadrži svaki programski jezik (C++, Basic, itd), i koje mi iz navedenih jezika poznajemo kao naredbe tipa If-Then-Else, For petlje, While i Do While petlje, itd. U ovom jeziku, tj. Scratch-u 3 izgled pomenutih naredbi može na prvi pogled izgledati malo neobičan i čudan. Ovdje ćemo bez problema imati beskonačnu for petlju (koja ovdje glasi "forever"), i slične naredbe, čija sintaksa i način pisanja odudaraju od standardnih načina pisanja pomenutih naredbi. Naime, ovdje nema pisanja koda ručno, tipa da napišemo naredbu If (nešto), radi (nešto drugo). Ne, ovdje su naredbe gotove (u smislu forme naredbi If-Then-Else ili bilo koje druge od njih). Iako neobičan pristup na prvi pogled, korisnik će vrlo brzo da se adaptira na ovo okruženje i da bude zadovoljan jednostavnošću ambijenta i upotrebe naredbi. Vrlo brzo će korisnik u slučaju potrebe naredbi sa zvukom da se sjeti da je to kratica roze boje, i tačno će znati šta iz tog izbornika želi. Naime, skup naredbi sadržanih u pojedinim kraticama je ograničen. Tih naredbi nema previše. Zainteresovani korisnik će vrlo lahko da prođe kroz svaku od pomenutih kratica i da sagleda šta to ona sadrži. Naredbe koje sadrže logičke operatore: poređenja, konjukcije, disjunkcije, itd. su sadržane u izborniku Operators koji je zelene boje. Korisnik koji želi da ih koristi i koji ih bude koristio će vrlo brzo da ima asocijaciju zelenom bojom na ovaj skup naredbi.

U izborniku Variables možemo da deklariramo varijable (kao i u svakom programskom jeziku), samo što ovdje ne moramo da vodimo računa o tome da li varijablu definišemo kao Int (cio broj srednjeg opsega) ili kao String (niz znakova), i sl. Ko-

rištenje je dakle vrlo jednostavno. Nama, za naše potrebe manipulacije Sprite-om, neophodne su naredbe kretanja (Motion-Plava kratica). Da bi manipulirali kretanjem, trebaju nam naredbe kontrole (Control-Narandžasta kratica) u kombinaciji sa događajima (Events-Žuta kratica) i logičkim operatorima (Operators-Zelena kratica). Varijable nam trebaju da definišemo određene promjenjive koje možemo koristiti na principu: uključi-isključi, tipa da neka promjenjiva bude postavljena da ima vrijednost 0 dok joj ne naredimo da promijeni vrijednost u 1. Govorimo o takvim i sličnim upotrebama promjenjivih.

Takođe, nešto bez čega je nemoguće zamisliti programiranje su nizovi (nizovi promjenjivih). Ovdje, u Scratch-u 3 koristimo nizove ali ih tako ne zovemo. Ovdje ih zovemo listama i liste su sadržane u izborniku Variables. Izbornik Sensing, kako i sam naziv kaže, odnosi se na određene vrste osjetila koja se mogu jako dobro iskoristiti. Tipa, ako letjelica1 priđe letjelici2 neka se desi nešto, ili ako boja ta i ta dodirne boju tu i tu, neka se desi nešto drugo. Ovo je vrlo zahvalan concept jer smo mi itekako svjesni da u klasičnim programskim jezicima nismo imali pristup ovakvim vrstama naredbi. Zbog toga je ovdje i riječ o vizuelnom programiranju a ne o klasičnom programiranju. Logično je onda da imamo različit asortiman objekata koji nisu bili i nisu ni trebali da budu dostupni u klasičnom programiranju.

Kroz navedene elemente naredbi ćemo prolaziti postupno, odnosno po potrebi. Vrlo brzo ćemo postati svjesni šta nam treba, koja naredba bi mogla biti upotrijebljena, na koji način bismo mogli riješiti to i to, itd. Dakle, paleta naredbi će vrlo brzo postati dio nas. Dodatne palete naredbi mogu biti naknadno uključene.

Možemo probati neku od naredbi.

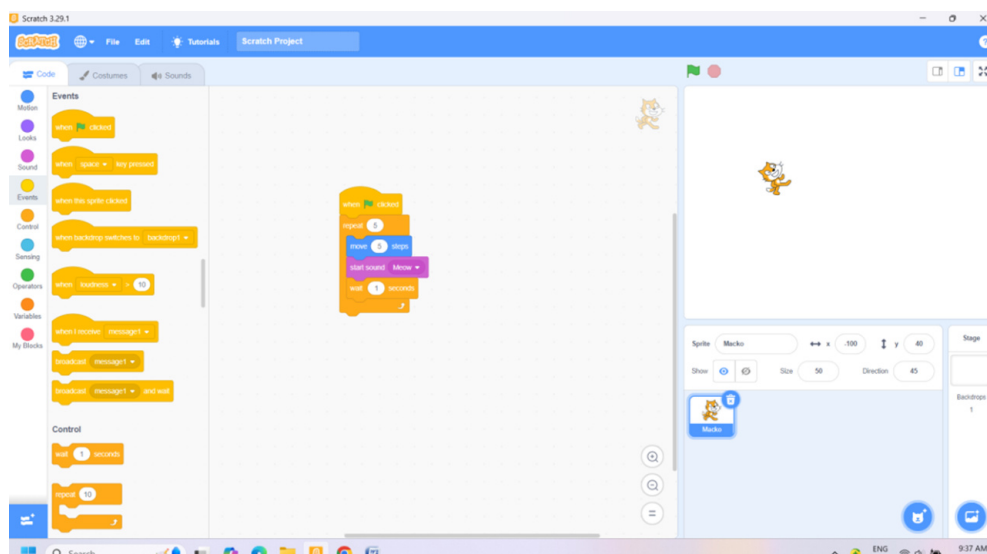
Odemo na izbornik Control, odaberemo naredbu: repeat (10 puta), kliknemo na nju lijevom tipkom miša, i dok držimo pritisak na lijevoj tipki miša, prevučemo pomenutu naredbu na srednji prozor, tj. na dio za kodiranje. Možemo broj 10 promijeniti u broj 5, tj. da se navedeni događaj odvija 5 puta. Dalje, pitanje je šta želimo. Želimo da se nešto desi 5 puta. Tako, možda želimo da se mačak kreće 5 koraka, da se čuje određeni zvuk, i da nakon toga sačeka jednu sekundu. Ako želimo ovakav niz događaja, idemo na naredbe kretanja, odaberemo naredbu: move 10 steps, i prenesemo je i uklopimo u postojeću naredbu (primijetimo da bi neka naredba mogla biti upisana u sklop neke postojeće naredbe, one moraju odgovarati jedna drugoj oblikom, moraju biti uklopive jedna u drugu). Promijenimo broj 10 u 5. Odemo zatim na naredbe za zvukove i naredbu: start sound meow prenesemo na dio za kod i uklopimo ispod prethodno unesene naredbe za kretanje. Napokon, odemo do naredbi za kontrolu, i naredbu: wait 1 seconds prenesemo i uklopimo nakon posljednje unijete naredbe. Da bi uneseni kod mogao biti izvršen, potreban nam je događaj koji će inicirati da se navedeni niz naredbi izvrši. Događaj može biti: pokretanje programa, klik nekom tipkom miša, pritisak na

neko slovo tastature, itd. Često se kao pokretač niza naredbi koristi naredba



iz skupine naredbi za kontrolu događaja.

Ova naredba bukvalno znači da: kada kliknemo zelenu zastavicu (zastavicu za pokretanje igre), da se onda desi, tj. izvrši niz naredbi koje iza ove komande slijede. Ovu naredbu ćemo prenijeti i zalijepiti na vrh već unesenih komandi (primijetiti da one oblikom odgovaraju jedna drugoj, i da oblik ove naredbe je takav da ispred nje ne može biti postavljena nekad druga naredba koja bi joj prethodila). Dobijamo sljedeći oblik ekrana



Klikom na zelenu zastavicu navedeni niz naredbi će biti izvršen.

Mačak će da se pomjeri 5 koraka, da pusti zvuk, i da sačeka 1 sekundu. Ovaj niz naredbi će se odigrati 5 puta.

Ukoliko kliknemo lijevom tipkom miša na prazan prostor srednjeg prozora (prozora gdje kodiramo), i držimo je, a zatim pomjeramo miš lijevo-desno, vidjet ćemo

da se kod pomjera lijevo-desno. Ovo nam govori koliko je unos koda fleksibilan, tj. da ne moramo voditi računa da li smo kod koji unosimo smjestili gore ili dole, da li smo ga smanjili ili povećali itd. Za povećanje odnosno smanjenje veličine koda koristimo tipke za + ili – koje smo već pomenuli i koje se nalaze u donjem desnom uglu srednjeg prozora.

Ukoliko su nam potrebni neki dodatni skupovi naredbi koji nisu po defaultu uključeni u već pomenute skupove naredbi, onda odemo na sličicu

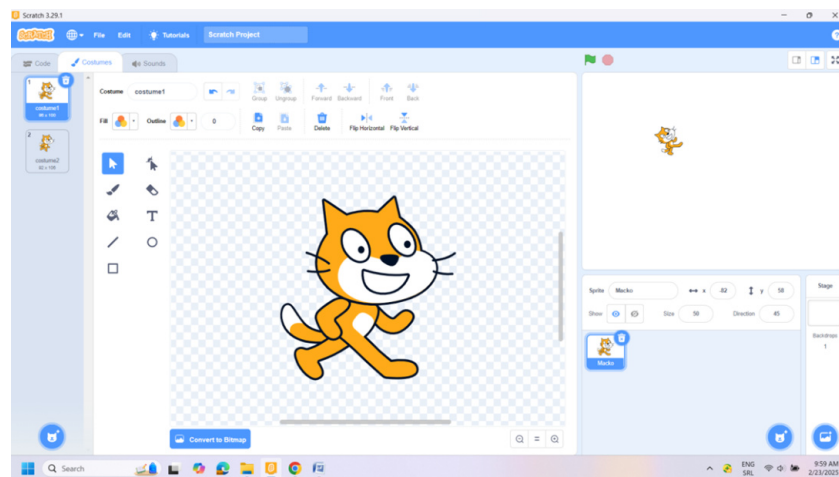


koja se nalazi u donjem lijevom uglu ekrana, i tamo odaberemo nešto od onoga što nam eventualno treba.

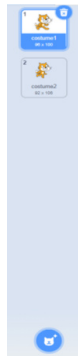
Primijetimo i to da je mačak nakon izvršenja niza naredbi prešao sa koordinata  $x = -100$ ,  $y = 40$ , na koordinate  $x = -82$ ,  $y = 58$ .

Za sada smo iznijeli određeni broj detalja o kratlici Code. Ostaje da vidimo kratice Costumes i Sounds.

Odemo na kraticu Costumes (ne zaboravimo da je i dalje aktivan Sprite Macko, tako da će klikom na kraticu Costumes da budu prikazani eventualni različiti kostimi za naš aktivni Sprite). Klikom na Costumes dobijamo prozor



Na lijevoj strani ekrana vidimo sada prozor



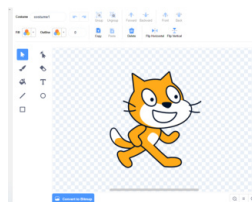
u kojem su prikazana dva postojeća kostima našeg Sprite-a Macka. Klikom na jedan ili drugi vidimo da je riječ o različitim pokretima Sprite-a, u cilju animacije prilikom njegovog kretanja.

Napomenimo da broj kostima zavisi od samog kreatora. Ako koristimo gotove Sprite-ove, onda će broj kostima da zavisi od tvorca kostima. Tako, neki Sprite-ovi će da imaju 1 ili više kostima, zavisno od toga šta je kreator sa njima planirao. U dnu prethodne slike vidimo opciju



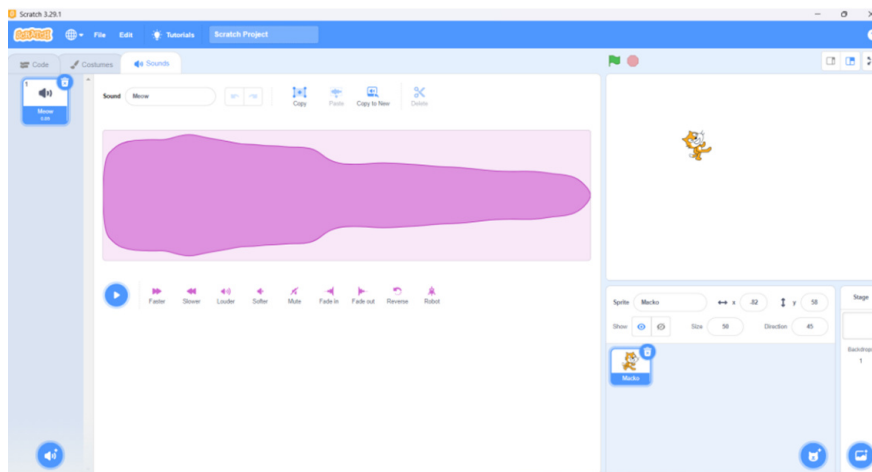
koja nam omogućava da za postojeći Sprite: odaberemo neki dodatni postojeći kostim, ili da nacrtamo novi kostim, ili da upload-ujemo neki kostim.

Primijetimo da je sada srednji ekran, koji je prethodno bio namijenjen za unos koda, sada zamijenjen prozorom



na kome možemo datu sliku Sprite-a, odnosno bilo koji od kostima datog Sprite-a, editovati. O editovanju slika ćemo detaljno govoriti u nastavku.

Konačno, imamo kraticu Sounds. Klikom na nju dobijamo prozor



Primijetimo da je i dalje aktivan Sprite Macko. Dakle zvukovi sa kojima operišemo odnose se na Sprite Macko. Sada su na lijevoj strani pobrojani zvukovi koje koristi Sprite Macko. Vidimo da ustvari imamo samo jedan zvuk trenutno i da se on zove: Meow

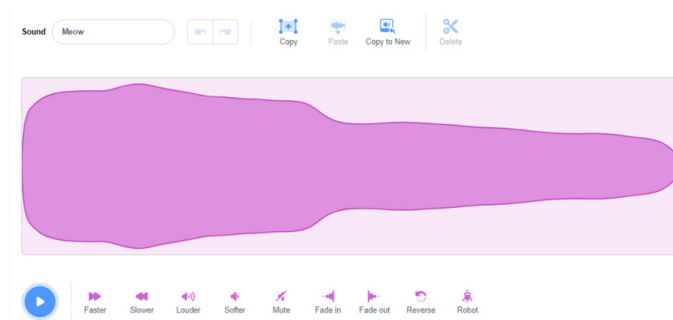


Osim toga, u dnu lijevo imamo ikonu

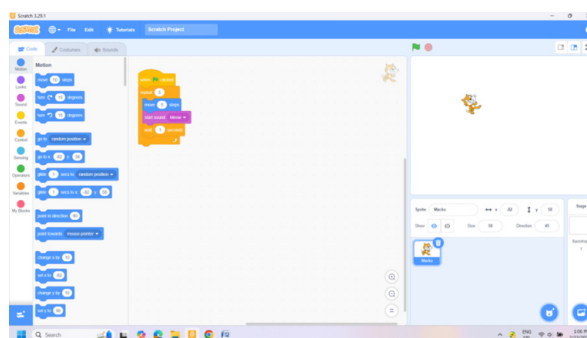


pomoću koje sada (kao i u već opisanim slučajevima), bismo postojeci zvuk (da ga dodamo Sprite-u koji je u upotrebi), da snimimo novi zvuk, ili da upload-ujemo neki zvuk.

Srednji prozor (koji je za kraticu Code bio namijenjen za kod, a za kraticu Costumes bio namijenjen za edit kostima), sada je zamijenjen prozorom



koji služi za edit zvuka (o editu zvuka ćemo govoriti detaljnije u nastavku).  
Vratimo se na kraticu Code, tj. na ekran



Primijetimo u desnom gornjem uglu ekrana ikone

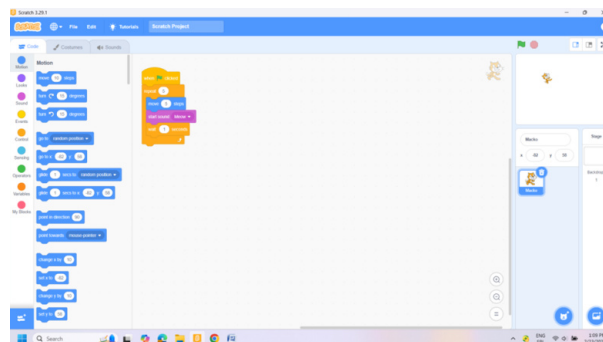


Posljednju od njih smo komentarisali iznad. Vidimo da je trenutno aktivna srednja ikona. Ako kliknemo na lijevu od njih, dobijamo izgled ekrana



## 1 OKRUŽENJE SCRATCH 3

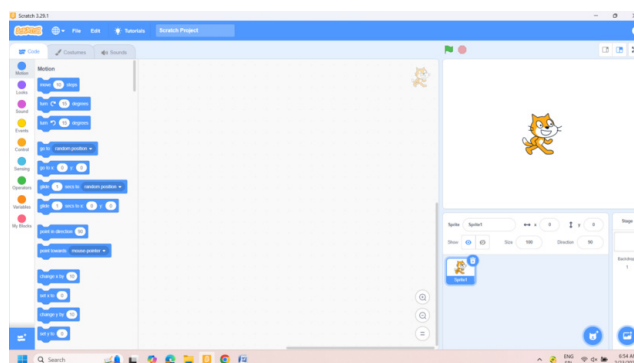
---



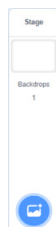
Ta ikona nam služi dakle da proširimo prozor za kod, što može biti korisno u slučaju pisanja većih, širih i zahtjevnijih kodova. Vratimo se na prethodno stanje, tj. kliknimo na srednju ikonu među ikonama



Koristimo tako ponovo naš polazni ekran



U ovom uvodnom dijelu, ostalo nam je još da se osvrnemo na dio koji se nalazi u desnom donjem uglu, tj. na dio

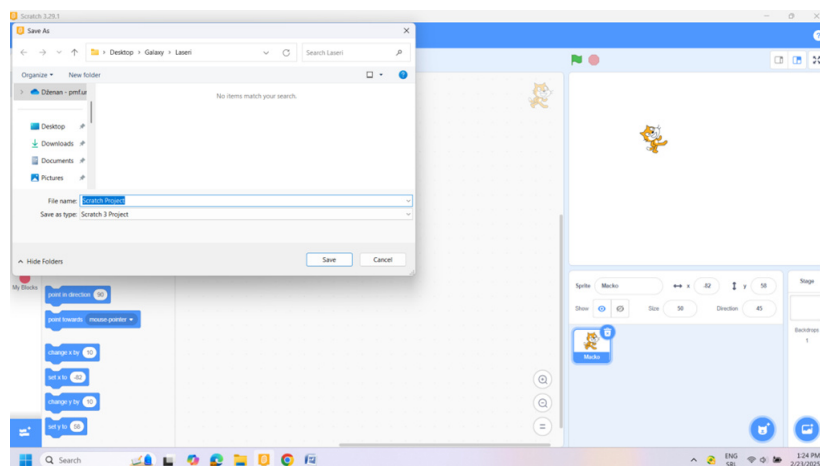


Na ovom mjestu biramo pozadine. Prazan mali ekran koji se nalazi na ovoj slici, ispod kojeg piše Backdrops, znači da još uvijek nismo ubacili niti jednu pozadinu. Klikom na ikonu

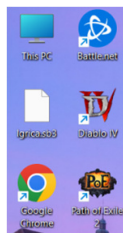


možemo da biramo neku od postojećih pozadina, da nacrtamo sami neku pozadinu, te da upload-ujemo neku pozadinu.

Ukoliko želimo da sačuvamo naš program (ili zovimo ga kod ili projekat), idemo na: File i: Save to your computer. Otvara se prozor



gdje biramo lokaciju i dajemo ime našem projektu. Odaberemo da program sačuvamo na Desktop, damo mu ime: igrica, i idemo na Save. Na Desktopu dobijamo dokument Igrica.sb3

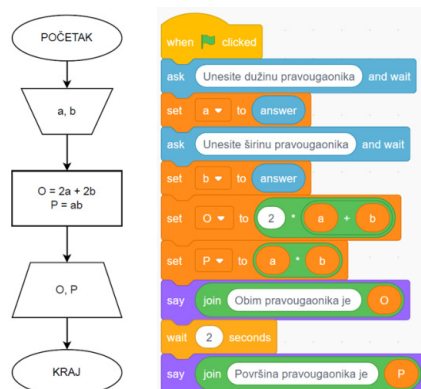


Ekstenzija .sb3 nam govori da je riječ o dokumentu kreiranom u programu Scratch 3.

## 1.1 Riješeni zadaci

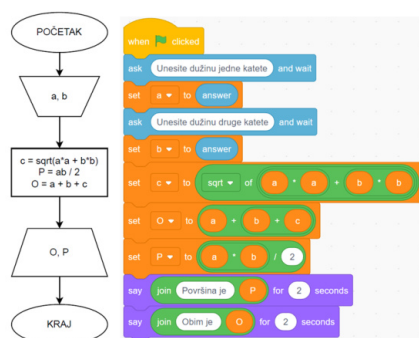
◇ **Zadatak 1.1.1** Korisnik unosi stranice pravougaonika  $a$  i  $b$ . Potrebno je izračunati i ispisati obim  $O = 2 * (a + b)$  i površinu  $P = a * b$  tog pravougaonika. Nacrtati dijagram toka i implementirati rješenje u Eticques - u.

**Rješenje:**



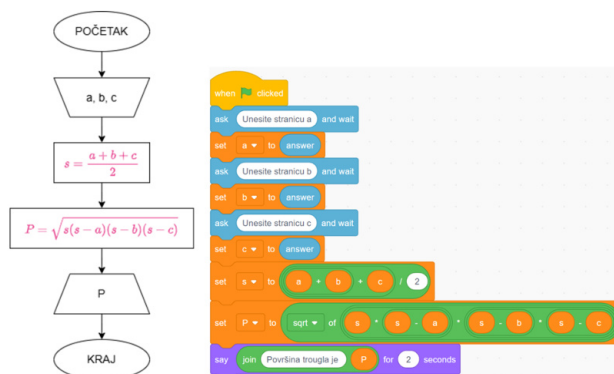
◇ **Zadatak 1.1.2** Korisnik unosi katete pravougloug trougla  $a$  i  $b$ . Potrebno je izračunati i ispisati obim trougla  $O = a + b + \sqrt{a^2 + b^2}$ . Nacrtati dijagram toka i implementirati rješenje u Eticques - u.

**Rješenje:**



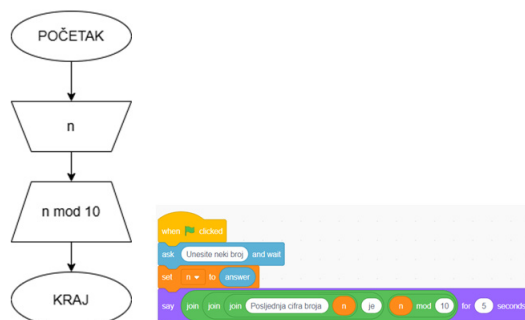
◇ **Zadatak 1.1.3** Korisnik unosi stranice trougla  $a$ ,  $b$  i  $c$ . Potrebno je izračunati i ispisati površinu trougla koristeći Heronovu formulu  $P = \sqrt{s(s-a)(s-b)(s-c)}$ , gdje je  $s = \frac{a+b+c}{2}$ . Nacrtati dijagram toka i implementirati rješenje u Eticques - u.

**Rješenje:**



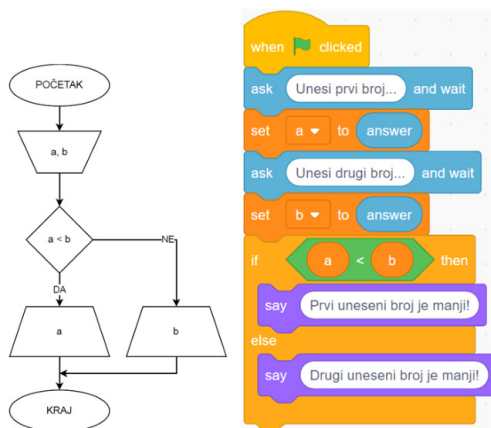
◇ **Zadatak 1.1.4** Korisnik unosi neki broj. Ispisati posljednju cifru tog broja. Nacrtati dijagram toka i implementirati rješenje u Etics - u.

**Rješenje:**



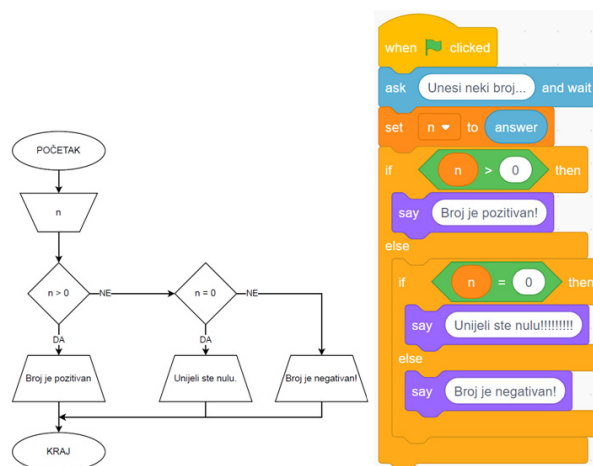
◇ **Zadatak 1.1.5** Korisnik unosi dva različita broja. Potrebno je ispisati manji među njima. Nacrtati dijagram toka i implementirati rješenje u Etics - u.

**Rješenje:**



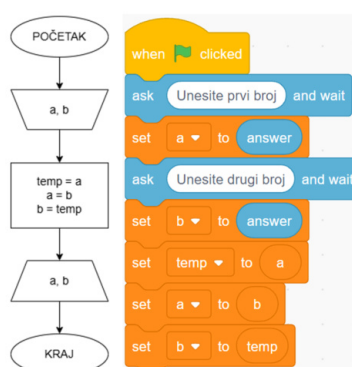
◇ **Zadatak 1.1.6** Korisnik unosi neki broj. Potrebno je ispisati da li je uneseni broj pozitivan ili negativan. Ako je unesena nula, onda je to potrebno naglasiti.

**Rješenje:**



◇ **Zadatak 1.1.7** Korisnik unosi dva broja  $a$  i  $b$ . Razmijeniti njihove vrijednosti.

**Rješenje:**



**1.2 Zadaci za vježbu**

♣ **Zadatak 1.2.1** Korisnik unosi tri broja. Potrebno je ispisati najveću unesenu vrijednost. Nacrtati dijagram toka i implementirati rješenje u Eticques - u.

♣ **Zadatak 1.2.2** Korisnik unosi broj. Potrebno je provjeriti da li je uneseni broj trocifren. Ako jeste, potrebno je taj broj podijeliti sa 8 i ispisati rezultat. Ako nije, potrebno je provjeriti da li je djeljiv sa 8 i ispisati odgovarajuću poruku.

♣ **Zadatak 1.2.3** Korisnik unosi koeficijente kvadratne jednačine. Potrebno je ispisati njena rješenja.



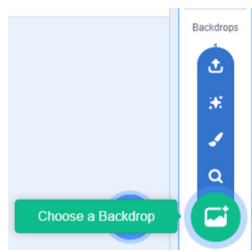


## 2 Događaji, uslovi, varijable

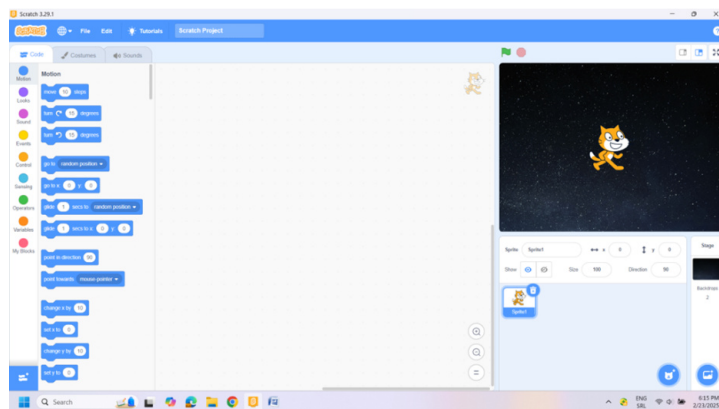
Idemo sada napraviti jednu igricu.

Pokrenimo Scratch 3.

Prvo, promijenit ćemo podlogu koju koristimo. Idemo na ikonu u Choose a Backdrop izborniku u donjem desnom uglu ekrana. Odaberemo podlogu Stars (radi jednostavnosti pronalaženja ukucamo Stars u Search)



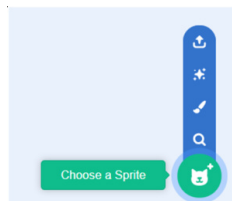
Nakon odabira dobijamo sljedeći izgled



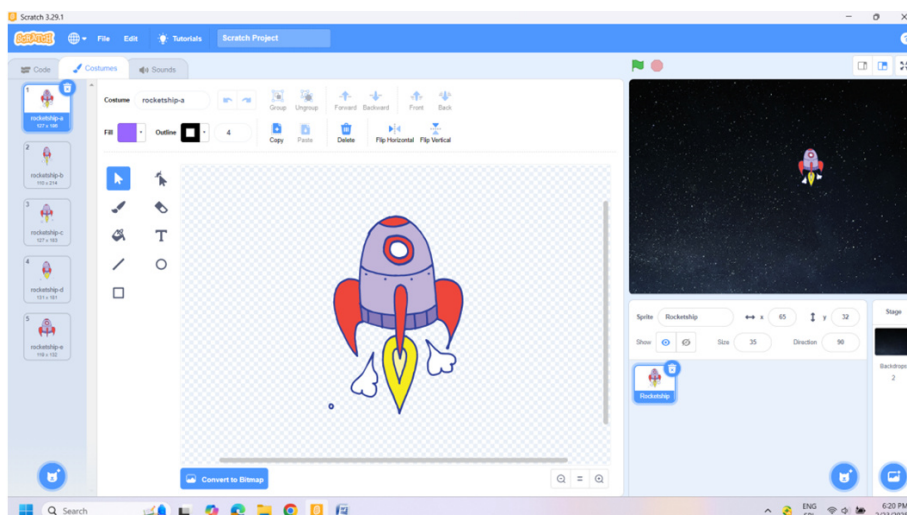
Drugo, obisat ćemo Spite mačka klikom na korpicu za smeće iznad njegove glave.

Potreban nam je novi Sprite.

Idemo na pretragu Sprite-ova



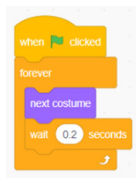
Tamo, odaberemo Sprite: Rocketship. Zbog njegove veličine smanjimo mu Size sa 100 na 35. Odemo na Costumes da vidimo da li imamo ponuđenih i koliko kostima za ovaj Sprite. Imamo situaciju kao na slici



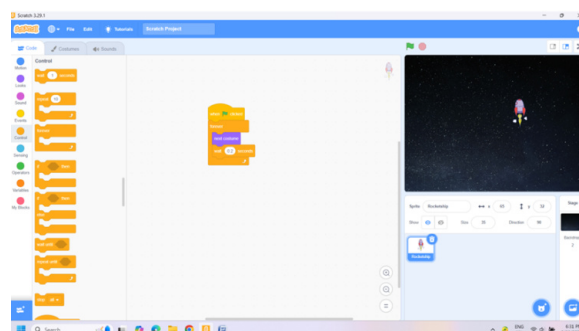
Vidimo da ovaj Sprite ima 5 kostima (lijevo). Peti kostim je nešto veći od prethodnih pa ćemo ga obrisati (želimo da koristimo kostime za animaciju, tj. da dočaramo kretanje letjelice, pa bi nam u tom slučaju taj posljednji kostim zbog svoje veličine odudarao).

Idemo onda dočarati kretanje naše letjelice tj. Sprite-a Rocketship.

Idemo na Code (vidimo da je aktivan Sprite RocketShip jer se njegova providna ikona nalazi u vrhu srednjeg prozora). Ubacimo sljedeće naredbe

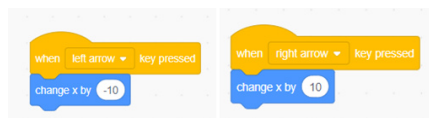


Ove naredbe znače da kada kliknemo zelenu zastavicu, izvršava se ostatak naredbi. Drugim riječima, naređujemo da se nešto vječno radi. To što želimo da se vječno radi jeste da se promijeni kostim Sprite-a i da se onda sačeka 0.2 sekunde. Znači kostim Sprite-a će se stalno mijenjati, tj. mijenjat će se svake 0.2 sekunde između 4 moguća kostima. Ako kliknemo zelenu zastavicu vidjet ćemo da se raketa okreće (okreće se jer ako pogledamo 4 kostima rakete koja imamo vidjet ćemo da na njima raketa zauzima različite položaje, tako da njihovim mijenjanjem svake 0.2 sekunde dobijamo doživljaj okretanja rakete).

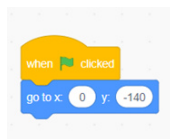


Sačuvamo našu igru na Desktopu pod određenim nazivom.

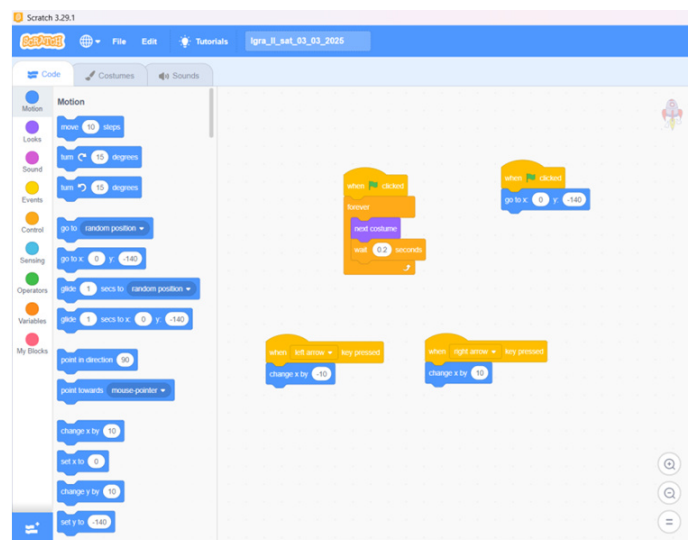
Možemo sada napraviti i mogućnost pomjeranja rakete lijevo desno. Unesemo onda i ovaj kod



Ovaj kod omogućava da se nakon pritiska na strelicu lijevo, raketa pomjeri za 10 podioka lijevo. Isto važi i za drugi dio koda. Pritiskom na strelicu desno, raketa će da se pomjeri za 10 podioka desno (probati ovo). Takođe, stavimo da su koordinate za raketu (koordinate koje će biti inicijalne koordinate) date sa:  $x = 0$ ,  $y = -140$



Imamo tako situaciju kao na slici

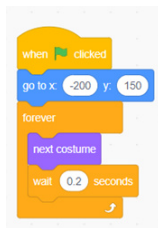


Treće, idemo dodati novi Sprite.

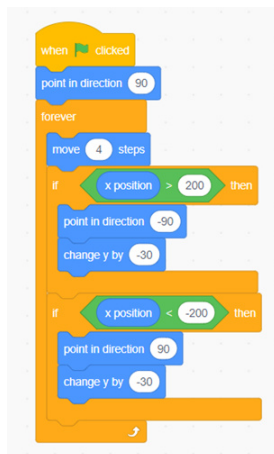
Odemo na Choose a Sprite i odaberemo Sprite pod nazivom Gobo. Sada imamo na ekranu dva Sprite-a, tako da vodimo račun na koji Sprite unosimo kod. Dakle, pratimo koja providna sličica se nalazi u gornjem uglu srednjeg ekrana. Promijenimo veličinu za Sprite Gobo na 40 (Size 40).

Sada ćemo unijeti kod da pozicioniramo Sprite Gobo, i da podesimo izmjene njegovih kostima (analogno kao u slučaju letjelice).

Dodamo kod za Sprite Gobo (obavezno prvo kliknuti na Sprite Gobo u donjem desnom uglu)



Dakle, kada kliknemo zelenu zastavicu, pozicioniramo Gobo-a na koordinate  $x = -200$ ,  $y = 150$ , što je ustvari gornji lijevi ugao. Nakon toga Gobo zauvijek mijenja svoje moguće kostime (od 3 moguća) svake 0.2 sekunde. Ako pokrenemo program, tj. kliknemo na zelenu zastavicu, onda vidimo da se položaj usta Gobo-a stalno mijenja. U biti želimo da napravimo igricu poznatu kao Galaksija (Space Invaders), u kojoj let-jelicom obaramo dolazeće protivnike. Dodajmo sada naredbe pomoću kojih će Gobo da se kreće desno-lijevo i da se polagano spušta prema dole, tj. prema nama. Unesemo sljedeće naredbe



Ovaj kod nam govori da kada kliknemo zelenu zastavicu, Gobo se usmjeri u pravcu 90, tj. u smjeru broja 3 na satu. Nakon toga dešava se beskonačna petlja, tj. vječno se

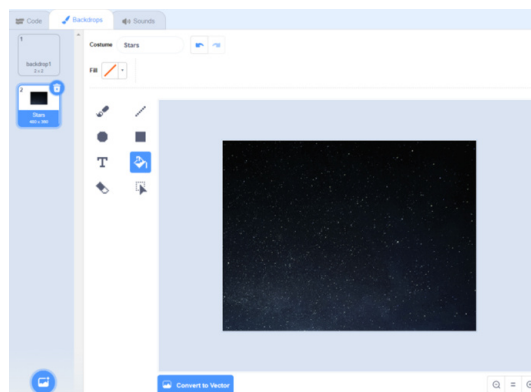
dešava sljedeće: prvo, Gobo se pomjeri 4 koraka (u pravcu u kojem smo ga usmjerili), a nakon toga se ispituje dokle je on stigao. Ako je otišao udesno po  $x$  osi za više od 200 (više od desnog kraja ekrana), onda ga usmjerimo u suprotnom pravcu, tj. u pravcu -90, odnosno u pravcu broja 9 na satu, i naredimo mu da se pomjeri niže za 30 (to je naredba da promijenimo  $y$  poziciju za -30). S druge strane, ako je otišao ulijevo po  $x$  osi za manje od -200 (manje od lijevog kraja ekrana), onda ga usmjerimo ponovo u pravcu 90, tj. broja 3 na satu, i spustimo ga niže za 30. Ako kliknemo na zastavicu, vidjet ćemo rezultat izvršavanja ovog koda. Kada želimo da zasustavimo izvršavanje, kliknemo na crveni krug. Primijetimo da dok je zelena zastavica aktivna, da je oko posljednjeg koda aktivna optraka, što znači da se taj dio koda izvršava. Ne zaboraviti povremeno sačuvati igru na Desktop pod postojećim nazivom.

Četvrto, idemo napraviti pozadinu sa natpisom Game Over.

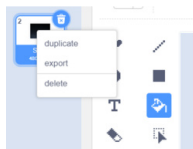
Radi jednostavnosti, igra će da bude završena u onom momentu kada Alien (Gobo) dodirne našu letjelicu. Idemo desno i kliknemo na već postojeću pozadinu (crna mala sličica ispod "Stage")



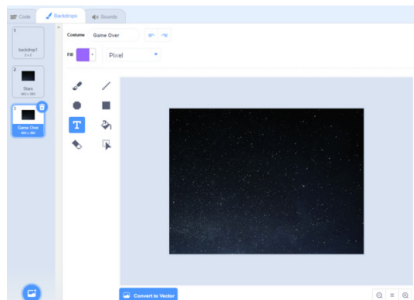
Dobijemo prozor kao na slici



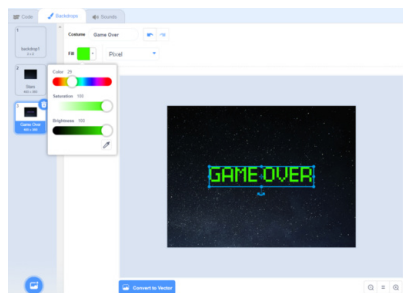
Kliknemo desni klik na već aktivnu pozadinu Stars i idemo na duplicate



Dobijamo novu, treću pozadinu Stars2 (program po automatizmu doda broj kada se nešto uduplava). Na opciji Costume (gore lijevo), umjesto Stars2 upišemo Game Over. Idemo na opciju Text (veliko T)

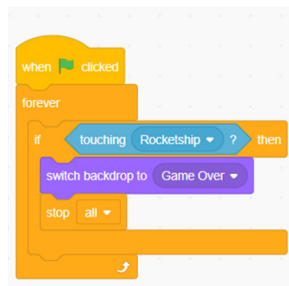


Kliknemo na crni ekran da se pojavi prozor za tekst, i u njemu upišemo: GAME OVER. Razvučemo ga po potrebi i centriramo (tako da se znak + na text box-u poklopi sa znakom + na pozadini). Odemo na Fill i podesimo žutu ili zelenkastu boju. Ukoliko je potrebno, koristimo komandu "back" (ako nešto pogriješimo)



Nakon što smo napravili podlogu: Game Over, obavezno kliknuti na opciju: Convert to Vector (vidjeti dno prethodne slike). U vektorskom obliku slika se ne može zamutiti, njene tačke se ne mješaju, boje se ne mješaju prilikom višestrukog pokretanja programa, tj. tokom njegovog izvršavanja. S druge strane, ako sliku ostavimo u Bitmap obliku (koji nije vektorski), sve pomenuto vezano za narušavanje kvaliteta slike će da se desi. Generalni savjet je da sve pozadine i sve Sprite-ove nakon njihovog kreiranja, podešavanja itd, pretvorimo u vektorski oblik.

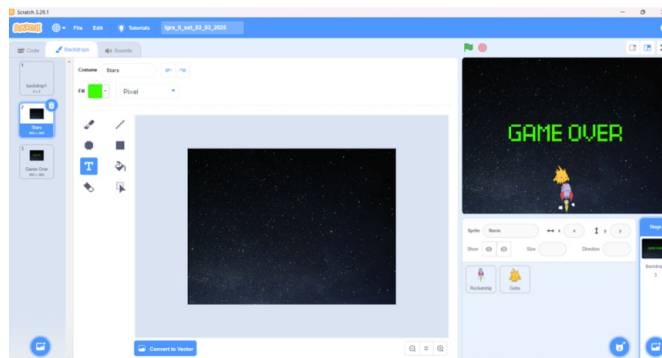
Sada smo u poziciji da dodatno isprogramiramo Gobo-a, tj. našeg Alien-a, tj. da dodamo na njega kod koji se odnosi na njegov eventualni dodir sa našom letjelicom, tj. raketom. Pseudo-govoreći, treba da na njega unesemo niz naredbi, takvih, da ako se on dodirne sa našom letjelicom, da se pozadina postavi na ovu novu pozadinu: Game Over koju smo napravili, i da se zaustavi igra. Unosimo tako sljedeći kod na Gobo-a (kliknemo na Gobo-a i onda na Code)



Prije izvršavanja koda, tj. klika na zelenu zastavicu, postavimo se na prethodnu podlogu, tj. na podlogu Stars. Ako sada izvršimo program, tj. kliknemo na zelenu zastavicu, i pustimo da se Gobo kreće, vidjet ćemo da će u momentu dodira Gobo-a sa našom letjelicom, da se pojavi podloga: Game Over, i igra će da stane. Znači, prethodni niz naredbi govori da treba nešto vječno (forever) da se dešava nakon našeg klika na zelenu zastavicu. Zašto vječno ? Zato što ne znamo kada će se dodir dogoditi. Tako, kada iskoristimo beskonačnu petlju (forever petlju), omogućavamo da se niz naredbi unutar nje izvrši u bilo kojem momentu, tj. u momentu kada se dodir dogodi. Forever petlja bukvalno radi stalno, i čeka na taj način da se nešto dogodi. Tako, ako se dogodi da Gobo (na kojem smo unijeli kod) dodirne Rocketship, onda će podloga da se promijeni na podlogu: Game Over, i sve će da se zaustavi (igra će da stane).

Ne zaboraviti povremeno dosnimiti igru pod već postojećim imenom

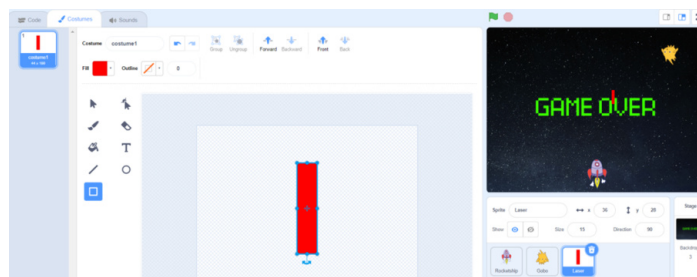




Peto, pravimo laser, kojim ćemo eliminisati Alien-e, tj. Gobo-e.  
Sami ćemo napraviti laser, tj. imitaciju lasera, kao jedan novi Sprite. Idemo na Choose a Sprite i onda na Paint

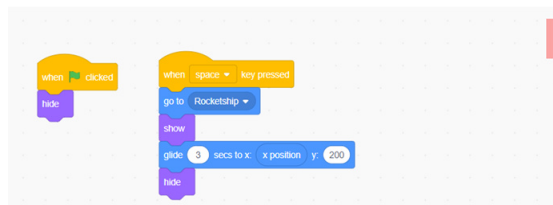


Novom Sprite-u damo ime Laser. Odaberemo da bude npr. crvene boje, idemo na Rectangle i nacrtamo ga (pravougaonog oblika). Kasnije mu podesimo veličinu. Možemo (u našem slučaju) staviti npr. da je veličina Lasera jednaka 15

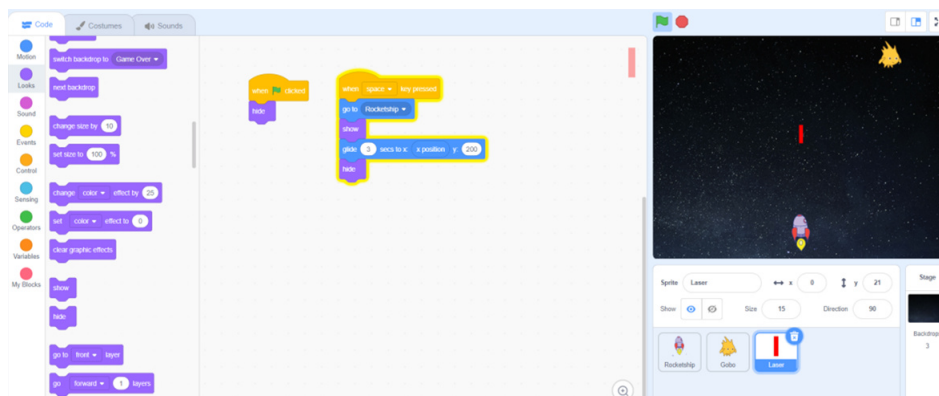


Ostaje da napravimo kod za Laser.

Unijet ćemo sljedeći kod na Sprite Laser



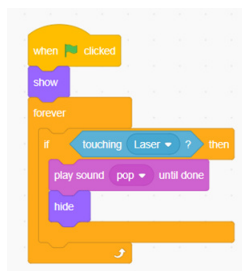
Jasno, Laser ne treba da bude aktivan, tj. ne treba da bude vidljiv ako drugačije ne želimo. Naime, ako "pucamo" iz naše letjelice, onda Laser treba da postane vidljiv, i da se kreće počevši od naše letjelice pa do vrha vidljivog ekrana, s tim da  $x$  koordinata ne treba da se mijenja, dok, jasno,  $y$  koordinata mora da se mijenja. Dakle, prvi dio koda nam govori to da kada pokrenemo program, tj. kada kliknemo na zelenu zastavicu, inicijalno, Laser treba da bude sakriven. Drugo, želimo da Laser postane aktivan pritiskom na tipku Space. Znači, drugi dio koda nam govori o aktivaciji Lasera. Kada pritisnemo tipku Space, Laser treba da ode do naše letjelice, da se pojavi tamo, i da klizi od te pozicije (ne mijenjajući  $x$  koordinatu) pa do vrha ekrana (vrijednost nekih 200 koordinate  $y$ ), nakon čega ponovo treba da se skloni, tj. sakrije. Ako pokrenemo program, tj. kliknemo zelenu zastavicu, vidjet ćemo da se dešava ovo što smo upravo opisali.



Ipak, primijetimo da i ako Laser dodirne Gobo-a, da ništa neće da se desi. Naime,

nismo još unijeli kod koji bi se eventualno odnosio na dodir Laser-a i Gobo-a (ili obrnuto).

Šesto, dodajmo kod na Gobo-a koji se odnosi na već pomenuti njegov dodir sa Laserom

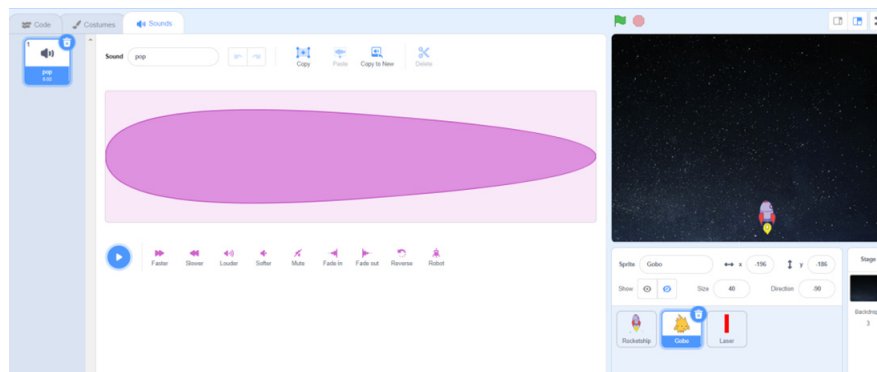


Kod je sada jasan.

Kada pokrenemo program, tj. kliknemo na zelenu zastavicu, Gobo treba da se pojavi (on će svakako inicijalno da se pojavi, kako se već i pojavljuje, ali ćemo malo kasnije vidjeti zašto moramo eksplicitno reći na ovome mjestu da želimo da se on pojavi).

Zatim, pokrećemo beskonačnu petlju (kao i do sada), i postavljamo If uslov, tj. uslov: ako dodiruje (Gobo) Laser, onda odsviraj zvuk: Pop do kraja i sakrij Gobo-a. Znači beskonačna petlja radi vječno (forever) a unutar nje se pitamo da li Gobo dodiruje laser. U momentu dodira odsviramo zvuk Pop i sakrijemo Gobo-a. Dodir Gobo-a sa Laserom znači da je Gobo mrtav. Zvuk Pop koristimo da dočaramo činjenicu da je Gobo eliminisan. Osim toga, ako je Gobo mrtav onda treba da ga sklonimo, tj. sakrijemo sa ekrana. Zbog toga koristimo naredbu "hide". Međutim, kada na ovom mjestu sakrijemo Gobo-a, mi nemamo više niti jedno mjesto u do sada unesenim kodovima na kojem pojavljujemo Gobo-a, tj. na kojem za njega koristimo naredbu "show". Znamo da ako pokrenemo ponovo igru da Gobo treba da se pojavi. Zbog toga ga moramo "pojaviti" negdje.

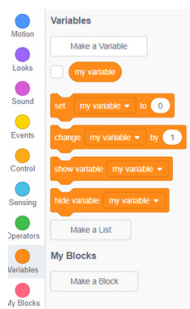
Jedna od potencijalnih lokacija da "pojavimo" Gobo-a jeste onda na početku posljednjeg koda. Dakle, Gobo-a treba da "pojavimo" u nekom od kodova neposredno nakon klika na zelenu zastavicu. Napomenimo da to nije morao biti posljednji kod. Mogao je to biti bilo koji od kodova koji su uneseni na Gobo-a. Napomenimo da zvuk Pop nismo mi unosili. On je već unesen "kažemo po defaultu", na neke od Sprite-ova koji su ugrađeni u Scratch 3. Da bi provjerili da li na nekom od Sprite-ova koje koristimo ima već unesen neki zvuk, kliknemo na taj Sprite, i onda na Sounds



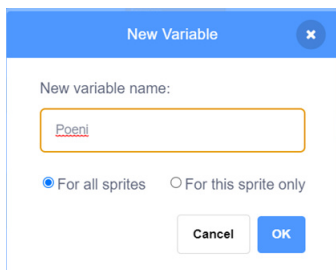
Sada možemo probati igru i uvjeriti se da ona radi ono što smo željeli i već objasnili.

Ukoliko eliminišemo Gobo-e, onda u skladu sa igračkim praksama, možemo željeti da ubacimo u igru i poene. Napravimo sada varijablu, tj. promjenjivu koja će odražavati broj osvojenih poena u igri. Ovu varijablu ćemo nazvati "Poeni" i inicijalno ćemo postaviti da njena vrijednost bude jednaka 0. Svakom eliminacijom Gobo-a, povećavat ćemo vrijednost promjenjive Poeni za 1.

Idemo dakle na Variables lijevo (narandžasto dugme), i kliknemo na opciju Make a Variable

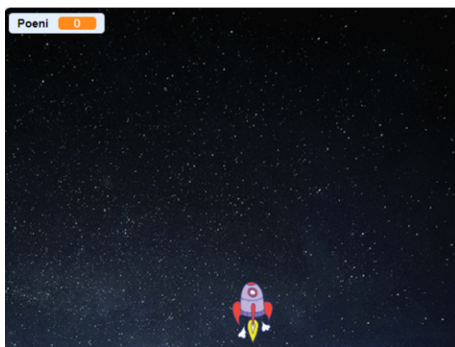


Damo ime Poeni našoj varijabli, i postavimo je da bude globalna varijabla, tj. postavimo je da bude aktivna za sve Sprite-ove (ovo znači da svi Sprite-ovi i kodovi mogu pristupiti ovoj varijabli)

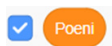


Kliknemo na OK.

Inicijalno, svaka varijabla koju napravimo je aktivna na ekranu igre



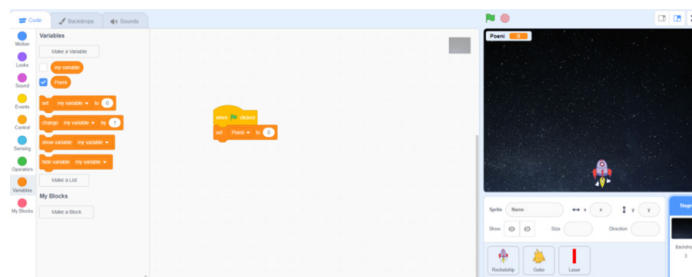
Naime, pored njenog imena je aktivna "kvakica"



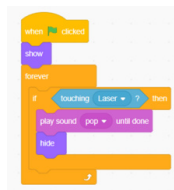
Ukoliko želimo da neka varijabla ne bude aktivna u prozoru igre, onda ukinemo kvakicu pored njenog imena. U slučaju varijable Poeni, ona nam je potrebna da broji poene, tako da sada kvakicu ostavljamo.

Kako smo već napomenuli, varijablu Poeni postavljamo da bude jednaka 0 na početku igre. To možemo uraditi na bilo kojem mjestu u igri, tj. u sklopu bilo kojeg od unesenih kodova koji počinju zelenom zastavicom. Naime, promjenjiva Poeni

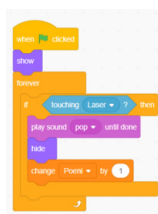
je globalna, tako da njeno postavljanje na vrijednost 0 će da funkcioniše sa bilo kojeg mjesta. Možemo otići na neutralno mjesto, npr. kliknemo na pozadine i odemo na Code, pa tamo postavimo promjenjivu Poeni da bude jednaka 0. Unesemo dakle kod



Konačno, potrebno je da unesemo kod koji će povećavati vrijednost promjenjive Poeni svakom sljedećom eliminacijom Gobo-a. Pošto povećavanje zavisi od Gobo-a, logično je da ovaj kod unesemo na Gobo-a. Dovoljno je da doradimo već uneseni kod. U kodu



kojeg već imamo na Gobo-u, dodamo još jednu naredbu, tako da imamo sljedeći adaptirani kod

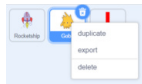


Dakle, ako Gobo dodirne Laser, on je onda mrtav, sviramo zvuk Pop, sklanjamo Gobo-a, i povećavamo vrijednost promjenjive Poeni za 1. Ta vrijednost će jasno biti evidentirana i na ekranu nakon eliminacije Gobo-a (probati ovo pokretanjem igre).

Sedmo, potrebno je da dodamo više Gobo-a (sjetimo se da smo našeg postojećeg Gobo-a inicijalno postavili na lokaciju  $x = -200$ ,  $y = 150$ ). Ostale Gobo-e koje budemo dodavali ćemo pomjeriti udesno i prema dole (u zavisnosti od toga koliko Gobo-a uopšte želimo da napravimo).

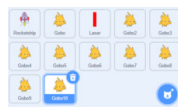
Recimo da želimo da imamo ukupno 10 Gobo-a na ekranu.

Da ne bismo ponavljali korake, dovoljno je da postojećeg Gobo-a umnožimo 9 puta. Kliknemo desni klik na ikonu postojećeg Gobo-a, i idemo na duplicate 9 puta

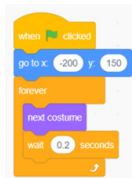


Napomenimo da će na ovaj način biti stvoreno još 9 Gobo-a, koji će imati dodane brojeve u svojim imenima, i pri čemu će kodovi koje smo već unijeli na prvog Gobo-a da ostanu nepromijenjeni i prisutni na svakom od novih 9 Gobo-a. Ti kodovi će biti prisutni na svakom od umnoženih Gobo-a (te kodove će jasno biti potrebno adaptirati u smislu polaznih pozicija novih Gobo-a).

Imena Gobo-a možemo ostaviti onakvim kakve ih je sam program poudio



Inicijalna pozicija prvog Gobo-a (Gobo) je kako smo već vidjeli  $x = -200$ ,  $y = 150$

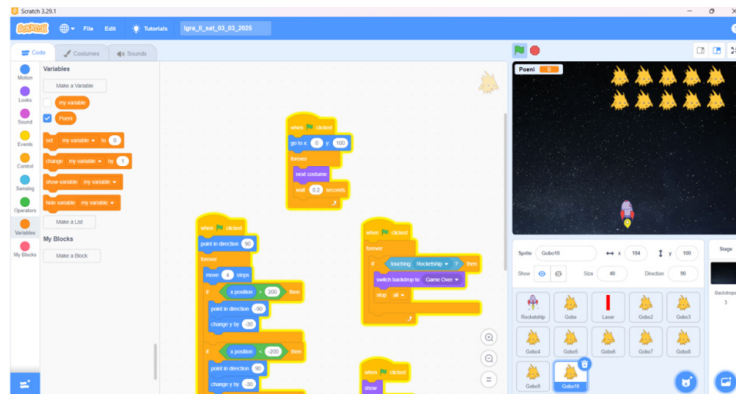


Ovaj kod se nalazi i na svakom od ostalih Gobo-a (Gobo2-Gobo10). Moramo ga adaptirati za svakog od njih. Za Gobo-e: Gobo2-Gobo10, inicijalne koordinate ćemo staviti da redom budu

|         |             |           |
|---------|-------------|-----------|
| Gobo2:  | $x = -150,$ | $y = 150$ |
| Gobo3:  | $x = -100,$ | $y = 150$ |
| Gobo4:  | $x = -50,$  | $y = 150$ |
| Gobo5:  | $x = 0,$    | $y = 150$ |
| Gobo6:  | $x = -200,$ | $y = 100$ |
| Gobo7:  | $x = -150,$ | $y = 100$ |
| Gobo8:  | $x = -100,$ | $y = 100$ |
| Gobo9:  | $x = -50,$  | $y = 100$ |
| Gobo10: | $x = 0,$    | $y = 100$ |

Izmijenimo ovo u odgovarajućim kodovima Gobo-a: Gobo2-Gobo10 i sačuvamo igru.

Testiramo igru

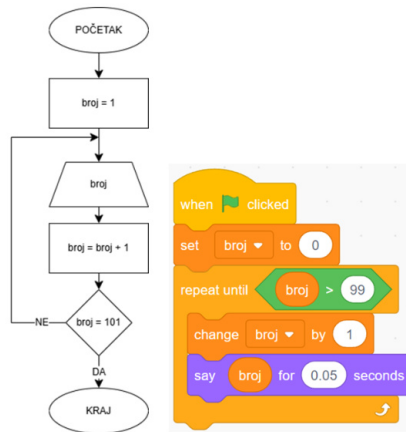


## 2.1 Riješeni zadaci

◇ **Zadatak 2.1.1** Napraviti program koji će ispisati prvih 100 prirodnih brojeva. Nacrtati dijagram toka i implementirati u Eticques-u.

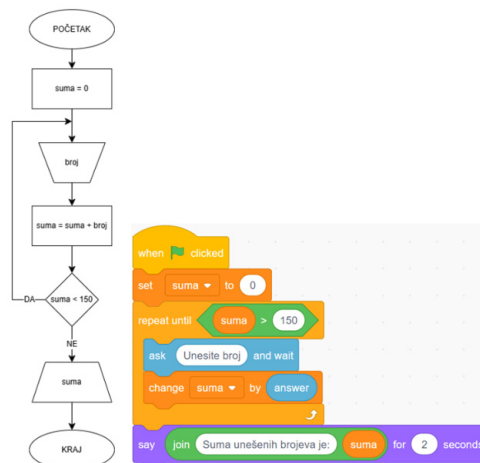


**Rješenje:**



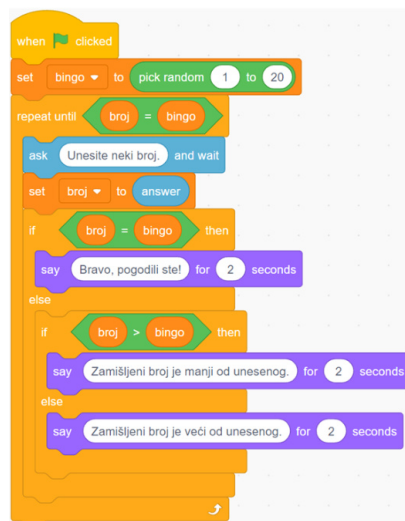
◇ **Zadatak 2.1.2** Napraviti program koji traži od korisnika da unosi brojeve sve dok se ne dostigne suma od 150 ili veća. Na kraju, ispisati sumu.

**Rješenje:**



◇ **Zadatak 2.1.3** Računar nasumično bira cio broj između 1 i 20. Korisnik pokušava pogoditi taj broj tako što unosi vrijednosti jednu po jednu. Nakon svakog pokušaja, program obavještava korisnika da li je pogodio broj, ili da li je taj broj veći ili manji od unesenog. Program se završava onda kada korisnik pogodi broj. Implementirati rješenje u Etics-u.

**Rješenje:**



## 2.2 Zadaci za vježbu

♣ **Zadatak 2.2.1** Potrebno je napraviti program koji ispisuje sve brojeve od 0 do unesenog broja. Korisnik na početku unosi gornju granicu, a program prikazuje sve brojeve redom, uz kratko zadržavanje između ispisa.

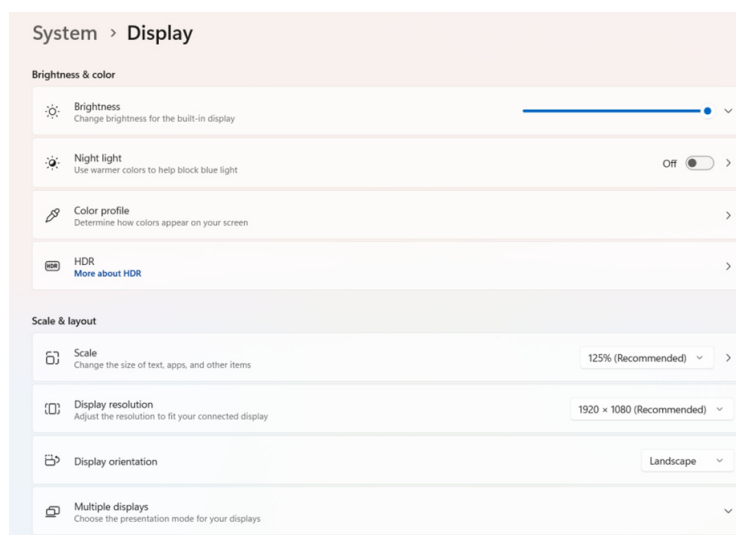
♣ **Zadatak 2.2.2** Računar nasumično bira cio broj između 1 i 20. Korisnik pokušava pogoditi taj broj, ali ima najviše 5 pokušaja. Nakon svakog pokušaja, program obavještava korisnika da li je pogodio broj, ili da li je taj broj veći ili manji od unesenog. Ako korisnik ne pogodi broj nakon 5 pokušaja, program završava porukom da korisnik nije uspio.

♣ **Zadatak 2.2.3** Računar nasumično bira cio broj između 1 i 100. Korisnik pokušava pogoditi taj broj. Program broji koliko pokušaja je bilo potrebno da se broj pogodi, i na kraju ispisuje ukupan broj pokušaja zajedno s porukom o uspjehu.



### 3 Od Scratch-a do Eticques-a

Sada se počinjemo ozbiljnije baviti programiranjem u Scratch-u, odnosno generalizaciji Scratch-a, tzv. Etik (Eticques) okruženju. Već smo spominjali da nam skromna rezolucija u Scratch-u neće biti dovoljna za naše potrebe. Program Etik nam dozvoljava da biramo rezoluciju kolika nam je potrebna. Jednostavno pogledamo na našim računarima i laptopima kolika je to rezolucija koju koristimo, i tu onda rezoluciju upotrijebimo u Etik-u. U našem slučaju, npr. idemo desni klik na desktop, i onda na Display settings. Dobijemo prozor gdje vidimo trenutnu korištenu rezoluciju na računar (laptopu)



Na Display resolution vidimo da je riječ o rezoluciji  $1920 \times 1080$ .

Program Etik koristimo online. Ovo je blagi nedostatak u odnosu na Scratch koji smo mogli download-ovati i koristiti ga lokalno. Dakle, ako imamo internet pri ruci onda nemamo problema ni sa okruženjem Etik. Link za program Etik je sljedeći

<https://sheeptester.github.io/scratch-gui/?width=1920&height=1080>

Primijetimo da unutar ovog linka pišu vrijednosti rezolucije koja nam treba, tj.  $1920 \times 1080$ . Te vrijednosti možemo odmah u linku promijeniti tako da ih postavimo

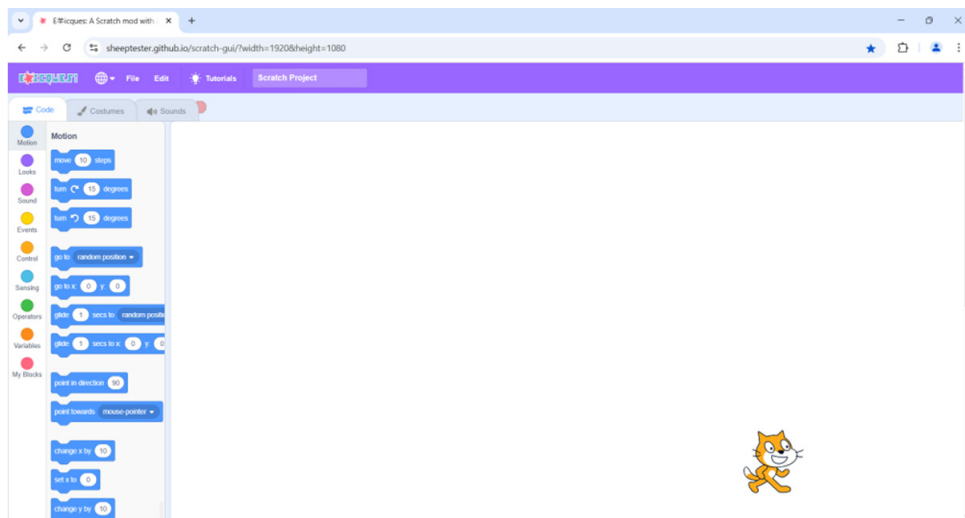
na vrijednosti koje koristi naš računar. Tako, ako neko koristi rezoluciju  $1680 \times 1050$ , onda bi nakon podešavanja, link za pristup Etik-u izgledao

<https://sheeptester.github.io/scratch-gui/?width=1680&height=1050>

Naravno, možemo pokrenuti Etik i u rezoluciji koja nam ne odgovara momentalno, pa onda u zaglavlju, tj. na lokaciji linka promijeniti vrijednosti, tj. postaviti ih da odgovaraju našem računar.

Napomenimo da dokumenti sačuvani u Etik-u imaju istu ekstenziju, tj. nastavak kao i oni u Scratch-u. Dakle, riječ je o nastavku .sb3. Drugim riječima, Scratch i Etik su bratska okruženja koja dijele iste osobine (kao i neki drugi programi). Prijelaz sa Scratch-a na Etik sa te strane ne bi trebao da bude problematičan.

Ako upotrijebimo prvi link od prethodna dva u nekom od pretraživača, npr. Chrom-u, dobijamo okruženje



Vidimo da je u biti sve ostalo isto kao u Scratch-u, samo što je rezolucija veća. Ta veličina može na prvi pogled djelovati nezgrapna i nezgodna za upotrebu. Korisnik mora stalno koristiti scroll trake. Ipak, nije potrebno mnogo vremena da se navikne na ovakav sistem rada.

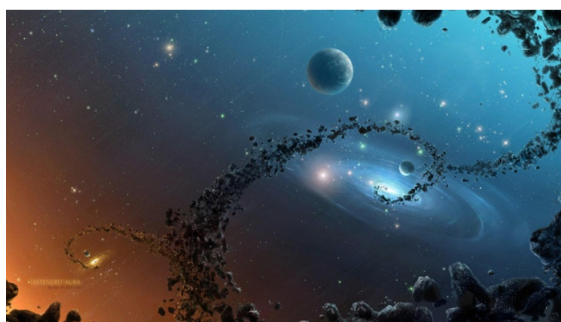
Ako idemo da sačuvamo naš dokument na File + Save to your computer, vidjet ćemo da se za razliku od Scratch-a ovdje čuvanje podataka obavlja automatski.

Naime, nije nam dozvoljeno da sami dajemo ime dokumentu. Ime koje Etik daje našem dokumentu je Scratch Project.sb3. Čuva ga na lokaciji Downloads. Nama nije teško otvoriti folder Downloads i promijeniti ime tako dobijenog dokumenta na ime koje nama odgovara. Dakle, desni klik na dokument i rename, damo ime koje želimo, npr. Gal (pri čemu nastavak .sb3 ne diramo). Dobijemo dokument Gal.sb3. Na ovom mjestu je najbolje zatvoriti Etik pa ga ponovo pokrenuti, te iz njega otvoriti sačuvani dokument u Downloads-u. Otvorimo dakle Etik ponovo i preko File + Load from your computer, otvorimo dokument Gal.sb3. Primijetimo da Etik koristi folder Downloads kao prirodno mjesto, tj. mjesto na kojem po defaultu čuva dokumente. Nama bi ta praksa trebala da bude prirodna ili da postane prirodna, jer i ako bismo dokumente iz foldera Downloads prebacivali na neke druge lokacije, svako naredno snimanje u Etik-u bi rezultovalo čuvanjem novog dokumenta u folderu Downloads. Znači, čuvanjem na drugoj lokaciji ne bi postigli puno (osim ako neki dokument na nekoj novoj lokaciji želimo da sačuvamo iz nekih privatnih razloga - da ga sačuvamo trajno i sl). Prednost čuvanja na lokaciji Downloads je ta što prilikom pokretanja Etik-a i load-ovanja nekog dokumenta, Etik nam po automatizmu nudi posljednji sačuvani dokument, što je najčešće i slučaj koji nama odgovara. Sada, nakon što smo promijenili ime dokumenta u Gal.sb3, i nakon što smo ga otvorili u Etik-u, ako ponovo odemo da sačuvamo dokument, vidjet ćemo da će ime dokumenta biti dato u odnosu na ime koje smo mi sada dali. Drugim riječima, novi dokument će biti sačuvan kao Gal(1).sb3. Na ovaj način možemo pratiti do koje verzije smo stigli. Naravno, kada se broj verzija znatno poveća, dovoljno je posljednjoj verziji ponovo dati neko kraće ime i nastaviti sa radom od te verzije. Možemo ponovo dati i ime Gal.sb3 i sl.

Prvo, treba nam sada bar jedna pozadina veličine  $1920 \times 1080$  (u našem slučaju). Pošto pravimo igru o galaksiji onda su nam potrebne i takve pozadine: kosmos, univerzum, galaksija, itd. Mi možemo prilikom pretrage slika zahtijevati da slike budu odgovarajuće veličine ( $1920 \times 1080$  ili druge). Ipak, ne mora značiti da će slike koje dobijemo pretragom zaista i biti takve veličine. Neke će biti manje rezolucije, neke veće. One slike koje nam veličinom odgovaraju možemo odmah zadržati. One koje su manje veličine od nama potrebne ne trebamo ni uzimati u obzir (prilikom load-ovanja takvih slika naš ekran neće biti adekvatno popunjen). Čak i ako dođe do popunjavanja ekrana, slika će biti razvučena što nije naš cilj. Isto tako, slike veće veličine ne mora značiti da će se uklopiti na pravi način. Takve slike je najbolje edit-ovati, i resize na veličinu koja nama treba. Drugim riječima, iz slika velikog formata dovoljno je isjeći dio veličine koja nama odgovara. Opisane varijante koje se mogu desiti će se zaista i desiti, tako da upravo opisano je najbolje probati u konkretnim situacijama. Pri tome je bitno napomenuti da format slike koju možemo koristiti je .jpg (kao najčešći, što ne znači da se ne mogu koristiti i drugi formati koje Etik dozvoljava). Prilikom

editovanja slike možemo koristiti klasično editovanje. Dovoljno je da imamo sliku sačuvanu negdje i onda je otvoriti i edit. Prilikom isjecanja određenog dijela veličine koja nam odgovara dovoljno je onda sačuvati dio koji nam treba.

Recimo da smo sačuvali sliku veličine  $1920 \times 1080$



Upload-ujemo ovu sliku u Etik kao Backdrops i dobijenom kostimu damo ime A1.

Pošto ćemo koristiti dosta objekata u radu: slike, zvukove, Sprite-ove, itd. najbolje je sve te elemente čuvati u odgovarajućim folderima. Možda koristiti jedan zajednički folder a onda u njemu imati posebne foldere za slike, zvukove, Sprite-ove, letjelice, lasere, razne vrste oružja itd.

Treba napomenuti da slike koje upload-ujemo u cilju pozadina, dosta opterećuju brzinu izvršavanja programa u smislu njegovog load-ovanja i snimanja. Ipak, takve stvari su prirodne i na njih se treba navići. Često će se desiti da prilikom snimanja Etik ponudi prozor na kome možemo birati čekaње da se snimanje izvrši ili izlazak iz programa. U takvim situacijama treba čekati. Dakle, nije riječ o nekoj blokadi programa i sl.

U gornjem desnom uglu idemo na prvu opciju među opcijama



Naime, želimo da na ekranu uočimo i ostale elemente (da slika ne zauzima glavni dio ekrana).

Obavezno sliku koju smo upload-ovali konvertovati u vektorski oblik.

Nije loše odmah nakon upload-ovanja: podloge, Sprite-a i sl, umanjiti veličinu dijela određenog za sliku upotrebom opcije "minus"





Govorimo u cilju dobijanja boljeg pregleda elemenata sadržanih na prozoru za sliku.

Backdrop1 kojeg imamo po defaultu možemo ukinuti. Neka stoji samo pozadina A1.

Takođe, Sprite mačka možemo ukinuti.

Sačuvamo naš dokument.

Sada želimo da dodamo letjelicu koju ćemo koristiti u našem programu. Nećemo koristiti ugrađene letjelice jer su one djetinjastog oblika. Isto važi i za zvukove, osim, naravno, ukoliko nam neki od njih treba.

Važno je napomenuti da Sprite-ove koje upload-ujemo treba da biramo da budu transparentni. Naime, ukoliko oni nisu transparentni, onda iza njih stoji i njihova pozadina, tako da takav Sprite nakon upload-ovanja vuče sa sobom i čitavu svoju pozadinu, tj. zaklanja veliki dio ekrana, što nama nije potrebno. Zbog toga, potrebno je tražiti Sprite-ove koji su transparentni. Isto važi i za razne vrste oružja koja ćemo koristiti ali i druge objekte koji ne smiju da budu zasjenjeni bilo čime ili da imaju iza sebe neku podlogu. Dobra stranica za pregled transparentnih objekata se nalazi na linku

<https://www.pngwing.com/>

Odlaskom na ovu stranicu dobijamo izgled



Ovdje je dovoljno u pretraživač ove stranice ukucati nešto što nas zanima (ne en-

gleskom jeziku najbolje), i pretražiti višestruko ponuđene podatke. Svakim klikom na neki od objekata otvaraju se kopije ove stranice sa novim objektima. Nakon više koraka korisnik će sigurno naći objekte koji u manjoj ili većoj mjeru odgovaraju njegovim potrebama. Napomenimo da transparentan objekat poznajemo po kvadratićima koji se nalaze u njegovoj pozadini. Ako u pretraživač unesemo npr. alien ship, dobit ćemo više zanimljivih objekata. Jedan od njih je dat slikom

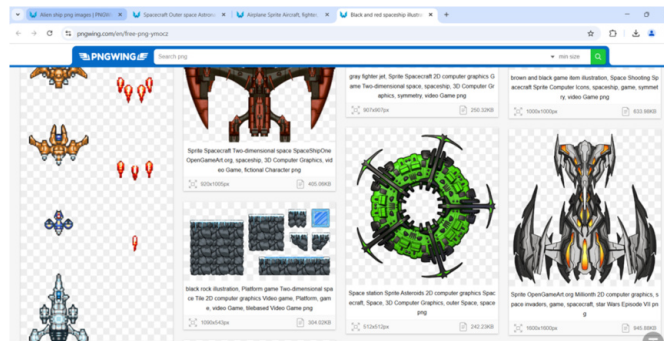


U pozadini vidimo kvadratiće koji znače da je riječ o transparentnom objektu. U nekim situacijama transparentan objekat poznajemo po njegovoj crnoj podlozi. Naime, kada objekat u čijoj podlozi se nalaze kvadratići sačuvamo na neku lokaciju na kompjuteru, ti kvadratići više neće biti vidljivi. Umjesto njih ili će biti crna podloga ili providna podloga. To su neki klasični pokazatelji da je riječ o transparentnom objektu.

Nama je sada potrebna letjelica (koju koristimo umjesto klasičnog Rocketship-a kojeg smo ranije koristili). Možemo, kako smo već rekli, u pretraživač posljednje stranice ukucati nešto što asocira na letjelice, tipa: alien ship ili airplane ili nešto sa sličnim značenjem. Nakon unosa "alien ship" i nekoliko klikova (vidjeti na vrhu stranice da je otvoreno više kopija pomenute stranice), uočavamo objekat koji nam je zanimljiv. Riječ je o letjelici

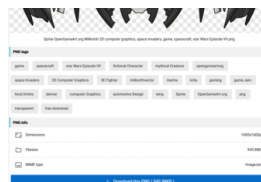


Ovu letjelicu vidimo u donjem lijevom uglu pretrage



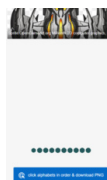
Kliknemo na letjelicu koju želimo.

Ukoliko nas stranica informiše da treba da sačekamo određeno vrijeme, onda tako i uradimo. Možda će biti potrebno da sagledamo određeni kratki video, pa onda i taj dio provedemo. Nakon pristupa slici, u dnu vidimo opciju za download



Kliknemo na download.

Otvori nam se prozor



na kojem se od nas traži da unesemo niz slova koji se od stranice zahtijeva da bi down-

load otpočeo (obično je ovaj korak potrebno uraditi prvi put, dok download ostalih slika koje nam trebaju počinje automatski). Uradimo kako se od nas traži. Kliknemo na simbole



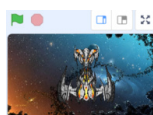
koji se nude, i otpočnemo download.

Slici koja je smještena u Downloads, promjenimo ime u M14 (najbolje da nazivi budu uniformno radi jednostavnosti), te je premjestimo u folder za slike koji smo ranije pominjali i koji ćemo koristiti za čuvanje objekata sa kojima radimo. Možemo, kako smo već rekli, posebno imati folder za slike letjelica, posebno za slike lasera, podloga, itd. U prvi moment ne izgleda tako, ali nakon određenog vremena uoči se potreba za nekom vrstom klasifikacije objekata i sistematičnosti njihovog čuvanja. Posebno treba čuvati zvukove do kojih je teže doći i koje je nezgodnije editovati nego kada je riječ o slikama.

Upload-ujemo sada letjelicu kao Sprite u naš kod.

Nakon upload-a, odmah odemo na Costume da vidimo kako slika izgleda, smanjimo je sa "minus", eventualno centriramo, i konvertujemo u vektorski oblik. Neka ovo bude procedura koje ćemo se pridržavati. Naime, ranije smo istakli da korištenje slike u Bitmap format-u može poremetiti izgled slike. Naime, pokretanjem Sprite-a u Bitmap obliku u smislu njegovog obavljanja određenih radnji: kretanja, ispaljivanja objekata u obliku razne vrste municije, itd. remeti raspored njegovih tačaka. Takav Sprite će nakon izvjesnog vremena da se pretvori u određenu jednobojnu masu na kojoj se više neće moći razaznavati elementi i tačnost koje smo imali na početku. Da bi prevenirali takve scenarije, bitno je, ponavljamo opet, odmah na početku sliku konvertovati u vektorski oblik. Dostignimo igru.

Primijetiti u gornjem desnom uglu prikaz



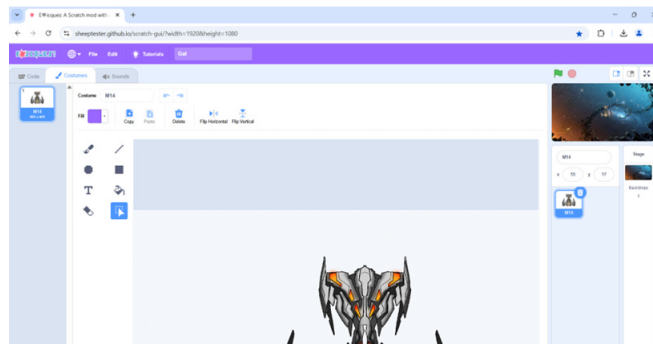
### 3 OD SCRATCH-A DO ETICQUES-A

---

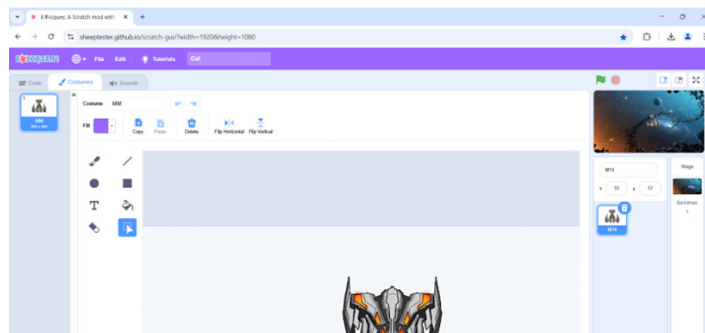
Vidmo da je naša letjelica zauzela ogroman dio prozora. Sve slike koje uploadujemo, inicijalno imaju Size postavljen na 100. Moramo dakle ovaj Size znatno smanjiti. Kliknemo na srednju opciju opcija



da bismo uopšte mogli pristupiti opciji Size. Višestrukom upotrebom horizontalnih i vertikalnih klizača postavimo se na opciju Size i postavimo je na 20 (možemo mijenjati po potrebi). Klikom na prvu od upravo navedenih opcija vratimo se na prethodno stanje. Imamo situaciju kao na slici

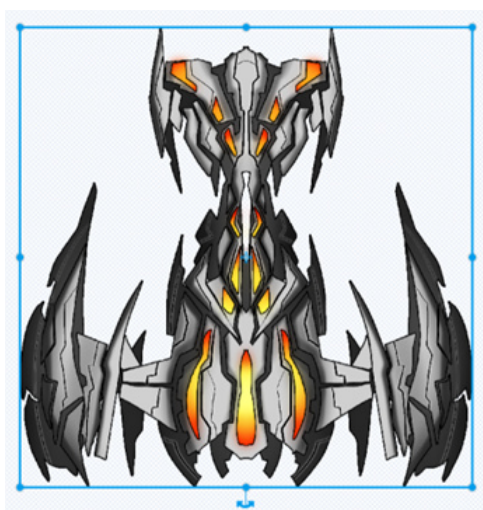


U prozoru Costumes (lijevo gore), ovom kostimu damo ime MM. Imamo



Prilikom rada u okruženju Etik ali i sličnim njemu, nije loše ostaviti aktivnim stranice i eventualne aplikacije koje dodatno koristimo. Naime, često će nam zatrebati stranica za download transparentnih slika, stranica za zvukove, stranica za konvertovanje raznih fomata u neke druge formate, itd. Riječ je o tome da često moramo imati dosta materijala pri ruci u mnogim situacijama.

Unesene slike možemo po potrebi smanjiti i ručno



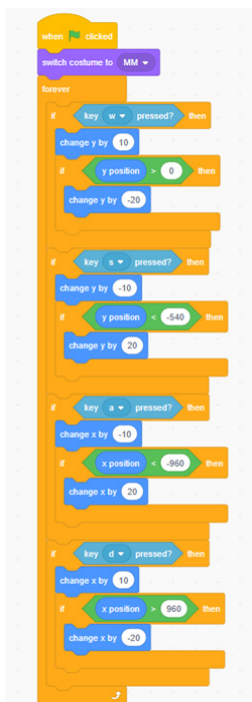
Naravno, da bi to bilo izvodivo, slika mora biti u vektorskom obliku.

Nakon eventualnog ručnog podešavanja veličine slike, ne zaboraviti centrirati sliku, tj. znak + same slike poklopiti sa znakom + pozadine slike (nije teško uočiti plavi i crni poimenuti znak + nakon pomijeranja slike).

Sada, kao i ranije, prvo treba da podesimo kretanje naše letjelice.

Ukoliko se prisjetimo onoga što smo radili ranije, onda imamo na umu da smo letjelice pomjerali lijevo-desno, i da te kretanje nisu bile tako glatke i tečne. Ovdje ćemo poraditi na pomenutim nedostacima. Takođe, letjelica treba da se može kretati i gore i dole (do određene mjere), tj. do visine koju mi smatramo pogodnom. Oni koji igraju igre su svjesni da kretanje objekata ne treba da bude limitirano samo na kretanje lijevo-desno, gore-dole (tipke w-a-s-d). Naime, običaj je da pritiskom određenih kombinacija pomenutih tipki na tastaturi, letjelici treba da bude omogućeno i kretanje ukoso, tj. gore-lijevo, gore-desno, dole-lijevo, dole-desno.

Dok smo pozicionirani na letjelicu M14, idemo na Code i unesemo sljedeći kod



Ovdje, za svaki slučaj, naredimo da se kostim MM pojavi neposredno nakon pokretanja programa. Zatim, zahtijevamo da se beskonačno odvija niz naredbi koje slijede, odnosno koje su sadržane u beskonačnoj petlji. Važno je napomenuti da je desna granica vidljivog ekrana jednaka nekih 960 jedinica (po  $x$  osi), lijeva granica jednaka -960 jedinica (po  $x$  osi), gornja granica jednaka 540 jedinica (po  $y$  osi), te donja granica jednaka -540 jedinica (po  $y$  osi). Mi ne želimo da se naša letjelica smije kretati po cijeloj ploči. Naime, na gornjem dijelu ploče treba da budu raspoređeni neprijatelji, tako da taj dio rezervišemo za njih. Zbog toga je prirodno da ograničimo kretanje naše letjelice u rasponu -960 do 960 po  $x$  osi (cijeli raspon ekrana), te u rasponu -540 do 0 po  $y$  osi (donji dio ekrana).

Tako, ako pritisnemo tipku  $w$ , letjelica će se pomjeriti prema gore za 10. Ako se pri tome desi da letjelica pređe gornju granicu 0 koju smo postavili, onda treba da se vrati prema dole za 20 (pomjeramo je za -20 po  $y$  osi). Analogno funkcionišu ostale tipke. Ako pritisnemo tipku  $d$ , letjelica će se pomjeriti udesno za 10. Ako se pri tome desi da letjelica pređe desnu granicu 960 koju smo postavili, onda ona treba da se vrati ulijevo

za 20 (pomjeramo je za -20 po  $x$  osi). Pošto se navedene If naredbe (4 njih) nalaze unutar jedne beskonačne petlje, onda će eventualni istovremeni pritisak na neke dvije od 4 navedenih tipki, rezultirati pomjeranjem letjelice ukoso u odgovarajućem pravcu (probati ovo).

Sada treba da dodamo neprijatelje u igru. Cilj je da napravimo letjelice koje se pojavljuju na slici



Vidimo da nam je potrebno 12 tipova letjelica.

Neke od njih ćemo upotrijebiti, tj. umnožiti više puta, tako da ukupan broj neprijateljskih letjelica koje se pojavljuju na mapi, treba da bude jednak 50. Ako krenemo od gornjeg reda pa udesno red po red, ove letjelice ćemo zvati Gobo1 do Gobo50. Pored imena, Gobo1-Gobo50, kostime koje odgovaraju ovim letjelicama ćemo zvati MM1-MM50 (sa određenim izuzecima sa kojima ćemo se susresti i koje ćemo posebno istaći).

Letjelice o kojima govorimo moramo naći na stranici koju smo već koristili.

Potrebno ih je sačuvati (kao i našu glavnu letjelicu), promijeniti im imena (rename) u folderu Downloads (dati im neka razumna imena). Kada ih upload-ujemo, u igri ćemo im svakako nakon umnožavanja dati imena Gobo1-Gobo50 kako smo već napomenuli.

Letjelice koje se nalaze u Downloads folderu potrebno je sačuvati u nekom folderu koji smo predvidjeli za njih. Potrebno ih je imati i čuvati bez obzira da li smo ih već upload-ovali. Naime, desi se da se tokom rada neka slika u igri promijeni, ili da je greškom iskvarimo na ovaj ili onaj način. Zbog toga, te slike moramo imati sačuvane na sigurnoj lokaciji u slučaju da nam zatrebaju ponovo.



Sada, letjelice upload-ujemo kao Sprite-ove, umnožimo ih da dobijemo raspored kao na slici, i damo imena kako smo rekli da ćemo dati.

Ne zaboraviti nakon upload-a koristiti opciju "minus" za smanjivanje veličine srednjeg prozora, te konvertovati slike u vektor format.

Najbolje ići letjelicu po letjelicu kao na datoj slici, te nakon upload-a podesiti imena, veličine, orijentaciju, itd.

Primijetimo npr. da je nakon upload-a, letjelica



okrenuta "naopačke" od onoga kako bismo željeli da ona izgleda.

Nakon što smo ovu sliku smanjili sa "minus" i konvertovali u vektorski oblik, potrebno je da je orjentišemo suprotno od orijentacije koju već imamo. Dovoljno je lijevo-gore otići na opciju flip vertical. Ime kostima promjenimo u MM1, smanjimo Size sa 100 na 50, i damo ime Gobo1. Za sada letjelicu rasporedimo manuelno u lijevi gornji ugao gdje i treba da se nalazi. Kasnije ćemo dati precizne startne pozicije svake od navedenih letjelica.

Dodajemo sada ostale letjelice.

Za svaku od njih potrebno nam je: ime, ime kostima, Size, te početna lokacija. Možemo onda te podatke navesti tabelarno za svaku od letjelica koje dolaze. Jasno, veličine letjelica koje su kopije već unesenih letjelica ostaju iste. U njihovom slučaju potrebno je samo promijeniti: ime, ime kostima te početnu lokaciju

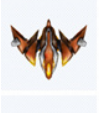
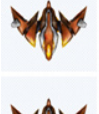
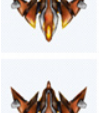
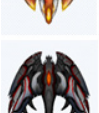
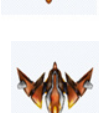
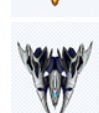
| Letjelica                                                                           | Ime   | Ime kostima | Size | Poč. lokacija |
|-------------------------------------------------------------------------------------|-------|-------------|------|---------------|
|  | Gobo1 | MM1         | 30   | (-750,280)    |
|  | Gobo2 | MM2         | 30   | (-600,350)    |

### 3 OD SCRATCH-A DO ETICQUES-A

---

|                                                                                     |        |      |    |            |
|-------------------------------------------------------------------------------------|--------|------|----|------------|
|    | Gobo3  | MM3  | 30 | (-450,280) |
|    | Gobo4  | MM4A | 40 | (-249,400) |
|    | Gobo5  | BB1  | 40 | (0,600)    |
|    | Gobo6  | MM6A | 40 | (249,400)  |
|    | Gobo7  | MM7  | 30 | (450,280)  |
|  | Gobo8  | MM8  | 30 | (600,350)  |
|  | Gobo9  | MM9  | 30 | (750,280)  |
|  | Gobo10 | MM10 | 10 | (-120,300) |
|  | Gobo11 | MM11 | 20 | (0,280)    |
|  | Gobo12 | MM12 | 10 | (120,300)  |
|  | Gobo13 | MM13 | 10 | (-690,150) |
|  | Gobo14 | MM14 | 10 | (-546,150) |

### 3 OD SCRATCH-A DO ETICQUES-A

|                                                                                     |        |      |    |            |
|-------------------------------------------------------------------------------------|--------|------|----|------------|
|    | Gobo15 | MM15 | 15 | (-381,150) |
|    | Gobo16 | MM16 | 10 | (-216,150) |
|    | Gobo17 | MM17 | 10 | (-72,150)  |
|    | Gobo18 | MM18 | 10 | (72,150)   |
|   | Gobo19 | MM19 | 10 | (216,150)  |
|  | Gobo20 | MM20 | 15 | (381,150)  |
|  | Gobo21 | MM21 | 10 | (546,150)  |
|  | Gobo22 | MM22 | 10 | (690,150)  |
|  | Gobo23 | MM23 | 15 | (-690,50)  |
|  | Gobo24 | MM24 | 15 | (-546,50)  |
|  | Gobo25 | MM25 | 15 | (-381,30)  |
|  | Gobo26 | MM26 | 15 | (-216,50)  |

### 3 OD SCRATCH-A DO ETICQUES-A

---

|                                                                                     |        |      |    |            |
|-------------------------------------------------------------------------------------|--------|------|----|------------|
|    | Gobo27 | MM27 | 15 | (-72,50)   |
|    | Gobo28 | MM28 | 15 | (72,50)    |
|    | Gobo29 | MM29 | 15 | (216,50)   |
|    | Gobo30 | MM30 | 15 | (381,30)   |
|   | Gobo31 | MM31 | 15 | (546,50)   |
|  | Gobo32 | MM32 | 15 | (690,50)   |
|  | Gobo33 | MM33 | 20 | (-540,-80) |
|  | Gobo34 | MM34 | 20 | (-420,-80) |
|  | Gobo35 | MM35 | 20 | (-300,-80) |
|  | Gobo36 | MM36 | 20 | (-180,-80) |
|  | Gobo37 | MM37 | 20 | (-60,-80)  |
|  | Gobo38 | MM38 | 20 | (60,-80)   |
|  | Gobo39 | MM39 | 20 | (180,-80)  |

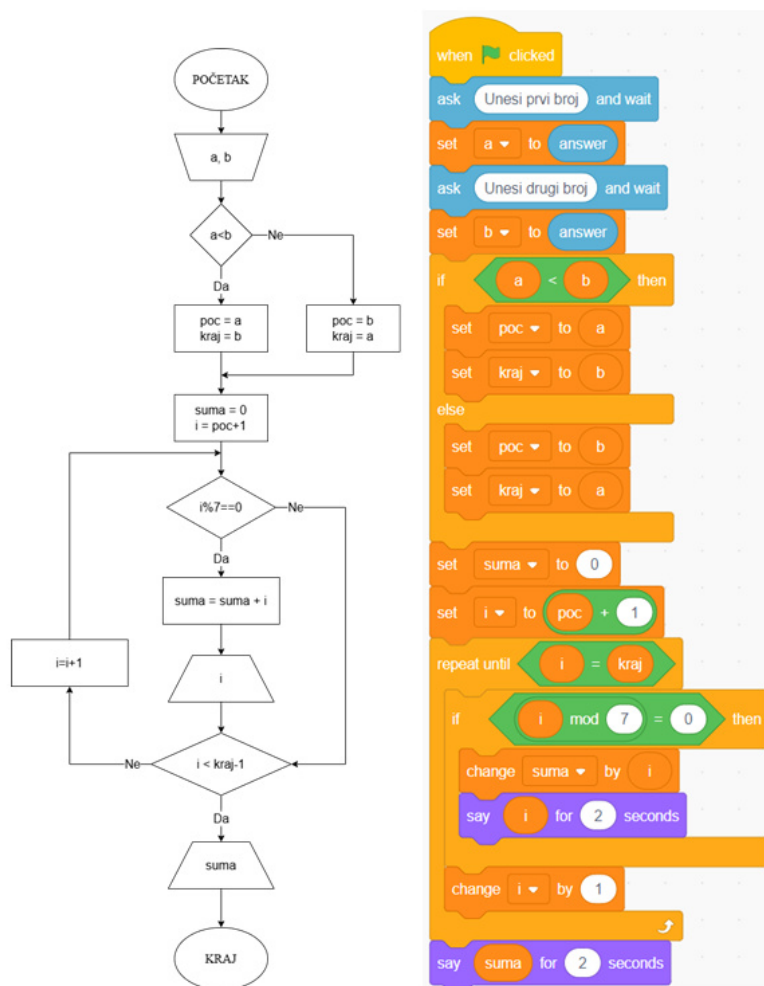
### 3 OD SCRATCH-A DO ETICQUES-A

|                                                                                     |        |      |    |             |
|-------------------------------------------------------------------------------------|--------|------|----|-------------|
|    | Gobo40 | MM40 | 20 | (300,-80)   |
|    | Gobo41 | MM41 | 20 | (420,-80)   |
|    | Gobo42 | MM42 | 20 | (540,-80)   |
|    | Gobo43 | MM43 | 20 | (-420,-180) |
|    | Gobo44 | MM44 | 20 | (-300,-180) |
|   | Gobo45 | MM45 | 20 | (-180,-180) |
|  | Gobo46 | MM46 | 20 | (-60,-180)  |
|  | Gobo47 | MM47 | 20 | (60,-180)   |
|  | Gobo48 | MM48 | 20 | (180,-180)  |
|  | Gobo49 | MM49 | 20 | (300,-180)  |
|  | Gobo50 | MM50 | 20 | (420,-180)  |

### 3.1 Riješeni zadaci

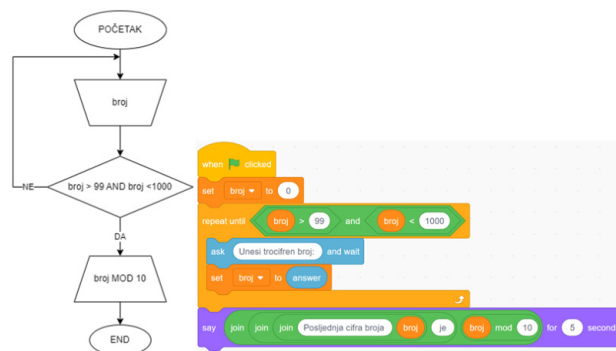
◇ **Zadatak 3.1.1** Korisnik unosi dva broja. Potrebno je izračunati i ispisati sumu brojeva djeljivih sa 7 između ta dva broja, te ispisati te brojeve.

**Rješenje:**



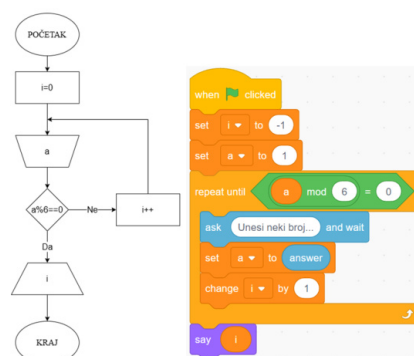
◇ **Zadatak 3.1.2** Korisnik unosi brojeve sve dok ne unese trocifren broj. Kada unese trocifren broj, program treba da ispiše njegovu cifru jedinica.

**Rješenje:**



◇ **Zadatak 3.1.3** Korisnik unosi brojeve sve dok ne unese paran broj djeljiv sa 3. Kada unese takav broj, program treba da ispiše koliko je brojeva uneseno prije tog broja.

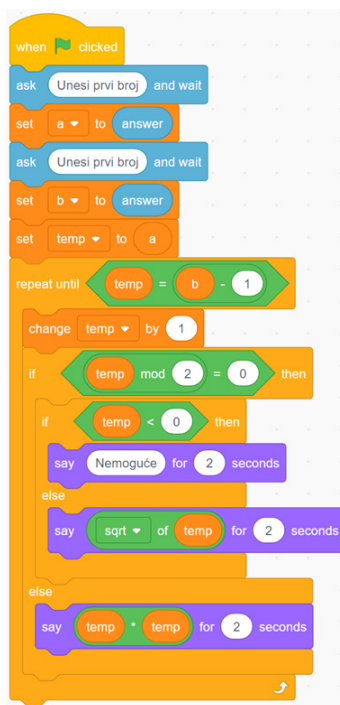
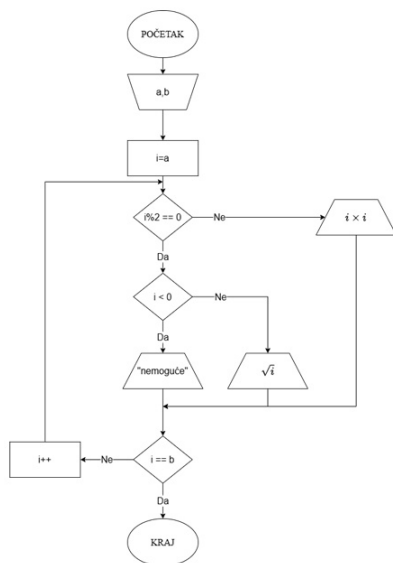
**Rješenje:**



◇ **Zadatak 3.1.4** Korisnik unosi dva broja (pretpostavka je da je prvi broj manji, pa nije potrebna provjera). Potrebno je za sve brojeve između ta dva broja uraditi sljedeće:

ako je broj neparan, kvadrirati ga i ispisati rezultat,  
 ako je broj paran, korjenovati ga i ispisati rezultat,  
 ako je paran broj negativan ispisati "nemoguće".

**Rješenje:**



## 3.2 Zadaci za vježbu

♣ **Zadatak 3.2.1** Korisnik unosi obim kruga. Potrebno je izračunati i ispisati poluprečnik i površinu kruga.



♣ **Zadatak 3.2.2** Korisnik unosi broj. Ako uneseni broj nije prirodan, ispisati odgovarajuću poruku. U suprotnom, ako je uneseni broj neparan, ispisati njegovu recipročnu vrijednost, a ako je paran ispisati kvadrat njegovog prethodnika.

♣ **Zadatak 3.2.3** Korisnik unosi prirodan broj  $n$ . Izračunati i ispisati sumu prirodnih brojeva u intervalu od 1 do  $n$  koji su djeljivi sa zbirom svojih cifara (npr. 234 je djeljiv zbirom svojih cifara  $2 + 3 + 4 = 9$ ).



## 4 Petlje, skripte

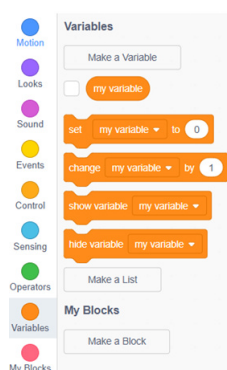
Sada želimo da napravimo to da se neprijateljske letjelice kreću.

Klasika je da te letjelice trebaju da idu desno - lijevo, da u momentu kada dodirnu desni rub ekrana pomjere se prema dole i da nastave da se kreću lijevo, odnosno da kada dodirnu lijevi rub ekrana pomjere se prema dole i da nastave da se kreću desno.

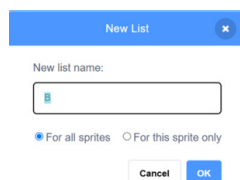
Mi imamo 50 neprijateljskih letjelica. Svaka od njih treba da zna da u jednom momentu treba da se pomjeri prema dole i da nastavi da se kreće u suprotnom smjeru od onoga u kojem se kretala prije pomjeranja prema dole. Dakle, za svaku od njih nam treba informacija, tj. neki pokazivač koji treba da joj naredi kada pomjeranje prema dole i promjena smjera treba da se odigraju.

Ukoliko bi bila riječ o jednoj neprijateljskoj letjelici, onda bi bilo dovoljno da definišemo neku promjenjivu kojoj bi eventualno mijenjali vrijednost između 0 i 1 (ili tačno i netačno), na principu uključi - isključi, i koja bi na osnovu te promjene diktirala ponašanje same letjelice. Ipak, sada imamo 50 letjelica čije ponašanje treba da regulišemo. Naravno, jedna od opcija bi bila da definišemo 50 prekidača za svaku od letjelica, što može biti zamorno u smislu ponavljanja jednog te istog posla, i zahtijevno u smislu manipulacije prekidačima. Zbog jednostavnosti manipulacije, definisat ćemo niz, tj. listu od 50 elemenata. Prvi element te liste označavat će prekidač prve letjelice, drugi element liste označavat će prekidač druge letjelice, itd.

Da bi napravili listu, kliknemo lijevo na naredbe, tj. na narandžasti krug (Variables), i uočimo na dnu tih naredbi opciju: Make a List



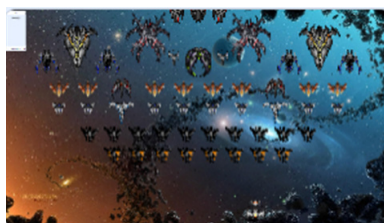
Kliknemo na: Make a List. Dobijemo prozor za pravljenje nove liste



Našoj listi ćemo dati ime "B". Znači, upamtiti da lista B služi za kretanje Sprite-ova, preciznije za kretanje neprijateljskih letjelica. Između dvije opcije: For all sprites i For this sprite only, biramo prvu jer želimo da podaci u navedenoj listi budu dostupni svuda u igri (ako nam zatrebaju). Nakon klika na OK dobijemo listu B lijevo



Pored nje je kvakica, što znači da je lista B uključena u smislu da bude vidljiva na ekranu (uočiti na ekranu igre gore - desno da je na njemu vidljiva lista B na nekoj lokaciji, obično u vrhu ekrana lijevo)



Ako kliknemo na treću opciju opcija



tj. na Full Screen Control, vidjet ćemo bolje gdje se nalazi lista B na našem ekranu, i vidjet ćemo da je ona prazna



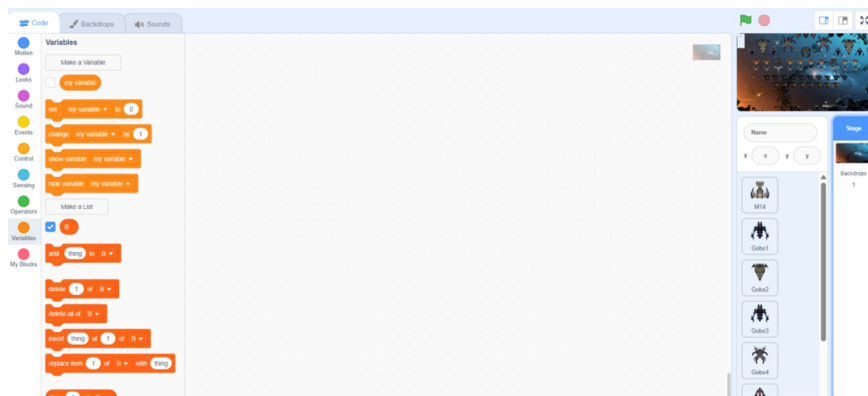
Naravno, nismo još ni naredili da bude popunjena nekim elementima.

Rezon kretanja neprijateljskih letjelica bi bio taj da broj 1 označava kretanje desno, a da broj 0 označava kretanje lijevo.

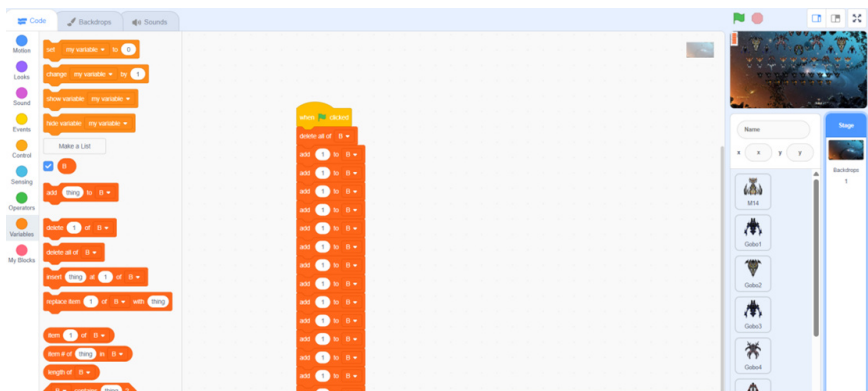
Mi želimo da se naše letjelice inicijalno kreću prvo desno, što znači da našu listu B prilikom pokretanja (klikom na zelenu zastavicu) trebamo prvo počistiti (u slučaju da u njoj imamo već nešto), a zatim u nju dodati 50 jedinica (po jedna za svaku od neprijateljskih letjelica redom). Na taj način sugerišemo da kretanje treba da bude prvo u smjeru desno. Naravno, nakon dodira neke letjelice sa desnim rubom ekrana, treba da pomjerimo tu letjelicu prema dole (za određenu vrijednost), i da njen odgovarajući broj 1 u listi B zamijenimo brojem 0. Jasno, nakon dodira te iste letjelice sa lijevim rubom ekrana, treba da pomjerimo letjelicu prema dole i da odgovarajući broj 0 u listi B promijenimo na 1, itd. Na ovaj način smo opisali algoritam kretanja, tj. dali pseudo - kod kojeg je još potrebno implementirati.

Čišćenje liste B i dodavanje 50 jedinica je postupak koji nema veze sa nekom konkretnom letjelicom, nema veze sa nekim konkretnim Sprite-om, dakle taj postupak je neutralan pa odgovarajući kod možemo unijeti na neutralnu lokaciju. Ranije smo već govorili da je lokacija pozadina na neki način neutralna lokacija.

Dakle, kliknemo desno na jedinu pozadinu koju imamo (iznad Backdrops), i postavimo se na njen kod. Dobijemo ekran



Unesemo kod



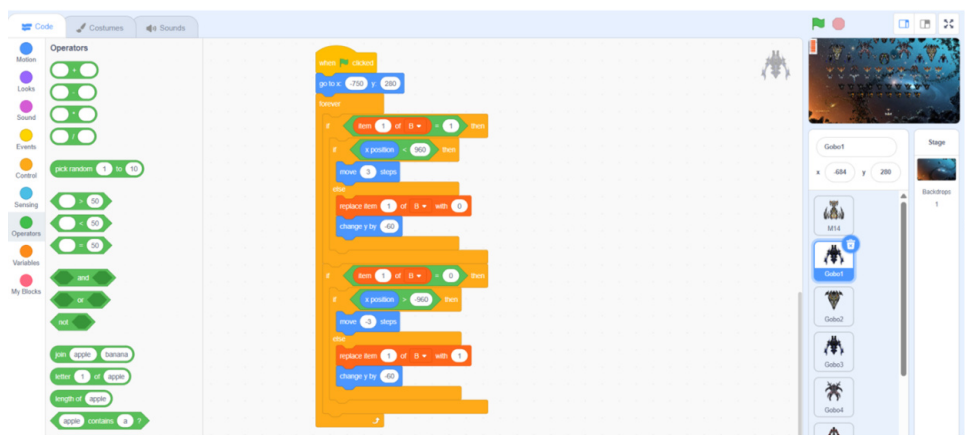
Naravno, ovdje treba 50 puta da se ponovi naredba: add 1 to B. Radi jednostavnosti, dovoljno je tu naredbu prenijeti iz liste naredbi u kod a onda umnožiti dovoljan broj puta. Možemo je umnožiti prvo jednaput, pa onda dvije te naredbe umnožiti, pa zatim 4, pa 8, itd. Procijeniti najbolji sistem rada.

Možemo dosnimiti sada ovo što smo uradili odlaskom na File + Save to your computer (dobijemo verziju Gal (1).sb3). Ako uvećamo ekran pomoću Full Screen Control i pokrenemo igru klikom na zelenu zastavicu, vidjet ćemo u gornjem lijevom uglu ekrana da je naša lista B popunjena sa 50 jedinica



Jasno, svakim narednim klikom na zelenu zastavicu, elementi liste B se pobrišu a onda se u nju doda 50 jedinica. Ovo je potrebno, jer nakon što iskodiramo kretanje letjelica, svaka od njih će u nekom momentu (prije zaustavljanja igre) da bude na nekoj poziciji do koje je došla kretanjem ili lijevo ili desno, što znači da će u listi B u tom momentu da se nalaze negdje nule a negdje jedinice. Tako, da bi osigurali da početak igre zaista bude početak, onda je, kako smo već rekli, potrebno osigurati da se elementi liste B svaki put prilikom pokretanja postave da svi budu jednaki 1.

Postavimo se sada na prvog Gobo-a i dodajmo sljedeći kod za njegovo kretanje



Uočimo ovdje nekoliko bitnih elemenata.

Prvo, sjetimo se da smo ranije naveli precizne startne lokacije svih neprijateljskih letjelica. Prva od njih je imala poziciju (-750,280). Ovdje smo kao prvu naredbu onda zadali postavljanje prve letjelice na poziciju (-750,280). Jasno, svakim pokretanjem programa, prva letjelica će prvo da bude postavljena na zadatu lokaciju. Znači, ma gdje se ona nalazila nakon što kliknemo zelenu zastavicu ponovo, ona će da se postavi na startnu lokaciju.

Ostatak koda odnosi se na kretanje letjelice. Kretanje mora da se odvija beskonačno pa imamo jednu beskonačnu "forever" petlju. U njoj imamo dvije If naredbe. Prvi If se pita da li je prvi element liste B jednak 1. Drugi If se pita da li je prvi element liste B jednak 0. Drugim riječima ovi If-ovi se pitaju da li treba letjelica u tom momentu da se kreće desno ili lijevo. Ako je to desno (broj 1), i ako nije došla do desnog ruba ( $x$  pozicija je manja od 960), onda neka se pomjeri udesno 3 koraka (3 steps). U suprotnom, tj. ako je došla do desnog ruba ili prešla desni rub ( $x$  pozicija je veća ili jednaka 960), onda treba da promijeni smjer (ono što je bilo 1 neka postane 0) i da se pomjeri prema dole za 60 (promijeni  $y$  za -60). Na isti način komentarišemo i drugi If. Ako letjelica treba da se kreće lijevo (broj 0), i ako nije došla do lijevog ruba ( $x$  pozicija je veća od -960), onda neka se pomjeri ulijevo za 3 koraka (-3 steps). U suprotnom, tj. ako je došla do lijevog ruba ili prešla lijevi rub ( $x$  pozicija je manja ili jednaka -960), onda treba da promijeni smjer (ono što je bilo 0 neka postane 1) i da se pomjeri prema dole za 60 (promijeni  $y$  za -60).

Na ovaj način smo opisali kretanje prve neprijateljske letjelice. Možemo pokrenuti kod pa vidjeti da se navedeno kretanje odvija onako kako želimo.

Sada je potrebno dodati kodove na ostale neprijateljske letjelice.

Jasno, njihovo kretanje treba da se odvija po istom algoritmu, pa je dovoljno "prenijeti" kod sa letjelice Gobo1 na ostale letjelice (kliknemo na vrh koda i držimo pritisnutu tipku miša dok prenosimo kod na ikonu letjelice na koju želimo da se kod prenese). Uradimo to za sve letjelice. Doslavimo program.

Ono što u prenesenim kodovima moramo promijeniti jesu startne pozicije letjelica Gobo2 - Gobo50.

Kako smo već rekli, startne pozicije svih letjelica su sljedeće

```
(-750,280), (-600,350), (-450,280), (-249,400), (0,600), (249,400), (450,280), (600,350), (750,280)
(-120,300), (0,280), (120,300)
(-690,150), (-546,150), (-381,150), (-216,150), (-72,150), (72,150), (216,150), (381,150), (546,150), (690,150)
(-690,50), (-546,50), (-381,30), (-216,50), (-72,50), (72,50), (216,50), (381,30), (546,50), (690,50)
(-540,-80), (-420,-80), (-300,-80), (-180,-80), (-60,-80), (60,-80), (180,-80), (300,-80), (420,-80), (540,-80)
(-420,-180), (-300,-180), (-180,-180), (-60,-180), (60,-180), (180,-180), (300,-180), (420,-180)]
```



Uradimo navedeno, promijenimo dakle startne pozicije letjelica Gobo2 - Gobo50. Dosnimimo program.

Dodatno što moramo promijeniti jesu brojevi prekidača.

Naime, u kodu za Gobo1 smo koristili broj 1 kao oznaku za njegov prekidač (item 1 of B). Sada, za Gobo2 npr, tamo gdje je stajalo item 1 of B, mora da stoji item 2 of B. Za Gobo3, tamo gdje je stajalo item 1 of B, mora da stoji item 3 of B, itd. Uradimo to. Dosnimimo program.

Pokrenemo program da probamo da li sve radi kako treba. Ako ima bilo kakvih grešaka i odstupanja, korigujemo i ponovo dosnimimo program.

Ovdje moramo nešto dodatno primijetiti. Naime, kretanje letjelica sa ovako navedenim startnim pozicijama se odvija glatko. Nema odstupanja, letjelice ne preklapaju jedna drugu, nakon dodira desnog ili lijevog ruba ekrana ne dolazi do bilo kakvih odstupanja.

S druge strane, uočimo da startne pozicije navedenih letjelica nemaju sve "lijep" izgled. Neke su tipa (-750,280), tj. koordinate su im djeljive sa 10. S druge strane imamo pozicije tipa (-381,150). One ne izgledaju "lijepo". Zbog čega je to tako, i zašto nam je to bitno ? Uočimo još da sva pomjeranja (lijevo ili desno) idu po 3 koraka. Da ne bi prilikom dodira letjelice sa desnim ili lijevim rubom došlo do odstupanja u kretanju, potrebno je da sve  $x$  koordinate početnih pozicija budu djeljive sa 3 (dakle sa brojem koraka za koje pomjeramo letjelicu). Dakle, broj -750 je djeljiv sa 3. Isto tako je i broj -381 djeljiv sa 3, ali i sve ostale  $x$  koordinate početnih pozicija. Možemo čisto radi testa promijeniti neku od  $x$  koordinata tako da ne bude djeljiva sa 3, i prilikom pokretanja programa vrlo brzo uočiti neželjeno pomjeranje ulijevo ili udesno posmatrane letjelice.

Idemo sada napraviti dugme, tj. Sprite za Start.

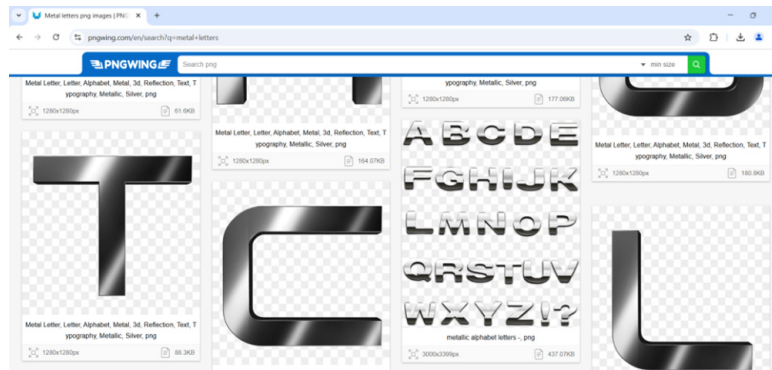
Moramo ga prvo napraviti fizički a onda i iskodirati.

Jasno, možemo koristiti neku od gotovih transparentnih slika za start koju nađemo. Ipak, mi ćemo praviti različite riječi koje će nam trebati unutar igre, pa nije loše da takve riječi budu uniformne u smislu korištenog fonta. Zbog toga je dobro naći alfabete koji nam odgovaraju i iz kojih, isjecanjem slova, možemo generisati riječi koje su nam potrebne.

Potrebna su nam transparentna slova koja možemo naći na linku za transparentne slike koji smo već koristili

<https://www.pngwing.com/en/search?q=laser>

Ako na datom linku potražimo metal letters kao slova metalnog oblika i izgleda, neće nam trebati dugo da ih uočimo



Sačuvamo ih sebi na sigurnu lokaciju pod imenom: metal.

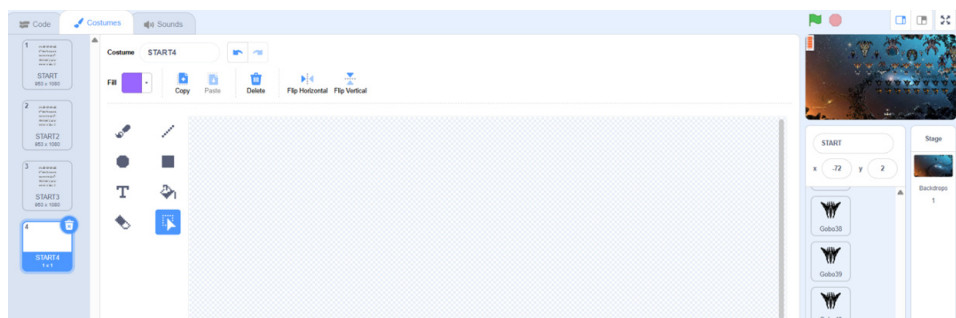
Idemo na Upload Sprite i tu ih upload-ujemo.

Odemo na kostime da ih vidimo.

Mi smo do sada sve elemente koje smo u igri koristili, odmah konvertovali u vektorski oblik. U ovom slučaju to nećemo raditi odmah, nego nakon isjecanja. Naime, da bismo iz datog alfabeta mogli isjeći slova, potrebno je da alfabet bude u Bitmap obliku.

Ovom Sprite-u damo ime START, a isto takvo ime damo i kostimu na lijevoj strani.

Na lijevoj strani umnožimo kostim nekoliko puta, čisto da ga imamo u rezervi ako nešto poremetimo sa prvim od njih. Takođe, umnoženi kostimi nam trebaju i za druge svrhe. Posljednji od njih počistimo odlaskom na Delete u gornjem lijevom uglu (da počistimo njegov sadržaj). U taj prazni kostim ćemo smiještati isječena slova



Slova koja nam trebaju za riječ START možemo isijecati sa nekog od prethodnih kostima, npr. prvog (ili drugog ili trećeg ukoliko se prilikom rada sa prvim kostimom dogodi nešto što ne bismo željeli).

Kliknemo na prvi kostim, i pozicioniramo klizačima slova, koja nam trebaju da budu dostupna. Označimo mišem slovo koje nam treba, ali tako da označeni dio ne dodiruje neko od okolnih slova, ali da po mogućnosti ne dodiruje ni rubove slova koje isjecamo (ako je moguće). Označeni dio, tj. slovo "S" kopiramo i zalijepimo na četvrti prazni kostim. Četvrti kostim nakon toga pretvorimo u vektorski oblik da bi preneseno slovo mogli pomjerati (ali i ostala slova koja budu dolazila). Vratimo se isjecanju, kopiranju i lijepljenju sljedećeg slova, itd. Primijetimo da će svako sljedeće slovo koje budemo zalijepili na četvrti kostim odmah da bude u željenom vektorskom obliku. Izvršimo isjecanje, kopiranje i lijepljenje ostalih potrebnih slova. Slova koja se ponavljaju je dovoljno jednom isjeći, kopirati i zalijepiti, a onda ga je dovoljno na četvrtom kostimu umnožiti potrebnim brojem puta (npr. slovo T).

Sada, slova koja su u vektorskom obliku, i koja su zasebne slike na četvrtom kostimu je potrebno poredati (poravnati) da čine riječ START. Ukoliko je potrebno neka od njih povećati ili smanjiti, onda to možemo uraditi njihovim razvlačenjem ili stezanjem na licu mjesta. Nekada i kada se čini da je sve poravnato kako treba, to u stvarnosti nije tako. Nekada se neravnine uoče tek nakon pokretanja igre, pa se može desiti da navedeni postupak moramo ponoviti nekoliko puta da bi sve bilo onako kako to želimo.

Ukoliko smo zaključili da su slova poravnata, i da su razmaci među njima adekvatni, onda je dovoljno na četvrtom kostimu otići na opciju Convert to Bitmap, a neposredno nakon toga i na opciju Convert to Vector, da bi date slike prvo pretvorili u bitmap oblik, koji onda čini da sva slova zajedno čine jednu sliku, dok drugi klik čini da ta novodobijena slika bude u vektorskom obliku. Ako nakon obavljenih klikova postavimo kursor miša na START, vidjet ćemo da se on ponaša kao jedna slika koju sada možemo pomjerati. Pomjerimo je tako da bude centrirana (da se crni plus + u pozadini podudara sa plavim plus-om + naše slike).

Ostali, tj. prethodni kostimi nam više nisu potrebni, pa ih pobrišemo. Ostavljamo dakle samo kostim koji sadrži riječ START. Ukoliko ime preostalog kostima nije START, onda ga postavimo da bude START. Dosnimimo igru.

Opisani postupak pravljenja riječi ćemo koristiti i u slučaju riječi koje nam budu potrebne, pa ga je potrebno znati. Riječ START vizuelno je prevelika pa postavimo njen Size da bude jednak 20 (jasno, u slučaju potrebe možemo ga dodatno promijeniti). Riječ START se na ekranu pojavljuje negdje, obično na sredini ekrana. Ona nama kao pojava ne smeta, jer ne utiče na kretanje letjelica. Ipak, ukoliko nam vizuelno smeta, možemo je na malom ekranu (u gornjem desnom uglu) pomjeriti manuelno na neku drugu lokaciju koju za sada smatramo adekvatnijom.

Potrebni su nam sada i neki dodatni zvukovi. Link za zvukove je sljedeći

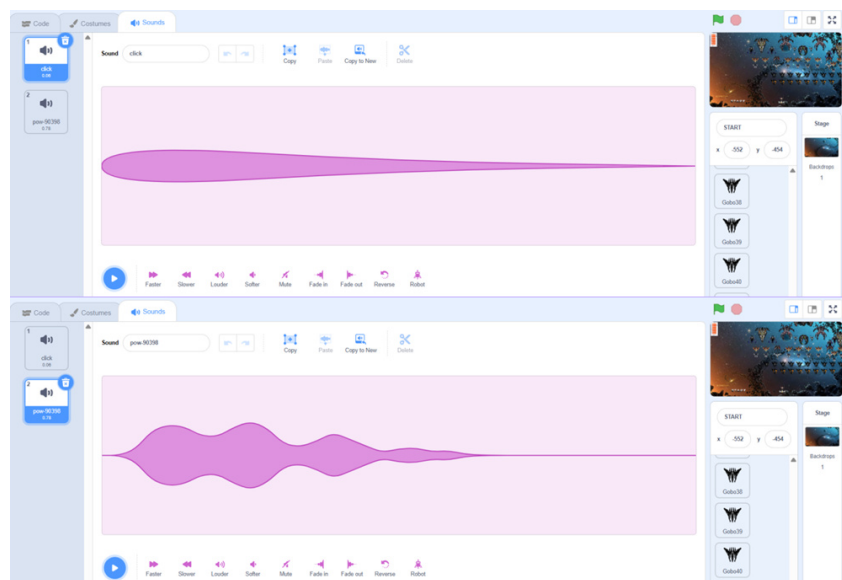
<https://pixabay.com/sound-effects/search/energy%20rotation/?pagi=3>

Ovdje je dovoljno u pretraživač u gornjem lijevom uglu ukucati na engleskom riječ koja nam treba, tj. ime zvuka koji nam otprilike treba, pa izvršiti pretragu. Neki zvukovi su duži po trajanju. Nekad će nam trebati neki duži a nekada neki kraći zvuk.

Prvi zvuk koji nam treba je zvuk klika, ali ne klika nastalog pritiskom miša na neku tipku ili dugme, nego zvuk koji se često pojavljuje prelaskom samo miša preko nekog dugmeta. To je obično neki blagi zvuk klika. Pri tom prelasku se obično veličina dugmeta promijeni. Obično se nailaskom kursora miša na dugme, uz proizvodnju zvuka klika, i samo dugme blago poveća, a onda se, nakon što se kursor miša skloni sa pomenutog dugmeta, ono smanji na polaznu veličinu. Ovaj efekat želimo da iskodiramo prilikom prelaska kursora miša preko riječi START.

Drugi zvuk koji nam treba je zvuk koji treba da se proizvede nakon što zaista kliknemo mišem na riječ START. Taj zvuk treba da je ozbiljniji, blago jači, ali i po mogućnosti originalniji. Naime, pravimo svoju igru, koristimo naše objekte onoliko koliko možemo.

Navedene zvukove ćemo zvati: click i pow-90398

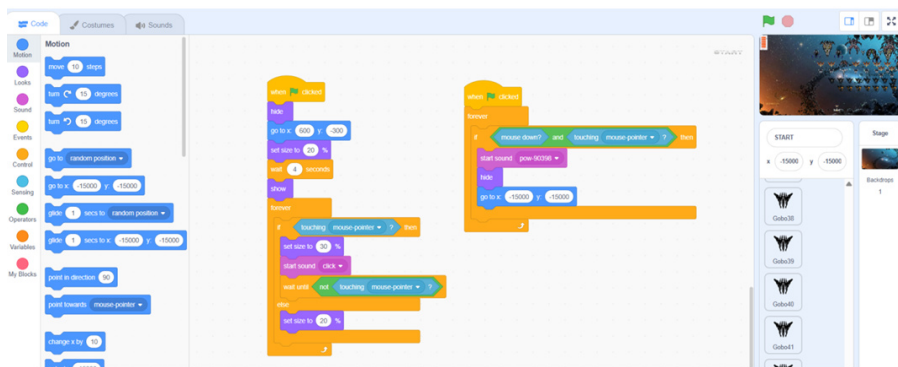


Na posljednjim slikama se mogu uočiti opcije za ubrzavanje i usporavanje zvuka, opcije za pojačavanje i smanjivanje tona, opcije koje nam omogućavaju da zvuk ima blagi početak i blagi kraj (Fade in i Fade out). Opcije Fade in i Fade out znaju biti korisne kod zvukova čiji su počeci i završeci grubi ili prejaki, te kod zvukova kod kojih mi vršimo isijecanje. Da bi eliminisali grube prijelaze koristimo pomenute opcije. Neke zvukove koje download-ujemo je potrebno pojačati a neke stišati, neke produžiti a neke ubrzati. Nekada nam odgovara obrnuti slijed zvuka (opcija Reverse). Takođe, ako se poslužimo opcijom Robot nekada ćemo dobiti željenu modifikaciju tona. Ove napomene su bitne jer najčešće zvuk koji download-ujemo sa interneta neće po defaultu da zadovoljava naše potrebe. Najčešće će taj zvuk da bude modifikovan od strane nas višestrukom upotrebom navedenih opcija.

Navedene zvukove: click i pow-90398 upload-ujemo.

Idemo sada unijeti kod na Sprite START.

Ovaj kod ćemo kasnije, kada dodamo nove elemente u igru, djelimično obogatiti. Dakle, kod koji unosimo sada možemo smatrati pripremnim kodom koji će da bude potpun kasnije (obavljati će sve predviđene funkcije u nastavku)



Ovdje imamo dva koda.

Dugme START ne treba da bude vidljivo odmah (tako želimo), nego treba da se pojavi nekoliko trenutaka nakon pokretanja igre. Znači, klikom na zelenu zastavicu, sakrijemo Sprite START, i postavimo ga na lokaciju (600,-300) (dakle, iako je START nevidljiv on će biti postavljen na zadatu lokaciju). Nakon toga mu postavimo veličinu na 20. Sjetimo se da smo mi već postavili veličinu za START na 20. Ovo dodatno

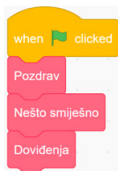
postavljanje činimo da se dodatno osiguramo (nije loše na početku postaviti stvari ponovo na željenu veličinu). Zatim, sačekamo 4 sekunde (to je vrijeme koje želimo da se pričeka prije pojave dugmeta START), i konačno prikažemo dugme. Nakon toga, pokrenemo beskonačnu petlju koja treba da obavlja posao koji smo opisali ranije. Naime, ako dugme, tj. Sprite START dodirne kursor miša, ono treba da se poveća na veličinu 30 i da proizvede zvuk click. Teba da se zadrži u tom obliku (30) sve do momenta kada ono ne bude dodirivalo kursor miša više. S druge strane, ako dugme START ne dodiruje kursor miša, onda treba da bude postavljeno na veličinu 20. Ovakav kod nam omogućava da prelaskom miša preko dugmeta START dobijamo zvuk click, te da se pri tom prelasku START povećava i smanjuje na veličine 30 i 20, redom u zavisnosti od toga da li kursor miša dodiruje START ili ne.

Drugi kod treba da nam omogućava ulazak u igru (nešto što ćemo napraviti u nastavku). Drugim riječima, ako dugme tj. Sprite START dodiruje kursor miša, pri čemu smo izvršili klik miša (mouse down), onda treba da se proizvede zvuk pow-90398 (pri tome treba da uđemo u igru - ovo programiramo kako smo već rekli kasnije), zatim treba dugme START da sakrijemo, i da ga "odložimo" na neku udaljenu lokaciju na kojoj nam neće smetati (odlaganje objekata ćemo stalno činiti po potrebi). To je npr. lokacija (-15000,-15000). Može to biti i neka druga udaljena lokacija. Jasno, svakim ponovnim pokretanjem igre, START će da bude postavljen na polaznu lokaciju (600,-300). Napomenimo da klik na START može da se desi u svakome trenutku, pa zbog toga upravo opisani kod mora da leži unutar beskonačne petlje.

Sačuvamo izmjene i probamo igru.

## 4.1 Riješeni zadaci

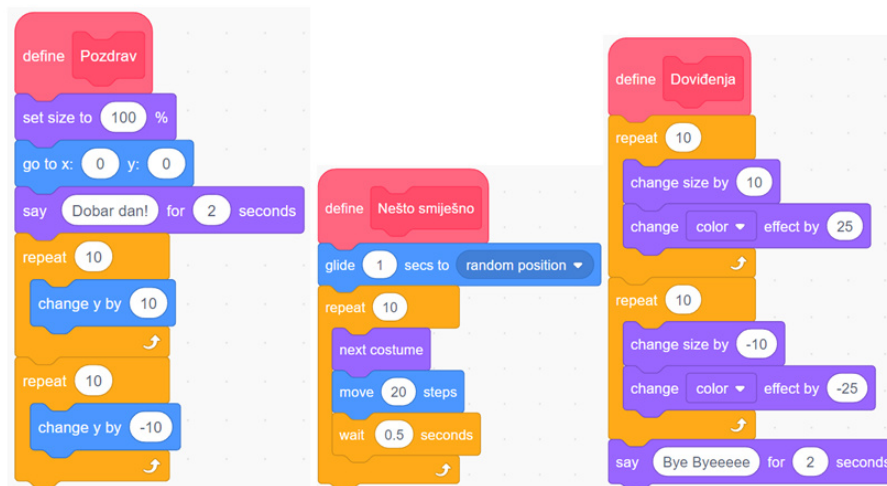
◇ **Zadatak 4.1.1** Dati su sljedeći blokovi:



Osmisliti i definisati sadržaj svakog bloka tako da oni rade nešto smisleno, po slobodnom izboru (npr. ispis poruka, upotreba zvuka, promjena izgleda lika, pokret ili

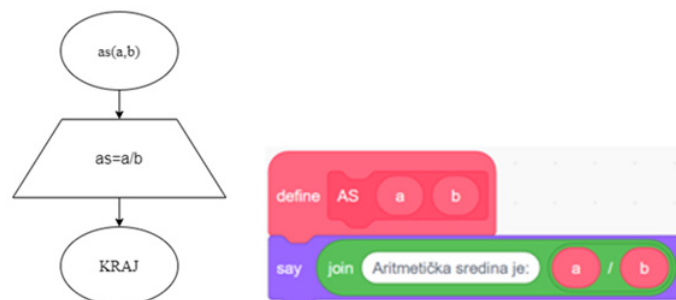
kombinacija više akcija).

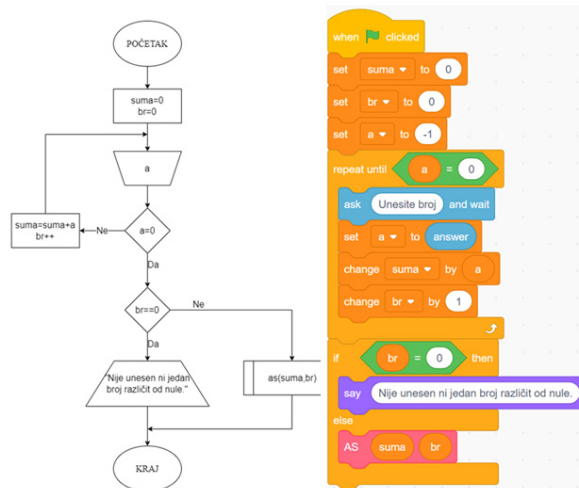
### Rješenje:



◇ **Zadatak 4.1.2** Korisnik unosi brojeve sve dok ne unese broj 0. Potrebno je izračunati aritmetičku sredinu unesenih brojeva (koristiti funkciju).

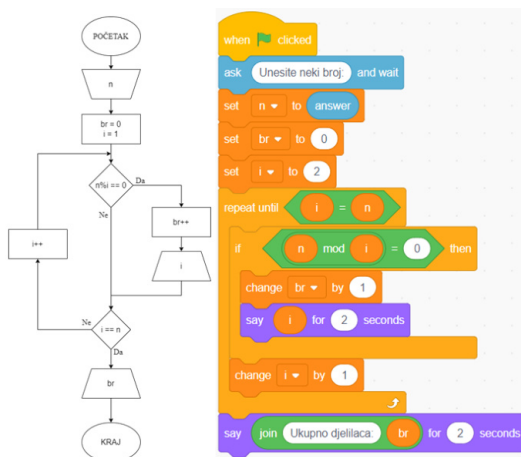
### Rješenje:





◇ **Zadatak 4.1.3** Korisnik unosi jedan broj. Potrebno je prebrojati i ispisati sve njegove djelioce.

**Rješenje:**

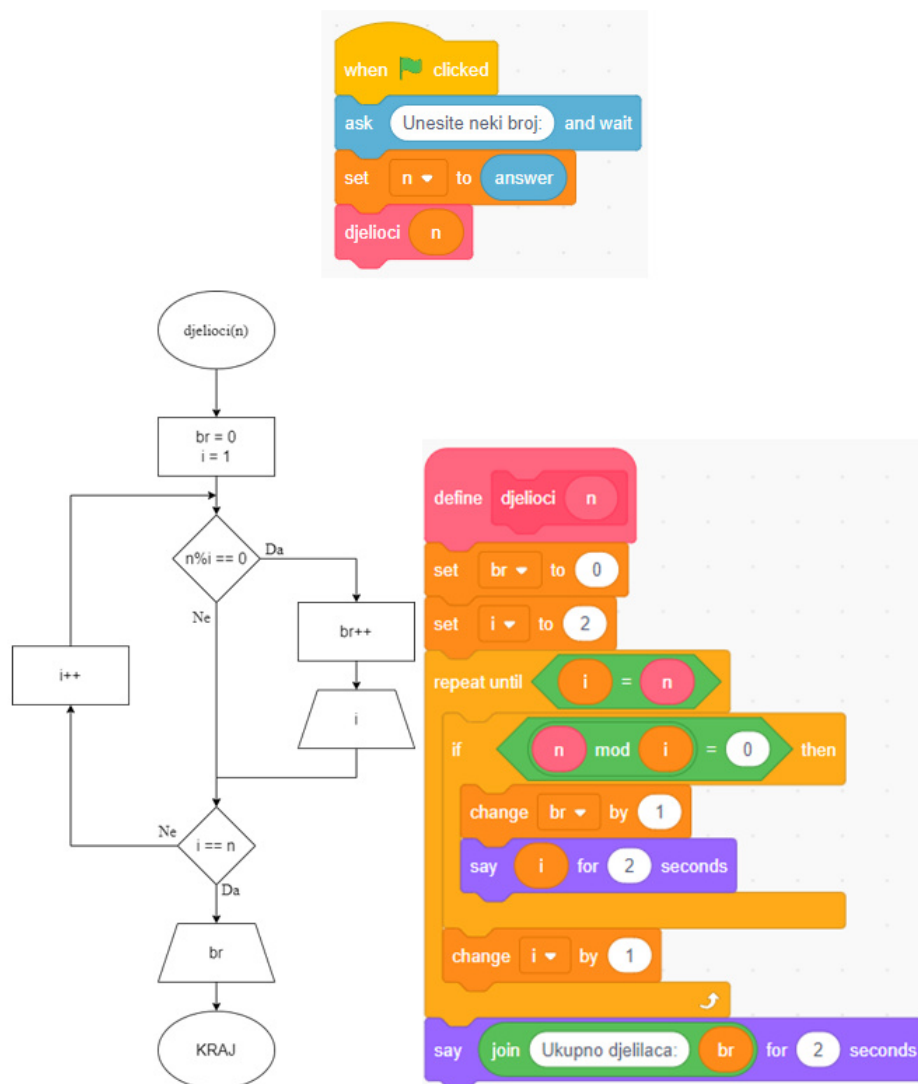


Zadatak smo mogli postaviti i riješiti nešto drugačije:



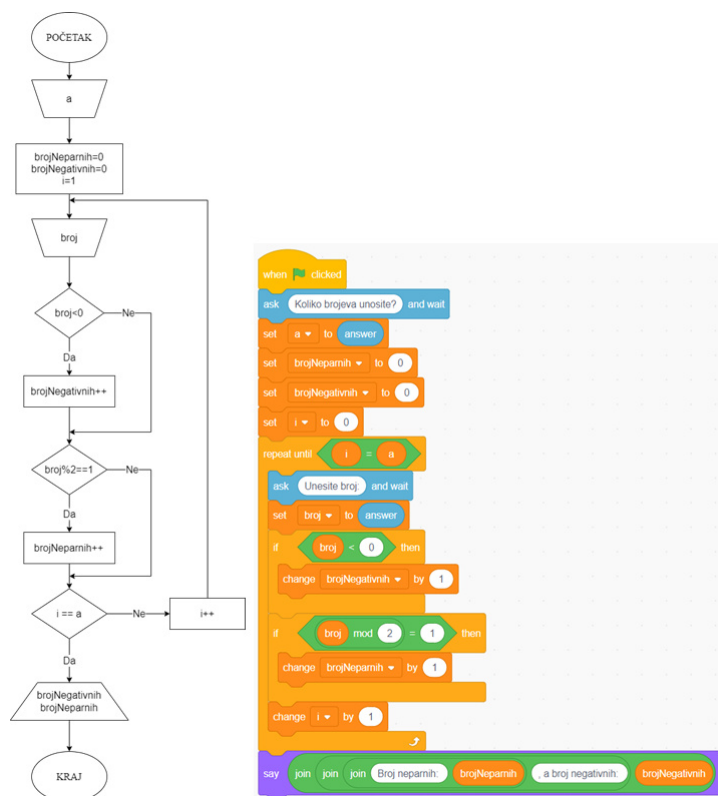
Korisnik unosi jedan broj. Kreirati blok "djelioci" koji prima jedan parametar  $n$ . Blok definisati tako da prebrojava i ispisuje sve djelioce parametra  $n$ .

Eventualno rješenje bi sada imalo oblik



◇ **Zadatak 4.1.4** Korisnik unosi jedan broj  $i$  i zatim unosi toliko brojeva. Potrebno je prebrojati koliko je korisnik unio negativnih i koliko neparnih brojeva.

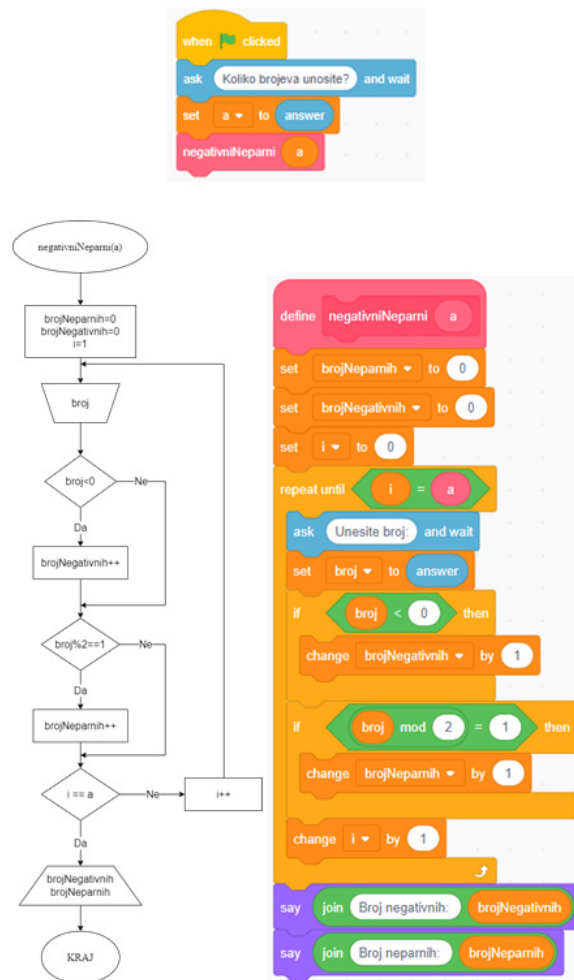
**Rješenje:**



Jasno je da smo i ovaj zadatak mogli postaviti i riješiti drugačije:

Korisnik na početku programa unosi koliko brojeva želi unijeti. Potrebno je kreirati vlastiti blok "negativniNeparni" koji prima jedan parametar  $a$ . Blok definisati tako da traži unos  $a$  brojeva. Nakon toga je potrebno prebrojati koliko je korisnik unio negativnih, a koliko neparnih brojeva.

Moguće rješenje je sljedeće



## 4.2 Zadaci za vježbu

♣ **Zadatak 4.2.1** Korisnik unosi temperaturu u stepenima Celzijusa. Potrebno je izračunati i ispisati tu temperaturu izraženu u:

$$\text{Kelvinima: } K = C + 273.15$$

Fahrenheitima:  $F = C \times 1.8 + 32$

Potrebno je kreirati vlastiti blok pod nazivom "pretvoriTemperaturu" koji prima temperaturu u Celzijusima kao ulaz, računa odgovarajuće vrijednosti i ispisuje rezultate.

♣ **Zadatak 4.2.2** Korisnik (student) unosi jedan broj. Potrebno je uporediti uneseni broj sa brojem indeksa studenta, i u zavisnosti od rezultata poređenja, uraditi sljedeće:

ako je broj veći od broja indeksa, pomnožiti ga sa dužinom svoga imena i ispisati rezultat,

ako je broj manji od broja indeksa, oduzeti od njega dužinu svoga prezimena i ispisati rezultat,

ako je broj jednak broju indeksa, ispisati poruku: "Čestitamo, to je tačan broj !".

Potrebno je kreirati vlastiti blok pod nazivom "obradiBroj" koji prima uneseni broj kao ulaz, provodi poređenje i izvršava odgovarajuću akciju prema pravilima datim zadatkom.

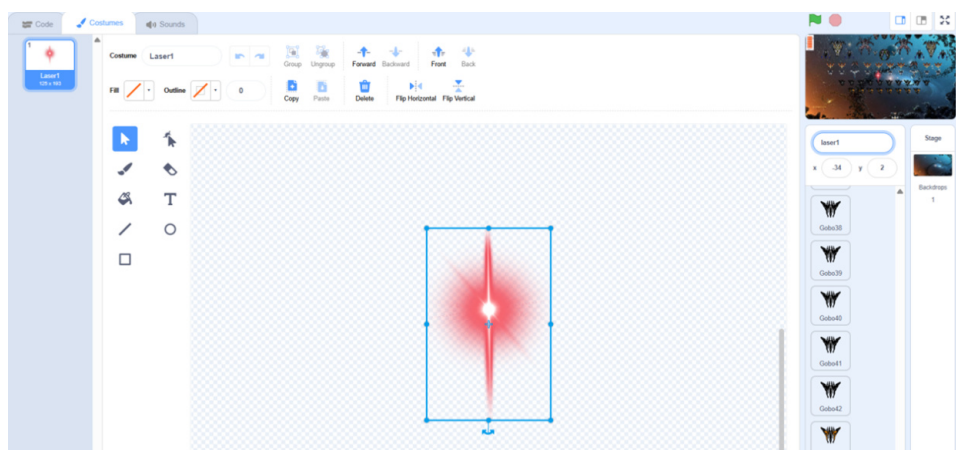
♣ **Zadatak 4.2.3** Korisnik unosi brojeve sve dok ne unese broj koji je jednak proizvodu dužina njegovog imena i prezimena (za ime Meho Mehic riječ je o proizvodu  $4 \times 5 = 20$ ). Potrebno je izračunati i ispisati koliko je brojeva uneseno i koja im je suma, bez uzimanja u obzir posljednjeg unesenog broja.

## 5 Liste objekata

Sada ćemo napraviti laser, tj. oružje koje naša letjelica koristi u borbi protiv 50 neprijatelja.

Primijetimo da laser ne možemo praviti nezavisno od pravljenja smrti neprijatelja. Dakle, pogodak laserom nekog od 50 neprijatelja treba da rezultira njegovom eliminacijom. Naravno, na nama je sada da odredimo koliko ćemo ići u dubinu po pitanju detalja opisanog procesa. Prvo, bitno je da napravimo kostur, tj. osnovnu konstrukciju, a onda da tu konstrukciju usavršavamo i dograđujemo.

Za laser ćemo koristiti Sprite



Ovaj, kao i dosadašnje Sprite-ove koje smo koristili, možemo naći na već korištenoj stranici sa transparentnim slikama

<https://www.pngwing.com/en/search?q=laser>

Naravno, potrebno je malo pretrage. Osim toga, Sprite koji nađemo ne mora značiti da je u istom formatu i obliku kao ovaj koji koristimo. Naime, obično Sprite koji inicijalno nađemo, ne zadovoljava naše potrebe, pa ga je potrebno dodatno modifikovati. Modifikacija se obično odnosi na veličinu Sprite-a, prostor koji zauzima na ekranu u smislu traga koji ostavlja za sobom (plavi kvadratić oko Sprite-a koji se pojavljuje na ekranu nakon klika na Sprite). Trag ne treba ni da je preveliki niti premali.

Obično su tragovi preveliki što onda znači da se takvi Sprite-ovi mogu lahko sudariti sa drugim Sprite-ovima, a da nam to nije u cilju i interesu. Dakle, Sprite-ove koji se koriste za oružja obično moramo modifikovati nakon što ih nađemo na internetu. Osim toga, često se dešava da Sprite-ovi koje nađemo na internetu nisu centrirani, tj. ili su pomjereni malo ulijevo ili udesno, što onda zahtijeva dodatno podešavanje.

Naš Sprite ćemo zvati laser1, a ime kostima ćemo zvati Laser1.

Prilikom ispaljivanja lasera iz naše letjelice treba da se čuje zvuk. Zvuk koji ćemo koristiti je sljedeći (ovo je zvuk koji upload-ujemo dok smo na Sprite-u laser1)



Zvat ćemo ga Z2.

Ovo je neki zvuk lasera ili nečega što zvuči kao laser.

Kao i u slučaju Sprite-ova, i zvukove nalazimo na već korištenoj stranici

<https://pixabay.com/sound-effects/search/energy%20rotation/?pagi=3>

Dovoljno je u pretraživač navedene stranice ukucati: laser, beam, energy, blast, i sl. Dakle, dovoljno je ukucati neki pojam koji bi asocirao na ono što nam treba. Zvukovi kao i Sprite-ovi ne moraju da odgovaraju našim potrebama. Neki su predugi, neki prekratki. Neki su prejaki neki pretihi. Obično je potrebno dodatno podešavanje.

Sada, prije nego počnemo kodirati laser1, trebamo da znamo šta uopšte hoćemo.

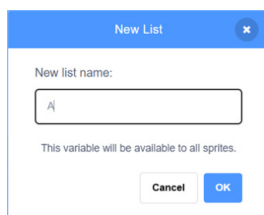
Treba nam nešto što će označavati život odnosno smrt svakog od 50 neprijatelja. Dakle, slično kao kretanje neprijatelja. Kretanje bilo kojeg od njih udesno smo označavali sa 1, dok smo kretanje u suprotnom smjeru, tj. ulijevo označavali sa 0. Naime, za kretanje smo definisali listu, koju smo nazvali B, i koja je sadržavala 50 elemenata, tj. item-a. Tamo, item 1 je korišten za kretanje prve letjelice, tj. za kretanje Gobo1.

Ako je item 1 liste B bio jednak 1, Gobo1 se kretao udesno. S druge strane, ako je item 1 liste B bio jednak 0, onda se Gobo1 kretao ulijevo. Isto je važiolo i za ostale item-e liste B. Tako, ako je item 23 liste B bio jednak 1, Gobo23 se kretao udesno, itd. Ove kretnje za Gobo1-Gobo50 su jasno bile nezavisne jedna od druge. Sada nam je potrebna slična logika za život odnosno smrt svakog od Gobo1-Gobo50.

Definisat ćemo listu A da ima 50 elemenata, tj. itema.

Svaki od 50 elemenata liste A će biti jednak ili 1 ili 0, i na taj način će označavati da li je odgovarajući Gobo živ ili mrtav. Tako, ako item 43 liste A bude jednak 1, to će značiti da je Gobo43 živ. S druge strane, činjenica da je item 43 liste A jednak 0, će značiti da je Gobo43 mrtav.

Na neutralnoj lokaciji, tj. na podlogama već imamo kod koji se odnosi na listu B. Nakon pokretanja igre, prvo pobrišemo sve elemente liste B, a onda dodamo 50 jedinica. Isto ćemo uraditi i sa listom A. Dakle, prvo definišemo listu A



i onda na podloge unesemo kod



Potrebno je 50 puta da ponovimo naredbu: add 1 to A.

Znači, u startu, smatramo da su svih 50 neprijatelja živi. Zbog toga inicijalno stavljamo da su svi elementi liste A jednaki 1. U momentu kada eliminišemo nekog od neprijatelja, npr. Gobo43, u tom momentu odgovarajući element liste A, tj. item

43, postavljamo da bude jednak 0. Ukoliko počne nova misija, tj. novi level, odnosno nakon što svi elementi liste A postanu jednaki 0, onda ih ponovo sve postavimo da budu jednaki 1. Dakle, tek onda kada svi Gobo1-Gobo50 budu eliminisani, onda ih sve oživimo, itd. Napominjemo da upravo navedeno ćemo raditi postupno, ali pominjemo to samo da bi znali šta je naš plan djelovanja.

Vratimo se sada malo našem oružju, tj. razmotrimo neke osobine vezane za laser1.

Laser ćemo aktivirati pritiskom na tipku Space.

Ukoliko pritisnemo tipku Space, laser treba da se pojavi na vrhu naše letjelice, bez obzira gdje se ona po  $x$  osi nalazila, i da se kreće pravolinijski, vertikalno, odozdo prema gore, prema vrhu ekrana (nekom brzinom). Kada laser pređe gornju granicu ekrana, treba da nestane, tj. treba da ga sakrijemo na neku lokaciju (da ne bi nastavio da putuje vertikalno do beskonačnosti). Znači, prije pritiska na tipku Space, laser treba da se nalazi na nekoj lokaciji koju smo unaprijed odredili za njega. To je neka lokacija van vidljivog dijela ekrana. Nakon pritiska na tipku Space, pozivamo laser da se sa te lokacije postavi na vrh naše letjelice, i da putuje prema gore dok ne pređe gornju granicu ekrana. Kada pređe tu granicu, mi taj laser treba ponovo da vratimo na početnu lokaciju koju smo unaprijed odredili za njega. Dok laser putuje prema vrhu ekrana, dodatni pritisci na tipku Space ne treba da proizvedu bilo kakav dodatni efekat. Dakle, ne treba da se pojavi neki dodatni laser ako pritisnemo ponovo tipku Space dok je postojeći laser na ekranu. Čim laser nestane iza gornjeg dijela ekrana, pritisak na tipku Space treba da otpočne opisani postupak nanovo, tj. ponovo treba da se pozove laser sa njegove lokacije, da se postavi na vrh letjelice, te da putuje ka vrhu ekrana.

Primijetimo da ovdje ostaje da se razradi još jedan detalj vezano za laser.

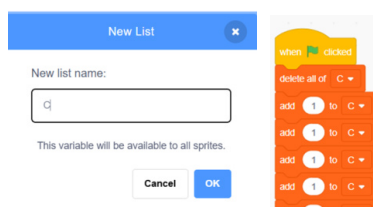
Ukoliko laser na svome putu prema vrhu ekrana dodirne nekog od neprijatelja, taj neprijatelj treba da se skloni, tj. treba da nestane (njegov item liste A treba da se postavi na vrijednost 0), ali isto tako i sam laser treba da nestane. Naime, nakon eliminacije nekog od neprijatelja, ne bi bilo uredu da taj laser nastavi da putuje prema vrhu ekrana, i da na svome putu dodatno eliminiše sve na što naiđe. Dakle, moramo regulisati nestajanje lasera sa ekrana u momentu kada dodirne nekog od neprijatelja.

Ovdje možemo rezonovati na sljedeći način.

Ako laser dodirne npr. Gobo25, onda u kodu za Gobo25 treba da dobijemo informaciju da je laser zaista dodirnuo Gobo25, te da nam ta informacija koristi da ukinemo laser sa ekrana. Isto važi i u slučaju dodira lasera sa bilo kojim od preostalih Gobo-a. Vidimo da treba da imamo onoliko informacija koliko je Gobo-a, dakle 50 njih. Jasno je onda da nam treba neka nova lista od 50 elemenata, čiji će elementi ponovo biti jednaki ili 1 ili 0 (kao i u slučaju već napravljenih lista A i B). Ovoj novoj listi ćemo dati ime C. Kako da koristimo listu C? Prvo, definišemo je na isti način kao i liste A i B, tj. definišemo je i iskodiramo na neutralnoj lokaciji, tj. na podlogama. Uradimo



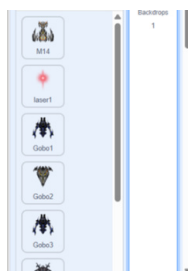
onda odmah taj dio



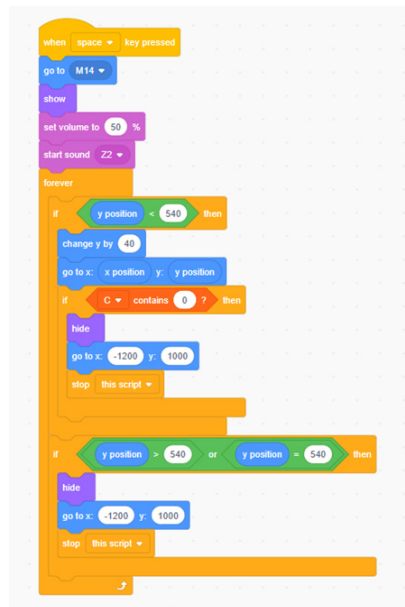
Vidimo da smo sve elemente liste C inicijalno postavili da su jednaki 1. U momentu kada laser dodirne nekog od Gobo-a, odgovarajući element liste C postavimo da bude jednak 0. Znači, u kodu za bilo kojeg od Gobo-a, treba da imamo upit: da li je taj Gobo dodirnuo laser. Ako jeste, onda odgovarajući element liste C postavimo da bude jednak 0. S druge strane, u kodu za laser, treba cijelo vrijeme da se pitamo da li je neki element liste C postao jednak 0. Ako jeste, to znači da je laser dodirnuo nekog od Gobo-a. A ako to jeste slučaj, onda u tom momentu treba da laser sklonimo sa mape, i da ga prebacimo na već unaprijed određenu lokaciju za njega. Drugim riječima, ukoliko je neki element liste C posatao jednak 0, nas ne zanima koji je to element konkretno, nama je bitna samo ta informacija da je neki od njih postao jednak 0, jer ona znači da je laser dodirnuo nekog od Gobo-a, u tom momentu mi treba da sklonimo laser sa mape na način kako smo to već opisali.

Sada smo u poziciji da unesemo prvi kod za laser (koji ćemo onda dorađivati).

Prvo, na listi Sprite-ova, pomjerimo laser prema gore, da bude neposredno ispod naše letjelice, da ne bismo stalno morali scroll dole-gore, i sl



Postavimo se na laser, idemo na kod, i unesemo sljedeći kod (ne zaboraviti povremeno dosnimavati već urađeno)



Ovaj kod radi ono što smo već opisali.

Nakon pritiska na tipku Space, laser dođe do naše letjelice (M14), i pokaže se.

Zvuk Z2 se nekome može činiti prejak, pa postavimo jačinu zvuka na 50, i otpočnemo taj zvuk (sjetimo se razlike između start-zvuk i play-zvuk: ako koristimo start-zvuk, naredbe koje dolaze nakon start-zvuk se odmah izvršavaju dok zvuk nastavlja da se čuje u pozadini, dok ako idemo sa play-zvuk, naredbe koje dolaze nakon play-zvuk moraju da sačekaju da se zvuk prvo odsvira do kraja). Nakon što smo začuli zvuk, laser treba da počne da se kreće prema gore. Znači treba nam jedna beskonačna petlja sa dva upita. Prvi: da li je  $y$ -pozicija lasera manja od 540 (što znači da laser još nije prešao gornju granicu), i drugi: da li je  $y$ -pozicija lasera veća ili jednaka 540 (što znači da je laser prešao gornju granicu). Drugi slučaj je jednostavniji. Naime, ako je laser prešao gornju granicu ekrana, treba da ga sakrijemo, i premjestimo na unaprijed rezervisanu lokaciju za njega. Ovdje kao tu lokaciju koristimo lokaciju (-1200,1000). S druge strane, ako je u pitanju prvi scenario, tj. ako je  $y$ -pozicija lasera manja od 540, što znači da laser i dalje putuje, i treba da putuje po ekranu, onda treba da razmotrimo kako on to putuje, i šta može da se desi na toj putanji. Ovdje ćemo staviti da je brzina njegovog kretanja 40 (kao 40 koraka). Zbog toga prvo naređujemo da se  $y$  poveća za 40 (pomjeramo se prema gore za 40), a onda tražimo da se laser pomjeri na lokaciju

( $x$ -pozicija, $y$ -pozicija). Primijetimo da ovdje  $x$  ne diramo, tj.  $x$  ostavljamo onakvo kakvo smo ga zatekli. Naime, slanjem lasera na letjelicu M14, mi prihvatamo da je  $x$ -pozicija lasera jednaka  $x$ -poziciji letjelice. Tako, u momentu kada ispalimo laser, on će da starta sa  $x$ -pozicije na kojoj se nalazi letjelica M14. Napokon, prilikom kretanja lasera po ekranu, laser može da dodirne nekog od Gobo-a. Rekli samo da ćemo činjenicu da se neki Gobo sudario sa laserom označavati time što ćemo odgovarajući element liste C postaviti da bude jednak 0. Logično je sada da se tokom kretanja lasera prema gore pitamo: da li je neki element liste C postao jednak 0, ili što je isto: da li lista C sadrži 0. Ako to jeste slučaj, onda to znači da je neki Gobo dodirnuo laser. A ako je to tako, onda laser treba da sakrijemo, i da ga sklonimo ponovo na lokaciju  $(-1200,1000)$ . Na ovaj način smo u potpunosti opisali upravo unijeti kod.

Možemo isprobati igru i dosnimiti je.

Maksimiziramo igru, ispalimo laser, i zaustavimo igru dok se laser nalazi negdje na sredini ekrana



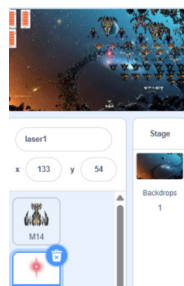
Želimo da vidimo koliki je taj naš laser.

Zanima nas da li je njegova veličina adekvatna, i koliki trag ostavlja za sobom prilikom svoga kretanja.

Što se tiče veličine, dovoljno je vizuelno da pogledamo da li je veličina lasera adekvatna.

Vezano za trag, minimiziramo ekran, i kliknemo nekoliko puta na laser. Vidjet ćemo da se na malom ekranu u gornjem desnom uglu oko lasera pojavljuje plavi

kvadrat, tj. plavi trag



Taj trag ne smije da bude prevelik, jer se pod dodirrom lasera sa nekim od Gobo-a upravo podrazumjeva dodir pomenutog traga. Ako je trag prevelik onda će na svome putu da dodiruje i ono što želimo ali i ono što ne bi trebalo da dodiruje.

Vidimo da bi generalno i veličina lasera ali i trag koji ostavlja za sobom trebali da budu manji. Ukoliko na Size za laser smanjimo njegovu veličinu, onda će i trag koji on ostavlja da se smanji. Postavimo onda Size za laser da bude jednak 30. Vidimo da se veličina lasera značajno smanjila. Takođe, dok je laser još uvijek zaustavljen na sredini ekrana, možemo klikom na laser vidjeti koliko se smanjio trag koji on za sobom ostavlja.

Dosnimimo igru.

Primijetimo da kada ispaljujemo laser, da se ustvari ništa ne dešava u smislu eliminacije neprijateljskih letjelica. Zašto je to tako ? Zato što smrti neprijateljskih letjelica još uvijek nismo iskodirali.

Posvetimo se sada kodovima koji se odnose na eliminaciju neprijateljskih letjelica.

Razmotrimo malo jedan takav događaj, tj. događaj susreta nekog od Gobo-a sa laserom.

Znači, ako neki Gobo didirne laser, onda treba da tog Gobo-a proglasimo mrtvim. Rekli smo da smrt Gobo-a označavamo tako što odgovarajući item liste A postavimo da ima vrijednost jednaku 0 (ako je Gobo živ onda je ta vrijednost jednaka 1). Dakle, ako Gobo dodirne laser, vrijednost odgovarajućeg elementa liste A postavimo na 0. Dalje, ako Gobo dodirne laser, onda i laser treba da sklonimo sa ekrana, tj. da ga sakrijemo i prebacimo na unaprijed rezervisanu lokaciju. Vidjeli smo da postupak sakrivanja lasera i prebacivanja na rezervisanu lokaciju obavlja dio koda kojeg smo unijeli na laser. Za početak tog procesa potrebna nam je informacija da je neki element liste C postao jednak 0. Sada, jasno, ako Gobo dodirne laser onda vrijednost odgovarajućeg elementa liste C treba da postavimo na 0. Takođe, samog Gobo-a koji je sada mrtav

treba da sakrijemo, i napokon da ga prebacimo na unaprijed rezervisanu lokaciju za njega. Ovdje moramo napomenuti da postupak sakrivanja Gobo-a i njegovog sklanjanja na rezervisanu lokaciju može da bude djelimično problematičan. Naime, mi sakrijemo Gobo-a, i prebacimo ga na rezervisanu lokaciju, ali se zbog brzine rada računara, i brzine izvršavanja operacija, desi da Gobo ostane vidljiv na ekranu. Dakle, desi se da se druga između pomenute dvije operacije: sakrivanje i premještanje izvrši prije prve. Da bismo riješili pomenutu problematiku, nije loše postupak sakrivanja Gobo-a ponoviti nekoliko puta prije njegovog premještanja na rezervisanu lokaciju, ali isto tako i nakon tog premještanja. Takođe, između uzastopnih premještanja nije loše ubaciti naredbe da se sačeka po nekoliko djelića sekunde. Nekih 3-4 ponovljenih sakrivanja rezultira u onome što nama tačno treba. Kako već spomenusmo, potrebno je i za Gobo-e (njih 50), odrediti lokacije na koje ih odlažemo nakon njihove eliminacije. Te lokacije možemo koristiti i na samom početku igre, tj. na te lokacije možemo pozicionirati naše Gobo-e na samome startu prije rasporeda na ploču, i onda ih po potrebi sa tih lokacija pozivati, a nakon njihove eliminacije i vraćati. Napomenimo da će te lokacije da igraju bitnu ulogu u nastavku razvoja igre, pa im i sa te strane posvećujemo važnost. Dodatno, te lokacije će da imaju funkciju prilikom inicijalnog rasporeda neprijateljskih letjelica po mapi, pa i to dodatno napominjemo s ciljem da odabir samih lokacija ne bi kod korisnika i programera stvarao bilo kakvu konfuziju. Drugim riječima, te lokacije ćemo odmah koristiti, a razlog zbog kojeg smo odabrali upravo te lokacije će biti jasan i objašnjen u nastavku razvoja programa. Sada smo u stanju da upravo rečeno pretpočimo u kod.

Kliknemo na Gobo1, zatim na kod, i unesemo sljedeći kod (kod za eliminaciju Gobo1 laserom)



Znači, ako Gobo1 dodirne laser, postavi item 1 liste A na 0 (znak da je Gobo1 mrtav), postavi item 1 liste C na 0 (znak da sklonimo laser sa ekrana).

Jasno, dodir Gobo-a i lasera se može desiti u bilo kojem trenutku pa čitav opisani kod mora da leži unutar jedne beskonačne petlje.

Nakon što smo item 1 liste A i item 1 liste C postavili na 0, ostaje da sakrijemo Gobo1 i sačekamo neko vrijeme (sakrivanje i čekanje obavimo dva puta sa nekim vremenima čekanja). Nakon toga prebacimo Gobo1 na za njega rezervisanu lokaciju. Ta lokacija je (-10000,1000). Kako smo već rekli, nakon prebacivanja na rezervisanu lokaciju, ponovo sakrijemo Gobo1 nekoliko puta što popratimo odgovarajućim čekanjem.

Opisani kod za Gobo1 nam je potreban i za ostale Gobo-e, tj. za Gobo2-Gobo50. Primijetimo da će kod u biti da ostane isti, s tim što moramo promijeniti iteme lista A i C koje postavljamo na 0. Za Gobo1 radilo se o itemu 1. Jasno, za Gobo2 to mora da bude item 2, itd. Dodatno, moramo eventualno promijeniti lokacije koje rezervišemo za odlaganje ubijenih Gobo-a. Prvo, prenesimo upravo uneseni kod za Gobo1, na ostale Gobo-e, pa onda pristupimo izmjenama. Da ne bi došlo do preklapanja kodova na Gobo2-Gobo50, potrebno je nakon prenosa koda sa Gobo1, kodove na Gobo2-Gobo50, adekvatno razmjestiti da kodovi budu vidljivi. Dosnimimo igru.

Lokacije za odlaganje Gobo-a ćemo navesti u sljedećoj tabeli:

| Gobo   | Lokacija      | Gobo   | Lokacija      | Gobo   | Lokacija      |
|--------|---------------|--------|---------------|--------|---------------|
| Gobo1  | (-10000,1000) | Gobo17 | (-10000,1000) | Gobo33 | (-10000,1000) |
| Gobo2  | (-10000,1000) | Gobo18 | (10000,1000)  | Gobo34 | (-10000,1000) |
| Gobo3  | (-10000,1000) | Gobo19 | (10000,1000)  | Gobo35 | (-10000,1000) |
| Gobo4  | (-10000,1000) | Gobo20 | (10000,1000)  | Gobo36 | (-10000,1000) |
| Gobo5  | (0,10000)     | Gobo21 | (10000,1000)  | Gobo37 | (-10000,1000) |
| Gobo6  | (10000,1000)  | Gobo22 | (10000,1000)  | Gobo38 | (-10000,1000) |
| Gobo7  | (10000,1000)  | Gobo23 | (-10000,1000) | Gobo39 | (-10000,1000) |
| Gobo8  | (10000,1000)  | Gobo24 | (-10000,1000) | Gobo40 | (-10000,1000) |
| Gobo9  | (10000,1000)  | Gobo25 | (-10000,1000) | Gobo41 | (-10000,1000) |
| Gobo10 | (-10000,1000) | Gobo26 | (-10000,1000) | Gobo42 | (-10000,1000) |
| Gobo11 | (0,10000)     | Gobo27 | (-10000,1000) | Gobo43 | (10000,1000)  |
| Gobo12 | (10000,1000)  | Gobo28 | (10000,1000)  | Gobo44 | (10000,1000)  |
| Gobo13 | (-10000,1000) | Gobo29 | (10000,1000)  | Gobo45 | (10000,1000)  |
| Gobo14 | (-10000,1000) | Gobo30 | (10000,1000)  | Gobo46 | (10000,1000)  |
| Gobo15 | (-10000,1000) | Gobo31 | (10000,1000)  | Gobo47 | (10000,1000)  |
| Gobo16 | (-10000,1000) | Gobo32 | (10000,1000)  | Gobo48 | (10000,1000)  |
|        |               |        |               | Gobo49 | (10000,1000)  |
|        |               |        |               | Gobo50 | (10000,1000)  |

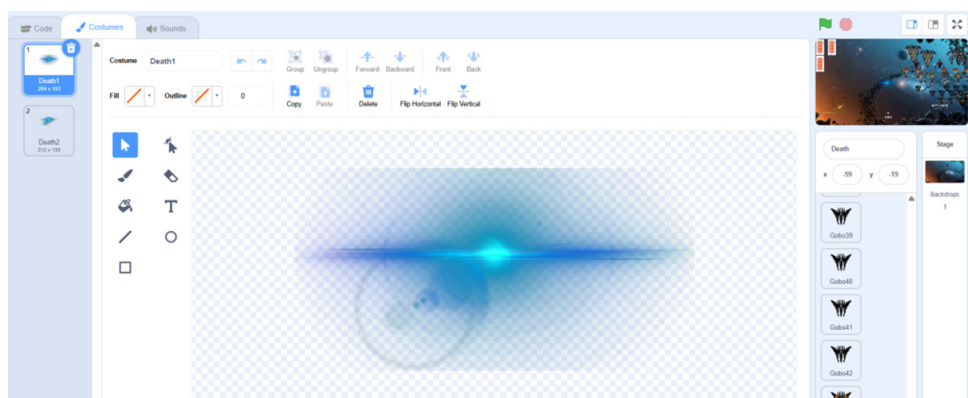
Ako pokrenemo igru, probamo laser, vidjet ćemo da on regularno funkcioniše u slučaju da ne pogađamo niti jednog od neprijatelja. Ipak, ako pogodimo nekog od

neprijatelja, laser više neće funkcionisati kako treba. Samo će da se pojavi za trenutak na vrhu naše letjelice, ali ne da ide i dalje. Zašto se ovo dešava ?

Primijetimo da mi u momentu kada laserom pogodimo nekog Gobo-a, tada u kodu za tog Gobo-a, postavimo vrijednost odgovarajućeg itema liste C da bude jednaka 0, i nigdje je više ne vratimo na vrijednost 1. Ta vrijednost 0 koju smo postavili onda ostaje da stoji u listi C na nekom mjestu. Sada, ako se na momenat vratimo na kod lasera, uočavamo dio gdje pitamo: da li lista C sadrži 0 ? Pa, u slučaju da C zaista sadrži 0, što je kod nas sada slučaj cijelo vrijeme, onda se po nastavku koda lasera, laser odmah sakriva i prebacuje na rezervisanu lokaciju (-1200,1000). Zbog toga imamo doživljaj da se laser samo na momenat pojavi na vrhu letjelice i onda nestane. U čemu je onda problem ? Problem je u tome da mi nakon što smo laserom dodirnuli Gobo-a, i postavili odgovarajuću vrijednost liste C da bude jednaka 0, nismo nikada više tu vrijednost vratili da bude jednaka 1. Da smo je vratili na 1, laser bi nastavio normalno da funkcionise. Dakle, zaključak je da mi nakon što smo u kodu za Gobo-a (bilo kojeg od njih), nakon dodira sa laserom postavili odgovarajuću vrijednost liste C da bude jednaka 0, tu istu vrijednost moramo nakon vrlo kratkog vremena vratiti da bude jednaka 1.

Ovaj problem nam daje priliku ne samo da ga riješimo, nego da dodamo još elemenata u našu igru, tj. da je obogatimo.

Uvest ćemo dodatni Sprite koji će označavati eksploziju Gobo-a koja nastaje pri njegovom dodiru sa laserom. Pri tome će da se proizvodi odgovarajući zvuk. Koristimo sljedeći Sprite

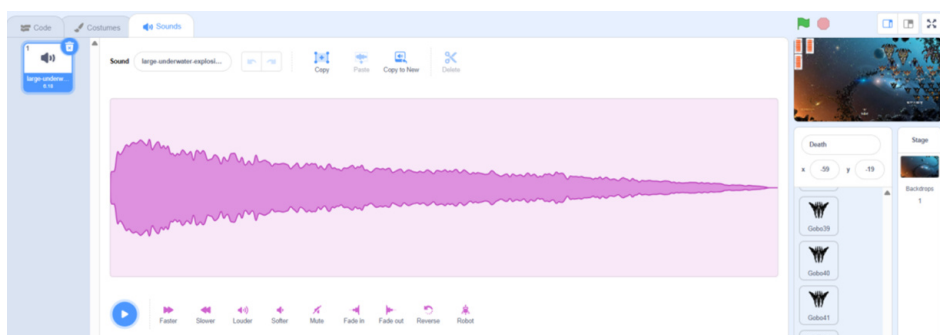


Ime ćemo mu dati Death. Vidimo da on ima dva kostima: Death1 i Death2.



Dovoljno je na Sprite-ovima upload-ovati Sprite Death1, promijeniti mu ime na Death, a onda na kostimima tog Sprite-a upload-ovati drugi kostim Death2.

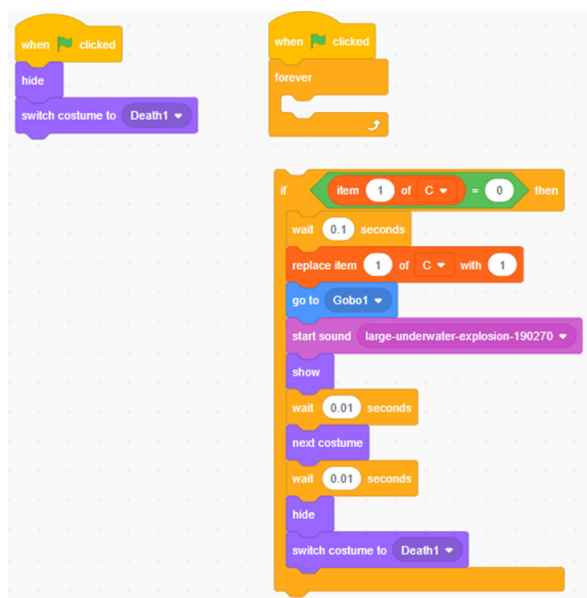
Potreban nam je i sljedeći zvuk kojeg upload-ujemo dok smo na Sprite-u Death



Možemo mu ostaviti ime koje je već ponuđeno.

Sada se možemo posvetiti kodiranju Sprite-a Death.

Dok smo na kodu Sprite-a Death, unesimo sljedeći kod



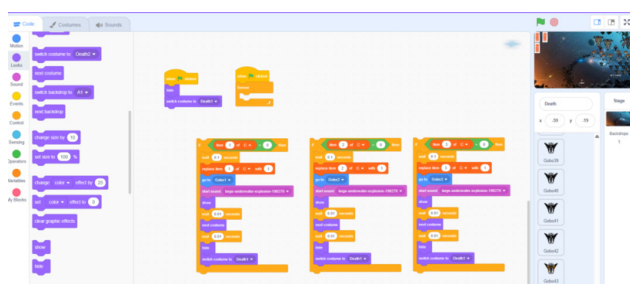


Prvo, lijevi dio koda, dok još laser nije dodirnuo niti jednog Gobo-a, eksplozija treba da je sakrivena. Osim toga, prirodno je da eksploziju inicijalno postavimo na prvi kostim.

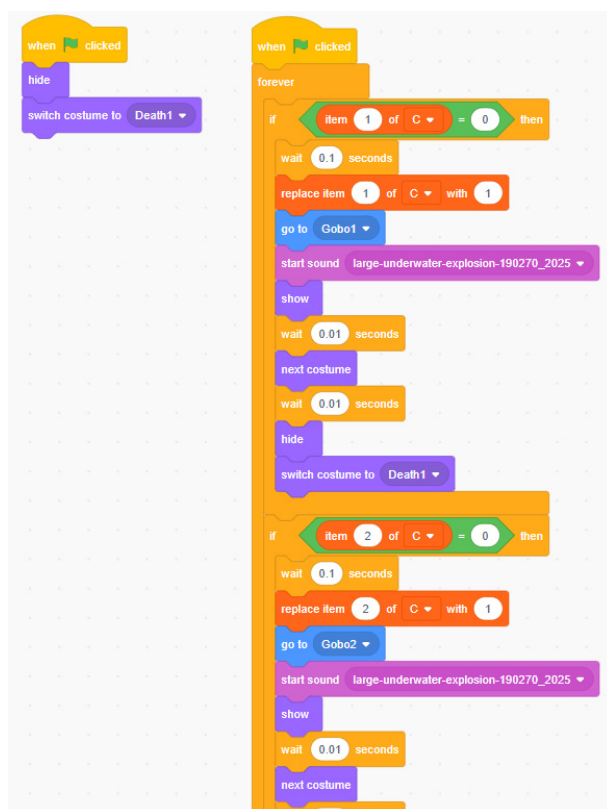
Za drugi dio koda imat ćemo jednu beskonačnu petlju, u kojoj će da se nalazi 50 If naredbi, svaka će da se odnosi na odgovarajućeg Gobo-a. Ovih 50 naredbi će biti nadovezane jedna na drugu, i kompletna struktura će da bude ubačena u beskonačnu petlju. Ovdje smo kao primjer naveli If naredbu koja se odnosi na prvog Gobo-a. Ostale će da budu analogne, sa promjenama koje moramo uraditi da bi odgovarale Gobo-ima: Gobo2-Gobo50. Dakle, šta radi ova If naredba:

Ako je item 1 liste C jednak 0, tj. ako je Gobo1 dodirnuo laser, onda: sačekaj neko kratko vrijeme, npr. 0.1 sekundi (čekamo da bi kod na laseru imao vremena da skloni laser sa ekrana), i onda ponovo postavi item 1 liste C na vrijednost 1 (da bi laser mogao pri budućim pritiscima na tipku Space da se normalno aktivira i kreće). Nakon toga: idi do Gobo1 (misli se na čitav Sprite eksplozije), počni svirati zvuk eksplozije, pojavi se. Pošto smo se u prvom dijelu koda postavili na prvi kostim tj. kostim Death1, to znači da će se u ovom momentu pred nama umjesto Gobo1 (jer smo njega već sklonili u kodu za Gobo1), pojaviti eksplozija Death1. Zadržimo se kratko vrijeme, tj. 0.01 sekundi, pređemo na sljedeći kostim (Death2), zadržimo i njega 0.01 sekundi, sakrijemo ga, i naravno, ponovo se postavimo na polazni kostim Death1. Na ovaj način smo opisali kako eksplozija treba da radi. Vidimo da se ona odigra jako brzo. Dva kostima se promijene u svega 0.02 sekundi, pri čemu se čuje zvuk eksplozije. Vizuelno, imamo osjećaj da laser dolazi do Gobo1, nakon toga Gobo1 nestaje, pojavljuje se eksplozija koja u svoja dva kostima animacije traje kratko, dok se u pozadini čuje zvuk eksplozije. Nakon toga sve nestaje.

Navedeni If koji se odnosi na Gobo1, sada moramo umnožiti 50 puta, i promijeniti odgovarajuće elemente. Tako za Gobo2 moramo imati naredbu: item 2 liste C, te Gobo2 umjesto Gobo1 tamo gdje treba, itd. za ostale Gobo-e



Kada završimo sa umnožavanjem i odgovarajućim izmjenama, sve If naredbe nadovežemo međusobno (iza prve dolazi druga pa na nju treća, itd), i onda čitavu strukturu ubacimo u beskonačnu petlju iznad. Dakle, treba da imamo kod



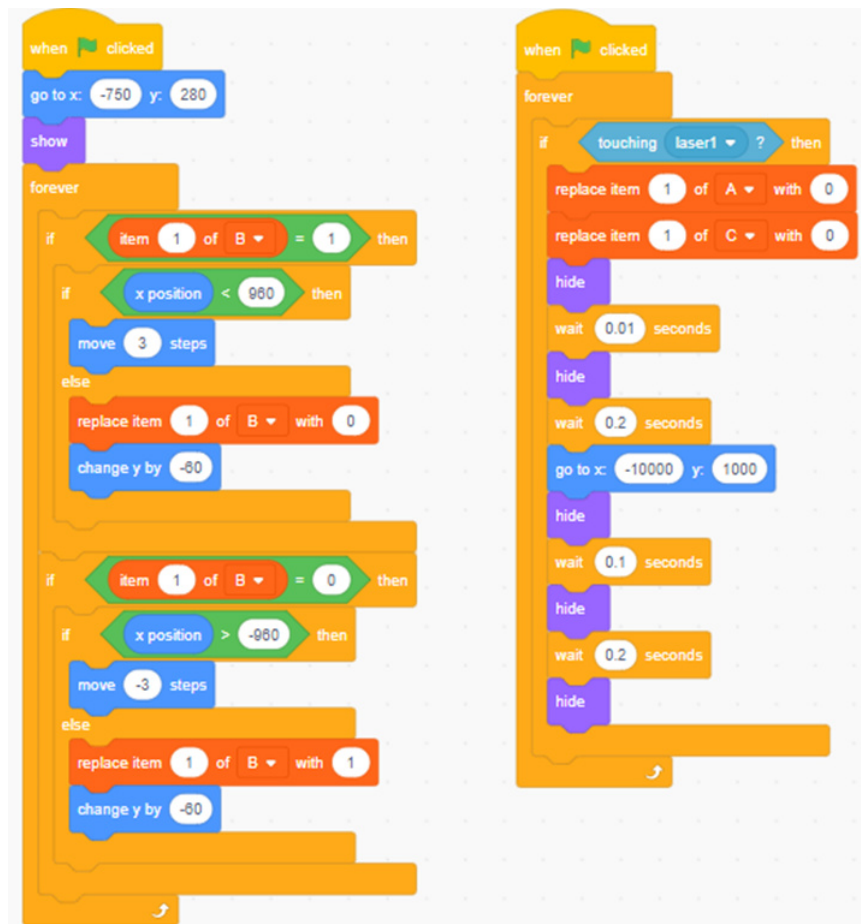
Sačuvamo igru.

Možemo pokrenuti igru i probati je.

Vidimo da sada stvari funkcionišu onako kako smo zaista i željeli.

Jedino što trenutno može da nedostaje je to da prilikom pokretanja igre, neki od Gobo-a koje smo već eliminisali (i na taj način sakrili), ne moraju da budu vidljivi prilikom pokretanja igre.

Da bismo prevazišli ovaj problem, dovoljno je na kodu svakog od Gobo-a, nakon njegovog slanja na početnu lokaciju, prikazati ga. Tako, kod za Gobo1 treba da izgleda ovako (analogno za ostale Gobo-e)

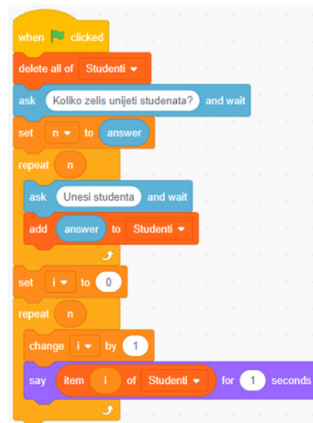


Ovdje je samo dodata naredba: show na lijevoj strani. Analogno uraditi za ostale Gobo-e, sačuvati igru i probati je.

## 5.1 Riješeni zadaci

◇ **Zadatak 5.1.1** Kreirati listu pod nazivom "Studenti". Tražiti od korisnika da unese broj studenata u listi (neka je to npr. broj  $n$ ). Zatim, tražiti od korisnika da unese  $n$  imena. Konačno, ispisati svih  $n$  imena.

**Rješenje:**



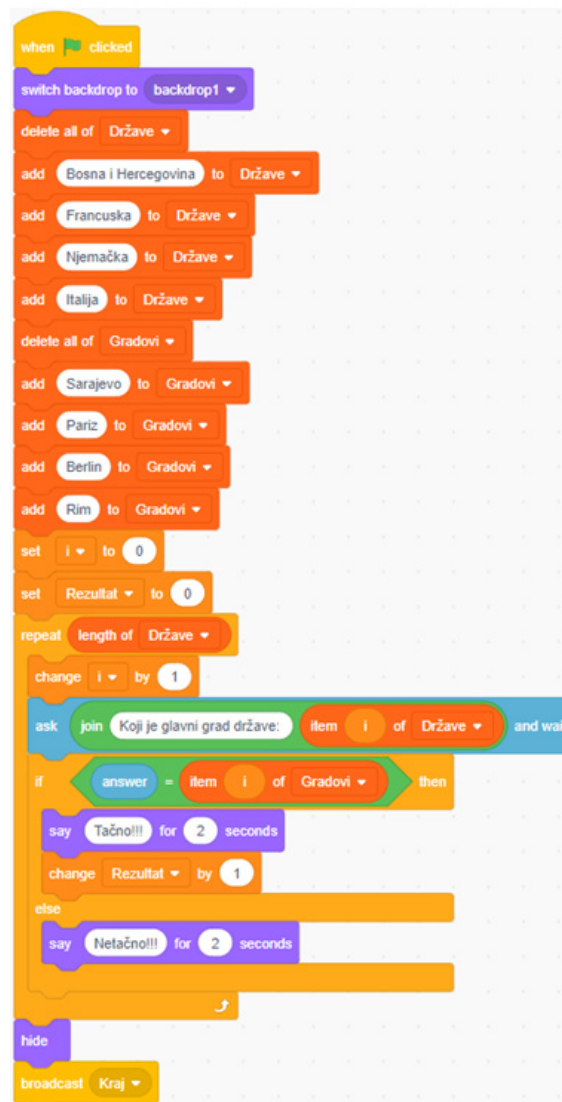
◇ **Zadatak 5.1.2** Tražiti od korisnika da unese 5 brojeva i spremiti ih u listu. Zatim, izračunati i ispisati njihovu sumu.

**Rješenje:**



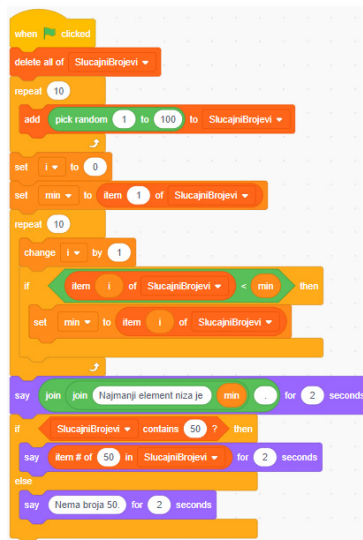
◇ **Zadatak 5.1.3** Kreirati kviz "GlavniGradovi" koristeći liste. Jedna lista treba da sadrži države, a druga njihove glavne gradove.

**Rješenje:**



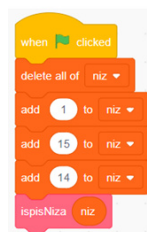
◇ **Zadatak 5.1.4** Kreirati niz (listu) pod nazivom "SlučajniBrojevi" i popuniti ga sa 10 slučajnih vrijednosti između 1 i 100. Naći i ispisati najmanji element liste. Provjeriti da li u listi postoji broj 50 i ispisati odgovarajuću poruku.

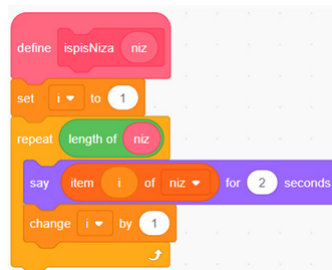
**Rješenje:**



◇ **Zadatak 5.1.5** Napraviti funkciju "ispisNiza" koja kao argument prima niz brojeva, a zatim ispisuje njegove elemente.

**Rješenje:**





## 5.2 Zadaci za vježbu

- ♣ **Zadatak 5.2.1** Generisati slučajni niz brojeva. Provjeriti da li niz sadrži nulu.
- ♣ **Zadatak 5.2.2** Napraviti funkciju koja kao argument prima niz brojeva, a zatim ispisuje sumu njegovih elemenata.
- ♣ **Zadatak 5.2.3** Napraviti funkciju koja računa faktoriyel  $n!$  unesenog broja  $n$ .
- ♣ **Zadatak 5.2.4** Napraviti program koji određuje trocifrene brojeve (ako ih ima) koji su jednaki zbiru faktoriijela svojih cifara.
- ♣ **Zadatak 5.2.5** Program treba da generiše dva slučajna trocifrena broja i zatraži od korisnika da unese rezultat oduzimanja manjeg od većeg broja. Nakon što korisnik unese odgovor, program treba da ispiše sljedeći komentar u zavisnosti od istinitosti odgovora: "Odgovor je tačan.", odnosno: "Odgovor nije tačan. Tačan odgovor je: (ispisati tačnu vrijednost)." Postupak treba ponoviti deset puta. Program na kraju ispisuje i broj pogrešnih odgovora.
- ♣ **Zadatak 5.2.6** Program traži od korisnika da unese neki cijeli broj  $n$ , a zatim generiše prvih  $n$  članova Fibonaccijevog niza.





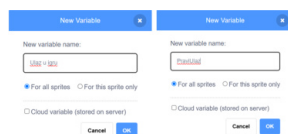
## 6 Vizuelno objektno orjentisano programiranje

Sada smo u poziciji da doradimo dugme za Start, a onda shodno tome i ostale elemente u igri vezane za njega.

Dodat ćemo dvije varijable u igru, po imenu: Ulaz u igru, i PraviUlaz. Definisat ćemo ih tačno ovako kako smo ih naveli (da imamo uniforman kod).

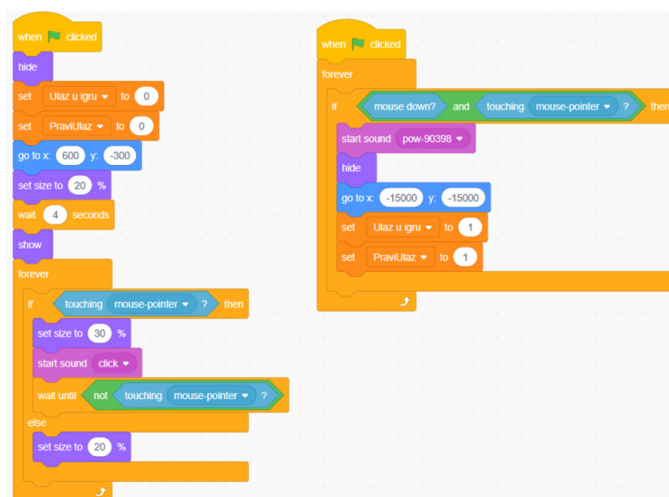
Znači, idemo na Variables (narandžasto dugme), pa na Make a Variable.

Prvi put unesemo Ulaz u igru, drugi put PraviUlaz.



Definisani ulazi: Ulaz u igru i PraviUlaz će nam trebati u nastavku za različite vrste ulaza u igri.

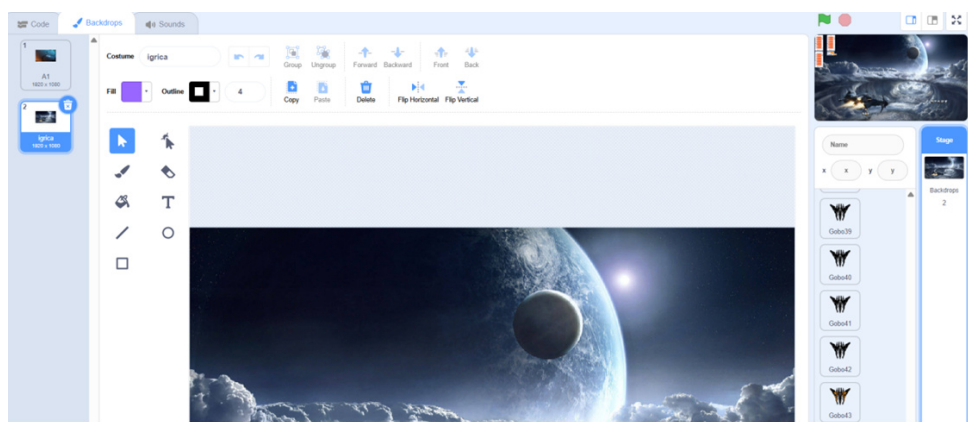
Kliknemo na Sprite START pa na Code. Moramo adaptirati postojeći kod dugmeta START pomoću definisanih ulaza. Adaptiramo kodove (dva koda koja imamo na dugmetu START) da imaju sljedeći oblik:



Ideja iza ovog koda je jednostavna. Oba ulaza su na početku postavljena na vrijednost 0, ali u momentu kada kliknemo na dugme START, oba ulaza postavljamo da budu jednaki 1. Vidjet ćemo za šta tačno će na ovi ulazi trebati.

Trebat će nam podloga koju ćemo zvati: igrica.

Klasično, kliknemo na podloge (Backdrops) desno, pa onda lijevo odemo na Upload Backdrop, te upload-ujemo navedenu podlogu (sliku)



Nakon upload-a ne zaboraviti sliku konvertovati u vektorski oblik.

Prvo, naša želja je da navedena podloga bude kao neki vid osnovne podloge, tj. podloge koja će biti aktivna prije pokretanja igre, tj. koja će biti aktivna do momenta dok ne kliknemo na dugme START. Potrebno je da to što želimo potvrdimo i u samom kodu.

U realizaciji navedenog će nam trebati i sljedeći zvuk



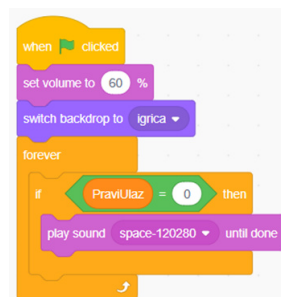
space-120280.wav

To će da bude zvuk, odnosno melodija, koja će da se čuje u pozadini dok se nalazimo na osnovnoj podlozi, tj. koja će da se čuje u pozadini sve dok ne kliknemo na dugme START.

Sada idemo na podloge, pa na Sounds, i unesemo dati zvuk



Dok smo na podlogama, idemo na kod, pa unesemo sljedeći kod

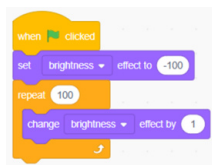


Ukoliko se nekome čini da je zvuk 100 preglasan, postavimo ga na 60. Na startu tako postavimo da aktivna podloga bude podloga: igrica. Zatim, dok god je varijabla PraviUlaz jednaka nuli, neka svira melodija koju smo upload-ovali. Pošto mi ne znamo kada tačno ćemo željeti da pomenuta melodija prestane da se čuje, to dio koda koji se odnosi na oglašavanje navedene melodije stavljamo u beskonačnu petlju. Kako koristimo opciju play sound a ne opciju start sound, to će jasno navedena melodija da se čuje svom svojom dužinom, tj. od početka pa do kraja, a pošto se nalazi u beskonačnoj petlji, to će ona nakon završetka ponovo početi da se oglašava ispočetka, itd. Dakle, polazna podloga je: igrica (dok drugačije ne naredimo), te navedena melodija: space-120280 će da se čuje u pozadini dokle god je varijabla PraviUlaz jednaka nuli. Kada će onda varijabla PraviUlaz da promijeni svoju vrijednost? Sjetimo se da smo na dugmetu START naredili da PraviUlaz postane jednako 1 nakon što kliknemo na dugme START. Tako, melodija space-120280 će da se čuje u pozadini do momenta

dok ne kliknemo na dugme START. Kada kliknemo na dugme START, tada se obje varijable: Ulaz u igru, te PraviUlaz postavljaju na vrijednost 1, i tada treba nešto novo da se dogodi, a šta, zavisi od nas, tj. zavisi od toga šta mi tačno iskodiramo da se desi u tom momentu. Bavimo se u nastavku time.

Primijetimo da dokle god ne kliknemo na dugme START, melodija space-120280 će da se oglašava iznova i iznova. Nakon što kliknemo na dugme START melodija space-120280 će da odsvira do kraja nakon čega više neće počinjati da se oglašava.

Kao neku vrstu estetskog vizuelnog efekta, na podloge dodajemo i sljedeći kod



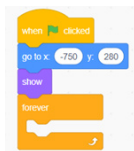
Tako, na početku postavimo da je osvijetljenje jednako -100 (potpuni mrak). Nakon toga napravimo petlju od 100 iteracija, gdje u svakom koraku mijenjamo osvijetljenje za 1. Jasno, nakon 100 koraka ćemo iz potpunog mraka (-100) da dođemo do normalnog osvijetljenja (0).

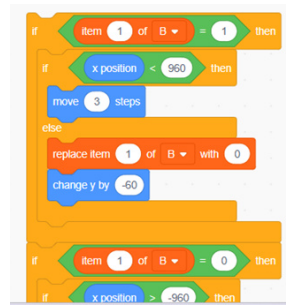
Uočimo da nakon pokretanja igre, osvijetljenje se promijeni sa -100 do 0. Pri tome dugme START postaje aktivno nakon 4 sekunde (pogledati kod dugmeta START). Uz to, u pozadini se čuje melodija space-120280. Uočimo i to da su neprijateljske letjelice već aktivne. Kreću se po ranije unesenom algoritmu. Naime, one ne bi trebale da su aktivne u ovoj fazi. One bi trebale da postanu aktivne tek nakon klika na dugme START. Moramo to onda doraditi. Uradit ćemo to u nekoliko koraka.

Prvo, moramo modifikovati kodove neprijateljskih letjelica, u smislu da se pojavljuju tek nakon klika na dugme START.

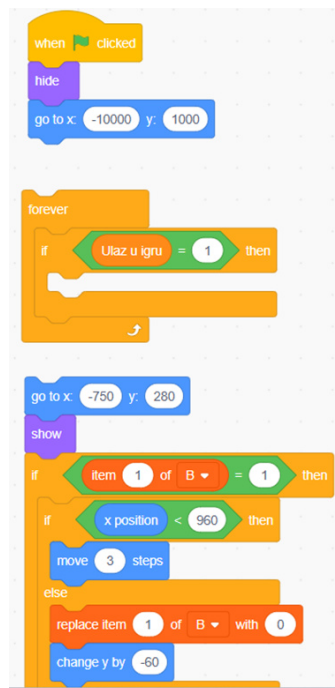
Idemo na Gobo1, i na kod.

U kodu koji se odnosi na kretanje letjelice (kod sa listom B), izvučemo If naredbe van petlje forever



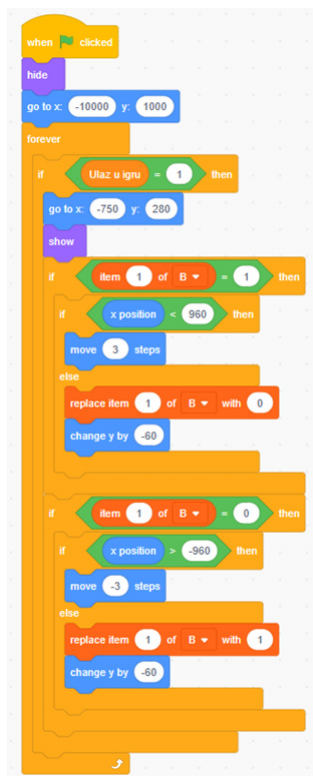


U petlju forever ubacimo dodatni uslov If, u kome ispitujemo da li je Ulaz u igru jednako 1. Naime, sada je vrijeme da iskoristimo varijablu Ulaz u igru. Tako, ako je Ulaz u igru jednako 1, tek tada letjelice treba da se pojave pred nama. Prije uklapanja elemenata, kod sada treba da ima sljedeći oblik



Jasno, prije klika na dugme START, Gobo1 (kao i ostali), treba da je sakriven, i pohranjen na lokaciji rezervisanoj za njega (-10000,1000) (pogledati spisak lokacija za ostale Gobo-e (Gobo2-Gobo50)).

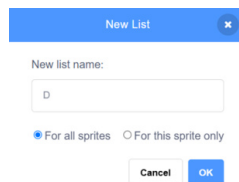
Zatim, kada otpočnemo forever petlju, u njoj treba da zatražimo uslov da je Ulaz u igru jednak 1. To će nam značiti da smo kliknuli na dugme START. Posljednji dio koda (treći) treba da ubacimo unutar pomenutog If-a. To će značiti da ako smo kliknuli na START, Gobo1 treba da ode na lokaciju (-750,280), i da se tek tada prikaže, nakon čega počinje njegovo kretanje desno-lijevo. Tako, nakon uklapanja, kod za Gobo1 treba da ima sljedeći oblik



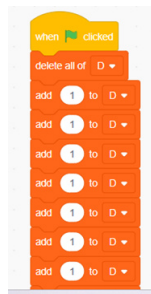
Na prvi pogled djeluje da je ovaj kod zaista ono što nam treba. Ipak, ako pokrenemo igru, vidjet ćemo da to i nije baš tako. Djelimično ćemo imati zadovoljeno ono što želimo. Kad pokrenemo igru, Gobo1 neće biti prisutan na ekranu. On će da se pojavi tek nakon što kliknemo na dugme START. Ono što ne funkcioniše kako treba

jeste da on stoji na jednom mjestu, tj. na lokaciji (-750,280). Naime, naredba da Gobo1 ide na lokaciju (-750,280) je ubačena unutar forever petlje, tako da će svakim prolaskom kroz tu petlju, Gobo1 da se iznova i iznova postavlja na lokaciju (-750,280), bez obzira što nakon te naredbe dolaze naredbe o kretanju desno-lijevo. Faktički zbog brzine izvršavanja naredbi, naredbe za kretanje desno-lijevo nemaju vremena da se odigraju, jer ponovni prolazak kroz forever petlju iznova i iznova postavlja Gobo1 na početnu lokaciju (-750,280). Nama je potrebno da se Gobo1 samo jednom postavi na poziciju (-750,280), i da nakon toga otpočne njegovo kretanje desno-lijevo. Za ovo nam je potreban neki prekidač, odnosno prekidači ako govorimo o svih 50 Gobo-a. Dakle, treba nam neka nova lista prekidača. Zvat ćemo je D.

Idemo na liste, tj. na Make a List, i napravimo listu D



Kao i sa dosadašnjim listama: A, B i C, i listu D na podlogama postavimo tako da joj elementi inicijalno budu jednaki 1. Znači, idemo na podloge a zatim na kod, pa dodamo sljedeći kod

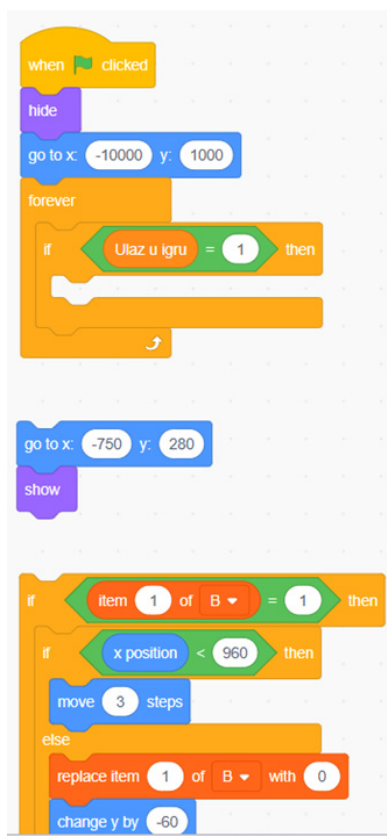


Ponovo, kao i u slučaju prethodnih listi, prvo elemente liste D počistimo (svaki put nakon pokretanja), i dodamo u listu D tačno 50 jedinica. Voditi računa o tome da povremeno dosnimimo igru.

Činjenicu da je item 1 liste D jednak 1 ćemo koristiti kao informaciju da Gobo1 postavimo na početnu poziciju (-750,280). Čim ga postavimo na tu poziciju, odmah ćemo item 1 liste D postaviti na vrijednost 0. Na taj način ćemo spriječiti da se Gobo1 konstantno postavlja na početnu poziciju. Kada ćemo ponovo item 1 liste D postaviti na vrijednost 1? Jasno, dok je Gobo1 živ, item 1 liste D treba da bude jednak 0. Dakle, item 1 liste D ćemo ponovo postaviti na vrijednost 1 tek kada eliminišemo Gobo1 sa mape.

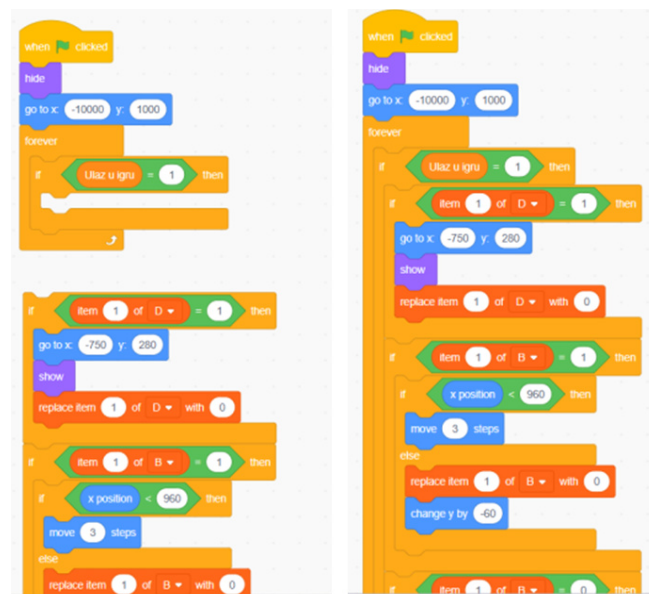
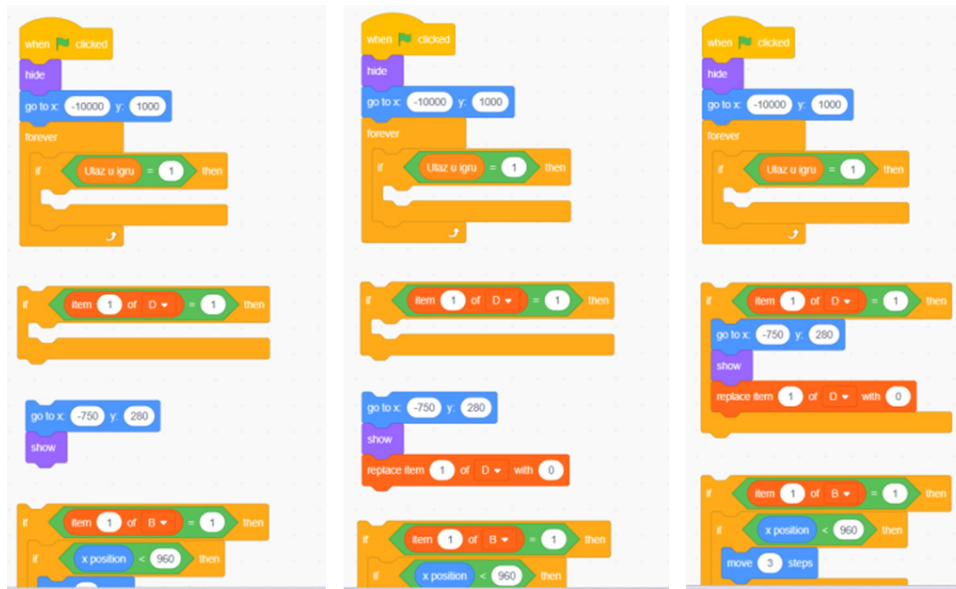
Modifikujemo onda kod za Gobo1.

Izvučemo ponovo kod koji se nalazi unutar If naredbe za Ulaz u igru

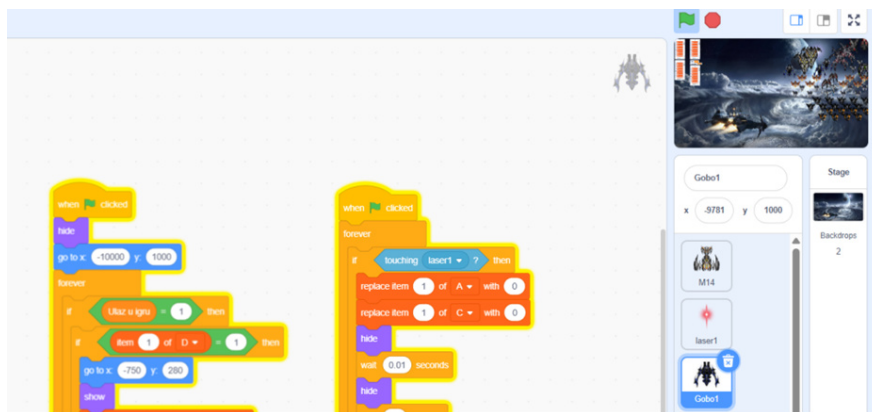


Napravimo novu If naredbu koja se odnosi na upit: da li je item 1 liste D jednak 1. Izvršimo dodavanja naredbi i uklapanja kao na slikama:





Ako sada pokrenemo igru, i pratimo Gobo1, vidjet ćemo da igra uglavnom radi ono što smo željeli. Nakon pokretanja igre, Gobo1 se ne pojavljuje. Tek kada kliknemo na START Gobo1 se pojavljuje i počinje da se kreće. Ako ga eliminišemo laserom, on nestaje. Ipak, može se vidjeti da se on i dalje kreće (kada minimiziramo ekran)

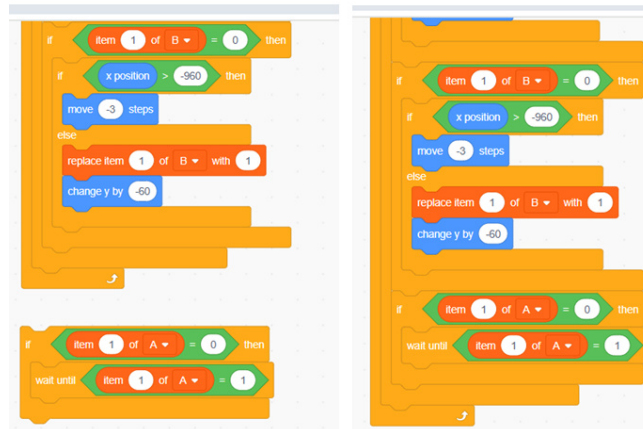


Naime, na koordinatama za Gobo1 se može vidjeti mijenjanje vrijednosti  $x$  a onda i  $y$ . To znači da je dio koda koji se odnosi na kretanje Gobo-a i dalje aktivan. Ova situacija ne mora ali i može da nas zanima. Drugim riječima, logično bi bilo da nakon eliminacije Gobo-a, taj Gobo postane neaktivan. On bi u momentu eliminacije trebao da se pozicionira na unaprijed rezervisanu lokaciju i da tamo čeka. Sada vidimo da u već modificovani kod možemo dodati naredbu koja nam omogućava već rečeno. Rezon za Gobo1 bi bio: ako je Gobo1 mrtav, neka sačeka dok ponovo ne bude živ. Sjetimo se da živ-mrtav za Gobo-e označavamo elementima 1-0 liste A. Tako, upravo opisani rezon za Gobo1, preveden na jezik liste A, bi bio: ako je item1 liste A jednak 0, neka sačeka dok ne postane jednak 1. To je ustvari sljedeća konstrukcija naredbi



Ovaj kod treba da se doda na kraj prethodno modificovanog koda. Možemo ga postaviti na lokaciju ispod prethodno modificovanog koda i tako pripremiti za ukla-

panje



Primijetimo da na prvoj slici, na kraju koda, imamo 4 završetka naredbi (4 smeđkasta reda). Novi kod koji se odnosi na listu A uklapamo tako da ga postavimo nakon prva dva pomenuta završetka (ili što je isto, prije zadnja dva pomenuta završetka), tako da dobijemo situaciju kao na prethodnoj slici.

Ako sada pokrenemo igru, vidjet ćemo da nakon eliminacije Gobo1, on ne nastavlja da se kreće, nego ostaje da čeka na unaprijed rezervisanoj lokaciji (-10000,1000).

Dosnimimo igru.

Sada je potrebno da posljednji kod sa Gobo1 zalijepimo na sve ostale Gobo-e i da ga na njima modifikujemo za svakog Gobo-a ponaosob. Možemo se dvoumiti između načina da prolazimo kroz sve Gobo-e pojedinačno i vršimo modifikaciju kodova, ili da, kako smo već rekli, navedeni kod zalijepimo i onda vršimo izmjene. Zbog broja koraka koje smo uložili u modifikaciju posljednjeg koda sigurno je optimalnije zalijepiti kod sa Gobo1 na ostale, a onda vršiti njegovu modifikaciju. Pri tome, za neophodne informacije o lokacijama, možemo koristiti već postojeće kodove koje imamo na Gobo2-Gobo50, a onda ih izbrisati. Uradimo kako smo opisali za sve Gobo2-Gobo50.

Liste: A, B, C, D, kao i varijable: Ulaz u igru, PraviUlaz, možemo ukinuti da se ne pojavljuju na ekranu (ukinemo kvakicu pored njihovog imena). Dosnimimo i testiramo igru.

Pozicionirajmo se na Gobo1 i na njegov kod. Drugi dio koda (desni kod koji se odnosi na dodir Gobo-a sa laserom za sada ima oblik)



U ovom kodu nismo vršili izmjene, pa ga prikazujemo minijaturno (isti je kao i ranije).

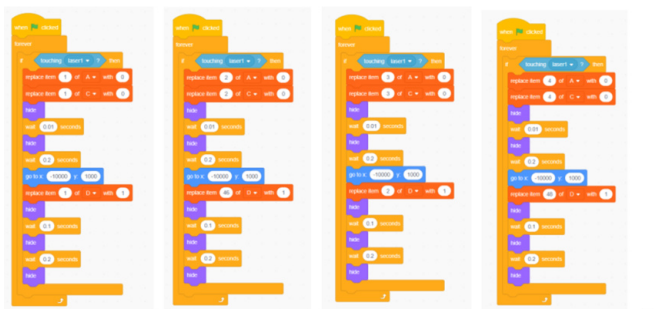
Kako smo već objasnili logiku za listu D, ako govorimo o npr. Gobo1, na početku je item 1 liste D jednak 1. Ova informacija nam je omogućavala da postavimo Gobo1 na početnu lokaciju (-750,280). Dakle, ako je item 1 liste D jednak 1, postavi Gobo1 na početnu lokaciju. Nakon toga, odmah promijeni vrijednost za item 1 liste D na 0. Nakon toga smo rekli kada je potrebno ponovo postaviti vrijednost za item 1 liste D da bude jednaka 1. To je onda kada Gobo1 bude eliminisan. Vidimo da je u posljednjem kodu, potrebno još na nekome mjestu postaviti vrijednost za item 1 liste D da bude jednaka 1, a onda analogno uraditi za ostale Gobo-e. Možemo tu promjenu vrijednosti sa 0 na 1 postaviti na mjesto kao u sljedećem kodu



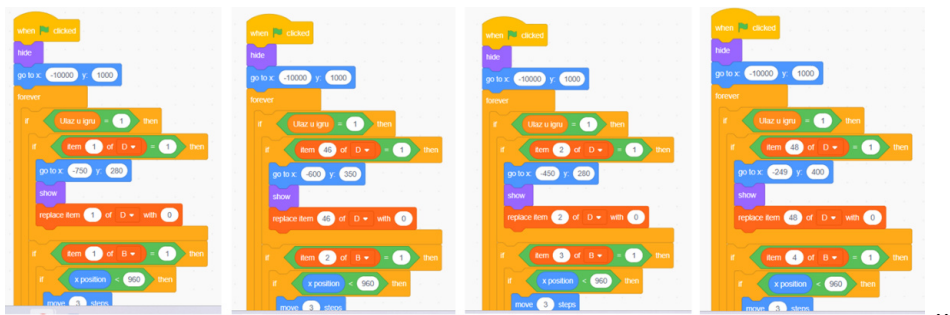
Mi ćemo u nastavku htjeti da doradimo pojavljivanje neprijateljskih letjelica, tj. Gobo-a na nešto detaljniji i vizuelno zanimljiviji način. Pri tome, nećemo te letjelice redati klasičnim redoslijedom od Gobo1 do Gobo50. Naime, radi vizuelnog doživljaja, uvest ćemo nešto drugačiji redoslijed redanja neprijatelja na ploču. Za svaku od naših letjelica Gobo1-Gobo50 ćemo sada dati tabelu sa odgovarajućim rednim brojem. Taj redni broj će da označava broj pod kojim se ta letjelica pojavljuje. Tako, ako uz Gobo1 stoji broj 1, onda to znači da se Gobo1 prvi pojavljuje na ekranu, ako uz Gobo2 stoji broj 46, to znači da se Gobo2 pojavljuje 46-ti po redu na ekranu, itd.

| Letjelica | Redni broj | Letjelica | Redni broj | Letjelica | Redni broj |
|-----------|------------|-----------|------------|-----------|------------|
| Gobo1     | 1          | Gobo18    | 19         | Gobo35    | 30         |
| Gobo2     | 46         | Gobo19    | 18         | Gobo36    | 31         |
| Gobo3     | 2          | Gobo20    | 6          | Gobo37    | 32         |
| Gobo4     | 48         | Gobo21    | 17         | Gobo38    | 33         |
| Gobo5     | 50         | Gobo22    | 16         | Gobo39    | 34         |
| Gobo6     | 49         | Gobo23    | 20         | Gobo40    | 35         |
| Gobo7     | 4          | Gobo24    | 21         | Gobo41    | 36         |
| Gobo8     | 47         | Gobo25    | 7          | Gobo42    | 37         |
| Gobo9     | 3          | Gobo26    | 22         | Gobo43    | 45         |
| Gobo10    | 9          | Gobo27    | 23         | Gobo44    | 44         |
| Gobo11    | 11         | Gobo28    | 27         | Gobo45    | 43         |
| Gobo12    | 10         | Gobo29    | 26         | Gobo46    | 42         |
| Gobo13    | 12         | Gobo30    | 8          | Gobo47    | 41         |
| Gobo14    | 13         | Gobo31    | 25         | Gobo48    | 40         |
| Gobo15    | 5          | Gobo32    | 24         | Gobo49    | 39         |
| Gobo16    | 14         | Gobo33    | 28         | Gobo50    | 38         |
| Gobo17    | 15         | Gobo34    | 29         |           |            |

Desne kodove za Gobo1-Gobo50 moramo sada adaptirati u odnosu na upravo navedenu listu. Za Gobo1 smo već dodali naredbu koja se odnosi na listu D, moramo onda i za ostale Gobo-e to uraditi kao na slikama (u skladu sa prethodnom tabelom)



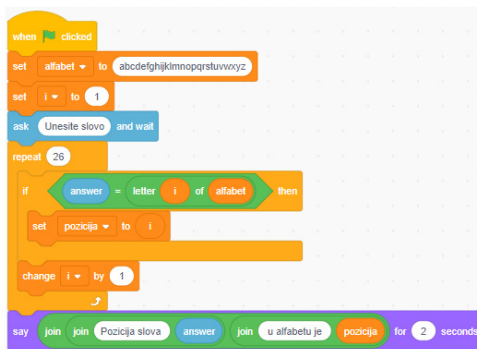
Konačno, nakon što smo desne kodove za Gobo1-Gobo50 adaptirali u skladu sa navedenom tabelom, moramo to učiniti i sa lijevim kodovima (na mjestima gdje se pojavljuje lista D). Znači, lijevi kodovi za Gobo1-Gobo50 treba da izgledaju kao na slikama (gledati samo dio za listu D)



## 6.1 Riješeni zadaci

◇ **Zadatak 6.1.1** Napraviti program koji određuje poziciju unesenog slova u engleskoj abecedi. Korisnik unosi jedno malo slovo, a program ispisuje odgovarajući redni broj (npr, za slovo *a* ispisuje vrijednost 1, za slovo *z* ispisuje vrijednost 26, itd).

**Rješenje:**



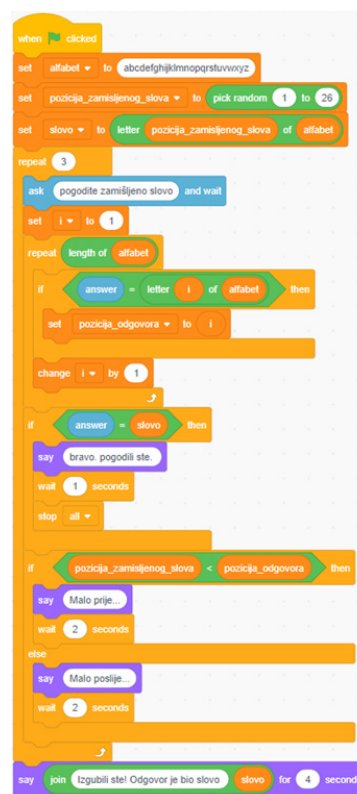
◇ **Zadatak 6.1.2** Računar nasumično bira jedno malo slovo engleske abecede. Korisnik ima tri pokušaja da pogodi koje je slovo odabrano. Nakon svakog pokušaja, program provjerava da li je korisnik pogodilo slovo. Ako jeste, program ispisuje poruku "Bravo, pogodili ste!" i staje sa radom. Ako nije, program upoređuje poziciju unesenog slova i slova odabranog programom, te daje korisniku naznaku:

"Malo prije..." ako je odgovor abecedno ispred slova odabranog programom,

"Malo poslije..." ako je odgovor abecedno iza slova odabranog programom.

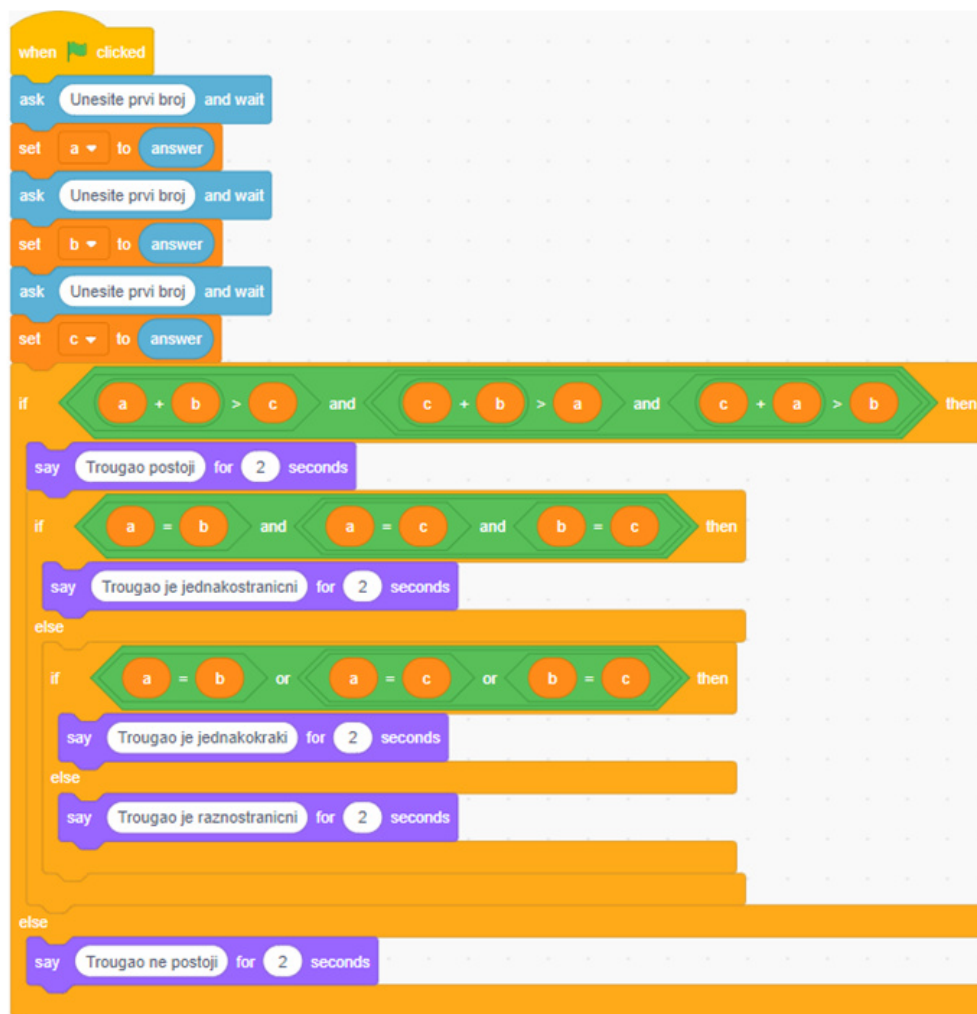
Ako korisnik ne pogodi slovo ni nakon tri pokušaja, program ispisuje poruku o neuspjehu, te otkriva koje je slovo programom bilo odabrano.

**Rješenje:**



◇ **Zadatak 6.1.3** Korisnik unosi tri broja:  $a$ ,  $b$  i  $c$ . Potrebno je provjeriti da li postoji trougao čije su stranice uneseni brojevi. Ako postoji, ispisati da li je trougao jednakos-tranični, jednakokraki ili raznostranični. Ako ne postoji, ispisati odgovarajuću poruku.

**Rješenje:**



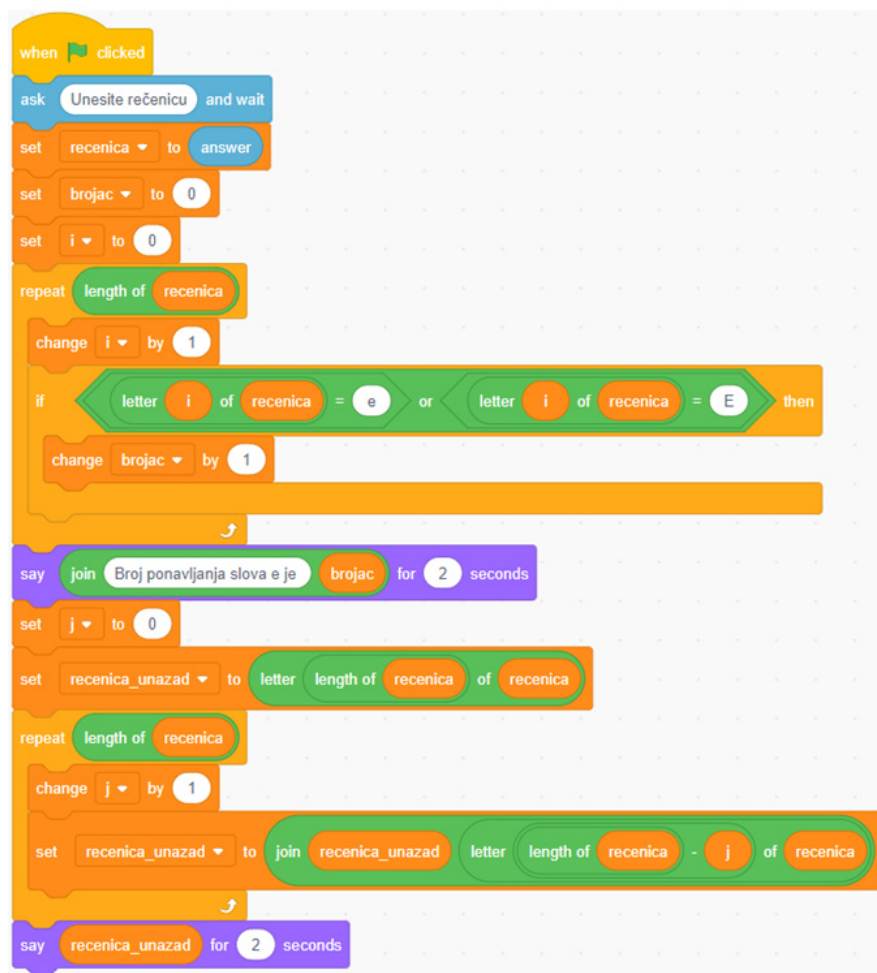


◇ **Zadatak 6.1.4** Korisnik unosi rečenicu. Potrebno je prebrojati koliko puta se u rečenici pojavljuje slovo e (veliko ili malo), te ispisati rezultat. Zatim, ispisati unesenu rečenicu unazad. Npr,

Ulaz: Danas je lijep dan.

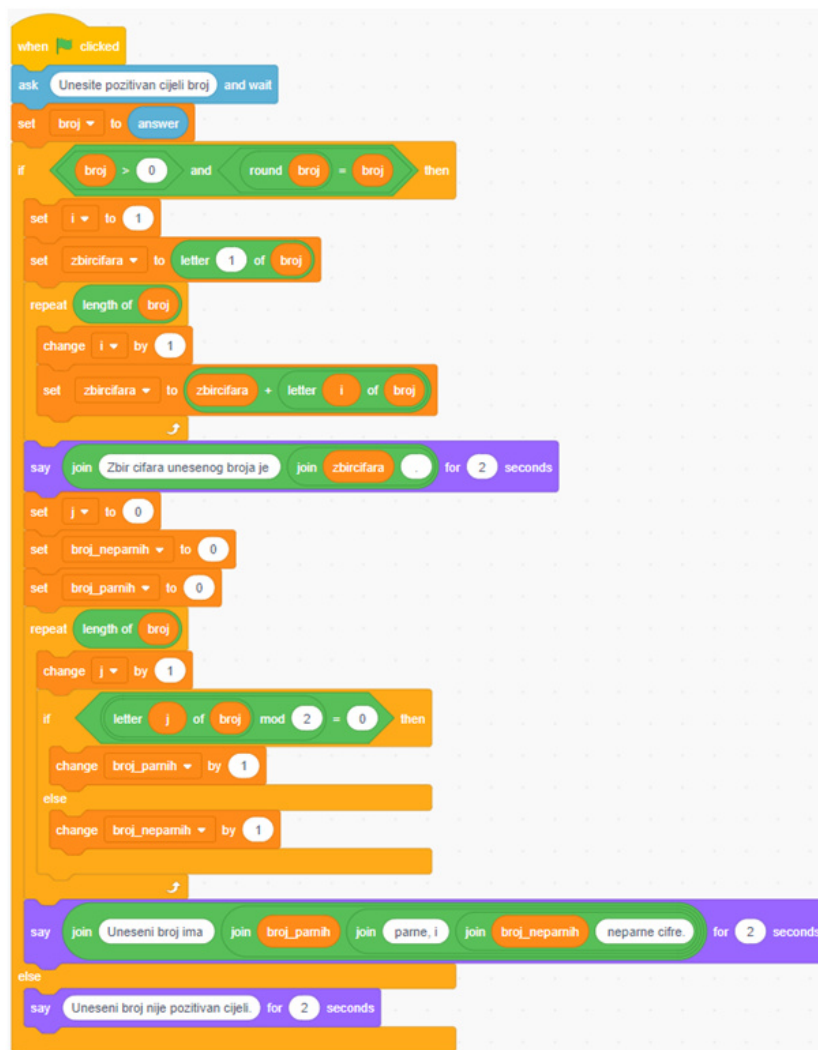
Izlaz: Broj ponavljanja slova e je 2. nad pejl sanaD.

**Rješenje:**



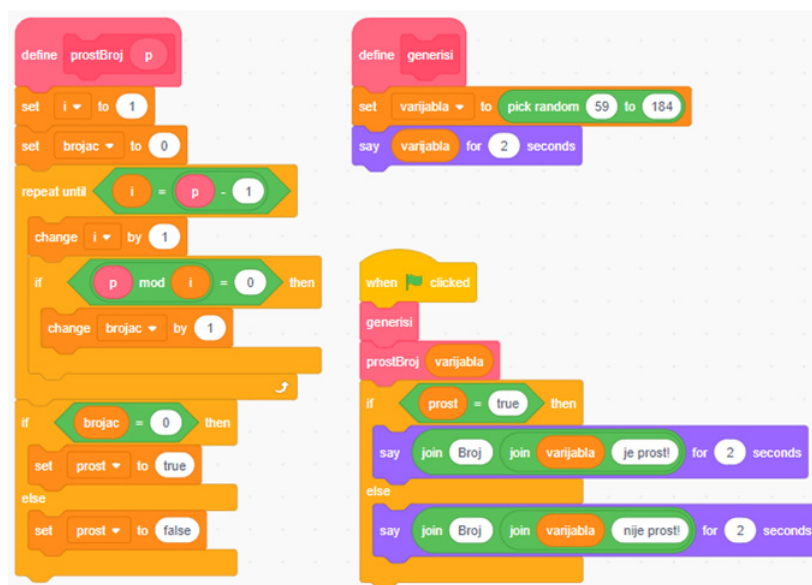
◇ **Zadatak 6.1.5** Korisnik unosi pozitivan cio broj. Potrebno je izračunati zbir njegovih cifara, a zatim prebrojati koliko cifara je parno, a koliko neparno. Ispisati rezultate. Ako uneseni broj nije cio pozitivan, ispisati odgovarajuću poruku.

**Rješenje:**



◇ **Zadatak 6.1.6** Potrebno je nasumično generisati broj iz intervala 59 do 184 i provjeriti da li je taj broj prost. Ako jeste, ispisati poruku da je broj prost. U suprotnom, ispisati da nije. Za provjeru da li je broj prost, koristiti vlastiti blok "prostBroj". Za generisanje broja koristiti vlastiti blok "generisi".

**Rješenje:**



## 6.2 Zadaci za vježbu

♣ **Zadatak 6.2.1** Korisnik unosi riječ. Potrebno je ispisati dužinu te riječi, tj. broj slova koje ta riječ sadrži.

♣ **Zadatak 6.2.2** Korisnik unosi riječ. Program treba ispisati prvo i posljednje slovo te riječi.

♣ **Zadatak 6.2.3** Korisnik unosi riječ. Potrebno je ispisati sva slova te riječi, jedno po jedno.

♣ **Zadatak 6.2.4** Korisnik unosi riječ i jedno slovo. Program treba ispisati koliko puta se to slovo pojavljuje u unesenoj riječi.

♣ **Zadatak 6.2.5** Izmijeniti Zadatak 6.1.2 tako da na početku programa korisnik unosi broj pokušaja koje želi da ima.

♣ **Zadatak 6.2.6** Korisnik unosi riječ. Program treba da zamijeni svako drugo slovo u unesenoj riječi zvjezdicom "\*" te da ispiše novo-dobijenu riječ.

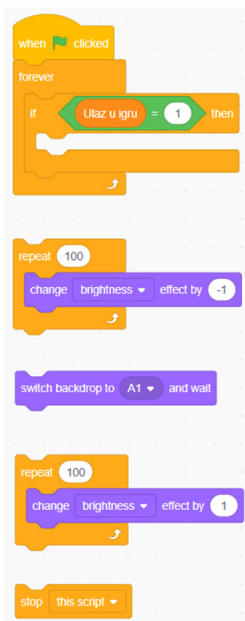
## 7 Osvjetljenje i zvuk

U kodu za START smo promjenjive Ulaz u igru i PraviUlaz inicijalno postavili na vrijednosti 0. Osim toga, nakon klika na dugme START smo im promijenili vrijednosti na 1. Pri tome, na podlogama, iskoristili smo vrijednost 0 promjenjive PraviUlaz da se zadržimo na podlozi koju smo zvali "igrica", tako da u pozadini svira melodija space-120280 (do momenta dok ne kliknemo na dugme START).

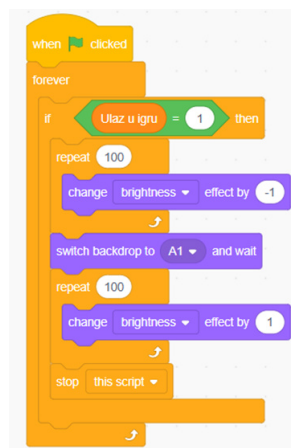
Kada kliknemo na dugme START, vrijednosti promjenjivih Ulaz u igru i PraviUlaz postavljamo na 1, tako da pomenuta melodija space-120280 će nakon što bude odsvirana do kraja, prestati da se čuje, tj. neće doći do njenog ponovnog pokretanja. Ovo ima smisla jer želimo nakon klika na START da pređemo u neku novu fazu igre, gdje bi trebala da bude aktivna neka nova podloga, te da svira neka nova melodija.

Idemo se prvo pozabaviti novom podlogom nakon klika na dugme START.

Želimo da se nakon klika na dugme START, postojeća podloga zatamni, da nakon toga postavimo novu podlogu (koristit ćemo podlogu A1), te da se ta podloga nakon toga zasvijetli. Iskoristit ćemo sada promjenjivu Ulaz u igru. Na podloge unosimo onda sljedeći kod po dijelovima



Pošto ne znamo kada ćemo kliknuti na dugme START, to koristimo forever petlju. Ako kliknemo na dugme START (promjenjiva Ulaz u igru će postati 1), tj. ako je Ulaz u igru jednako 1, onda treba da se desi: postojeću podlogu "igrica" zatamnimo (to je prva petlja od 100 koraka), zatim promijenimo podlogu na A1, te zasvijetlimo novu podlogu (to je druga petlja od 100 koraka). Naravno, pošto sve ovo treba da se desi samo jednom, i to nakon klika na dugme START, to nakon izvršavanja navedenog koda zaustavimo pomenutu skriptu. Uklopljena varijanta navedenog koda je



Možemo probati igru, tj. navedeni efekat.

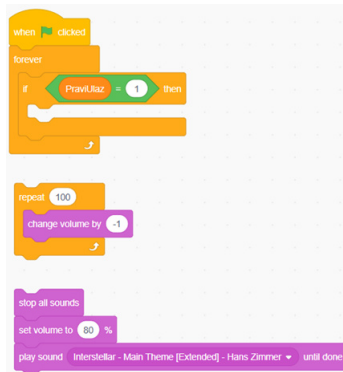
Igru možemo pokrenuti i dok je prozor igre minimiziran a mi se nalazimo na pomenutom kodu. Jasno se onda može vidjeti do kada traje život navedenog koda, tj. kada se pomenuta skripta zaustavlja (čim se nova podloga potpuno zasvijetli, pomenuti kod prestaje sa radom).

Dosnimimo igru.

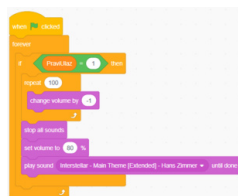
Podesimo sada i promjenu melodije koja se čuje u pozadini nakon klika na dugme START. Ta promjena treba da bude blaga, tj. postojeća melodija koja se čuje treba da blago utihne, nakon čega treba da počne nova melodija. Ovdje ćemo iskoristiti promjenjivu PraviUlaz. Koristit ćemo sljedeću melodiju (temu) iz filma Interstellar

Interstellar - Main Theme [Extended] - Hans Zimmer

Ovu melodiju prvo upload-ujemo dok se nalazimo na podlogama. Neuklopljeni kod na podlogama je sljedeći



Logika koda je sljedeća. Ne znamo kada ćemo kliknuti na dugme START pa koristimo forever petlju. Znamo da kada kliknemo na dugme START, promjenjiva PraviUlaz postaje 1. Tako, ako je PraviUlaz jednako 1, tj. ako smo kliknuli na START, onda: prvo stišamo postojeću melodiju polagano (to je petlja od 100 koraka). Nakon toga zaustavimo zvukove (zaustavimo postojeću melodiju - inače bi ona nastavila da svira iako je stišana, a onda kada bi povećali ton, ona bi se ponovo čula). Konačno, postavimo jačinu tona na željenu vrijednost za melodiju Interstellar (ovdje je dovoljna vrijednost 80), te naredimo da svira melodija Interstellar. Primijetimo da kada se Interstellar melodija završi, da onda ponovo počinje forever petlja, pa pošto je PraviUlaz i dalje jednako 1, to će se nakon završetka melodije Interstellar zvuk ponovo polagano stišati, zaustavit će se zvukovi, nakon čega će ponovo početi da svira Interstellar melodija. Na ovaj način omogućavamo da melodija Interstellar svira onoliko dugo koliko je to potrebno. Mi želimo da se ova melodija čuje dok traje prvi nivo (prvi level). Nakon što pređemo na novi nivo, željet ćemo da se promjeni i ova melodija. Ipak, mi ne znamo kada će se to dogoditi, tako da na upravo opisani način omogućavamo da Interstellar melodija bude aktivna dokle god je to potrebno. Uklopljeni kod je sljedeći



Probamo igru i sačuvamo izmjene.

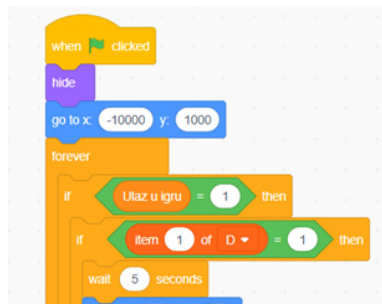
Vidimo da su neprijateljske letjelice aktivne odmah nakon klika na dugme START. Ipak, s obzirom na upravo izvršena zatamnjenja i osvjetljavanja podloga, one ne bi trebale da su aktivne odmah nakon klika na dugme START. One bi trebale da postanu aktivne nakon što se nova podloga osvijetli. Znači trebalo bi da se sa pojavljivanjem neprijateljskih letjelica sačeka određeno vrijeme nakon klika na dugme START, tj. da se sačeka onoliko koliko je potrebno da se nova podloga u potpunosti osvijetli.

Sjetimo se da smo već odredili raspored pojavljivanja letjelica. On je sljedeći:

|        |    |        |    |        |    |        |    |
|--------|----|--------|----|--------|----|--------|----|
| Gobo1  | 1  | Gobo14 | 13 | Gobo27 | 23 |        |    |
| Gobo2  | 46 | Gobo15 | 5  | Gobo28 | 27 | Gobo40 | 35 |
| Gobo3  | 2  | Gobo16 | 14 | Gobo29 | 26 | Gobo41 | 36 |
| Gobo4  | 48 | Gobo17 | 15 | Gobo30 | 8  | Gobo42 | 37 |
| Gobo5  | 50 | Gobo18 | 19 | Gobo31 | 25 | Gobo43 | 45 |
| Gobo6  | 49 | Gobo19 | 18 | Gobo32 | 24 | Gobo44 | 44 |
| Gobo7  | 4  | Gobo20 | 6  | Gobo33 | 28 | Gobo45 | 43 |
| Gobo8  | 47 | Gobo21 | 17 | Gobo34 | 29 | Gobo46 | 42 |
| Gobo9  | 3  | Gobo22 | 16 | Gobo35 | 30 | Gobo47 | 41 |
| Gobo10 | 9  | Gobo23 | 20 | Gobo36 | 31 | Gobo48 | 40 |
| Gobo11 | 11 | Gobo24 | 21 | Gobo37 | 32 | Gobo49 | 39 |
| Gobo12 | 10 | Gobo25 | 7  | Gobo38 | 33 | Gobo50 | 38 |
| Gobo13 | 12 | Gobo26 | 22 | Gobo39 | 34 |        |    |

Znači, prvo treba da se pojavi Gobo1, zatim, Gobo3, pa Gobo9, pa Gobo7, itd. Dakle, nakon što se podloga A1 u potpunosti osvijetli, treba da se pojavi Gobo1. Zatim, nakon jako kratkog vremena, treba da se pojavi Gobo3, itd. Vidimo da je potrebno da odredimo vrijeme koje je potrebno da Gobo1 sačeka nakon klika na dugme START da bi se pojavio na ploči.

Možemo probati dodati neko vrijeme čekanja u kod za kretanje Gobo1

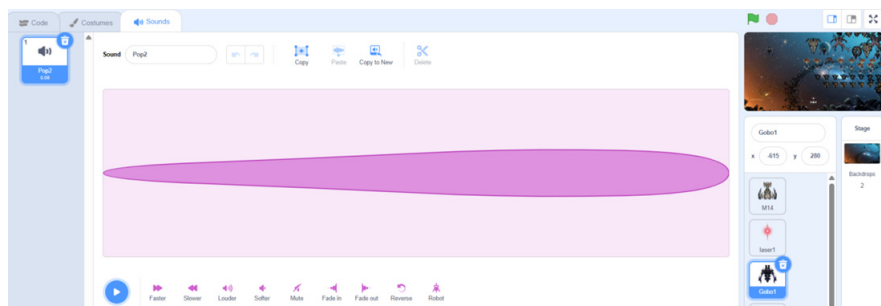




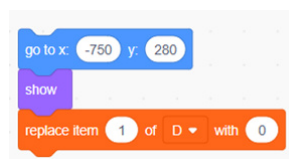
Ovdje smo dodali 5 sekundi čekanja prije pojave Gobo1. Ako probamo igru vidimo da ni to nije dovoljno, tj. vidimo da se Gobo1 i dalje pojavljuje prije nego se podloga A1 totalno osvijetli. Ovdje jednostavno metodom pokušaja i pogrešaka nalazimo da je optimalno vrijeme čekanja 7.5 sekundi. Sada, pošto želimo da se svaki od ostalih Gobo-a pojavljuje nešto kasnije u odnosu na onog prije njega, to ćemo na svakog sljedećeg Gobo-a dodavati po dodatnih 0.1 sekundi čekanja. Prilikom pojave Gobo-a potrebno je da čujemo neki zvuk, kao neki "zip". Taj zvuk treba da je jako kratak. Koristit ćemo zvuk po imenu

Pop2

Ovaj zvuk upload-ujemo na zvukove za sve Gobo-e. Uradimo to



Postavimo se ponovo na Gobo1 i na njegov kod. Zanima nas i dalje dio koji se odnosi na njegovo kretanje. Uočimo dio koda koji se nalazi unutar If naredbe: If item 1 of D = 1 then



Ovaj kod ćemo sada modifikovati u skladu sa već opisanim željama o pojavljivanju naših Gobo-a. Jasno, prvo ćemo provesti modifikaciju za Gobo1 a onda za sve ostale Gobo-e, i to idući redom onako kako bi ti Gobo-i trebali da se pojavljuju.

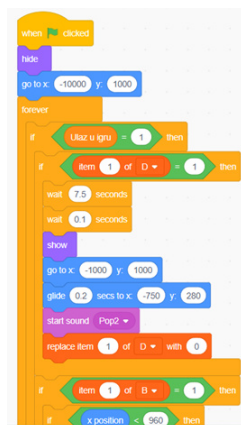
Ovim redoslijedom možemo lahko da brojimo dodatne sekunde čekanja za pojavljivanje Gobo-a. Naime, ovim redom je dovoljno da na svakog sljedećeg Gobo-a dodamo još 0.1 sekundi čekanja. Posljednji kod izdvojimo iz pomenute naredbe If prije njegove modifikacije. Ovdje još treba da se prisjetimo da smo sve Gobo-e na početku razvrstali tako da se oni inicijalno nalaze na nekim svojim početnim lokacijama. Tako, Gobo1 smo stavili na lokaciju (-10000,1000). Možemo ovo zvati lijeva lokacija (zbog minusa). Dalje, Gobo5 smo stavili na lokaciju (0,10000). Ovo ćemo zvati gornja lokacija. Isto tako, Gobo6 smo stavili na lokaciju (10000,1000). Zvat ćemo je desna lokacija (zbog plusa). Preciznije, sve Gobo-e smo postavili na ove tri lokacije. Neki su tako lijevo, neki su gore, neki su desno. Ustvari, one Gobo-e koje želimo da se pojavljuju s lijeva smo stavili lijevo, one Gobo-e koje želimo da se pojavljuju odozgo smo stavili gore, a one Gobo-e koje želimo da se pojavljuju zdesna smo stavili desno. Pri tome su ove lokacije (lijevo, gore, desno) prilično daleko (veliki su brojevi). Dakle, inicijalne lokacije su poprilično daleko od naše ploče za igru. Tako, prije nego "pojavimo" Gobo-e, potrebno je da ih prvo malo približimo sa inicijalnih udaljenih lokacija. Prilikom približavanja i dalje je potrebno da ih zadržimo na istim stranama, samo bliže (oni koji su bili lijevo i dalje da budu lijevo, oni koji su bili gore da i dalje budu gore, oni koji su bili desno da i dalje budu desno). Tako, Gobo-e koji su lijevo ćemo približiti na lokaciju (-1000,1000), Gobo-e koji su gore ćemo približiti na lokaciju (0,2000), te Gobo-e koji su desno ćemo približiti na lokaciju (1000,1000). Nakon toga ćemo ih "pojaviti", te pustiti da klize (glide) sa približene lokacije na odgovarajuću poziciju na ploči, da bi stvorili utisak njihovog dolaska. Tako, posljednji kod za Gobo1, modifikujemo u kod



Tako, za Gobo1, nakon klika na dugme START sačekamo 7.5 sekundi, i onda još dodatnih 0.1 sekundi (za sljedećeg Gobo-a, tj. za Gobo3 čekat ćemo dodatnih 0.2 sekundi, za Gobo-a nakon njega, tj. za Gobo9 čekat ćemo dodatnih 0.3 sekundi, itd).

Nakon završenog čekanja pojavimo Gobo-a (bez obzira što je on još uvijek na udaljenoj lokaciji (-10000,1000)), približimo ga na lokaciju (-1000,1000). Sa ove lokacije ga relativno brzo (0.2 sekunde) klizeći postavimo na početnu lokaciju na ploči (-750,280). Uz to, treba da se čuje zvuk Pop2, i to samo kao start, da ne bi došlo do bilo kakvog čekanja. Ostatak koda je isti kao ranije, tj. naredbu po kojoj item 1 liste D mijenjamo u 0 ostaje isti. Ovaj kod, prije uklapanja (u If naredbu iz koje smo uzeli nemodifikovanu verziju), najbolje je prvo zalijepiti na sljedećeg Gobo-a koji treba da se pojavi, tj. na Gobo3, da bismo sebi olakšali modifikaciju koda za Gobo3, pa onda taj modificirani kod za Gobo3 (prije ukplapanja u njegovu If naredbu) zalijepiti na sljedećeg Gobo-a, tj. na Gobo9, itd.

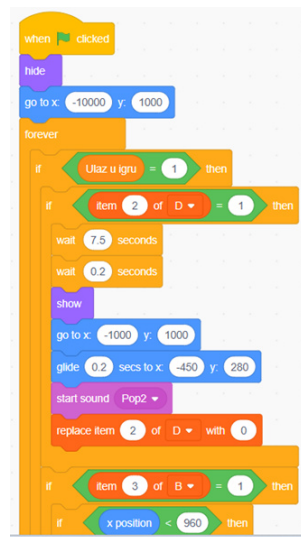
Uklapljeni kod za Gobo1 sada ima oblik



Kod koji smo zalijepili na Gobo3, nakon modifikacije treba da ima oblik

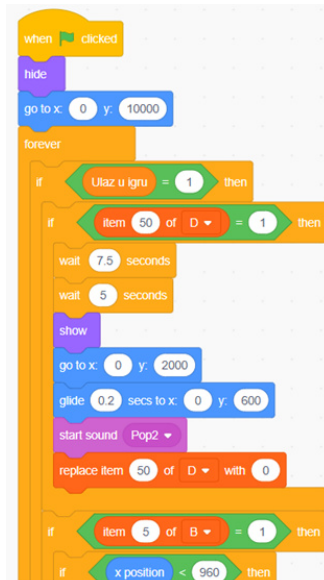


Kod koji se trenutno nalazi unutar odgovarajuće If naredbe izbacimo i zamijenimo ga upravo navedenim kodom. Tako, uklopljeni kod za Gobo3 sada ima oblik



Opisani postupak modifikacije nastavljamo za Gobo-e koje slijede, tj. za Gobo9, Gobo7, itd (držeci se tabele pojavljivanja). Uradimo to i dosnimimo igru.

Ako pokrenemo igru da vidimo kako funkcioniše, vidjet ćemo da se neprijateljske letjelice zaista redaju onako kako smo zamislili, pri čemu se čuje odgovarajući zvuk. Ono što odmah možemo vidjeti da nije uredu je to da letjelice koje se razmjesti na svoje pozicije, ne čekaju ostale da budu razmještene pa da počnu da se kreću desno- lijevo, nego njihovo kretanje počinje neposredno nakon razmještanja. Jasno, ovakav razvoj događaja nam ne odgovara. Letjelice se razmještaju, odmah kreću, dakle mi- ješaju, preklapaju. itd. Potrebna nam je logika po kojoj treba da prevaziđemo dobijeni fenomen. Naime, nama bi trebalo da se sve neprijateljske letjelice prvo razmjesti, a tek nakon razmještanja da zajedno počnu da se kreću. Drugim riječima, svi Gobo-i bi trebali da sačekaju dok se i posljednji među njima, tj. Gobo5, ne razmjesti, pa tek onda da svi zajedno počnu da se kreću. Pošto bi svi oni zajedno trebali da počnu da se kreću, to bi nam trebao neki brojač, tj. neki vid odbrojavanja, nakon čijeg bi odbro- javanja sve letjelice trebale da se sinhronizuju u smislu početka zajedničkog kretanja. Potrebno je da razmislimo koliko bi to odbrojavanje trebalo da traje. Pogledajmo kod kretanja zadnjeg Gobo-a koji se razmjesti na ploču. To je Gobo5. Njegov kod kretanja je



Vidimo da je ukupan broj sekundi čekanja za Gobo5 jednak  $7.5 + 5 = 12.5$  sekundi.

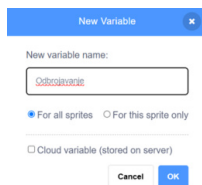
Drugim riječima, nakon klika na dugme START, čekamo 12.5 sekundi da se Gobo5 razmjesti na ploču.

Zašto posmatramo konkretno Gobo5 ?

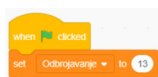
Zbog toga što taj Gobo najduže čeka da bude razmješten na ploču. Naime, svi Gobo-i prije njega bivaju ranije razmješteni na ploču. Jasno, ako želimo da pravimo neko odbrojavanje onda se moramo ravnati prema Gobo-u koji je zadnji razmješten na ploču. U našem slučaju to je Gobo5. Pošto se do njegovog razmještanja čeka 12.5 sekundi, onda je jasno da odbrojavanje o kojem govorimo (odbrojavanje nakon kojeg treba svi neprijatelji da počnu da se kreću), treba da traje nekih 13 sekundi. Dakle treba nam neka promjenjiva, kojoj ćemo vrijednost postaviti na 13, i koja će nakon klika na dugme START da počne da otkucava unazad, tj. od 13 prema 0. U momentu kada joj vrijednost postane 0, tada svi neprijatelji treba da počnu da se kreću. Dakle, dokle god je promjenjiva o kojoj govorimo različita od nule, svi Gobo-i treba da čekaju.

Na ovaj način smo opisali algoritam, tj. logiku koja nam je potrebna, i koju sada treba da pretočimo u kod. Promjenjivu o kojoj govorimo ćemo zvati Odbrojavanje.

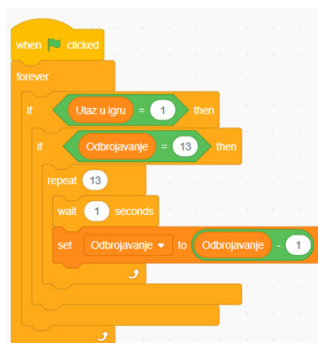
Idemo na varijable, i definišemo varijablu (promjenjivu) po imenu Odbrojavanje



Ostavimo je za sada da bude aktivna na ekranu igre (aktivna kvakica), da možemo pratiti ponašanje letjelica. Idemo na podloge i unesemo kod

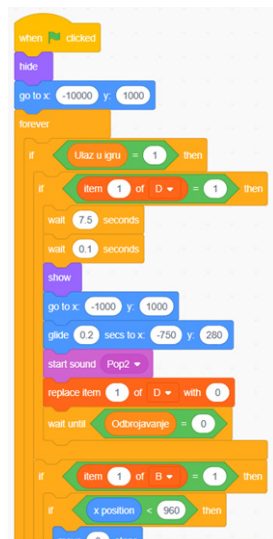


Dakle, inicijalno postavimo vrijednost varijable Odbrojavanje da bude jednaka 13. Kada da počne odbrojavanje do nule ? Jasno, onda kada kliknemo na dugme START. Šta nam znači da smo kliknuli na dugme START ? Govore nam promjenjive Ulaz u igru i PraviUlaz čije vrijednosti postaju jednake 1 nakon klika na START. Nama je svejedno koju ćemo od ove dvije varijable sada iskoristiti. Možemo npr. iskoristiti varijablu Ulaz u igru. Tako, ako kliknemo na dugme START, tj. ako je varijabla Ulaz u igru jednaka 1, onda treba da počne odbrojavanje unazad, tj. vrijednost promjenjive Odbrojavanje treba da svake sekunde smanjujemo za 1, dok ne dođemo do vrijednosti 0. Unosimo tako sljedeći kod na podloge

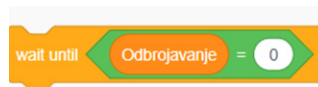


Mi ne znamo kada će se naš klik na START dogoditi. Ne znamo kada će se taj događaj odigrati. Zbog toga koristimo beskonačnu petlju forever. Ako kliknemo na dugme START, tj. ako vrijednost varijable Ulaz u igru postane jednaka 1, onda odbrojavanje treba da počne. Osim toga, dodatno, odbrojavanje može početi samo ako je vrijednost varijable Odbrojavanje jednaka 13. Nema smisla da počinjemo odbrojavanje ako je vrijednost varijable Odbrojavanje jednaka 7 ili 5 i sl. Zbog toga ide dodatni uslov da je vrijednost za Odbrojavanje jednaka 13. Ako su navedena dva uslova zadovoljena, onda možemo početi odbrojavanje. Njega jednostavno programiramo kao jednu petlju od 13 iteracija, gdje u svakoj iteraciji sačekamo jednu sekundu a nakon toga vrijednost varijable Odbrojavanje smanjimo za 1. Dakle, nakon 13 sekundi, vrijednost varijable Odbrojavanje će biti jednaka 0.

Ako pokrenemo igru vidjet ćemo da je vrijednost varijable Odbrojavanje jednaka 13 sve dok ne kliknemo na dugme START. Nakon tog klika počinje odbrojavanje. Takođe, možemo uočiti da se zadnja neprijateljska letjelica, tj. Gobo5, postavi na ploču momenat (0.5 sekundi) prije nego vrijednost varijable Odbrojavanje postane jednaka 0. Dakle, logika je ta da sada iskoristimo vrijednost 0 varijable Odbrojavanje da sinhronizujemo početak kretanja neprijateljskih letjelica. Dovoljno je u kodu za kretanje neprijateljske letjelice (svake od njih 50) zadati naredbu po kojoj letjelica treba da sačeka (prije nego počne sa kretanjem) dok vrijednost varijable Odbrojavanje ne postane 0. Modifikovani kod za kretanje Gobo1 sada ima oblik



Dakle, dovoljno je na kraj naredbe If (koju smo cijelo vrijeme razmatrali) dodati naredbu



Ovo uradimo za sve Gobo-e (bez ikakvih modifikacija). Sačuvamo izmjene i probamo igru.

## 7.1 Riješeni zadaci

◇ **Zadatak 7.1.1** Korisnik unosi godinu. Potrebno je provjeriti da li je ta godina prestupna (godina je prestupna ako je djeljiva sa 4 i nije djeljiva sa 100, ako se desi da je djeljiva sa 100 onda da bi bila prestupna mora biti djeljiva i sa 400). Program treba ispisati poruku da li je unesena godina prestupna ili nije.

**Rješenje:**

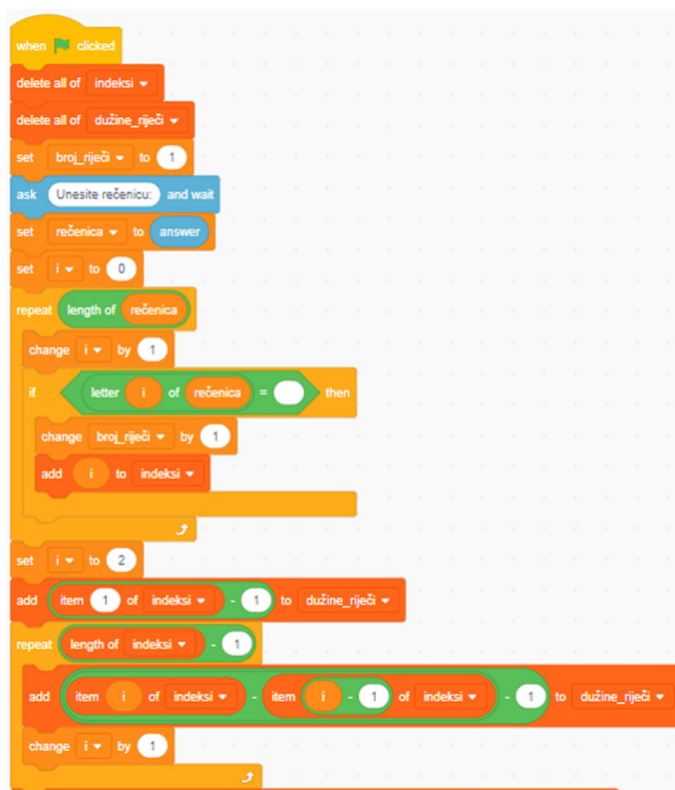


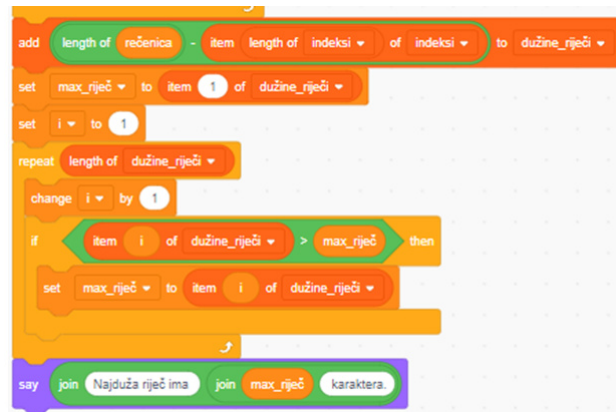
◇ **Zadatak 7.1.2** Korisnik unosi rečenicu. Potrebno je pronaći najdužu riječ u toj rečenici i ispisati koliko karaktera ona ima.

**Rješenje:** Na početku programa potrebno je izbrisati sve vrijednosti iz listi: "indeksi" i "dužine\_riječi", kako bi se osiguralo da program radi ispravno prilikom svakog



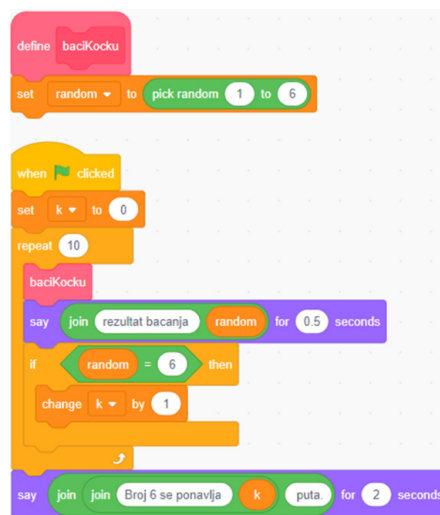
pokretanja. Lista "indeksi" koristi se za bilježenje pozicija svih razmaka u unesenoj rečenici, dok lista "dužine\_riječi" služi za pohranu dužina riječi u toj rečenici. Nakon što korisnik unese rečenicu, program prolazi kroz svako slovo, i kad naiđe na razmak, zabilježi njegovu poziciju u listu "indeksi". Tako se određuje gdje počinju i završavaju riječi. Broj riječi u rečenici određuje se na osnovu broja razmaka - pretpostavlja se da postoji barem jedna riječ, a zatim se broj riječi povećava za svaki pronađeni razmak. U narednom dijelu programa, pomoću prikupljenih pozicija razmaka, izračunava se dužina svake riječi: dužina prve riječi se računa od početka rečenice pa do prvog razmaka, dužina svake naredne kao dužina između dva razmaka, a dužina posljednje riječi kao dužina od posljednjeg razmaka pa do kraja rečenice (ne uzimamo u obzir znakove interpunkcije). Te dužine se upisuju u listu "dužine\_riječi". Konačno, program prolazi kroz tu listu i pronalazi najveću vrijednost, koja predstavlja onda broj karaktera najduže riječi. Ova vrijednost se zatim ispisuje kao rezultat.





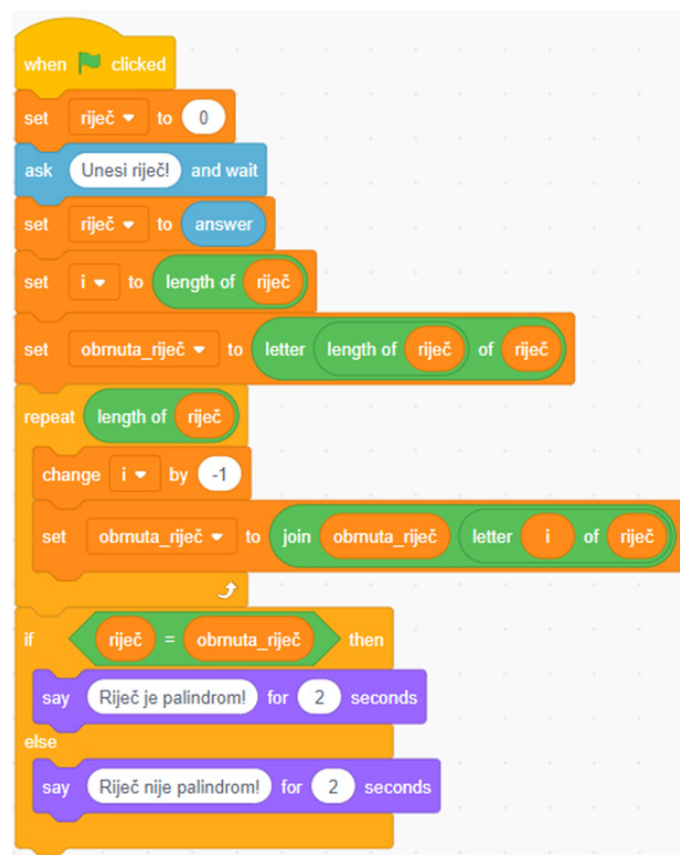
◇ **Zadatak 7.1.3** Potrebno je simulirati bacanje kocke 10 puta. Nakon svakog bacanja ispisuje se rezultat (broj od 1 do 6). Na kraju programa, ispisuje se koliko puta se pojavio broj 6. Za bacanje kocke koristiti vlastiti blok "baciKocku".

**Rješenje:**

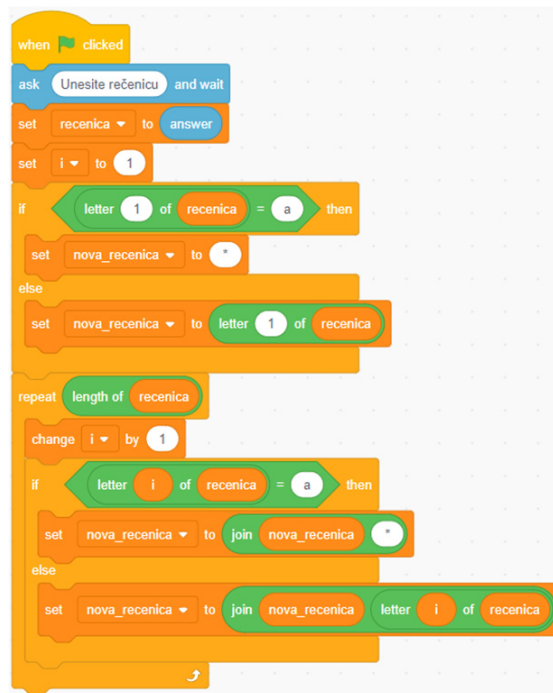


◇ **Zadatak 7.1.4** Korisnik unosi riječ. Potrebno je provjeriti da li je riječ palindrom, tj. da li se čita isto s lijeva na desno i zdesna na lijevo. Program treba ispisati odgovarajuću poruku u zavisnosti od rezultata provjere.

**Rješenje:**



◇ **Zadatak 7.1.5** Korisnik unosi rečenicu. Potrebno je napraviti novu rečenicu u kojoj će svako slovo "a" biti zamijenjeno zvjezdicom "\*", dok će ostali karakteri ostati nepromijenjeni. Rezultat treba biti nova, opisana rečenica.

**Rješenje:****7.2 Zadaci za vježbu**

♣ **Zadatak 7.2.1** Potrebno je simulirati bacanje dvije kocke istovremeno. Program treba ispisati vrijednosti dobijene na kockama i izračunati njihov zbir. Ako se na obje kocke dobije isti broj (npr. 3 i 3), program treba ispisati poruku: "Dupla! Bacanje se ponavlja.", te ponoviti bacanje sve dok dobijeni brojevi ne budu različiti. Na kraju se ispisuje zbir dobijenih vrijednosti posljednjeg bacanja.

♣ **Zadatak 7.2.2** Korisnik unosi rečenicu. Potrebno je provjeriti da li rečenica ima tačno pet riječi i da li se završava tačkom. Ako oba navedena uslova nisu zadovoljena, ispisuje se poruka: "Rečenica nije ispravna." Ako oba navedena uslova jesu zadovoljena, ispisuje se: "Rečenica je pravilno oblikovana."

## 8 Naprednije modifikacije objekata

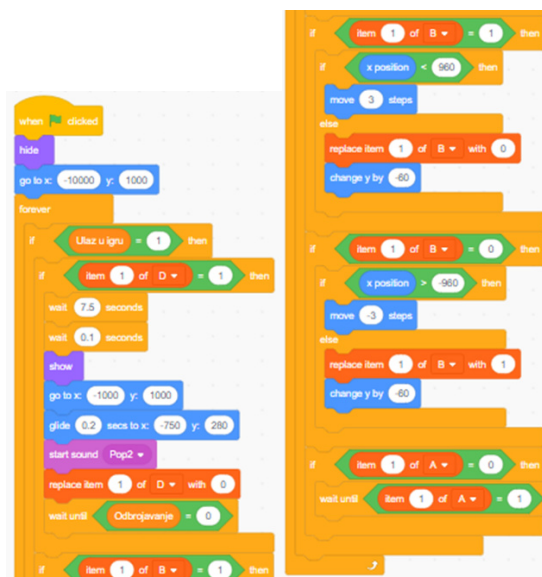
Primijetimo da u momentu kada eliminišemo sa ploče sve neprijatelje, ništa novo se ne dogodi. Naime, nismo ništa ni iskoristili dodatno što bi dalo instrukcije igri da uradi nešto novo. Znamo da bi nakon eliminacije svih neprijatelja trebali da pređemo na novi nivo igre koji bi bar podrazumijevao ponovni raspored neprijatelja na ploču. Općenitije, podrazumijevao bi i mnogo više od toga: da imamo napravljen brojač poena koji bi evidentirao rast poena eliminacijom neprijatelja, da je uveden brojač nivoa koji bi se povećavao eliminacijom kompletne garniture neprijatelja, da imamo promjenu melodije pojavom novog nivoa, da imamo promjenu podloge dostizanjem novog nivoa, da neprijatelji eventualno mijenjaju boju dolaskom na novi nivo koja bi mogla biti usklađena promjenom podloge nivoa. Bavit ćemo se u nastavku adresiranim pitanjima.

Prvo što uočavamo jeste to da se za sada nakon eliminacije svih neprijatelja sa ploče oni ne pojavljuju ponovo na njoj. Trebalo bi da se nakon njihove eliminacije oni ponovo rasporede na ploču da bi igra mogla da ide dalje. Zašto se to za sada ne dešava, i kako da riješimo to pitanje. Dovoljno je da detaljnije pogledamo kod nekog od Gobo-a, npr. kod za Gobo1. Postavimo se na kod za Gobo1, i posmatramo desni kod, tj. kod koji se odnosi na eliminaciju tog Gobo-a. Taj kod je sljedeći



Vidimo da u momentu kada Gobo1 dodirne laser, da item 1 liste A postavljamo na 0 (što je znak da je Gobo1 eliminisan). Isto važi i za ostale Gobo-e. Tako, kada Gobo2 dodirne laser, mi item 2 liste A postavljamo na 0. Šta nam je onda pokazatelj da su svi Gobo-i eliminisani ? To je onda činjenica da su svih 50 elemenata liste A postavljeni da budu jednaki 0, tj. da lista A ne sadrži niti jednu jedinicu (1) (znamo da smo svih 50 elemenata liste A inicijalno postavili da budu jednaki 1). Znači činjenica da lista A ne sadrži niti jednu jedinicu govori da su svi Gobo-i eliminisani sa ploče, i da bi ih zbog toga ponovo trebalo rasporediti na ploču.

Pogledajmo onda i lijevi kod za Gobo1, tj. kod koji se odnosi na njegov raspored na ploču, te za njegovo kretanje po ploči. U pitanju je kod

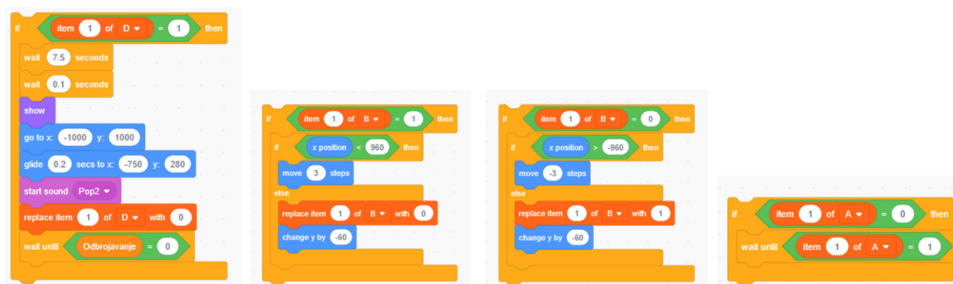


Radi jednostavnosti izražavanja, možemo ga trenutno zvati Lijevi kod, dok ćemo prethodni kod zvati Desni kod. Uočimo da je glavnina koda u Lijevom kodu sadržana unutar strukture



Kako smo ranije isticali, forever petlja čeka "pogodan" događaj, jer ne znamo kada će se on dogoditi, a pod tim događajem podrazumijevamo ili klik na dugme START pa da igra uopšte otpočne sa radom, ili neki drugi događaj koji će omogućiti da igra nastavi sa radom. Znamo da klik na dugme START znači postavljanje promjenjive Ulaz u igru na 1, što onda omogućava da posljednji uslov If bude zadovoljen i da igra započne sa radom (raspoređivanje letjelica na ploču a nakon toga i njihovo kretanje po njoj). Isto tako, neki drugi događaj (kojeg još uvijek nemamo) bi trebao da inicira ponovni raspored letjelica na ploču a onda i njihovo kretanje.

Pogledajmo koje su to naredbe sadržane u posljednjoj strukturi forever-if. U ovoj strukturi imamo 4 bloka



Prvi blok se odnosi na inicijalni raspored letjelice Gobo1 na ploču popraćen adekvatnim zvukom. Primijetimo da prilikom inicijalnog rasporeda koristimo listu D. Rekli smo da lista D služi samo da nam omogući inicijalni raspored letjelice jednom ali ne i više puta. Naime, ako bi letjelica nastavila u nedogled da se inicijalno raspoređuje, imali bismo utisak da se letjelica i ne pomjera, tj. da samo stoji na inicijalnoj poziciji. Za prevazilaženje opisanog problema smo i koristili listu D, tj. iz toga razloga smo je i uveli. Govorimo o smislu da 50 elemenata liste D inicijano postavimo da bude jednako 1 (svaka jedinica za po jednog Gobo-a). Tako, ako je item 1 liste D jednak 1, to nam je znak da želimo da Gobo1 razmjestimo na ploču. Čim ga razmjestimo, item 1 liste D postavljamo na 0 (da se raspoređivanje na ploču ne bi odigravalo u nedogled). Postavlja se pitanje: kada ima smisla da ponovo postavimo item 1 liste D da bude jednak 1? Logika bi bila ta da se to desi nakon što smo Gobo1 eliminisali sa ploče, jer ako je Gobo1 eliminisan, onda bi to trebalo da znači da ga u jednom momentu, tj. onda kada svi Gobo-i budu eliminisani, ponovo možemo pozvati i rasporediti na ploču (kao i sve ostale Gobo-e). Naime, u momentu kada je Gobo1 eliminisan, potrebno je da elemente koji se odnose na njega resetujemo, tako da budu spremni da u adekvatnom momentu budu ponovo aktivirani. Taj momenat je momenat eliminacije i posljednjeg

Gobo-a sa ploče. Ako pogledamo Desni kod, onda tamo možemo vidjeti da smo u njemu i postupili na upravo opisani način. Naime, tamo smo (nakon što je Gobo1 dodirnuo laser), tj. nakon što je Gobo1 eliminisan, item 1 liste D postavili da bude jednak 1 (da bude spreman za eventualni ponovni inicijalni raspored za Gobo1). Na ovaj način smo detaljno opisali strukturu prvog bloka (od četiri navedena) a u vezi sa Desnim kodom.

Drugi blok se odnosi na kretanje za Gobo1 udesno (i promjenu toka kretanja).

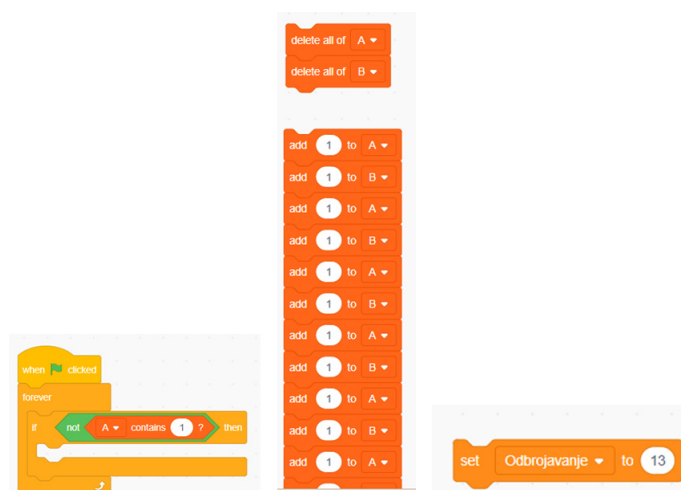
Treći blok se odnosi na kretanje za Gobo1 ulijevo (i promjenu toka kretanja).

Posmatrajmo napokon četvrti blok. U ovom bloku se pitamo da li je item 1 liste A jednak 0, tj. da li je Gobo1 eliminisan sa ploče. Vidimo da ako to jeste slučaj, da sljedeća naredba glasi da kod sačeka sa izvršavanjem sve dok item 1 liste A ne postane ponovo jednak 1. Dakle, nakon što je Gobo1 eliminisan, čitav njegov Lijevi kod čeka sa radom (kompletna 4 navedena bloka). To znači da inicijalni raspored za Gobo1 na ploču čeka sa radom, da njegovo kretanje desno-lijevo čeka sa radom. Šta se čeka i zašto ? Čeka se da item 1 liste A ponovo postane jednak 1, tj. čeka se da Gobo1 ponovo postane živ. Ista priča važi i za ostale Gobo-e. Dakle, ako je neki Gobo eliminisan, njegov Lijevi kod čeka sa radom sve dok posmatrani Gobo ponovo ne postane živ.

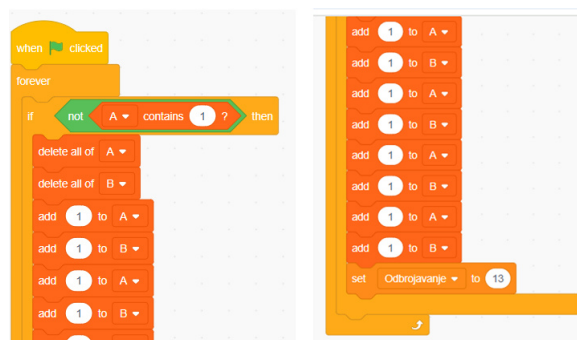
Provedeno razmatranje nam daje ideju o tome šta treba da radimo u nastavku. Bilo koji Gobo, koji je eliminisan, čeka da ga oživimo, i započnemo proceduru njegovog ponovnog rasporeda na ploču. Ovo važi za bilo kojeg Gobo-a. Kada ta procedura ponovnog rasporeda treba da započne sa radom ? Treba da započne nakon što smo eliminisali i posljednjeg Gobo-a sa ploče. Kako smo upravo obrazložili u četvrtom bloku Lijevog koda, ako je Gobo eliminisan, odgovarajući item liste A je postavljen na 0, i Lijevi kod čeka sve dok dotični item liste A ponovo ne postane 1. Dakle, da bi Lijevi kod ponovo počeo sa radom, moramo pomenuti item liste A postaviti da bude jednak 1. Drugim riječima, da bi neprijateljske letjelice ponovo rasporedili na ploču, moramo (nakon što su svi elementi liste A postali jednaki 0, tj. nakon što zaključimo da lista A ne sadrži više niti jednu jedinicu), napraviti kod koji će ponovo elemente liste A postaviti da budu jednaki 1. U momentu kada ti elementi postanu jednaki 1 (njih 50), Lijevi kodovi svih Gobo-a će nastaviti sa radom, tj. njihov raspored na ploču, a onda i kretanje desno-lijevo će ponovo da budu reaktivirani. Postavljanje elemenata liste A da budu ponovo jednaki 1 možemo uraditi na neutralnom mjestu, tj. na podlogama. Idemo na kod podloga i unosimo sljedeće kodove koje na kraju uklopimo u jedan (kodove na desnoj strani nadovežemo u jedan, i uklopimo u uslov If



na lijevoj strani)



Znači, početak i kraj uklopljenog koda imaju oblik



Primijetimo da ovdje ne resetujemo samo elemente liste A da budu jednaki 1, nego i elemente liste B. Naime, kada resetujemo elemente liste A da budu jednaki 1, to znači da želimo da ponovo započne raspored letjelica na ploču. Jasno je onda da to inicira da i njihovo kretanje treba da krene ispočetka, tj. nadesno. Kako znamo lista B je lista kretanja letjelica, pa je potrebno i nju tako resetovati, tj. staviti njene elemente ponovo da budu jednaki 1. Naime, nakon što smo eliminisali sve Gobo-e, mi ne znamo na

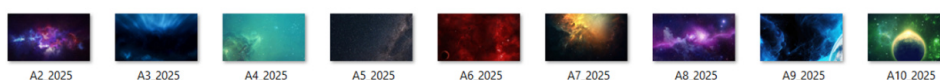
kojem mjestu se desila njihova eliminacija, tj. ne znamo da li se ona desila dok su se oni kretali desno ili lijevo, odnosno ne znamo koju vrijednost su zauzeli elementi liste B (1 ili 0), tako da je onda jasno da reset kretanja podrazumjeva ponovno postavljanje svih elemenata liste B da budu jednaki 1. Pošto smo inicijalno raspoređivanje letjelica uskladili sa odbrojavanjem promjenjive Odbrojavanje od vrijednosti 13 ka 0 unazad, to reset podrazumjeva i ponovno postavljanje vrijednosti promjenjive Odbrojavanje na 13.

Sačuvamo igru i probamo je.

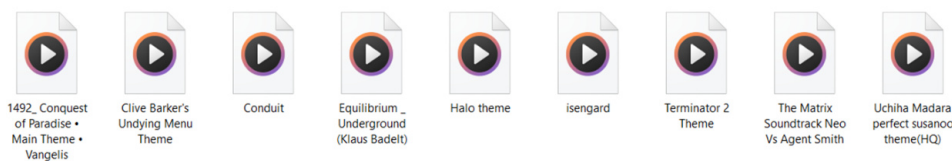
Primijetimo da nakon što smo eliminisali sve Gobo-e, promjenjiva Odbrojavanje ponovo poprima vrijednost 13, nakon čega kreće odbrojavanje, a kada njena vrijednost postane jednaka 0, letjelice su se ponovo rasporedile na ploču.

Promjenjivu Odbrojavanje smo inicijalno postavili na vrijednost 13 da bi imali vremena ne samo da rasporedimo letjelice nego da promijenimo i podlogu i melodiju, uz adekvatne promjene osvjetljenja i jačine zvuka. Sada je jasno da je potrebno doraditi i to da svaka nova eliminacija svih letjelica sa ploče povlači za sobom i promjenu pozadine kao i promjenu melodije. Potrebne su nam tako nove melodije i nove podloge.

Koristit ćemo sljedeće podloge A2-A10



i sljedeće melodije



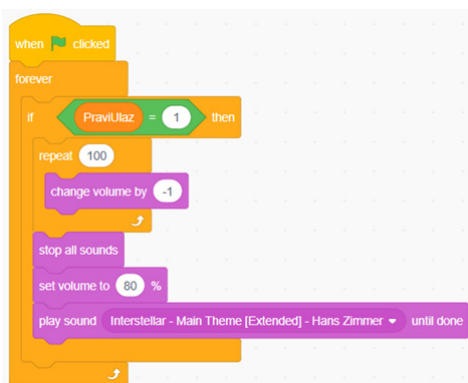
Redoslijed podloga je opisan rednim brojem u imenu podloge, dok ćemo navedene melodije koristiti po redu:

2. Halo theme
3. Uchiha Madara perfect susanoo theme(HQ)
4. Conduit
5. Equilibrium \_ Underground (Klaus Badelt)
6. 1492\_ Conquest of Paradise • Main Theme • Vangelis
7. The Matrix Soundtrack Neo Vs Agent Smith
8. Terminator 2 Theme
9. isengard
10. Clive Barker's Undying Menu Theme

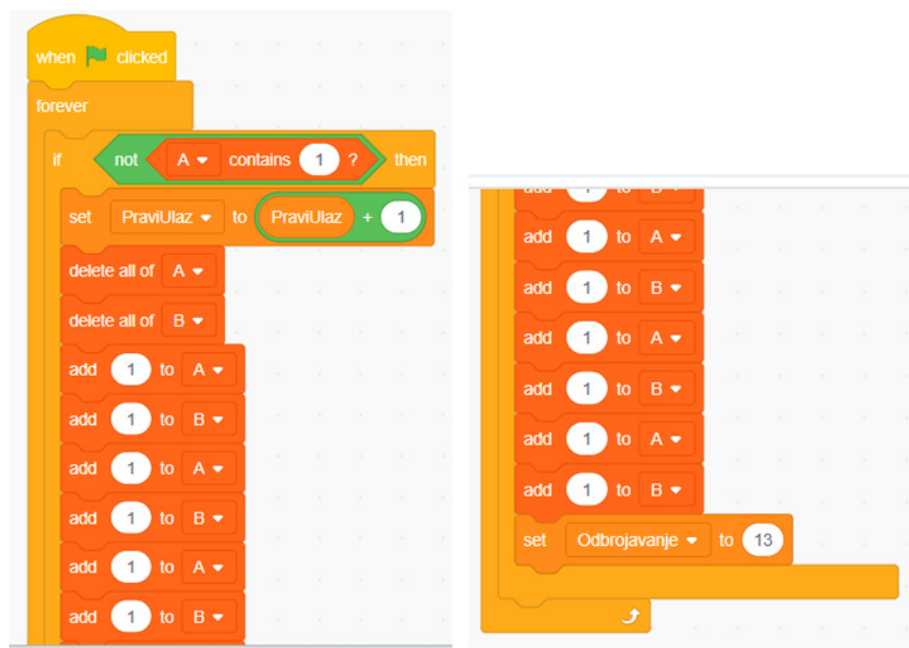
Ideja je da koristimo ukupno, tj. bar 50 različitih podloga uz 50 različitih melodija. Ono što nam predstavlja blagi problem je količina memorije koju tim činom zauzimamo. Zbog navedenog razloga koristit ćemo za sada 10 navedenih podloga i melodija. Navedene podloge i melodije upload-ujemo opisanim redom. Podloge obavezno konvertujemo u vektorski oblik. Melodijama ćemo ostaviti navedene nazive (Sound) dok ćemo nazive kostima navedenih podloga (Costume) primenovati u samo A2-A10.

Sačuvamo izmjene.

Da bi omogućili promjenu melodije prilikom prelaska na novi nivo, iskoristit ćemo već postojeću promjenjivu PraviUlaz. Sjetimo se da smo je već jednom iskoristili da postavimo novu melodiju nakon klika na dugme START. Naime, imali smo sljedeći kod



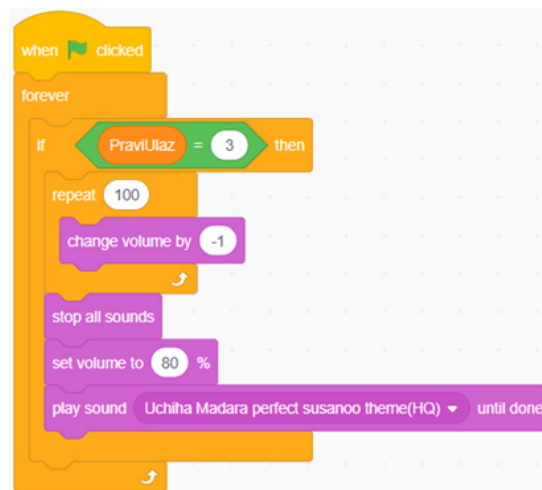
Ovaj kod nam je omogućavao da promijenimo uvodnu melodiju na melodiju Interstellar. Bitno je uočiti da smo pri tome koristili uslov: ako je PraviUlaz jednako 1. Naime, promjenjiva PraviUlaz je inicijalno postavljena na vrijednost 0, a kada kliknemo na dugme START postavljamo da joj je vrijednost jednaka 1. Kada jednom uđemo u igru putem dugmeta START, nama više nije neophodno da vrijednost promjenjive PraviUlaz bude jednaka 1. Možemo je mijenjati. Ideja bi sada bila da prilikom prelaska na novi nivo, svaki put povećamo vrijednost promjenjive PraviUlaz za 1, te da tu novu vrijednost iskoristimo za promjenu melodije (prelaz sa stare na novu melodiju). Zbog toga ćemo u prethodno uvedeni kod za reset lista A i B, dodati naredbu za uvećanje promjenjive PraviUlaz. Sada, taj, modifikovani kod ima oblik



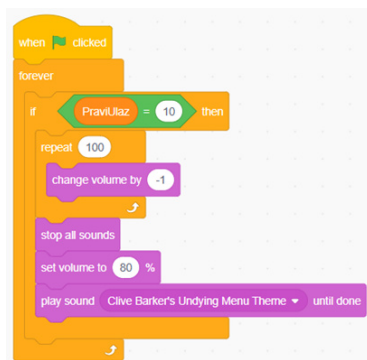
Primijetimo u kojem momentu povećavamo vrijednost promjenjive PraviUlaz za 1. To radimo onda kada u listi A više nema jedinica, što znači da to radimo onda kada su svi Gobo-i eliminisani sa ploče, odnosno radimo to onda kada treba da pređemo na novi nivo (što je upravo ono što nama treba). Znači, kada prelazimo na novi nivo, želimo da mijenjamo melodiju. Na kod podloga dodajemo sljedeći kod, odnosno kopiramo-zalijepimo već navedeni kod za Interstellar a onda ga modifikujemo u oblik



Uočimo da ovdje koristimo melodiju Halo. Uradimo onda isto sa ovim kodom. Kopiramo-zalijepimo ovaj kod za Halo na podloge, modifikujemo ga u oblik



Nastavljamo ovako dalje dok ne iskoristimo sve navedene melodije i to onim redom kako smo ih naveli. Posljednji kod će imati oblik  
Sačuvamo izmjene.



Ako pokrenemo igru i probamo je, vidjet ćemo da se unesene melodije zaista smjenjuju onako kako smo naredili, i to nakon svake eliminacije svih neprijatelja. Smjena je popraćena smanjenjem tona kako je predviđeno kodom. Ipak, uočavamo da promjenu melodija ne prati promjena podloga. Taj dio moramo sada doraditi.

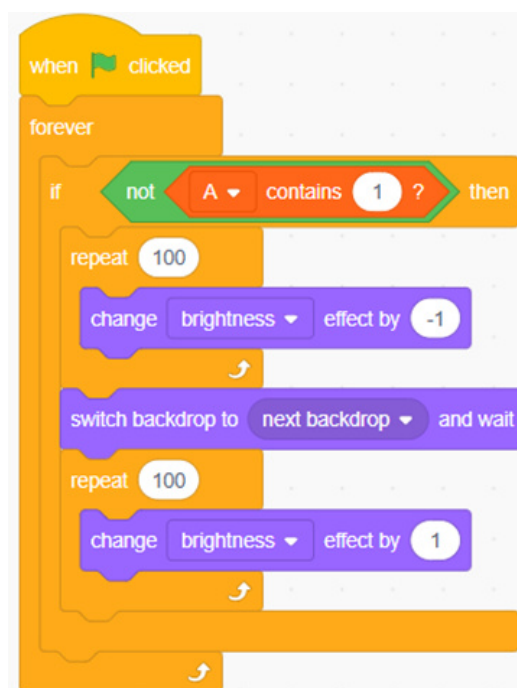
Sjetimo se da smo promjenjivu Ulaz u igru iskoristili da promijenimo inicijalnu podlogu igre zvanu "igrica", na podlogu A1. Naime, imali smo kod



Logika je bila ta da onda kada kliknemo na dugme START (a upravo tada mijenjamo vrijednost varijable Ulaz u igru sa vrijednosti 0 na vrijednost 1), promijenimo podlogu sa početne inicijalne "igrica", na podlogu A1. Pri tome dolazi do postupnog

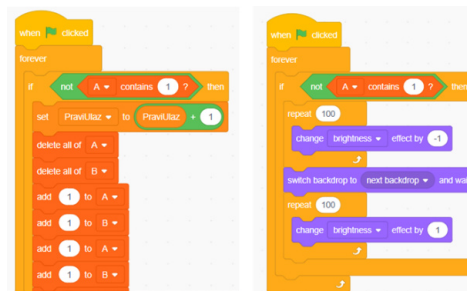
zatomnjenja podloge, promjene podloge, i onda ponovnog osvjetljenja.

Primijetimo da problem promjene podloge možemo gledati i nešto generalnije, tj. neovisno o varijabli Ulaz u igru ili nekoj drugoj varijabli. Naime, promjene podloga sa A1 na A2, pa sa A2 na A3, itd. možemo posmatrati vezano za eliminaciju Gobo-a sa ploče. Onda kada su svi Gobo-i eliminisani, promijenimo podlogu (na način kako smo tu promjenu izveli sa podloge "igrica" na podlogu A1). Dakle, onda kada lista A ne sadrži niti jednu jedinicu, pokrećemo promjenu podloge (dovoljno je da naredimo da se podloga prebaci na sljedeću podlogu). Tako, na kod podloga unosimo sljedeći kod (kopiramo-zalijepimo kod za prelaz sa podloge "igrica" na podlogu A1 i onda ga modifikujemo: obavezno izbacimo iz tog koda naredbu za zaustavljanje skripte, jer sada želimo da skripta vječno radi, tj. da prelazi sa A1 na A2, pa sa A2 na A3, itd)



Sačuvamo izmjene.

Probamo igru. Može se desiti da se stvari po pitanju podloga ne budu odvijale onako kako smo to željeli. Naime, za sada, imamo dva koda koja treba da se dese istovremeno, odnosno dva koda koja pozivamo zadovoljavanjem istog uslova. To su kodovi



Lijevi kod je kod za reset lista A i B, a desni kod je kod za opštu promjenu podloga. Oba ova koda rade onda kada je zadovoljen uslov da lista A ne sadrži niti jednu jedinicu. Ipak, ako pogledamo prvi od njih, tj. lijevi, onda u njemu vidimo da ako zadovoljimo pomenuti uslov, da momentalno nakon toga, elementi lista A i B se brišu, i postavljaju na jedinice, što znači da ako desni kod bar malo kasni sa izvršavanjem za lijevim, onda se on ne mora ni izvršiti, jer uslov da lista A ne sadrži niti jednu jedinicu više ne mora biti zadovoljen. Zbog ovog razloga, da bi se osigurali, dovoljno je u lijevom kodu staviti neko malo vrijeme čekanja prije postavljanja svih elemenata lista A i B na jedinice. Naime, dovoljno je staviti čekanje tipa 0.01 sekundi da bi desni kod imao vremena da započne sa izvršavanjem. Tako, bolja varijanta lijevog koda je sljedeća



Sačuvamo izmjene i testiramo igru. Stvari bi trebale da se odvijaju onako kako to želimo.

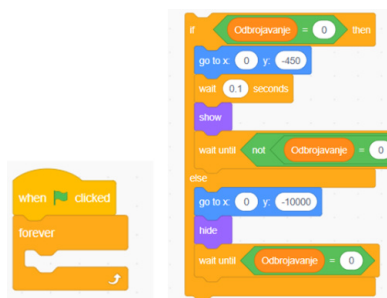
Ono što još možemo primijetiti je to da je naša letjelica M14 aktivna od samog



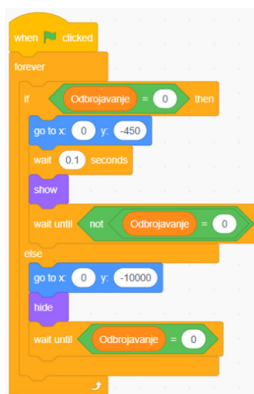
pokretanja igre. Isto kao što smo neprijateljske letjelice programirali tako da se pojavljuju tek nakon ulaska u igru, te da se redaju do momenta dok vrijednost promjenjive Odbrojavanje ne postane jednaka 0, tako bi trebala da funkcioniše i letjelica M14. Naime, nema smisla da ona bude aktivna prije nego se neprijatelji poredaju. Znamo da su se svi neprijatelji poredali onda kada vrijednost promjenjive Odbrojavanje postane jednaka 0. Tada bi trebala da se pojavi letjelica M14. Osim toga vrijednost promjenjive Odbrojavanje ponovo postaje različita od nule u momentu kada smo eliminisali sve neprijatelje sa ploče. Tada bi jasno i letjelica M14 trebala da bude neaktivna. Drugim riječima, letjelica M14 bi trebala da bude aktivna samo onda kada je vrijednost promjenjive Odbrojavanje jednaka nuli. U fazama u kojima je vrijednost promjenjive Odbrojavanje različita od nule, letjelica M14 bi trebala da bude sklonjena na neku za nju unaprijed rezervisanu lokaciju. Možemo koristiti npr. lokaciju (0,-10000). Osim toga, kao i do sada, kao startnu poziciju letjelice M14 možemo koristiti poziciju (0,-450). Idemo na kod letjelice M14 i kod kretanja modifikujemo dodavanjem naredbe za sakrivanje, tako da imamo sljedeći kod



Zatim, na kod letjelice M14, dodamo sljedeći kod (koji će regulisati njeno pojavljivanje na način kako smo upravo opisali)

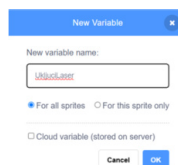


Imamo tako uslov If-then-else. Ako je Odbrojavanje jednako 0, onda postavimo letjelicu M14 na lokaciju (0,-450) na ploči, sačekamo 0.1 sekundu da se osiguramo da su se sve neprijateljske letjelice razmjestile, te omogućimo pojavljivanje letjelice M14. Ona treba da bude vidljiva dokle god imamo bar jednog neprijatelja na ploči, što znači da ona treba da bude vidljiva sve dok Odbrojavanje ne bude ponovo različito od nule. U suprotnom, tj. ako Odbrojavanje nije jednako 0, onda to znači ili da nismo ušli još u igru ili da smo eliminisali sve neprijatelje, pa tada letjelica M14 ne treba da bude na ploči, tj. treba da bude na rezervisanoj lokaciji (0,-10000). Tamo treba da se zadrži do momenta dok Odbrojavanje ne postane jednako 0. Uklopimo desni kod u forever petlju i dobijamo konačni kod

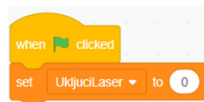


Sačuvamo izmjene i probamo igru.

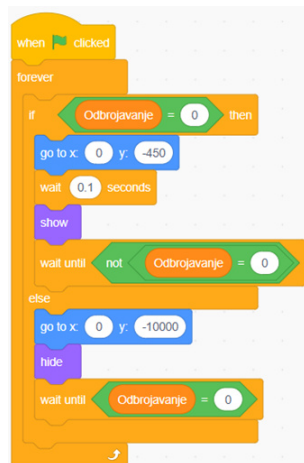
Primijetimo da iako smo riješili pitanje sklanjanja letjelice M14 sa ploče u momentima kada ona ne treba da bude na ploči, da laser i dalje radi u momentima kada ne bi trebao da radi. Potrebno je dakle da i laser doradimo u smislu koda tako da ne bude aktivan onda kada i letjelica M14 nije aktivna. Pitanje ćemo riješiti upotrebom prekidača (uključiti-isključiti). Definišimo novu promjenjivu po imenu UkljuciLaser



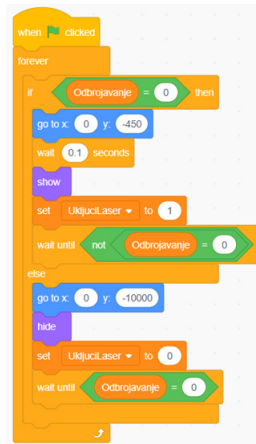
Inicijalno, vrijednost ove varijable ćemo postaviti da bude jednaka 0. Idemo dakle na kod podloga i unesemo kod



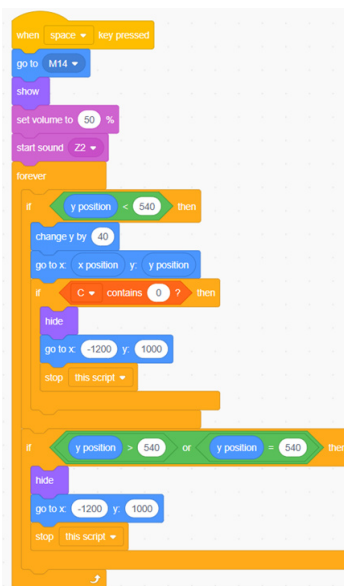
Vrijednost varijable UkljuciLaser jasno treba postaviti da bude jednaka 1 onda kada letjelica M14 treba da bude vidljiva na ekranu, a to su, kako smo već vidjeli, situacije kada je vrijednost varijable Odbrojavanje jednaka 0. Drugim riječima, rezon je da ako je vrijednost promjenjive Odbrojavanje jednaka 0, da onda vrijednost promjenjive UkljuciLaser treba da bude jednaka 1, te da ako je vrijednost promjenjive Odbrojavanje različita od nule, da onda vrijednost promjenjive UkljuciLaser treba da bude jednaka 0. Sjetimo se da smo na kodu letjelice M14 upravo imali dio koda gdje smo posmatrali slučajeve ako je Odbrojavanje jednako 0 ili jednako 1, tj. imali smo kod



Ovaj kod možemo iskoristiti da mijenjamo vrijednost varijable UkljuciLaser po potrebi. Tako, u slučaju da je Odbrojavanje jednako 0, stavljat ćemo da je UkljuciLaser jednako 1. U suprotnom, tj. ako je Odbrojavanje različito od 0, stavljat ćemo da je UkljuciLaser jednako 0. Tako, posljednji kod modifikujemo u kod



Potrebno je još da modifikujemo kod za laser. Sadašnji kod je



Potrebno je da ga doradimo na način da laser bude aktivan samo ako smo ušli u igru, tj. ako smo kliknuli na dugme START, odnosno ako je vrijednost varijable Ulaz

u igru jednaka 1, te ako je zadovoljen uslov da je vrijednost varijable UkljuciLaser jednaka 1. Modifikujemo tako navedeni kod sa dodatne dvije If naredbe u sljedeći kod (sve naredbe prethodnog koda izuzev prve, ubacimo unutar nove dvije If naredbe)



Sačuvamo izmjene i testiramo igru. Vidimo da ona zaista radi ono što smo i željeli da radi.

## 8.1 Riješeni zadaci

♦ **Zadatak 8.1.1** Postaviti pozadinu oblika kao na slici te dodati Sprite pod nazivom "Čarobni štapić" (takođe oblika kao na slici)

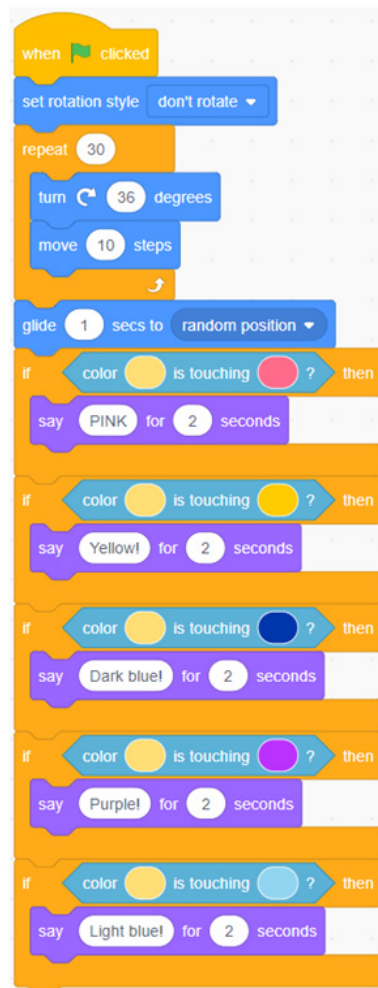


Napraviti program koji će pritiskom na zelenu zastavicu, uraditi sljedeće:

Štapićem napraviti tri mala kruga,

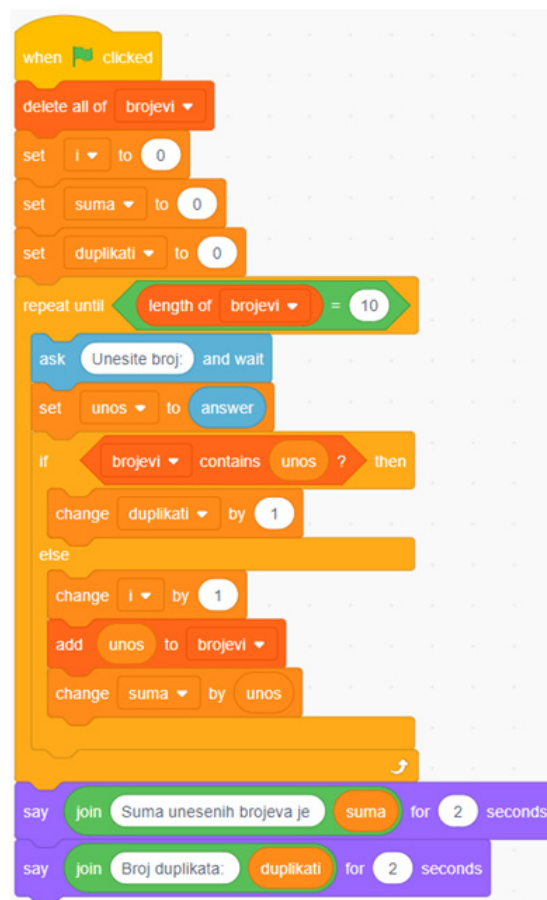
Otklizati štapić do nasumičnog mjesta na ekranu, te u zavisnosti od toga na koju boju pokazuje vrh štapića, ispisati naziv te boje koristeći "say" ("govori") blok.

**Rješenje:**



◇ **Zadatak 8.1.2** Napraviti program koji traži od korisnika da unese 10 brojeva. Sačuvati uneseni broj u listu samo ako taj broj ne postoji već u listi. Ispisati sumu unesenih brojeva te broj pokušaja unosa duplikata.

**Rješenje:**



◇ **Zadatak 8.1.3** Unijeti Sprite košarkaša po imenu Andie (iz nekog foldera), sa tri kostima: andie-a, andie-b, andie-d, te Sprite-ove lopte i koša (Sprite-ovi mogu biti oblika kao na slici)



Animirati lik košarkaša i lopte tako da:

Košarkaš udari loptom tri puta od pod (napraviti funkciju "vodi\_loptu" koja obavlja ovaj dio posla),

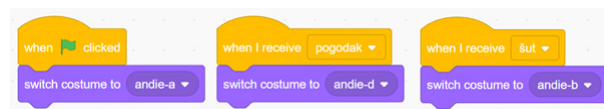
Šutira loptu prema košu (prilikom šuta kostim se promijeni u andie-b),

Lopta se kreće po zakrivljenoj putanji,

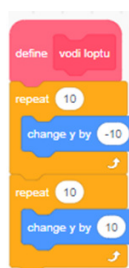
Kada lopta dođe do koša, pada na zemlju,

Prilikom pogotka, kostim košarkaša se mijenja u andie-d.

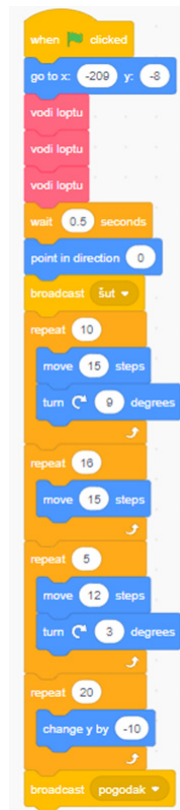
**Rješenje:** Kod za Andie-ja je:



Kod za loptu je:







## 8.2 Zadaci za vježbu

♣ **Zadatak 8.2.1** Postaviti pozadinu sa više boja i unijeti Sprite loptice. Napraviti program koji će nakon klikna na zelenu zastavicu:

Pomjeriti lopticu do sredine ekrana,

Nasumično odabrati pravac kretanja i obezbijediti kretanje loptice sve dok se ne dodirne zadata boja,

Nakon što loptica dodirne navedenu boju, ispisati poruku "Dodirnuta boja!" i zaustaviti kretanje.

♣ **Zadatak 8.2.2** Unijeti Sprite Metak i Sprite Meta koja je podijeljena u više obojenih zona (npr. plava, zelena, žuta, crvena). Program treba da radi 10 rundi. U svakoj rundi:

Računar nasumično bira jednu od ponuđenih boja i prikazuje poruku: "Pogodi (boja)!",

Korisnik ima 3 sekunde da metkom pogodi odabranu boju na meti,

Ako je pogođena odabrana boja, ispisuje se poruka: "Bravo, pogodak!" i broj poena se povećava,

U slučaju da metak ne pogodi odabranu boju ili ne pogodi metu, ispisuje se poruka: "Promašaj!".

Nakon 10 pokušaja, program prikazuje koliko je pogodaka korisnik ostvario.

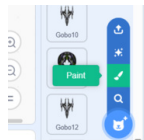
## 9 Povratne vrijednosti (u realnom vremenu)

U nastavku ćemo napraviti energiju našoj letjelici M14.

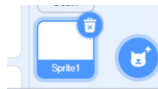
Znamo da su igre kod kojih glavni lik a i neprijatelji imaju neku vrstu energije HP (Health Points) koja sprečava da najbezazlenijim dodirima sa neprijateljima ili neprijateljskim oružjem lik gine, mnogo zanimljivije u odnosu na one koje nemaju tu osobinu.

Pretpostavimo da želimo napraviti energiju od 30 poena našoj letjelici M14.

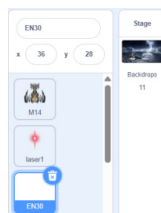
Idemo na Sprite-ove (Choose a Sprite) i na Paint da napravimo vlastiti Sprite energije



Dobijemo Sprite na dnu



Dat ćemo mu ime EN30 i premjestiti ga u gornji dio Sprite-ova, gdje se nalazi naša letjelica



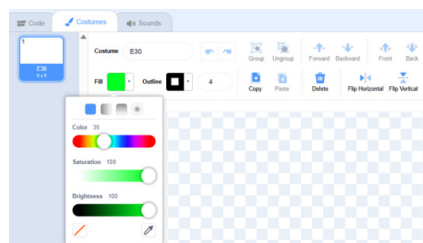
Na lijevoj strani dok smo u vektorskom obliku povećamo veličinu šeme (kvadratića) da možemo tačno odrediti 30 podioka koji su nam potrebni

## 9 POVRAT. VRIJEDN. (U REALNOM VREMENU)

---



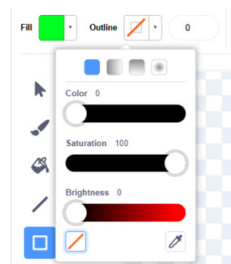
Damo ime kostimu E30, odaberemo zelenu boju (C35,S100,B100)



Idemo na Rectangle



Postavimo da je Outline opcija isključena, što znači da želimo da naš pravougaonik nema ivica

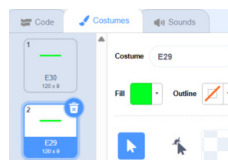


Nacrtamo pravougaonik dužine 30 kvadratića i širine 2 kvadratića

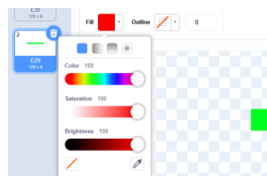


Ako imamo nekih odstupanja prilikom crtanja, onda odemo na Select, tako da možemo naknadno promijeniti veličinu pravougaonika, tj. podesiti ga da izgleda onako kako želimo. Ovo možemo uraditi jer se nalazimo cijelo vrijeme u vektorskom obliku.

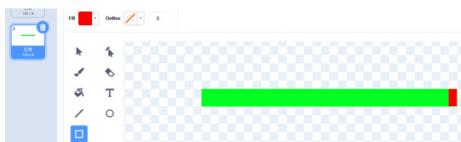
Dupliramo sada ovaj kostim, i novom kostimu damo ime E29



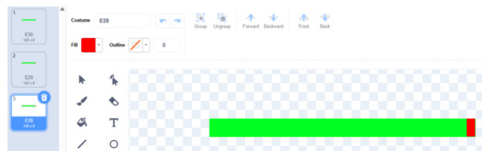
Dok smo na E29, podesimo crvenu boju (za popunjavanje) (C100,S100,B100)



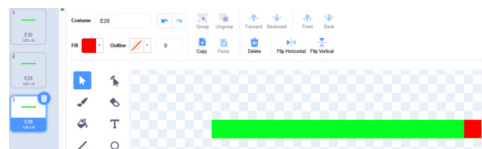
Po postojećoj slici sa desne strane nacrtamo pravougaonik crvene boje dužine 1 i visine 2



Dupliramo kostim E29 i novom kostimu damo ime E28



Postavimo se na Select i po postojećoj slici razvučemo crveni pravougaonik zdesna nalijevo tako da mu dužina bude jednaka 2 i visina jednaka 2



Nastavimo ovaj postupak dalje, sve dok ne dođemo do potpuno crvenog pravougaonika E0 dužine 30 i visine jednake 2

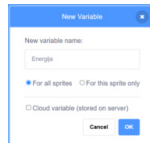


Sve ove slike su vektorskog oblika pa ne moramo vršiti konverziju formata. Sačuvamo napravljene izmjene.

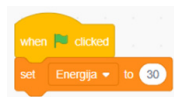
Pošto se bavimo maksimalnom energijom jednakom 30, potrebno je da definišemo i jednu promjenjivu koja će u sebi "sadržavati" energiju letjelice M14. Ta promjenjiva u sebi treba da sadrži broj 30 inicijalno, a svakim "ranjavanjem" naše letjelice ta vrijednost treba da se smanjuje. Može da se smanjuje za 1 ili 2 ili više u zavisnosti kakvim oružjem je pogođena naša letjelica. Jasno, u momentu kada ta promjenjiva bude poprimila vrijednost manju ili jednaku 0, to će značiti da smo eliminisani sa ploče i da treba da sklonimo letjelicu M14 sa ekrana, kao i da proglasimo kraj igre (Game Over). Promjenjivu ćemo nazvati Energija.

## 9 POVRAT. VRIJEDN. (U REALNOM VREMENU)

Idemo dakle na Code (dok smo na Sprite-u energije EN30). Idemo na Variables i definišemo varijablu (promjenjivu) Energija

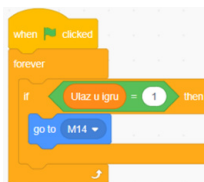


Unesemo kod na kod za EN30



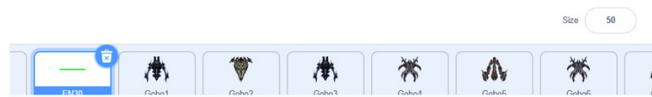
Postavili smo vrijednost varijable Energija na 30.

Pošto Sprite energije treba da vijerno prati letjelicu M14, to moramo napraviti kod koji to zaista i radi. Unosimo tako i sljedeći kod



Znači tek kada je Ulaz u igru jednako 1, tj. kada smo kliknuli na dugme START, Sprite energije treba da prati letjelicu M14. Sačuvamo izmjene i probamo igru.

Uočavamo da treba da veličinu (Size) Sprite-a EN30 smanjimo. Postavit ćemo je na 50



Ono što smo zaboravili uraditi je da kostime za Sprite EN30 centriramo.

Uradimo navedeno.

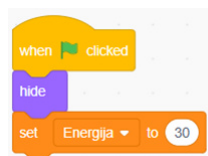
Prije toga, sjetimo se da smo kostime napravili pomoću dva pravougaonika, jednog zelenog i drugog crvenog koje nismo međusobno objedinili u jedan. Moramo to uraditi prije pomjeranja (jer bi inače morali pomjerati posebno zeleni a posebno crveni pravougaonik). Da bi izvršili objedinjavanje, dovoljno je da se postavimo na kostim i onda konvertujemo sliku u Bitmap format a onda je opet konvertujemo u vektorski oblik. Ovo uradimo za sve kostime za energiju EN30. Sašuvamo izmjene pa izvršimo centriranje kostima.

Ponovo sačuvamo izmjene.

Primijetimo da Sprite energije prelazi djelimično preko letjelice M14. Zbog toga ćemo ga djelimično translirati prema dole, tj. pomjeriti po  $y$  osi za nekih 60 jedinica. Prethodni kod tako modifikujemo u kod

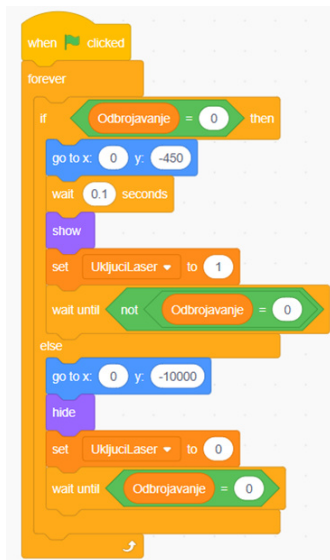


Takođe, Sprite energije ne bi trebao da bude vidljiv na početku igre. Zbog toga ćemo ga inicijalno sakriti, pa onda ponovo omogućiti da se pojavi onda kada to bude potrebno. Znači u kod za inicijalizaciju Energije dodamo naredbu za sakrivanje. Modifikovani kod je sljedeći kod

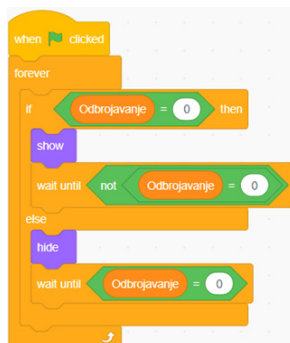


U ovakvom obliku igre, Sprite energije se nikada ne bi pojavio. Znači, moramo omogućiti da se taj Sprite pojavi. Logično je da se on pojavi onda kada se pojavi i letjelica M14. Dovoljno je da pogledamo kod pojavljivanja letjelice M14. To je kod





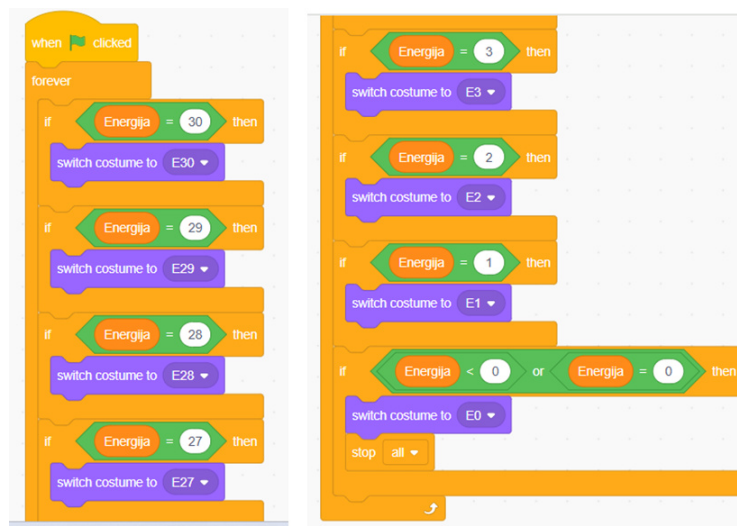
Ovaj kod kopiramo i zalijepimo na Sprite energije EN30, a zatim ga pojednostavimo, tj. modifikujemo u kod



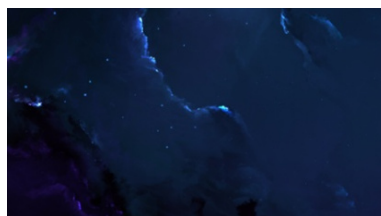
Ukratko, ako je Odbrojavanje jednako 0, neka se energija pojavi (već znamo da ona prati letjelicu M14), i neka ostane vidljiva sve dok Odbrojavanje ne postane različito od nule. U suprotnom, neka je energija sakrivena (dakle ako je Odbrojavanje različito od nule), i neka ostane sakrivena sve dok Odbrojavanje ne postane jednako 0. Probamo

igru, vidimo da ona radi ono što smo htjeli, i sačuvamo izmjene.

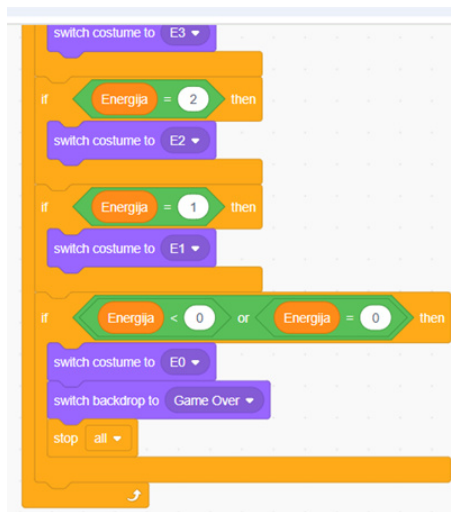
Vidimo da kostim energije koja prati letjelicu M14 ne mora biti adekvatno odabran (u smislu energije koju predstavlja). Kostim koji prati M14 treba da zavisi od vrijednosti promjenjive Energija. Tako, ako je Energija jednaka 30 onda prateći kostim treba da bude E30. Ako je Energija jednaka 29, onda prateći kostim treba da bude E29, itd, ako je Energija manja ili jednaka 0, prateći kostim treba da je E0. Jasno, ako je Energija manja ili jednaka 0, onda to znači da je letjelica M14 eliminisana. U tom momentu je prateći kostim za Energiju ustvari kostim E0, igra treba da stane (Game Over). Kod koji reguliše ova pitanja je sljedeći



Možemo dodati i pomoćnu pozadinu koju ćemo koristiti onda kada Energija postane manja ili jednaka 0. Koristit ćemo sljedeću podlogu



Upload-ujemo ovu podlogu. Ime kostima ove podloge zvat ćemo Game Over. U posljednji kod, prije: stop all naredbe ubacimo naredbu koja postavlja podlogu na podlogu Game Over. Dobijamo modificirani kod

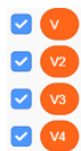


Sačuvamo izmjene napravljene u igri.

Vizuelne efekte napravljenih izmjena ćemo moći vidjeti tek kada napravimo razne vrste oružja neprijateljskim letjelicama, koje će onda biti u mogućnosti da ranjavaju letjelicu M14, i time impliciraju izmjene kostima prateće energije. U nastavku se bavimo izradom takvih oružja.

Za izradu neprijateljskih oružja nam treba kvalitetna priprema.

Prvo, definisat ćemo 4 liste sa po 50 mjesta. Zvat ćemo ih: V, V2, V3 i V4



Koristit ćemo neutralnu lokaciju, tj. prostor podloga da elemente navedenih lista očistimo i postavimo na vrijednosti 0. Idemo na podloge, pa na kodove, i unesemo sljedeće kodove



U nastavku ćemo vidjeti za šta tačno želimo da koristimo navedene liste. Naime, za sada možemo reći da želimo da napravimo oružje na četiri Gobo-a oblika



To su nam Gobo-i: Gobo1, Gobo3, Gobo7, Gobo9.

Svaki od ovih Gobo-a će imati po 4 pridružena oružja, ne u smislu njihove različitosti kao oružja nego u smislu mogućnosti 4 ispaljivanja jedne vrste oružja (takvo oružje moramo napraviti 4 puta radi nezavisnosti djelovanja i njihovog ispaljivanja). Zbog toga nam trebaju 4 prethodne liste. U svaku od navedenih lista smiještat ćemo određene podatke koji se odnose na odgovarajuće oružje. Tako, prve lokacije ove 4 liste će biti rezervisane za određene informacije o 4 oružja za Gobo1 (lista V za prvo oružje, lista V2 za drugo, lista V3 za treće i lista V4 za četvrto). Isto tako, treće lokacije navedene četiri liste će biti rezervisane za odgovarajuće informacije o 4 oružja za Gobo3, sedme lokacije za Gobo7, i devete lokacije za Gobo9.

Kako smo već rekli, svaki od Gobo-a: Gobo1, Gobo3, Gobo7, Gobo9 će imati

pridružena po 4 oružja. Zvat ćemo ih:

Lightning1A, Lightning1B, Lightning1C, Lightning1D (oružja za Gobo1),  
Lightning3A, Lightning3B, Lightning3C, Lightning3D (oružja za Gobo3),  
Lightning7A, Lightning7B, Lightning7C, Lightning7D (oružja za Gobo7),  
Lightning9A, Lightning9B, Lightning9C, Lightning9D (oružja za Gobo9).

Izgled ovog oružja će biti sljedeći



Kostime za;

Lightning1A, Lightning1B, Lightning1C, Lightning1D ćemo zvati L1,  
kostime za:

Lightning3A, Lightning3B, Lightning3C, Lightning3D imenom L3,  
kostime za:

Lightning7A, Lightning7B, Lightning7C, Lightning7D imenom L7,  
te kostime za:

Lightning9A, Lightning9B, Lightning9C, Lightning9D imenom L9.

Ime dokumenta koji sadrži navedeno oružje koje ćemo upload-ovati je "L oružje"

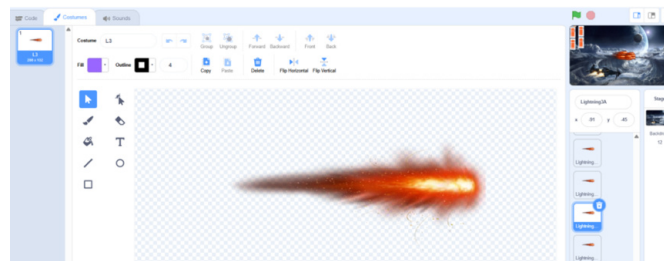


L oružje.svg

Nakon upload-a ćemo imena kostima postaviti onako kako smo već opisali (L1 za 4 oružja za Gobo1, L3 za 4 oružja za Gobo3, L7 za 4 oružja za Gobo7, L9 za 4 oružja za Gobo9).

Na Sprite-ove sada upload-ujemo navedeni dokument "L oružje", dupliramo ga 15 puta uz promjenu imena, i imena kostima na način opisan iznad

## 9 POVRAT. VRIJEDN. (U REALNOM VREMENU)

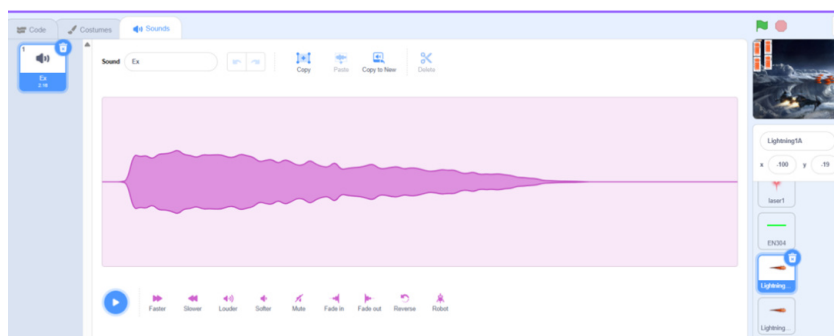


Svih 16 Sprite-ova pomjerimo fizički prema gore, da budu iznad svih Gobo-a.  
Veličinu (Size) za svako od navedenih 16 oružja ćemo staviti da je 40.  
Sačuvamo izmjene.  
Bit će nam potreban sljedeći zvuk na svakom od 16 upravo upload-ovanih oružja



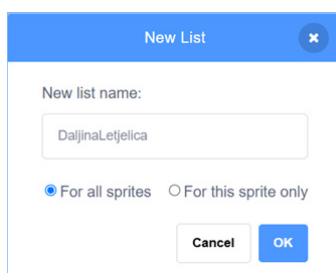
Ex zvuk.wav

Iako ga upload-ujemo pod imenom "Ex zvuk", nakon upload-a, ime zvuka (Sound) ćemo skratiti na samo "Ex". Ovo uradimo za svih 16 oružja. Dakle, upload-ujemo sada navedeni zvuk na svih 16 oružja Lightning1A-Lightning9D. Dovoljno je izvršiti upload navedenog zvuka na oružje Lightning1A, promijeniti mu ime (Sound) u "Ex" a onda ga kopirati i zalijepiti na svih ostalih 15 oružja Lightning1B-Lightning9D



Sačuvamo izmjene.

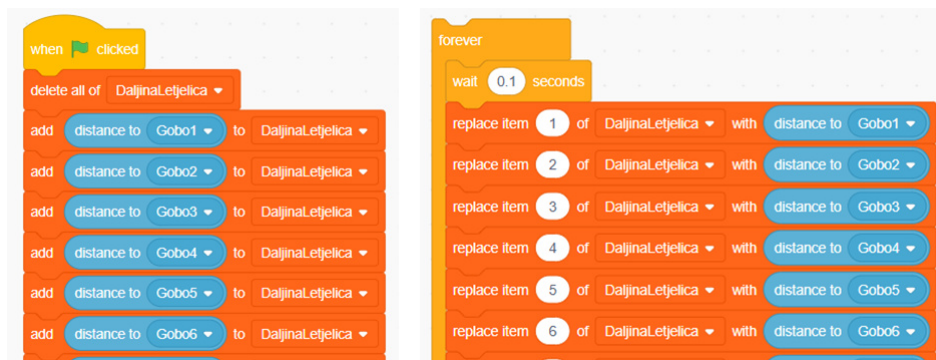
U sklopu pripreme za izradu oružja, potrebna će nam biti lista u kojoj ćemo sadržavati udaljenosti od naše letjelice M14 do svih neprijateljskih letjelica, tj. do Gobo-a. Listu ćemo zvati DaljinaLetjelica. Napravimo ovu listu



Lista DaljinaLetjelica treba da sadrži udaljenosti (50 njih) od naše letjelice pa do svih 50 Gobo-a. Ta lista treba da se update-uje u realnom vremenu, tj. da stalno sadrži najsvježije podatke. Zbog toga ćemo tu listu osvježavati najnovijim informacijama svakih 0.1 sekundi. Jasno da će nam za ovo trebati jedna beskonačna petlja (koja nikada ne prestaje sa radom) i koja osvježava podatke svakih 0.1 sekundi. Važno je sjetiti se da smo mi Gobo-e programirali tako da nakon što budu eliminisani sa ploče, idu na za sebe unaprijed rezervisanu (daleku) lokaciju. Ovo će značiti da ako je naša letjelica M14 na ploči, da će onda udaljenost do "živog" Gobo-a biti mala (jer su i letjelica M14 i pomenuti Gobo istovremeno na ploči - dakle blizu su), dok će udaljenost do eliminisanog Gobo-a biti velika (jer je M14 na ploči a pomenuti Gobo na dalekoj lokaciji). Ova logika će nam takođe koristiti da znamo da li je neki Gobo živ ili ne. Ako je udaljenost do M14 mala onda je on živ. On nije živ ako je ta udaljenost jako velika.

Do sada smo liste koje smo koristili definisali na neutralnim lokacijama, konkretnije na podlogama. Naime, inicijalni podaci koje smo unosili u liste su obično bili jednaki 0 ili 1, tj. nisu zavisili od konkretnog objekta, tj. Sprite-a. Pošto lista DaljinaLetjelica treba da sadrži udaljenosti od naše letjelice M14 pa do svih Gobo-a, onda podatke za listu DaljinaLetjelica moramo uzimati sa lokacije letjelice M14. Dakle, kod za listu DaljinaLetjelica unosimo na kod za letjelicu M14.

Idemo na letjelicu M14, na njen kod, i unosimo sljedeći kod  
Nadovežemo desni kod na lijevi.



Tako, inicijalno postavimo da su elementi liste DaljinaLetjelica ustvari udaljenosti letjelice M14 do svih Gobo-a (njih 50). Nakon toga, forever petljom, osvježavamo ove podatke svakih 0.1 sekundi. Ovdje kad se kaže: distance to Gobo1, npr. misli se na udaljenost letjelice M14 do Gobo1 jer se nalazimo na kodu za letjelicu M14.

Probamo igru. Možemo uočiti kako se udaljenosti u listi DaljinaLetjelica mijenjaju u zavisnosti od položaja letjelice M14 i položaja Gobo-a. Pošto se te udaljenosti mijenjaju svake 0.1 sekunde to dolazi do blagog usporavanja prilikom izvršavanja igre. Zbog toga savjetujemo da se lista DaljinaLetjelica ukine sa ploče, tj. da kvakica ne bude aktivna pored njenog imena (možemo je naravno uključiti po potrebi ako treba vizuelno da vidimo podatke koje ona sadrži)



Neprijateljska oružja koja ovdje pravimo (16 njih) nazvali smo Lightning1A-Lightning9D. U momentu kada nas neko od ovih oružja pogodi, tj. kada dodirne letjelicu M14, to oružje treba da nestane sa ploče, odnosno treba da ga sklonimo sa ploče na neku za njega rezervisanu lokaciju. Da bi omogućili sklanjanje sa ploče potrebno je da događaj dodira oružja sa letjelicom M14 opišemo na neki praktičan način. Najlakše je ponovo koristiti princip uključi-isključi ili 0-1.

Tako, za svako od oružja Lightning1A-Lightning9D ćemo definisati po jedan pokazivač, tj. jednu promjenjivu čiju ćemo inicijalnu vrijednost postaviti da bude jednaka 0. Ako neko od oružja dodirne letjelicu M14, onda ćemo njegov odgovarajući pokazivač staviti da bude jednak 1. Tako, ako se desi da je odgovarajući pokazivač



jednak 1, onda će nam to značiti da odgovarajuće oružje sklonimo sa ploče.

Navedenih 16 pokazivača, tj. promjenjivih ćemo zvati L7-L22 (promjenjive L1-L6 koristimo u druge svrhe).

Definišimo onda novih 16 promjenjivih po imenu L7-L22.

Ove promjenjive ne moraju biti aktivne na ploči pa kvakice pored njihovog imena možemo ukinuti.

Pošto se promjenjive L7-L22 odnose na dodir letjelice M14 sa oružjima Lightning1A-Lightning9D, to je onda prirodno da i inicijalni kod za L7-L22 unesemo na kod letjelice M14. Tako, idemo na M14, na kod, i unesemo sljedeći kod



Dodatno, sada, dok smo na kodu letjelice M14, možemo, iako još uvijek nismo napravili kod i algoritam za rad oružja Lightning1A-Lightning9D, napraviti kod koji se odnosi na eventualni dodir letjelice M14 i nekog od pomenutih oružja Lightning1A-Lightning9D. Šta treba da se desi prilikom jednog takvog dodira ? Treba da se Energija smanji za 1 (ovdje možemo smatrati da oružja Lightning1A-Lightning9D nisu oružja velike snage tako da dodir M14 sa njima uzima samo 1 podiok energije M14), te da se odgovarajući pokazivač (među pokazivačima L7-L22) postavi na vrijednost 1. Tako, na kod letjelice M14 unosimo sljedeći kod



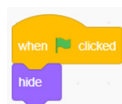
Desni kod uklopimo u forever petlju lijevog koda i sačuvamo izmjene.

Možemo sada, pošto kodiranje oružja Lightning1A-Lightning9D nije završeno, a oružja se nalaze vidljiva na ploči, prići im letjelicom M14 i vidjeti kako dodir sa njima utiče na našu energiju. Vrijednosti promjenjive Energija se smanjuju što je popraćeno vizuelnim efektom gubitka energije (gubimo zelenu boju koja postaje crvena na traci energije EN30)



Sada smo u poziciji da damo kod za oružja Lightning1A-Lightning9D. Uradit ćemo kod za jedno od oružja a onda taj kod kopirati i zalijepiti na ostala oružja, te ga tamo modifikovati.

Idemo na Lightning1A, na kod, i unosimo sljedeći kod (kodove)



Ovaj kod je jasan, jer oružje ne treba da bude vidljivo sve do momenta kada mi odlučimo da ono bude vidljivo. Ovaj kod je isti za sve Lightning1A-Lightning9D.



Ovaj kod predstavlja glavnu petlju koda oružja. Kodovi koji dolaze će biti uklopljeni unutar If uslova ovog koda. Znamo da nakon nekog vremena neprijatelj treba da ispali hitac prema M14. Onda opet, nakon nekog vremena treba da učini isto. Zbog toga nam treba prethodna forever petlja. Uslov da je Ulaz u igru jednako 1 nam znači da smo uopšte ušli u igru, tj. da samo kliknuli na dugme START. Znači ako smo ušli u igru, tj. kliknuli na dugme START, onda ima smisla da uopšte govorimo o ispaljivanju hitaca prema nama. Ovaj kod je isti za sve Lightning1A-Lightning9D.

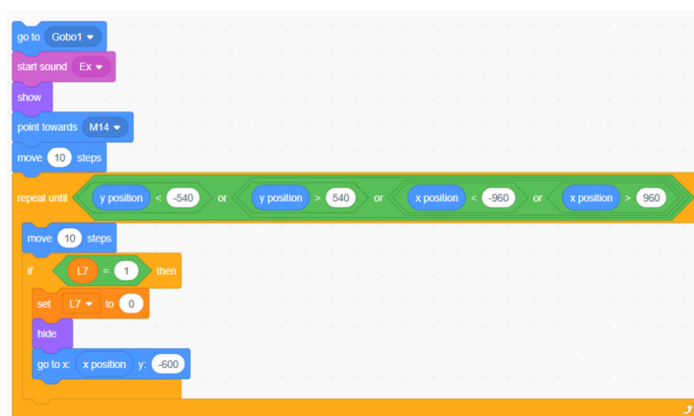


Ovdje, pitamo se šta treba sa oružjem prvo da radimo. Govorimo o oružju Lightning1A. Ako razmislimo gledajući kroz razne igre koje smo igrali tokom života, neprijateljske letjelice rijetko da pucaju na nas svojim oružjima odmah. Obično tu ima neko razumno proizvoljno vrijeme čekanja, nakon kojeg neprijatelj počne da puca na nas. Isti rezon koristimo onda i mi. U item 1 liste V smještamo neko proizvoljno vrijeme između 5 i 15 sekundi. To je vrijeme čekanja nakon kojeg će Gobo1 da ispali hitac Lightning1A prema nama. Znači, nekada će to biti 5 sekundi čekanja, nekad 9, a nekada 15. Manje od 5 ili više od 15 bi bilo premalo odnosno previše. Opet, na nama je da odredimo granice čekanja. Za oružje Lightning1B bi koristili item 1 liste V2, za Lightning1C item 1 liste V3, za Lightning1D item 1 liste V4. Jasno, onda, za oružje Lightning3A bi koristili item 3 liste V, itd, redom. Ovdje je uslov da je Odbrojavanje jednako 0 potreban jer taj uslov znači da se uopšte nalazimo na ploči, tj. da je bar jedan neprijatelj živ. Naime, ako je Odbrojavanje različito od nule, tada ili nismo uopšte ušli

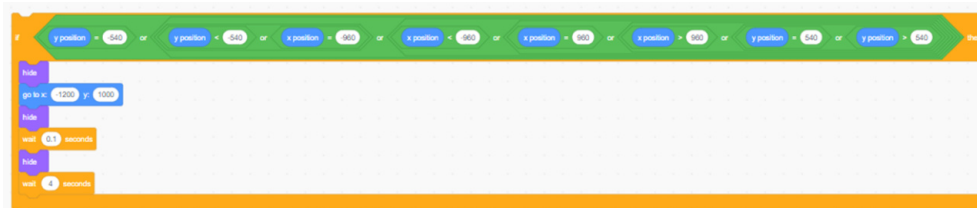
u igru ili se na ploči ne nalazimo ni mi (letjelica M14) niti bilo ko od Gobo-a. Znači, ne bi bilo smisla da provodimo odbrojanje do ispaljenja neprijateljskog hica ako na ploči nema nikoga, tj. uslov da je Odbrojanje jednako 0 je opravdan.



Pitamo se šta dalje nakon što je čekanje do ispaljenja hica završeno. Potrebno je da hitac krene prema M14. Jasno da bi on krenuo prema nama treba ponovo da je Odbrojanje jednako 0 (ako je različito od nule onda ili nismo ušli u igru ili na ploči nema nikoga). Dodatno, moramo zahtijevati da je Gobo1 prisutan na ploči, a to, kako smo ranije napomenuli, znači da je on blizu M14 na ploči u smislu udaljenosti. To kako vidimo možemo tumačiti uslovom da je item 1 liste DaljinaLetjelica manji od 5000 (mogli smo staviti 4000 ili 6000 i sl, jer znamo da kada je Gobo1 mrtav da je on tada puno dalje od M14 nego što je to 4000, 5000 ili 6000). Dakle, koristili smo da item 1 liste DaljinaLetjelica označava udaljenost od M14 do Gobo1. Napomenimo da će ovaj dio koda da izgleda isto i za oružja Lightning1B-Lightning1D, jer se i ova tri oružja vezuju za Gobo1. Za Lightning3A-Lightning3D bi koristili item 3 liste DaljinaLetjelica, itd, za ostala oružja.



Nakon što smo u prethodnom kodu ustanovili da hitac Lightning1A možemo uopšte ispaliti (od Gobo1) prema M14, idemo na realizaciju putanje hica. Sprite Lightning1A šaljemo do Gobo1, uz ispuštanje zvuka "Ex". Omogućimo da se Lightning1A pojavi. Usmjerimo ga ka M14, i naredimo da krene (prema M14) 10 koraka. Kretanje hica Lightning1A ponavljamo sve dok hitac ili ne pređe neku od ivica ekrana ( $x < -960$  ili  $x > 960$  ili  $y < -540$  ili  $y > 540$ ), ili ne dodirne M14. Ako hitac Lightning1A dodirne M14, onda po kodu letjelice M14 znamo da odgovarajući pokazivač L7 postaje jednak 1. Tako, ako L7 postane jednak 1 (što je znak da je Lightning1A dodirnuo M14), onda odmah L7 postavimo ponovo na 0 (pripremimo ga za ponovno ispaljenje), sakrijemo Lightning1A i sklonimo Lightning1A sa ekrana. Dovoljno je ne mijenjati  $x$  poziciju na kojoj se Lightning1A nalazi, nego samo ga "spustiti" niže van ekrana, tj. prebaciti ga niže na lokaciju ( $x$ -pozicija, -600). Za Lightning1B bi koristili pokazivač L8, za Lightning1C pokazivač L9, za Lightning1D pokazivač L10, za Lightning3A pokazivač L11, itd, za Lightning9D pokazivač L22.



Sada, ako je hitac Lightning1A izašao van okvira ekrana (bilo da je sam izašao, bilo da je dodirnuo M14 pa da smo ga mi u prethodnom koraku pomjerali niže van ekrana), treba da ga sakrijemo, sklonimo na rezervisanu lokaciju (koristit ćemo lokaciju (-1200,1000)), te da sakrivanje ponovimo nekoliko puta sa povremenim manjim i većim čekanjima. O neophodnosti višestrukog sakrivanja Sprite-ova smo govorili i ranije. Naime, zna se desiti da zbog brzine izvršavanja operacija, neke naredbe budu izvršene prije nego to njihov redoslijed zahtijeva. Otud potreba za ponovnim sakrivanjem objekata (onih objekata koje zaista želimo da sakrijemo). Ovaj kod je univerzalan za sve hice Lightning1A-Lightning9D.



Znamo da algoritam za hitac Lightning1A (kao i za ostale hice) treba da se ponavlja. Postupak koji se vrti je: čeka se neko proizvoljno vrijeme, hitac se kreće prema M14 (izlazi van ekrana ili pogađa M14), sklanja se na lokaciju (-1200,1000). Ovaj krug ima smisla da se vrti samo ako je Gobo1 na ekranu. Ako Gobo1 nije na ekranu, onda navedeni krug nema smisla da se odigrava. Kako znamo, činjenica da je Gobo1 na ekranu je da je item 1 liste DaljinaLetjelica manji od 5000. Tako, ako je Gobo1 van ekrana (ako je mrtav), tj. ako je item 1 liste DaljinaLetjelica veći od 5000, onda čitav kod za Lightning1A treba da sačeka dok Gobo1 ne bude živ, tj. dok item 1 liste DaljinaLetjelica ne bude manji od 5000. Isti ovakav kod ide i za hice Lightning1B-Lightning1D. Za hice Lightning3A-Lightning3D bi koristili item 3 liste DaljinaLetjelica (jer je tada u pitanju Gobo3), itd.

Uklopimo sada navedene kodove na način (ukupnimo elemente).

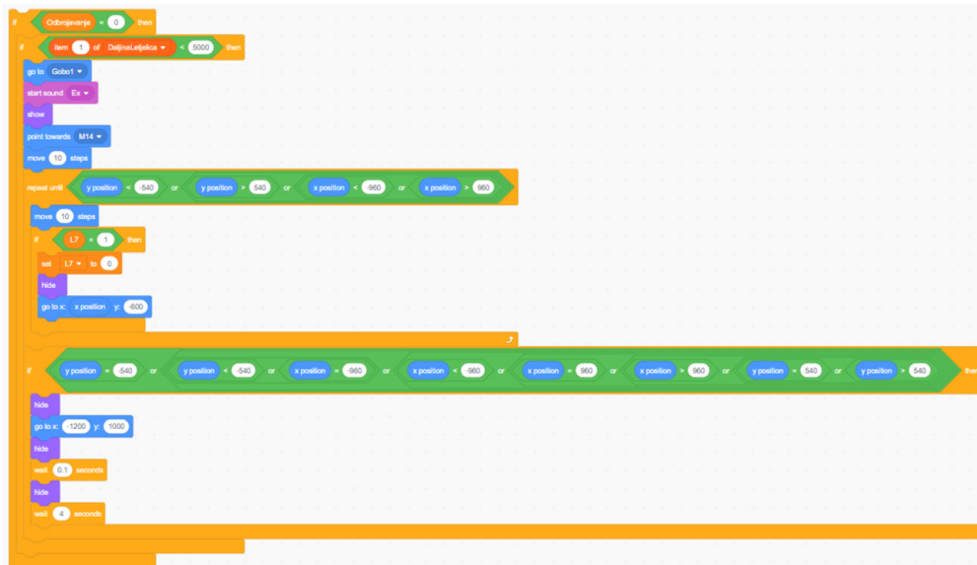
Osnovna petlja:



Prvi dio unutar osnovne petlje:



Drugi dio unutar osnovne petlje:



Treći dio unutar osnovne petlje:



Sva tri navedena dijela nadovežemo jedan na drugi kako slijede, i objedinjeni kod uklopimo unutar osnovne petlje. To je kod za Lightning1A.

Sačuvamo izmjene.

Kopiramo sada kod sa Lightning1A i zalijepimo na ostale Lightning1B-Lightning9D, te izvršimo izmjene prema upravo opisanim uputstvima. Ako probamo igru (i korigujemo eventualne greške) vidjet ćemo da ona radi upravo ono što smo i željeli.





## 9.1 Riješeni zadaci

◇ **Zadatak 9.1.1** Napraviti klasičnu igru odbijanja (Paddle Game) u kojoj igrač pomoću palice sprječava da loptica padne na dno ekrana.

Postaviti pozadinu i kreirati Sprite palice. Potrebno je omogućiti da palicu možemo pomijerati lijevo i desno pomoću strelica na tastaturi.

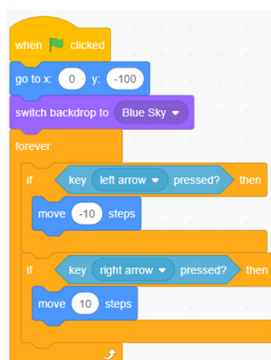
Dodati Sprite loptice kojeg postavljamo na odabranu startnu lokaciju i koji se neprekidno kreće po ekranu. Loptica treba da se odbija od ivica koristeći blok "if on edge, bounce". Kada dodirne dno ekrana, igra se završava (pozadina se mijenja u "GAME OVER"). Ako loptica dodirne palicu, ona treba da se odbije od nje i treba da se poveća vrijednost varijable "Rezultat" za 1. Rezultat treba da bude prikazan u gornjem dijelu ekrana (koristeći varijablu "Rezultat").

Proširiti igru dodavanjem još jedne loptice koja se kreće sporije od prve. Cilj je povećati težinu i dinamiku igre.



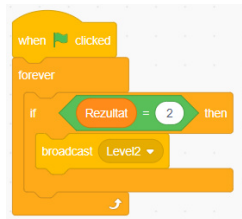


**Rješenje:** Kod za palicu:

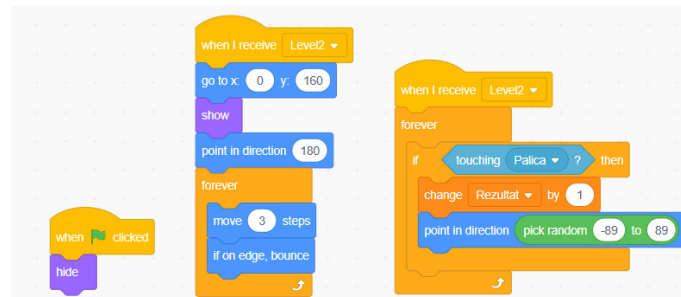


Kod za prvu lopticu:





Kod za drugu lopticu:



## 9.2 Zadaci za vježbu

♣ **Zadatak 9.2.1** U sklopu Zadatka 9.1.1 napraviti ekran koji velikim brojevima na originalan način prikazuje konačni rezultat, te dugme "Igraj ponovo" koje omogućava

restart igre.

♣ **Zadatak 9.2.2** Napraviti igru "Football" sa dva igrača, golovima i loptom.



## 10 Izrada složenijih struktura

Idemo sada da napravimo oružje za Gobo4 i Gobo6. Riječ je o neprijateljskim letjelicama oblika



Imamo samo dvije ovakve neprijateljske letjelice koje se pojavljuju na ploči. Ako ih tretiramo kao neku vrstu boss-ova koji se pojavljuju u igri, onda bi i njihovo oružje na neki način trebalo da bude jedinstveno. Ideja oružja za ove Gobo-e su munje (gromovi). Zamisao je da prije same upotrebe oružja ovi Gobo-i signaliziraju da će izvršiti napad da bi igrač imao vremena da reaguje i eventualno se skloni od takvog napada. Pošto želimo da je napad ovakvih letjelica tipa: unique, to je onda jasno da oružje o kome govorimo i koje želimo da napravimo neće biti moguće prevenirati, tj. neće biti moguće poduzeti kontra-mjere našom letjelicom M14 da takav napad poništimo nekim od naših oružja. Ono što preostaje će biti klasično izbjegavanje napada (evade). Kako rekosmo, Gobo4 i Gobo6 će signalizirati da će doći do napada. Ideja oružja je da Gobo4 i Gobo6 ispaljuju gromove po 4 puta, sa određenim kratkim stankama. Prije nego uopšte dođe do napada, Gobo4 i Gobo6 će imati određeno vrijeme pripreme za napad, u smislu da će postojati određeno vrijeme čekanja prije samog napada. Govorimo o nekom random vremenu koje ćemo uzeti na početku i nakon čijeg isticanja će napad da se dogodi. Kako smo i ranije govorili, dobra praksa igara je ta da napad neprijatelja ne treba da se dogodi smjesta nakon raspoređivanja neprijateljskih letjelica. Poželjno je da na početku postoji neko vrijeme čekanja nakon kojeg napad treba da otpočne. To vrijeme je obično random vrijeme iz nekog intervala, i nezavisno je od letjelice do letjelice, da bi se stvorio utisak nezavisnosti ponašanja i djelovanja letjelica, kao i osjećaj spontanosti i ne-rigidnosti. Nakon rezervisanja određenog vremena, to vrijeme počinje da odbrojava, i kada istekne, napad kreće. Zamisao oružja za Gobo4

i Gobo6 je taj da nakon isticanja pomenutog vremena, napad ne polazi smjesta, nego dolazi do pripreme oružja, pri kojoj igrač može da uoči neka dešavanja na letjelicama Gobo4 i Gobo6 koje aludiraju da treba nešto da se dogodi, a što mogu tretirati kao pripremu jednog takvog događaja. Naime, mi želimo da Gobo4 i Gobo6 ispaljuju munje vertikalno prema nama, cijelom dužinom ekrana uz proizvođenje adekvatnog popratnog zvuka. Želimo da imamo 4 takva ispaljivanja između kojih traje neko kratko vrijeme. Ako se sjetimo nekih SF filmova o ovoj tematici, svako ispaljivanje nekog main oružja zahtjeva neku vrstu pripreme za takav čin: rasklapanje određenih elemenata oružja, varnice, sakupljanje energije, popratna boja i zvuk, itd. I naša ideja ovdje je da imitiramo jedan takav događaj. Da bi Gobo4 i Gobo6 koristili munje protiv nas, potrebno je da prije toga prikupe određenu energiju. To prikupljanje će biti svojevrsna priprema za napad na nas. Prikupljanje potrebne energije možemo manifestovati na razne načine, raznim vrstama malih munja, energija, čestica, atoma, i sl., koji se pojavljuju u okolini Gobo4 i Gobo6. Stvar je sada u kreatoru igre koju varijantu događaja želi da izabere i animira. Mi ćemo odabrati jedan dosta jednostavan način da prikazemo ovaj događaj ali opet dovoljan da ukaže na ono što nam je interesantno. Osim Sprite-a koji koristimo za Gobo4 i Gobo6 koji smo pokazali iznad, koristit ćemo još neke elemente da usložnimo navedeni Sprite da bi on ukazivao na ono što smo upravo opisali da treba da ukazuje. Naime, napraviti ćemo nekoliko kostima pored navedenog kostima za Gobo4 i Gobo6 čijom rotacijom (animacijom) ćemo postići željeni efekat. Želimo da se oko našeg Sprite-a za Gobo4 i Gobo6, nakon isticanja početnog vremena čekanja, počne da pojavljuje neki energetski krug, uz proizvodnju određenog električnog zvuka, koji će periodično da se pojavljuje nekoliko puta ali isto tako i da nestaje, što će na određeni način aludirati na pomenuto sakupljanje energije. Za naše potrebe dovoljno će biti da se takav krug prikaže 4 puta. Da bi prikazali takav krug treba da ga nađemo kao transparentni Sprite, i uklopimo ga u postojeći Sprite za Gobo4 i Gobo6. Jasno, nakon toga, prilikom pokretanja igre, dovoljno će biti da osnovni Sprite za Gobo4 i Gobo6 smjenjujemo novim modifikovanim Sprite-om 4 puta uz proizvodnju električnog zvuka da bi postigli efekat pripreme main oružja. Drugim riječima, trebat će nam nekih 8 Sprite-ova: 4 osnovna i 4 modifikovana, koje ćemo smjenjivati prije nego dođe do upotrebe munja kao osnovnog oružja za Gobo4 i Gobo6. Nakon ovog dosta detaljnog opisa ideje oružja bavimo se implementacijom navedenih ideja.

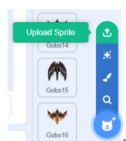
Prvo, na pngwng.com stranici koju smo i ranije koristili, potražimo transparentnu sliku adekvatnog energetskog kruga. Potražimo objekat pod nazivom energy circle. Dobit ćemo više odgovarajućih objekata, tako da na nama ostaje da odaberemo onaj koji nam vizuelno i estetski odgovara. Možemo npr. koristiti neki od Sprite-ova koje

smo pronašli



Možemo evo probati upotrijebiti prvi Sprite. Neka njegovo ime bude En\_Energy. Naravno, nakon upload-a ćemo potrebna imena modifikovati po potrebi.

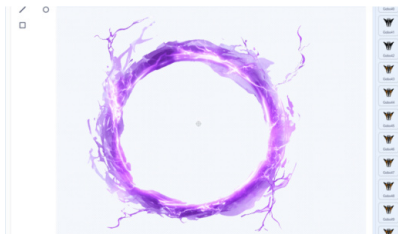
Idemo na Choose a Sprite pa na Upload Sprite, te upload-ujemo navedeni Sprite En\_Energy



Upload-ovanu sliku konvertujemo u vektorski oblik, a koristeći opcije

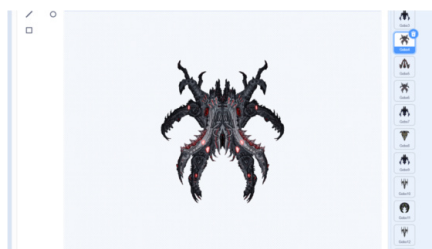


smanjimo njenu veličinu da je možemo u potpunosti vidjeti pred sobom na ekranu

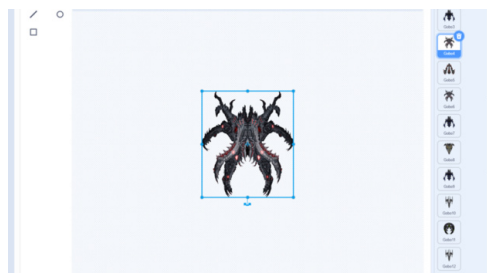


Mi želimo da se ovaj krug pojavi oko Gobo4 i Gobo6 ali tako da Gobo4 (ili Gobo6) bude u potpunosti vidljiv unutar ovog kruga. Isto tako želimo i da ovaj krug bude u potpunosti vidljiv. Dodatno, primijetimo da navedeni krug nije u potpunosti simetričan. Želja nam je da stvorimo utisak da se navedeni krug kreće ukруг, odnosno želimo da ostavimo utisak rotacije oko Gobo4 i Gobo6. Dovoljno će biti da u 4 pojavljivanja ovog kruga, koliko smo ih zamislili, izvršimo rotaciju ovog kruga za 90 stepeni. Navedena animacija će stvoriti utisak rotacije oko Gobo4 i Gobo6.

Ako pogledamo izgled za Gobo4 i Gobo6, možemo vidjeti da oni trenutno zauzimaju poprilično prostora na našoj lokaciji za kostime



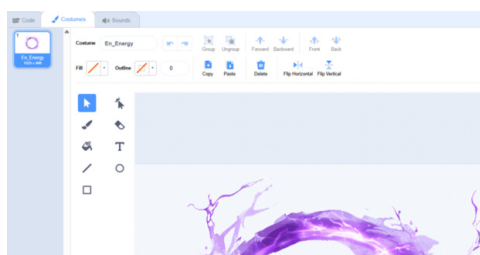
Da bismo mogli adekvatno uklopiti Gobo4 i Gobo6 sa opisanim krugom potrebno je da Gobo4 vizuelno smanjimo na ovom mjestu (ručno bez mijenjanja njegove veličine: Size). Veličinu Size možemo naknadno podesiti, tj. povećati ako je potrebno. Pri tome, nakon smanjivanja Gobo4, ne treba raditi isto za Gobo6, jer nikada ne možemo postići da identično smanjimo i Gobo4 i Gobo6. Najbolje je izvršiti smanjivanje za Gobo4 i podesiti sve kostime za njega a onda ih zalijepiti na Gobo6 (da osiguramo istu veličinu Sprite-ova). Nakon smanjivanja i centriranja, Gobo4 može izgledati npr. ovako



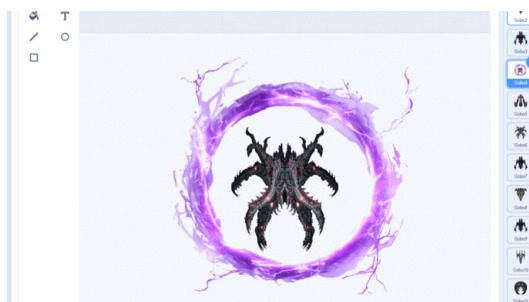


Dupliramo sada kostim za Gobo4 i damo mu ime MM4B (prvobitni se zove MM4A).

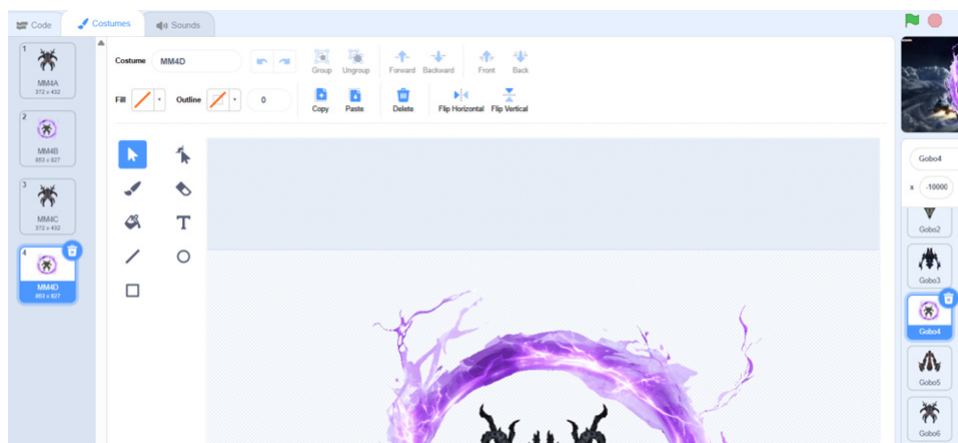
Odemo na Sprite En\_Energy, i kopiramo ga (uočimo opciju Copy na sljedećoj slici)



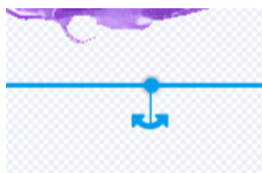
Zatim, vratimo se na Gobo4, i na MM4B zalijepimo ono što smo kopirali. Vidimo da je veličina kruga oko Gobo4 znatno veća u odnosu na letjelicu. Zbog toga smanjimo ručno veličinu kruga koliko je potrebno, i postavimo ga tako da se letjelica nalazi u sredini. Jasno je da ove izmjene možemo raditi jer su slike i letjelice i kruga u vektorskom obliku. Kada jednom podesimo veličinu kruga da bude adekvatna, onda je više ne mijenjamo, jer će nam trebati u nastavku. Za sada ne uklapamo slike jedne u drugu, jer će nam slika kruga trebati za sljedećih nekoliko kostima na kojima želimo da prikazemo rotaciju. Dakle, za sada, smanjena slika kruga u čijem centru je letjelica može da ima oblik



Sada dupliramo MM4A, dobijenom kostimu damo ime MM4C, premjestimo ga ispod kostima MM4B. Nakon toga dupliramo kostim MM4B, damo mu ime MM4D, i premjestimo ga ispod kostima MM4C. Imamo situaciju kao na slici



Primijetimo da i na kostimu MM4B i MM4D imamo istu sliku, tj. imamo isti položaj kruga oko letjelice. Kako smo već napomenuli, mi želimo da ostavimo utisak kretanja tog kruga. Zbog toga ćemo na kostimu MM4D, taj krug zarotirati udesno za 90 stepeni. Dakle, dok smo na MM4D, pozicioniramo se kursorom na krug, i na opcijama u dnu predviđenim za rotiranje

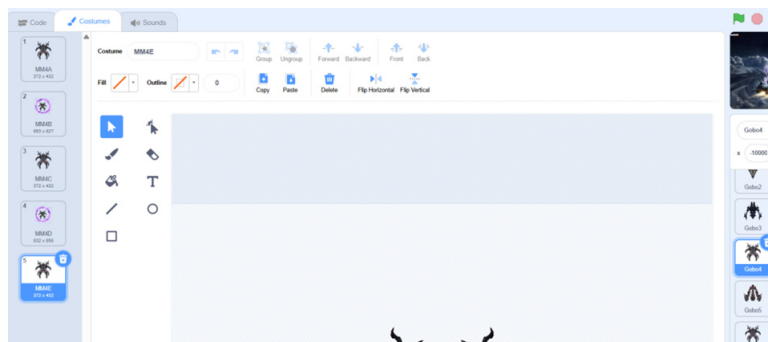


izvršimo rotaciju kruga udesno za 90 stepeni. Ta rotacija ustvari podrazumjeva da dio slike koji se nalazio u gornjem lijevom uglu, sada treba da se nalazi u gornjem desnom uglu (radi predodžbe o tome koliko treba da zarotiramo sliku udesno). Može se primijetiti da nakon rotacije, krug i letjelica ne moraju biti usklađeni u smislu da se letjelica nalazi u sredini kruga. Ponovo, onda, kao i ranije, nakon rotacije, podesimo krug tako da letjelica bude u njegovoj sredini. Pri tome, letjelicu ne diramo jer je ona centrirana, tj. ona ne smije da mijenja svoj položaj jer bi time bilo narušeno njeno kretanje koje kako znamo mora da bude glatko. Tako, nakon rotacije i podešavanja kruga (da letjelica bude tačno u njegovoj sredini), dobijamo finalni oblik za MMD4

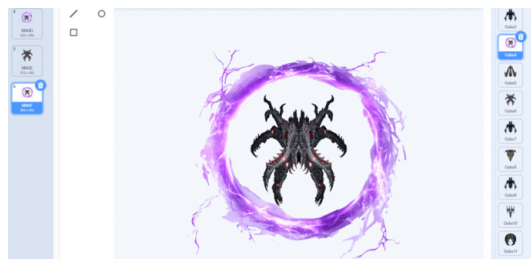


Ako kliknemo naizmjenično između MM4B i MM4D možemo steći utisak opisane rotacije. I dalje ne uklapamo slike u jednu jer će nam sada MM4D trebati u nastavku. Na kraju, kada dobijemo sve željene Sprite-ove, izvršit ćemo (prelaskom na Bitmap a zatim vektor oblik) objedinjavanje slika u jednu.

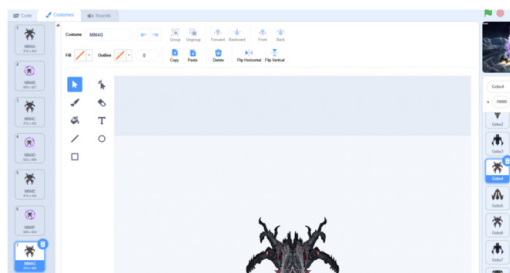
Sada, ponovo dupliramo MM4A, damo ime kostimu MM4E, i pomjerimo kostim ispod kostima MM4D



Zatim, dupliramo MM4D, damo ime kostimu MM4F, i pomjerimo kostim ispod kostima MM4E. Pošto je kostim MM4F identičan kostimu MM4D, to na njemu moramo izvršiti rotaciju kruga udesno za 90 stepeni. Izvršimo rotaciju, i ako se letjelica ne nađe u sredini dobijenog kruga, krug pomjerimo tako da letjelica ipak bude u sredini dobijenog kruga (bez pomjeranja letjelice). Dobijamo situaciju kao na slici



Sada, očekivano, ponovo dupliramo kostim MM4A, dobijenom kostimu damo ime MM4G, premjestimo ga ispod kostima MM4F



Dupliramo sada kostim MM4F, damo ime novom kostimu MM4H. Kostim MM4H ima isti oblik kao i kostim MM4F. Na kostimu MM4H izvršimo rotaciju kruga udesno za 90 stepeni. Ukoliko letjelica ne bude u sredini kruga, onda, bez pomjeranja letjelice, pomjerimo zarotirani dobijeni krug tako da se letjelica nađe u sredini kruga. Dobijamo situaciju kao na slici



Napokon, dupliramo još jednom kostim MM4A. Dobijenom kostimu damo ime MM4I, i taj kostim spustimo na dno, ispod kostima MM4H. Imamo



Nije nam potrebno više od ovih 9 kostima. Njihovom rotacijom, tj. animacijom postići ćemo efekat koji smo opisali iznad (efekat pripreme oružja).

Sada smo u situaciji da navedene kostime, tj. slike koje ih čine objedinimo u jednu sliku. Klikom miša se pozicioniramo na svaki od kostima. Dok smo na nekom kostimu idemo na Bitmap i odmah zatim na vektor oblik. Na taj način, kao i do sada, objedinjavanje slika u jednu će biti završeno (provjeriti obavezno da li na kraju na svakom od kostima piše: Convert to Bitmap - ako piše onda je objedinjavanje obavljeno pravilno). Dovoljno je da to uradimo samo za one kostime koji u svom sastavu imaju više slika. Pozicioniramo se na MM4A i sačuvamo izmjene.

Sprite En\_Energy nam više nije potreban pa ga možemo u potpunosti obrisati.

Kostime koje smo upravo napravili za Gobo4 želimo iste takve da imamo i na Gobo6. Jednostavan način da ih dobijemo i na Gobo6 je da ih exportujemo sa Gobo4 (ili kopiramo sa Gobo4). Pozicioniramo se dakle na Gobo4, i prolazeći kroz svaki od njegovih kostima, exportujemo ih. Svi ti exportovani kostimi će da završe u Download folderu odakle ih ponovo možemo upload-ovati na Sprite Gobo6. Tako i uradimo, exportujemo sve kostime sa Gobo4 (desni klik na kostim i onda export). Ako ne budemo zadovoljni preciznošću exportovanih fajlova (u smislu njihove precizne pojave na ekranu) uvijek možemo uraditi klasično kopiranje i lijepljenje kostima sa Gobo4 na Gobo6.

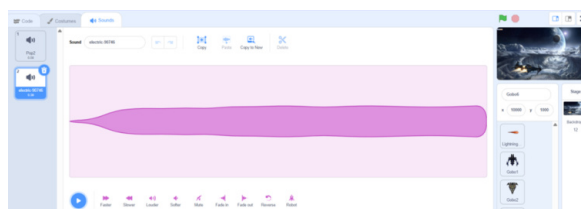
Sada idemo na Gobo6 i upload-ujemo kostime koje smo upravo exportovali. Polazni kostim koji smo imali za Gobo6, tj. kostim MM6A obrišemo. Kostimima koje smo upravo upload-ovali, promijenimo imena u MM6A, MM6B, ..., MM6I, redom. Pozicioniramo se na MM6A i sačuvamo izmjene.

Ovako kako smo postavili stvari sa kostimima na Gobo4 i Gobo6, može se uočiti ideja da izmjenama 9 napravljenih kostima vršimo alternaciju između normalnog stanja letjelice i stanja u kojem se oko nje nalazi energetski krug. Polazimo iz nor-

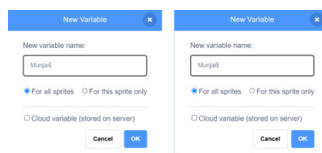
malnog stanja MM4A, na stanje MM4B, gdje krug zauzima svoj prvobitni oblik (uz ispuštanje adekvatnog zvuka). Zatim ponovo imamo normalno stanje letjelice MM4C pa stanje MM4D sa pomjerenim krugom (uz zvuk), itd. Dakle, 4 puta mijenjamo stanje sa normalnog na stanje sa krugom, pri čemu se na kraju vraćamo na polazno stanje MM4I. U tom momentu treba da se dogodi napad (ovdje smo uzeli kao primjer Gobo4 a ista priča važi i za Gobo6). Dakle, potreban nam je neki zvuk koji treba da se čuje prilikom izmjena stanja letjelica Gobo4 i Gobo6. Koristit ćemo zvuk po imenu

electric-90746

Upload-ujemo ovaj zvuk na Sounds za Gobo4 i Gobo6 te sačuvamo izmjene



Kako smo već opisali, ideja je da nakon odbrojanja, Gobo4 (ili Gobo6) započne sa pripremom za korištenje main oružja. Drugim riječima, gledajući za Gobo4, potrebna će nam biti neka promjenjiva, čiju ćemo vrijednost inicijalno postaviti da bude jednaka 0, kojoj ćemo nakon odbrojanja postaviti vrijednost da bude jednaka 1, što će značiti da već opisana priprema oružja otpočne sa radom (prolaz kroz 9 opisanih kostima), nakon čega ćemo ponovo vrijednost pomenute promjenjive postaviti da bude jednaka 0 (da bi bila spremna za novi ciklus aktivacije oružja). Dakle, trebaju nam dvije nove promjenjive, jedna za Gobo4 i druga za Gobo6. Ove promjenjive ćemo zvati Munja4 i Munja6. Definišimo ove promjenjive



Ove promjenjive ne moraju biti aktivne na ekranu, pa kvakice pored njihovog

imena možemo odmah ukinuti.

Sada, kada znamo plan rada oružja, tj. algoritam u glavnim crtama, u stanju smo da iskodiramo dio koji se odnosi na pripremu oružja, tj. prolazak kroz 9 kostima za Gobo4 odnosno Gobo6. Pošto se u ovom dijelu koriste kostimi za Gobo4 odnosno Gobo6, to je jasno da će ovi kodovi (za pripremu upotrebe main oružja) da budu pozicionirani na Code za Gobo4, tj. Gobo6. Idemo na Gobo4, na Code, i unosimo sljedeći kod (možemo ga pozicionirati lijevo od postojeća dva koda: lijevog - za pozicioniranje i kretanje Gobo4, desnog - za eliminaciju Gobo4)



Kod smo razložili na 4 dijela: prvi, drugi, treći, četvrti. Nakon pojašnjenja ove kodove ćemo objediniti u jedan kod.

Prvi dio predstavlja glavnu petlju koda. Početak pripreme oružja počinje nakon što dobijemo informaciju da je Munja4 jednako 1. Znači ako je Munja4 jednako 1, priprema oružja počinje sa radom. Jasno, mi ne znamo kada će Munja4 da postane jednako 1, zbog toga za "hvatanje" pravog momenta koristimo forever petlju. Primijetimo da informaciju da je Munja4 jednako 1 moramo dobiti izvana, tj. od nekog drugog koda. Kako ćemo vidjeti, riječ je o kodu oružja (govorimo o glavnom oružju za Gobo4 koje ćemo tek izgraditi). Ukoliko govorimo o Gobo6 onda umjesto promjenjive Munja4 riječ je o promjenjivoj Munja6.

Drugi kod govori o smjeni 9 kostima koje smo uradili. Pretpostavlja se da smo na početku pozicionirani na prvi kostim MM4A (za Gobo4) odnosno MM6A (za Gobo6). Nakon toga prolazimo kroz ostalih 8 kostima (dva kostima po jednoj iteraciji petlje). Dakle, imamo petlju od 4 iteracije, i u svakoj iteraciji promijenimo po dva kostima (ukupno 8 kostima). Uočimo da ovdje moramo postaviti uslove: da je item 4 liste DaljinaLetjelica manji od 5000, što, kako smo i ranije vidjeli, znači da je Gobo4 živ, tj. da je prisutan na ploči (inače ako bi pomenuta daljina bila mnogo veća od 5000 to bi značilo da ili letjelice još uvijek nisu raspoređene na ploču, ili da je Gobo4 eliminisan

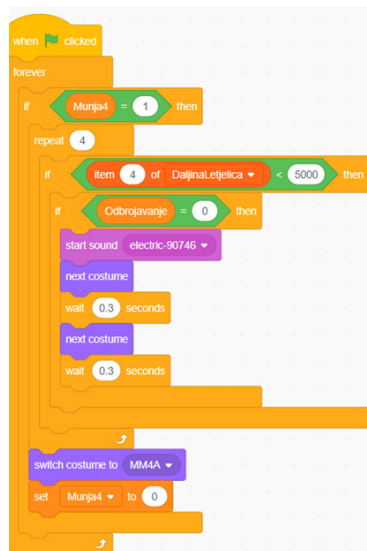


sa ploče), te uslov da je Odbrojavanje jednako 0 (ovo kako znamo znači da su letjelice prisutne na ploči, jer u suprotnom, tj. ako je Odbrojavanje različito od nule, onda to znači da ili nismo ušli u igru, ili da smo ušli u igru ali da su svi neprijatelji eliminisani sa ploče). Dakle, navedeni uslovi su postavljeni zbog toga što priprema navedenog oružja i ima samo smisla ako su ovi uslovi zadovoljeni.

Kod koji je treći po redu je ustvari kod koji treba da bude sastavni dio drugog koda, tj. treba da u njemu figuriše. Kako smo već djelimično opisali u prethodnom koraku, u svakoj od 4 iteracije prethodne petlje, treba da se smijene po dva kostima. Između tih smjena, treba da se sačeka neko vrijeme, recimo 0.3 sekunde, a na samom početku opisanih dešavanja treba da se čuje električni zvuk electric-90746. Na ovaj način smo kompletirali opis trećeg koda.

Konačno, četvrti kod je kod koji dolazi na kraju, tj. kao nastavak koji treba da se nadoveže na kraj repeat petlje iz drugog dijela. Jasno je da kada je proces pripreme završen, tj. kada smo prošli kroz sve navedene kostime, da treba ponovo da se postavimo na polazni kostim MM4A (za Gobo4), tj. MM6A (za Gobo6), te da promjenjivu Munja4 ponovo postavimo na vrijednost 0 (da bi bila spremna za ponovno djelovanje). Isto važi i za Gobo6, samo što je u njegovom slučaju riječ o promjenjivoj Munja6.

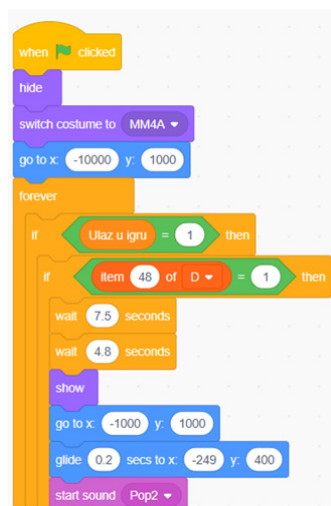
Uklopimo kod u jednu cjelinu. Riječ je o kodu





Opet, ističemo da ako je riječ o letjelici Gobo6, onda je u prethodnom kodu potrebno Munja4 zamijeniti sa Munja6, item 4 zamijeniti sa item 6, te kostim MM4A zamijeniti sa kostimom MM6A. Upravo navedeni kod možemo kopirati i zalijepiti na Gobo6, a onda dok smo na Gobo6, izvršiti navedene izmjene. Uradimo kako smo upravo opisali, tj. kopirajmo uravo navedeni kod i zalijepimo ga na kod za Gobo6, te izvršimo navedene izmjene. Sačuvamo izmjene.

Postavimo se ponovo na kod za Gobo4. Kako smo već istakli, Gobo4 ili Gobo6 treba da startaju sa početnim kostimima MM4A i MM6A. Potrebno je tako da u kodu postavimo da navedeni kostimi budu njihovi inicijalni kostimi. Možemo to uraditi na srednjem kodu, tj. na kodu koji do sada koristimo kao kod za pozicioniranje Gobo-a i kretanje desno - lijevo. Dovoljno je na vrhu srednjeg koda dodati naredbu za pozicioniranje na prvi kostim. Tako, modifikovani srednji kod za Gobo4 treba da ima oblik

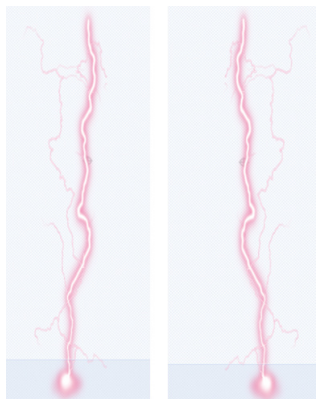


Jasno, u slučaju Gobo6, imali bi naredbu: switch costume to MM6A. Sačuvamo izmjene.

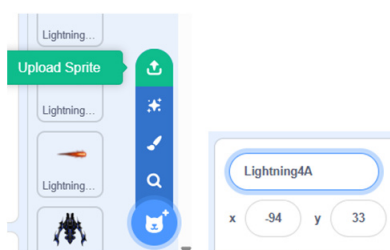
Sada se možemo pozabaviti izradom main oružja za Gobo4 (Gobo6).

Posmatrajmo Gobo4. Analogno će da vrijedi za Gobo6 sa odgovarajućim izmjenama.

Koristit ćemo dva kostima za oružje za Gobo4. Riječ je o kostimima koje ćemo zvati L4A i L4B za Gobo4, odnosno L6A i L6B za Gobo6

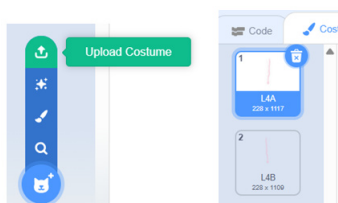


Inače, kompletno oružje za Gobo4 ćemo zvati Lightning4A, dok ćemo ono za Gobo6 zvati Lightning6A. Ovdje je bitno primijetiti da pomenuti kostimi predstavljaju ustvari jedan te isti kostim samo okrenut simetrično u odnosu na  $y$  osu. Naime, želja nam je da munja koja udara 4 puta, ne udara u jedno te isto mjesto. Zbog toga vršimo alterniranje, prvo jedan kostim pa drugi, pa nakon toga ponovo još jednom prvi pa drugi kostim. Ono što se još može uočiti je to da ovi kostimi prelaze donjim dijelom vidljivu površinu kostima. Ovo radimo zbog toga što želimo da prilikom pojave ovih munja na ploči imamo utisak da one idu skroz do dna. Naime, ne bismo ostavljali adekvatan utisak munje ako bismo ostavili da ona lebdi negdje u zraku, tj. da ide do nekog dijela vidljivog ekrana i onda samo nestaje bez ikakvog razloga i objašnjenja. Idemo na Choose a Sprite pa na Upload Sprite, pa upload-ujemo L4A. Ime Sprite-u damo da je Lightning4A

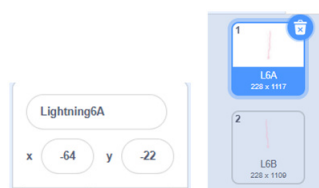


Zatim, odemo lijevo na Choose a Costume pa na Upload Costume, te upload-

ujemo kostim L4B



Dupliramo Sprite Lightning4A. Novom Sprite-u damo ime Lightning6A, a njegovim kostimima imena L6A i L6B

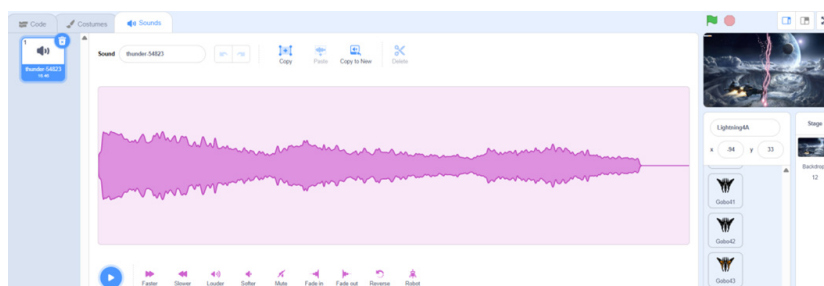


Pozicioniramo se na Lightning4A i sačuvamo izmjene.

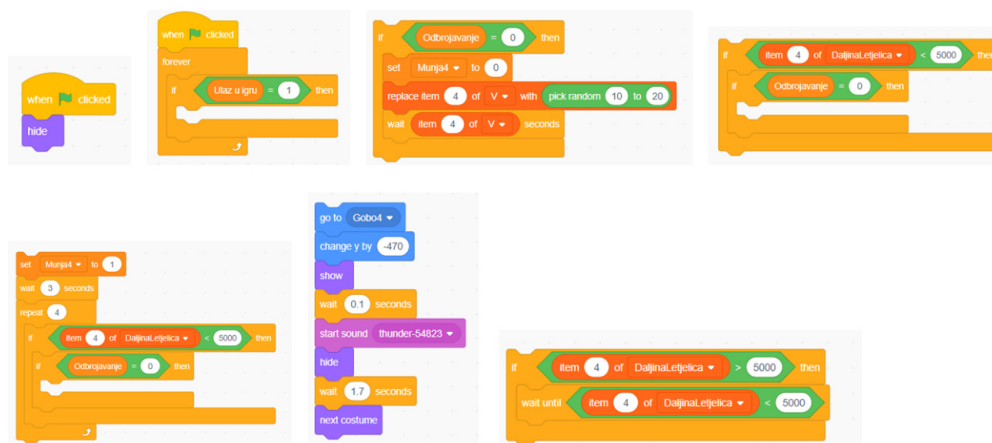
Trebat će nam zvuk za Lightning4A i Lightning6A. Koristit ćemo zvuk po imenu

thunder-54823

Upload-ujemo ovaj zvuk na Lightning4A i Lightning6A i sačuvamo izmjene



Sada možemo pristupiti kodiranju Sprite-a Lightning4A (analogno onda za Lightning6A). Pozicioniramo se na Lightning4A, pa na Code, i unosimo sljedeći kod



Kod smo razložili na elemente da bi lakše kroz njega prošli. Naravno, kao i u slučaju prethodnih kodova ovakve vrste, kod će kasnije biti objedinjen. Napisane kodove možemo tretirati kao: prvi, drugi, ..., sedmi.

Ono što se odmah može uočiti je to da upravo navedeni kodovi imaju itekakve sličnosti sa kodovima oružja koje smo ranije radili. Naime, logika ponašanja tih oružja je u određenim aspektima zajednička dok je u nekim drugim aspektima potpuno nova i različita.

Prvi kod je očekivan, jer munja ne treba da je vidljiva u momentu kada pokrećemo igru, i kao takva treba da ostane nevidljiva sve dok drugačije ne kažemo.

Drugi kod je takođe očekivan. Naime, svo dešavanje vezano za to da Gobo4 ili Gobo6 koristi munju kao oružje, ima smisla od momenta kada uđemo u igru. Dakle, nema smisla da govorimo o aktiviranju oružja ako nismo ni ušli u igru. Zbog toga je neophodan uslov da je Ulaz u igru jednako 1. Kao hvataljku kao i do sada koristimo forever petlju.

U trećem kodu postavljamo dodatni uslov da je Odbrojavanje jednako 0. Kako smo i do sada više puta ponovili, činjenica da je Odbrojavanje različito od nule znači ili da nismo ušli u igru ili da smo ušli u igru ali da smo sve neprijatelje eliminisali sa

ploče. Kako znamo, u takvim situacijama nema smisla govoriti o aktivaciji bilo kakvog oružja. Tako, ako je Odbrojavanje jednako 0, onda inicijaliziramo promjenjivu Munja4 da bude jednaka 0 (sjetimo se da pripremno oružje na Gobo4 i Gobo6 počinje da radi ako je Munja4, tj. Munja6 jednako 1). Zauzimamo četvrtu lokaciju u listi V nekim proizvoljnim vremenom iz intervala 10 do 20 sekundi. Sjetimo se da smo za prethodno oružje koristili 4 liste: V, V2, V3, V4. Naime, tada smo koristili 4 hica jednog te istog oružja. Sada koristimo samo jedan hitac pa nemamo potrebu da koristimo dodatne liste: V2, V3, V4. Očekivano, sačekamo da vrijeme koje smo rezervisali u listi V istekne (za Lightning6A bi koristili item 6 liste V i promjenjivu Munja6 u analognom kodu). Ovdje je važno primijetiti da smo vrijeme čekanja stavili da bude jednako nekom vremenu između 10 i 20 sekundi. Naime, oružje koje koristimo je po prirodi opasnije od oružja koje smo već pravili, pa s te strane treba i rijeđe da se pojavljuje u igri. Naravno, stvar je kreatora koliko će smatrati da je potrebno da vrijeme čekanja u stvarnosti bude.

Četvrti kod: nakon isteka čekanja u prethodnom dijelu, potrebno je da aktiviramo pripremu oružja. Sve što radimo na bilo kojem nivou prilikom izrade ovog oružja, mora biti popraćeno zadovoljenjem uslova: da je item 4 liste DaljinaLetjelica manji od 5000 (što znači da je Gobo4 na ploči), i zadovoljenjem uslova: da je Odbrojavanje jednako 0. Naime, može se desiti da mi vrlo brzo eliminišemo Gobo4, čak i dok odbrojavanje vremena čekanja traje. Glupo bi bilo vidjeti u igri da se munje pojavljuju na ekranu ili da odjekuju negdje u daljini dok je Gobo4 eliminisan. Dakle, ovo je razlog zašto stalno moramo postavljati pomenuta dva uslova.

Peti kod treba da dođe u sklopu uslova datih u četvrtom kodu. Znači nakon isteka vremena čekanja, pristupamo aktivaciji oružja, pri čemu prvo treba da aktiviramo pripremno oružje. Kako smo već iskodirali, aktivacija pripremnog oružja počinje dostavljanjem kodu za Gobo4 informacije da je Munja4 jednaka 1. Dakle, na ovom mjestu postavljamo Munja4 da bude jednako 1. Sada se u kodu za Gobo4 aktivira kod za pripremno oružje, koji će da traje nekoliko sekundi. Potrebno je da sačekamo da se taj dio izvrši. Zbog toga moramo sačekati neko vrijeme, a to vrijeme je otprilike jednako 3 sekunde. Otud naredba za čekanje 3 sekunde. Nakon toga aktiviramo glavno oružje. To radimo petljom od 4 iteracije. Pri svakoj iteraciji ponovo provjeravamo da li je Gobo4 živ, tj. da li je item 4 liste DaljinaLetjelica manji od 5000, i da li je Odbrojavanje jednako nula. Ako su zadovoljeni ovi uslovi, možemo pristupiti upotrebi munje, tj. možemo preći na sljedeći kod.

Šesti kod kako već vidimo predstavlja aktivaciju munje za Gobo4. Naređujemo

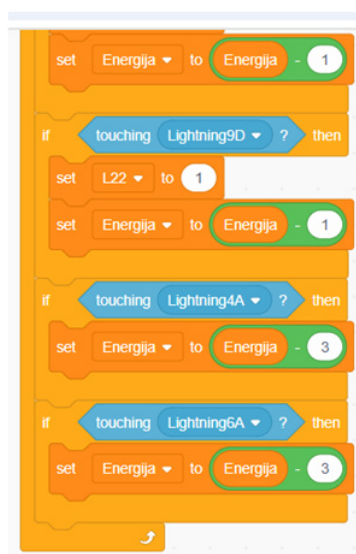
Konačno, sedmi kod je kod koji nam govori šta da radimo onda kada je Gobo4 mrtav. Dakle, ako je Gobo4 mrtav onda čitav kod treba da sačeka sa izvršavanjem dok Gobo4 ponovo ne postane živ. Pretočeno u jezik koda, ovo znači da ako je Gobo4 daleko, da treba da sačekamo dok on ne priđe blizu, odnosno da ako je item 4 liste DaljinaLetjelica veći od 5000, da kod treba da sačeka dok taj item ne postane manji od 5000. Kodove, osim prvog uklopimo u jedan. Dobijamo kod



U slučaju Gobo6, mijenjamo Munja4 u Munja6, item 4 liste DaljinaLetjelica u item 6 te iste liste, te naredbu go to Gobo4 u naredbu go to Gobo6. Možemo dakle upravo navedeni kod kopirati, zalijepiti na Lightning6A i napraviti navedene izmjene. Sačuvamo izmjene.

U ovom dijelu ono što je ostalo da uradimo jeste pitanje dodira naše letjelice M14 sa nekim od gromova Lightning4A ili Lightning6A. Kako smo već istakli, ova oružja treba da budu jača od ranije pravljenih oružja, tako da pri dodiru sa našom letjelicom M14 treba da nam oduzmu i više energije. Kod koji se odnosi na ovaj dio ćemo podesiti tako da naša letjelica M14 gubi 3 podioka energije (za razliku od ranijih kada smo gubili samo jedan podiok) pri dodiru sa nekom od munja Lightning4A, Lightning6A.

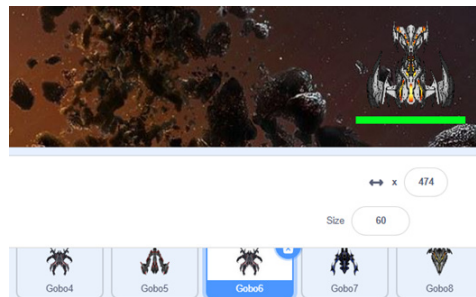
Idemo dakle na Sprite M14, na Code, i u kod za dodir letjelice M14 sa prethodnim oružjima, dodamo kodove za dodir M14 sa Lightning4A i Lightning6A. Inače, svakim novim dodavanjem oružja u igru ovaj kod moramo ponovo obogatiti. Nove kodove možemo dodavati na kraj. Tako, modifikovani kod na M14 je sljedeći (gledamo samo njegov kraj koji se odnosi na Lightning4A i Lightning6A)



Sačuvamo izmjene.

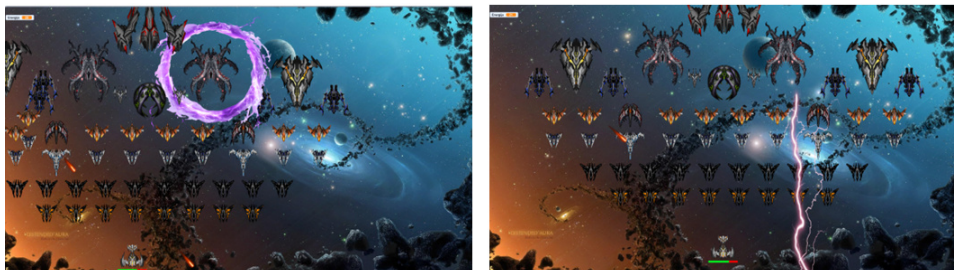
Prilikom pokretanja igre možemo vidjeti da se eventualno veličina Gobo4 i Gobo6 smanjila na veličinu koja nam ne odgovara. Sjetimo se da smo manuelno smanjivali njihovu veličinu da bi mogli dodati energetske krug oko njih. Sada veličinu možemo

podesiti, tj. povećati na Size za Gobo4 i Gobo6. Možemo tu veličinu postaviti tako da Size bude jednako 60



Sačuvamo izmjene i probamo igru.

Možemo uočiti da bi u kodu za Lightning4A i Lightning6A bilo dobro još pomjeriti prema dole (po  $y$  osi) pojavljivanje munja. Tako, umjesto: change  $y$  by -470 poželjno bi bilo staviti change  $y$  by -670. Sačuvamo izmjene. Probamo igru. Možemo vidjeti da se u njoj dešava upravo ono što smo i željeli



## 10.1 Riješeni zadaci

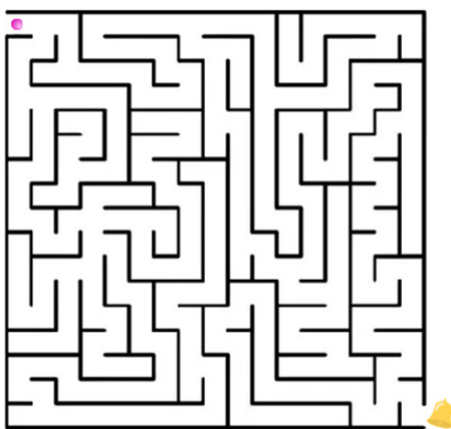
◇ **Zadatak 10.1.1** Napraviti igru u kojoj igrač upravlja lopticom kroz lavirint, od ulaza do izlaza.

Na početku, loptica se postavlja na ulaz u lavirint. Lavirint treba da sadrži crne zidove kroz koje loptica ne smije proći. Igrač koristi strelice na tastaturi za pomijeranje

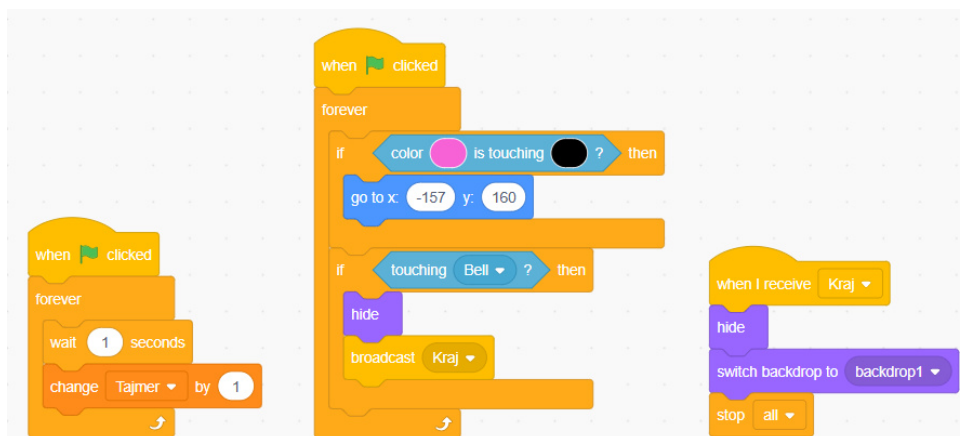


loptice lijevo, desno, gore i dole. Ako loptica dodirne crni zid, automatski se vraća na početnu poziciju.

Cilj je da loptica dođe do kraja lavirinta i da dodirne Sprite zvonce, čime se igra završava i ispisuje poruka o uspjehu.



### Rješenje:





## 10.2 Zadaci za vježbu

♣ **Zadatak 10.2.1** U Zadatku 10.1.1.1 proširiti postojeći program tako da se u lavirintu pojave dodatne prepreke koje se kreću (npr. lijevo-desno ili gore-dole). Igrač treba izbjegavati te prepreke dok pokušava doći do cilja. Prepreke mogu biti predstavljene novim Sprite-ovima (npr. pravougaonicima crvene boje) koji se kreću unutar određenih granica. Ako igrač (loptica) dodirne neku od prepreka, vraća se na početnu poziciju (kao i u slučaju dodira sa zidom lavirinta).

♣ **Zadatak 10.2.2** U momentu kada u Zadatku 10.1.1 igrač završi lavirint (prvi lavirint), tj. kada dodirne zvonce, automatski treba da pređe na drugi nivo s novim lavirintom, težim ili većim.

♣ **Zadatak 10.2.3** Dodati u Zadatak 10.1.1 brojač koji odbrojava od npr. 30 sekundi. Ako igrač ne dođe do cilja na vrijeme igra se završava porukom "Vrijeme isteklo!".

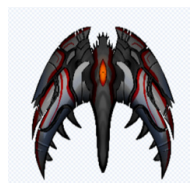
## 11 NPC algoritmi

Uveliko se već bavimo pitanjima dizajniranja, kretanja, interakcije objekata, te primjenom postojećih ali i razvojem raznih novih vrsta algoritama vezanih za pomenute objekte. Objekti nisu subjekti, pa na određeni način govorimo o klasičnim non-player karakterima (NPC). Kroz sve ovo što radimo mi ustvari diktiramo ponašanje NPC-ova, kako oni djeluju, kako misle, kakva im je interakcija u našem svijetu igre, što definitivno uključuje i podrazumjeva njihovo kretanje, pravljenje odluka ali i borbu.

Neprijateljske letjelice koje u nastavku želimo da naoružamo su tipa



Znamo da ovih letjelica ima 8. Njihova imena su: Gobo13, Gobo14, Gobo16-19, Gobo21, Gobo22. Znamo da navedeni brojni niz prekidaju letjelice tipa



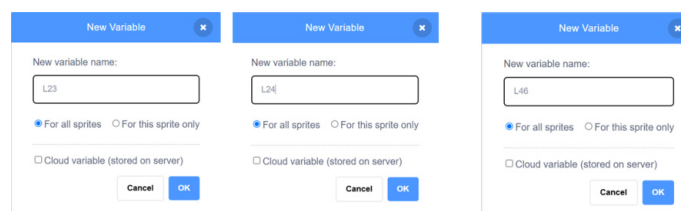
Imamo ukupno dvije ovakve letjelice a njihova imena smo označili sa Gobo15 i Gobo20. Dakle, u nastavku, bavimo se letjelicama prvog tipa.

Ideja nam je da za navedene letjelice napravimo oružje u obliku rotirajućih krugova, tj. rotirajućih energija. To mogu biti prazni diskovi kod kojih se samo ivica vidi ili popunjeni krugovi takvog oblika da je uočljivo njihovo rotiranje. Rotacija treba da stvori utisak da će oni pri dodiru sa našom letjelicom M14 da nanesu štetu. Oružje o kojem ovdje govorimo ne treba da bude oružje koje će biti izrazito jako (za razliku od gromova - munja za koje smo vidjeli da su jaka oružja i da nam oduzimaju 3 podioka energije). Oružje kojim se ovdje bavimo je više osmišljeno kao light oružje koje

u dodiru sa M14 oduzima samo jedan podiok energije. Govorimo dakle o kvantitetu naspram kvaliteta. Oružja tipa groma možemo smatrati kvalitetnijim. Rijeđe se aktiviraju, nema ih mnogo, ali nanose znatnu štetu. S druge strane pod kvantitativnim oružjima možemo smatrati ona kojih je brojačano mnogo ali koja nisu u velikoj mjeri opasna. Oružje koje ovdje razmatramo bi trebalo da bude jedno takvo oružje.

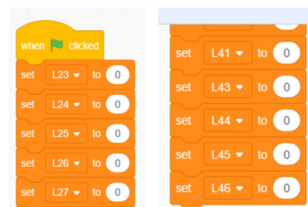
Zamisao je da svakoj od 8 navedenih letjelica pridružimo po 3 oružja istog tipa (ukupno 24 oružja). Možemo radi jednostavnosti govoriti o letjelici Gobo13. Dakle, Gobo13 će imati 3 oružja koja su nezavisna, tj. aktiviraju se nezavisno jedno od drugoga. Aktivacija će biti regulisana vremenom čekanja. Drugim riječima, svako od 3 oružja pridružena Gobo13 će imati neko startno vrijeme čekanja, nakon kojeg će započeti aktivacija oružja. Ta vremena će jasno da budu nezavisna (za svako od oružja koristimo random funkciju da rezervišemo neko inicijalno vrijeme čekanja). Sjetimo se da smo za skladištenje ovakvih vremena korisili liste: V, V2, V3 i V4. I ovdje ćemo postupiti isto. Dovoljno će biti da koristimo liste: V, V2 i V3. Kako je riječ o Gobo13, to ćemo za njega koristiti iteme 13 navedenih triju lista (što je očekivano).

Sjetimo se da smo prilikom izrade ranijih oružja ovakvog tipa koristili određene prekidače (radili su na principu 0 - 1). Njihova svrha je bila ta da nas informišu o statusu oružja: aktivno je (oznaka 0), te: nije aktivno (oznaka 1). Preciznije, inicijalna oznaka oružja je 0. Ukoliko oružje pogodi letjelicu M14 mijenjamo mu status u oznaku 1 (što znači da treba da ga sklonimo sa ploče). Ovdje ćemo koristiti isti rezon. Pošto imamo 8 letjelica sa po 3 oružja, dakle ukupno 24 oružja, trebat će nam isto toliko prekidača. Definišimo 24 promjenjive imena: L23, L24, ..., L46 (sjetimo se da smo već koristili prekidače L7 - L22)

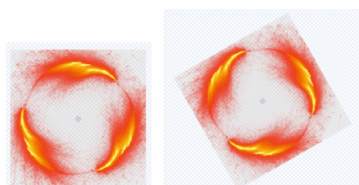


Ovi prekidači nam nisu neophodni u smislu njihove vidljivosti na ekranu, tako da kvakice pored njihovih imena možemo ukinuti.

Navedeni prekidači su definisani za sve Sprite-ove, tako da ih možemo inicijalizirati na bilo kojoj lokaciji. Po ustaljenoj praksi, to radimo onda na neutralnom mjestu, tj. na podlogama. Idemo na podloge, pa na kod, i unosimo sljedeći kod (nakon čega sačuvamo izmjene)



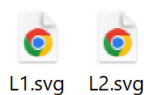
Za oružje koje gradimo ćemo koristiti dva kostima: L1 i L2 (možemo koristiti ove nazive za kostime svih 24 oružja)



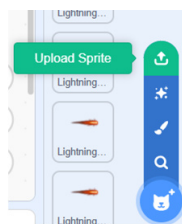
Jasno je da je kostim L2 nastao rotacijom kostima L1 za nekih 45 stepeni. Ukoliko vršimo stalnu izmjenu između ova dva kostima, jasno je da ćemo dobiti osjećaj rotacije. Nama je takođe jasno da prilikom pretrage transparentnih slika na netu ne možemo uvijek očekivati da dobijemo idealne slike bez nekakvih popratnih boja, efekata, i sl., odnosno bez nekakvih dodatnih elemenata za koje ne bismo htjeli da budu prisutni na slici. To imamo u slučaju navedenih kostima. Primijetimo da je osnovna boja slike narandžasta, da je podloga za kostime (dio Eticques-a) blijedo siva (ovo je oznaka za providnost - transparentnost), a da je dodatna podloga kostima nešto tamnije siva. Idealno bi bilo kada bi i ta dodatna podloga slike bila blijedo siva (svaka druga boja znači da slika nije u potpunosti transparentna). Naime, ovakve slike mogu biti problematične na ekranu jer su te dodatne boje vidljive korisniku. Često je potrebno probati neku sliku pa vidjeti kolike su njene mane i da li se one mogu na neki način zaobići. U slučaju konkretne slike, ako bismo je puštali na ekran u nekom statičnom obliku, onda bi njena dodatna podloga, koja nam ne odgovara, bila vidljiva oku, a to nije ono što se želi. Sretna okolnost je ta da mi ovu sliku rotiramo i to relativno brzo, tako da podloga slike (koja nam kako rekosmo smeta) više neće biti problem, nego će stvarati utisak sjene koja se stvara rotiranjem narandžastog kruga (zamislite da se navedena dva kostima međusobno izmjenjuju pri velikim brzinama). Dakle, dobijamo

jako dobar efekat iz nečega što nam inicijalno nije odgovaralo. Poenta je da pri izradi aplikacija (kakvom se mi bavimo ovdje) autor mora biti kreativan, te razne nepogodnosti znati okrenuti u svoju korist. Jasno je i to da smo mi navedene kostime mogli dobiti plaćanjem nekome za njihovu izradu. Na taj način bismo zaobišli sve upravo opisane negativne efekte free-online dokumenata. Ipak, ono što radimo ovdje je čisto u akademske a ne komercijalne svrhe, pa dodatni troškovi nisu neophodni.

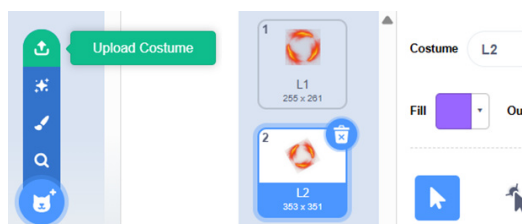
Upload-ujemo navedene kostime (sa lokacije na kojoj se nalaze)



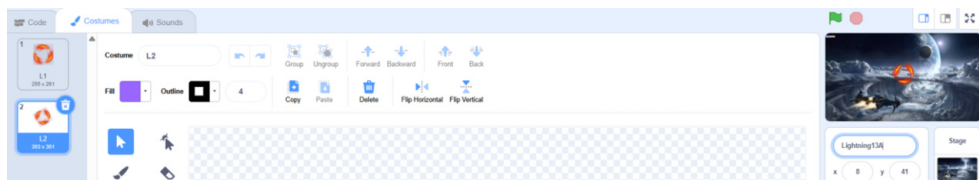
Dakle, idemo na Choose a Sprite, pa na Upload Sprite



te upload-ujemo L1. Zatim idemo lijevo na Choose a Costume, pa na Upload Costume, te upload-ujemo L2

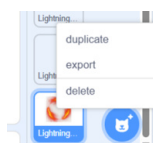


Ime koje ćemo dati ovom oružju (prvom oružju za Gobo13) će biti Lightning13A

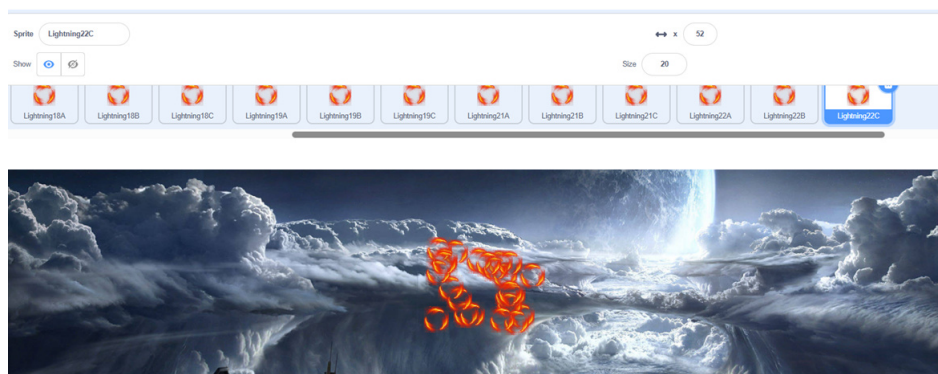


Jasno je onda da će imena za preostala dva oružja za Gobo13 da budu Lightning13B i Lightning13C. Takođe, iz konteksta je onda jasno kako će izgledati nazivi oružja za preostale Gobo-e. Tako, imena Lightning17A, Lightning17B, Lightning17C će biti rezervisana za Gobo17, itd.

Preostala 23 oružja su identična sa Lightning13A. Dovoljno je onda duplicirati Lightning13A 23 puta i na novo-dobijenim oružjima promijeniti nazive na način kako smo to upravo opisali. Uradimo to i sačuvajmo izmjene (voditi računa da preskačemo brojeve 15 i 20 zbog Gobo15 i Gobo20)



Size za napravljena oružja ćemo staviti da je jednak 20



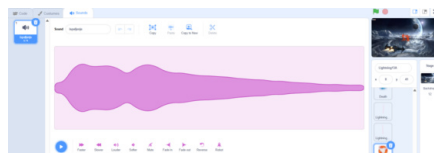
Sačuvamo izmjene.

Oružja koja smo do sada pravili su radila na principu da nakon isticanja njihovog vremena čekanja, oružje odlazi do odgovarajućeg Gobo-a, pojavljuje se i onda nastavlja da radi po principu predviđenom tačno za njega: ide prema dnu ekrana ili blinka u mjestu određeni broj puta (munje). Znači, između isteka vremena čekanja i odlaska do Gobo-a nismo imali neki dodatni algoritam za oružje.

U slučaju oružja koje trenutno pravimo postupit ćemo nešto drugačije. Naime, Gobo-i sa kojima sada radimo se svojim pozicijama nalaze dosta niže na ploči u odnosu na Gobo-e čija smo oružja do sada pravili. S te strane, ne bi bilo sasvim korektno da ovi Gobo-i ispaljuju oružja direktno sa svoje pozicije. Naime, ovih Gobo-a ima mnogo, a i njihovih oružja ima mnogo, pa s obzirom da su ovi Gobo-i dosta nisko na ploči, i da nastavljaju da se kreću naniže, to bi njihova oružja u tom slučaju vrlo brzo došla do nas, tj. do letjelice M14. Iz tog razloga ćemo kod posmatranih oružja dizajnirati tako da se svako takvo oružje nakon isteka vremena čekanja, pojavljuje pokraj odgovarajućeg Gobo-a, nakon čega klizi neko kratko vrijeme prema gornjem dijelu ploče. Na ovaj način ćemo stvoriti utisak da posmatrani Gobo ustvari raspoređuje svoja oružja po gornjem dijelu ploče. Naime, postupak klizanja (glide) će jasno ostaviti utisak o tome koji Gobo ispaljuje svoja oružja. Takođe, ovim putem ćemo izbjeći situaciju u kojoj su posmatrana oružja u startu blizu letjelici M14. Jasno, nakon raspoređivanja, oružje će nastaviti da putuje prema dole, tj. prema dnu ekrana. Pri tome ćemo omogućiti da se oružje blago usmjerava prema M14. Da bi se postigao navedeni efekat, potrebno je u kodu oružja narediti da se to oružje u svakoj iteraciji usmjerava prema M14 (i putuje tako neki broj koraka), ali da istovremeno ide prema dnu ekrana za određeni broj koraka, pri čemu je broj koraka usmjerenih prema dnu nešto veći od broja koraka usmjerenih prema M14. Koraci koji idu prema dole će da "vuku" oružje prema dnu, dok će koraci usmjereni prema M14 djelimično korigovati putanju prema M14.

Prilikom raspoređivanja oružja od Gobo-a do gornjeg dijela ploče (klizanjem), potrebno je da se čuje neki zvuk. Naime, potreban nam je zvuk koji asocira na to da se nešto otkačilo - ispalilo (kao da smo oružje otkačili - ispalili od Gobo-a). Koristit ćemo zvuk po imenu: ispaljenje

Upload-ujemo ovaj zvuk na svako od 24 oružja





Ovaj zvuk treba da se aktivira samo prilikom ispaljenja (odaslanja) oružja od Gobo-a. Sačuvamo izmjene.

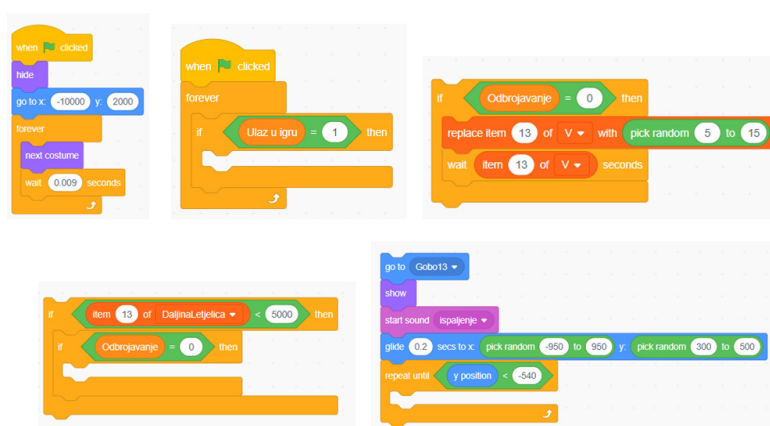
Inače, zvuk koji treba da se čuje tokom kretanja oružja prema dnu ekrana i prema nama (M14) je zvuk po imenu: blago

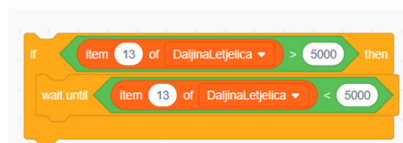
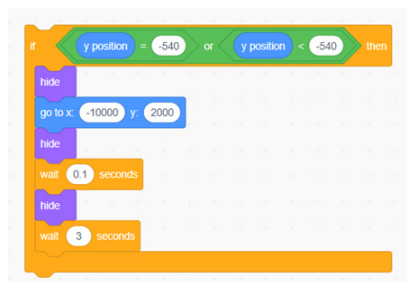
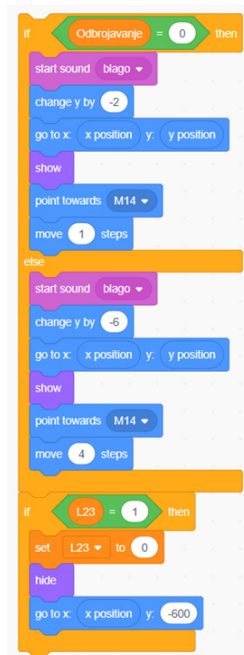


Naime, riječ je o blagom zvuku energije. Mi svakako govorimo o oružju koje je neka vrsta energije koja se rotira i kreće prema nama. Za takvo oružje popratni zvuk treba da je: energija, rotirajuća energija, energetska žaga, energetski disk, i slično, prilikom potrage za adekvatnim zvukom. Uplaod-ujemo ovaj zvuk na svako od 24 oružja i sačuvamo izmjene.

Sada smo u poziciji da možemo pristupiti izradi koda za Lightning13A.

Idemo na Lightning13A, pa na Code, i unesemo sljedeći kod (kдове)





Ove kodove ćemo zvati: prvi, drugi, ..., osmi. Kao i do sada objasniti ćemo detaljno način rada ovih kodova a nakon toga i metod njihovog uklapanja. Naravno, na kraju, navest ćemo koje izmjene je potrebno uraditi u objedinjenom kodu da bi on mogao biti upotrebljiv i na preostalih 23 oružja.

U prvom kodu sakrivamo oružje jer ono ne treba da bude vidljivo do mometa kada

zaista želimo da ono bude vidljivo. Oružje čuvamo na lokaciji rezervisanoj za njega, a to je lokacija (-10000,2000). Želimo da se cijelo vrijeme smjenjuju dva kostima oružja koje imamo (L1 i L2), i to bez obzira na to da li su oružja sakrivena ili otkrivena, i bez obzira na to da li se nalaze na ekranu ili na rezervisanoj lokaciji ili na nekom trećem mjestu. Brzina rotiranja će biti dosta velika, tj. 0.009 sekundi. Ovaj kod ne zahtijeva adaptacije za ostala oružja.

Drugi kod je glavna petlja cijelog koda oružja. Ako je Ulaz u igru jednako 1, tj. ako smo ušli u igru pritiskom na dugme START, onda ima smisla da ovo oružje funkcioniše (kao i oružja koja smo pravili ranije). Klik na START "lovimo" kao i do sada forever petljom. Kod ne zahtijeva adaptaciju za ostala oružja.

Treći kod je prvi kod koji se uklapa u prethodni kod, tj. prvi kod vezan za rad našeg oružja. Ovdje rezervišemo vrijeme čekanja u listu V (ona je prva od 4 liste: V, V2, V3, V4 koje koristimo u ove svrhe). Biramo vrijeme između 5 i 15 sekundi, što je adekvatno za ovo oružje. Dakle, čekamo da ovo vrijeme istekne da bi počeli sa aktivacijom oružja. Jasno, kao i u slučaju ranijih oružja koja smo pravili, potrebno je da Odbrojavanje bude jednako 0 (ako je Odbrojavanje različito od nule onda to znači da ili nismo ušli u igru, ili da smo ušli u igru ali da su svi neprijatelji eliminisani). Za oružje Lightning13B bi koristili item 13 liste V2 a za Lightning13C item 13 liste V3. Jasno, onda, za Lightning14A bi koristili item 14 liste V, za Lightning14B item 14 liste V2, itd.

Kada smo završili sa izvršavanjem trećeg koda, tj. kada je isteklo vrijeme čekanja, onda je potrebno da se oružje Lightning13A aktivira. Iz rada sa prethodnim oružjima je jasno da ovaj korak ima smisla da se poduzme samo onda kada je Gobo13 na ploči, tj. kada nije "predaleko" od naše letjelice M14. Kao što znamo to je slučaj ako je item 13 liste DaljinaLetjelica manji od 5000, i kada je istovremeno Odbrojavanje jednako 0. Dakle, ako su ova dva uslova ispunjena onda je Gobo13 sigurno na ploči, pa ima smisla započeti aktivaciju oružja. Upravo rečeno predstavlja opis četvrtog koda. Za oružja Lightning13B i Lightning13C ovaj kod se ne mijenja (jer se i ova oružja odnose na Gobo13). Jasno, onda, za oružja Lightning14A, Lightning14B, Lightning14C, u ovom kodu se posmatra item 14 liste DaljinaLetjelica. Analogno važi za ostala oružja.

Šta je to što prvo radimo prilikom aktivacije oružja Lightning13A ? To je sadržaj petog koda. Pošaljemo oružje do Gobo13, učinimo ga vidljivim, započnemo zvuk ispaljenja, te omogućimo da oružje klizi od Gobo13 do neke proizvoljne pozicije unutar

gornjeg dijela vidljive ploče. Ovdje koristimo funkciju `random` da odaberemo neku proizvoljnu lokaciju  $(x, y)$  unutar gornjeg dijela ploče. Pod gornjim dijelom ploče podrazumijevamo tačke  $(x, y)$  kod kojih je  $-950 < x < 950$  i  $300 < y < 500$ . Nakon rasporeda oružja u gornji dio ploče, treba da započne njegovo kretanje prema dole, za što nam služi petlja: `repeat until y < -540`. Naime, ima smisla da pomjeramo oružje sve dok ono ne nestane iza donjeg ruba ekrana. Primijetimo da `glide` u gornji dio ploče traje 0.2 sekunde, što je sasvim dovoljno da se stekne utisak da oružje polazi od Gobo13. Za oružja `Lightning13B` i `Lightning13C` ostaje isti ovaj kod. Za oružja `Lightning14A`, `Lightning14B`, `Lightning14C`, mijenjamo Gobo13 u Gobo14. Analogno za ostala oružja.

U šestom kodu se bavimo kretanjem Gobo13 prema dnu ploče. To je kod koji se uklapa u prethodni kod. Ugrubo, u ovom kodu imamo nekoliko cjelina, prvu: ako je Odbrojavanje jednako 0 (što znači da igra nije u fazi među - koraka kada se letjelice raspoređuju na ploču), drugu: ako je Odbrojavanje različito od nule (to je `else` dio koji znači da su svi neprijatelji eliminisani sa ploče), i treću: ako je pokazivač L23 jednak 1 (što znači da je oružje `Lightning13A` dodirnulo našu letjelicu M14). Razmotrimo šta raditi u svakoj od ovih cjelina. Ako smo u prvoj cjelini, tj. ako je Odbrojavanje jednako 0, onda to znači da `Lightning13A` treba da se kreće prema dole. Dakle, startujemo zvuk "blago", pomjerimo se prema dole (po  $y$  osi) za -2 koraka, prikazemo oružje (da se osiguramo, iako smo ga već prikazali iznad), usmjerimo oružje prema letjelici M14, krećemo se prema njoj 1 korak. Ovdje je bitno uočiti da prema dole idemo 2 koraka, a prema M14 samo jedan korak. Važno je da broj koraka prema dole bude veći od broja koraka prema M14. Na ovaj način oružje teži da ide prema dnu uz blagi naklon prema M14. U drugoj cjelini, ako je Odbrojavanje različito od 0, onda imamo situaciju da su svi neprijatelji mrtvi. To je među - faza. Ako se oružje nađe u ovoj fazi, onda ono definitivno ne treba da bude na ploči (može se desiti da je oružje ostalo u ovoj fazi nakon što smo eliminisali sve neprijatelje sa ploče). U ovakvoj situaciji oružje treba što prije da se skloni sa ploče da ne bi smetalo, pa u ovom slučaju ponovimo sve korake kao u prethodnom slučaju, ali kretanje oružja znatno ubrzamo (da bi što prije nestalo sa ploče). Tako, umjesto -2, prema dole se sada krećemo brzinom -6, osim toga umjesto 1, prema M14 sada idemo brzinom 4. Konačno, u trećoj cjelini, imamo situaciju da je L23 jednak 1. To znači da je oružje `Lightning13A` dodirnulo letjelicu M14. U tom slučaju oružje treba da sklonimo na dno ekrana. Dakle, L23 odmah postavimo da je jednako 0 (da bude spremno za buduću upotrebu oružja), oružje sakrijemo, i premjestimo ga na dno. To je pozicija  $(x, -600)$ . Naime, koordinatu  $x$  ne mijenjamo, nego samo oružje "spustimo" prema dole do  $y = -600$ . Ovaj šesti kod na neki način predstavlja i glavni dio koda ovog oružja. Vezano za adaptaciju koda za ostala oružja,

jasno, za Lightning13B bi koristili pokazivač L24, za Lightning13C pokazivač L25, za Lightning14A pokazivač L26, itd.

Sedmi kod je kod koji se nadovezuje na peti kod. Ovdje, ako oružje Lightning13A dođe do dna ekrana ili pređe dno ekrana, onda ga sakrijemo i pošaljemo na rezervisanu lokaciju (-10000,2000). Pri tome, kao i do sada, sakrivanje provedemo nekoliko puta sa određenim vremenskim stankama između, da bi bili sigurni da će sakrivanje zaista biti i provedeno. Kod je identičan za sva oružja.

Konačno, osmi kod je kod koji je analogan kodovima koje smo imali i kod ranijih oružja. Naime, ako je item 13 liste DaljinaLetjelica veći od 5000, to znači da je Gobo13 mrtav, pa u tom slučaju izvršavanje koda treba da sačeka dok Gobo13 ponovo ne postane živ, tj. dok item 13 liste DaljinaLetjelica ne postane manji od 5000. Ovaj kod je identičan za oružja Lightning13B i Lightning13C (jer se i ova oružja vezuju za Gobo13). U slučaju oružja Lightning14A, Lightning14B, Lightning14C bi umjesto item 13 liste DaljinaLetjelica koristili item 14 (jer se sva tri ova oružja vezuju za Gobo14). Analogno za ostala oružja.

Uklopimo kodove.

Prvi kod ostaje isti



Ostali kodovi uklopljeni u jedan imaju sljedeći oblik

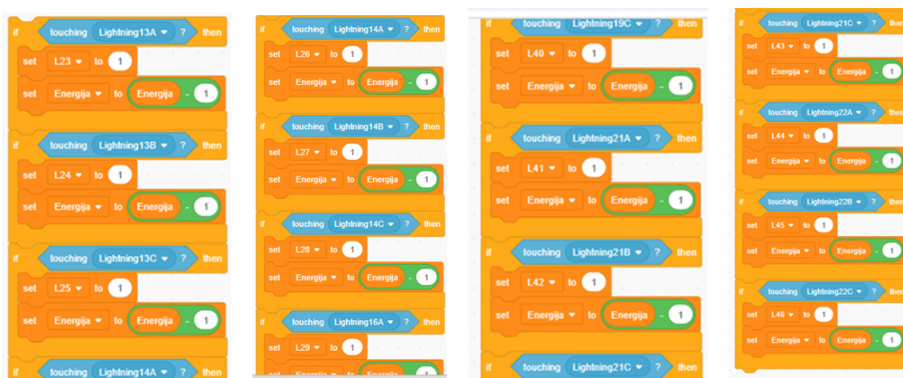


Sada, kodove sa Lightning13A zalijepimo na svaki od preostala 23 oružja. Napravimo izmjene kako smo iznad opisali i sačuvamo ih. Probamo igru

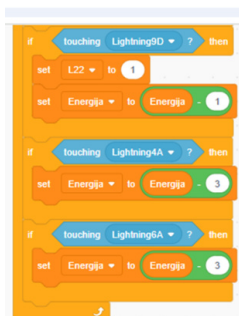


Nova oružja se zaista i ponašaju onako kako smo željeli. Idu prema dole, i blago teže prema nama, tj. prema letjelici M14. Na kraju nivoa se može vidjeti da ako eliminišemo sve letjelice, a neko od oružja koja smo upravo napravili ostane aktivno na ekranu, da onda to oružje ide znatno brže ka dnu ekrana.

Primijetimo da novo-nastala oružja ne ranjavaju našu letjelicu M14. Naime, još uvijek nisamo adaptirali kod da susret novih oružja sa M14 izazove bilo kakav efekat. Oružja koja smo upravo napravili nećemo smatrati previše opasnim oružjima, pa ćemo pri njihovom dodiru sa M14 letjelici M14 oduzimati samo jedan podiok energije. Uradimo još taj dio posla. Idemo na Sprite M14, pa na Code, i tamo unesemo kod



Ovaj kod uklopimo u generalni kod koji reguliše pitanja dodira letjelice M14 sa drugim objektima u igri. Dakle, upravo unijeti kod uklopimo u kod



Dovoljno je da ga u ovaj kod dodamo na kraj, iza završetka posljednje naredbe If. Sačuvamo izmjene i probamo igru.

Moćemo vidjeti da nova oružja sada ranjavaju našu letjelicu M14



Takođe, vidi se da neprijateljskih oružja za sada ima poprilično mnogo. Naime, za prethodna oružja smo već istakli da će biti takvog tipa da ih ne možemo prevenirati. Dakle, nemamo, niti ćemo praviti oružja za letjelicu M14 koja će biti sposobna da eliminišu pomenuta oružja. S druge strane, ideja je da za neprijateljska oružja koja smo upravo izgradili napravimo odgovarajuće anti - oružje.



## 11.1 Riješeni zadaci

◇ **Zadatak 11.1.1** Napraviti igru preskakanja na sljedeći način:

Kreirati pozadinu za početak i kraj igre (pobjedu ili poraz). Dodati likove: "pile" i "lopta".

Napraviti mogućnost skakanja pritiskom na tipku space ili strelicu "gore".

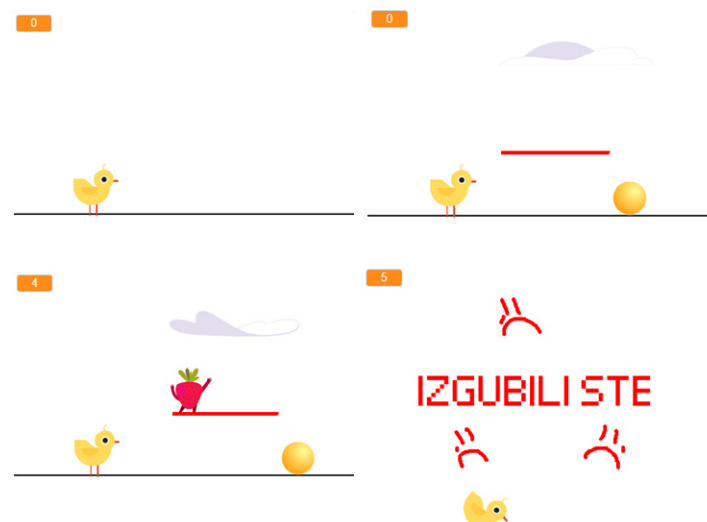
Simulirati nadolazeće prepreke, tj. omogućiti kreiranje klonova lopte koji idu prema piletu.

Kreirati varijablu "rezultat" čija je vrijednost na startu jednaka 0. Vrijednost treba da se poveća svaki put kada pile preskoči loptu.

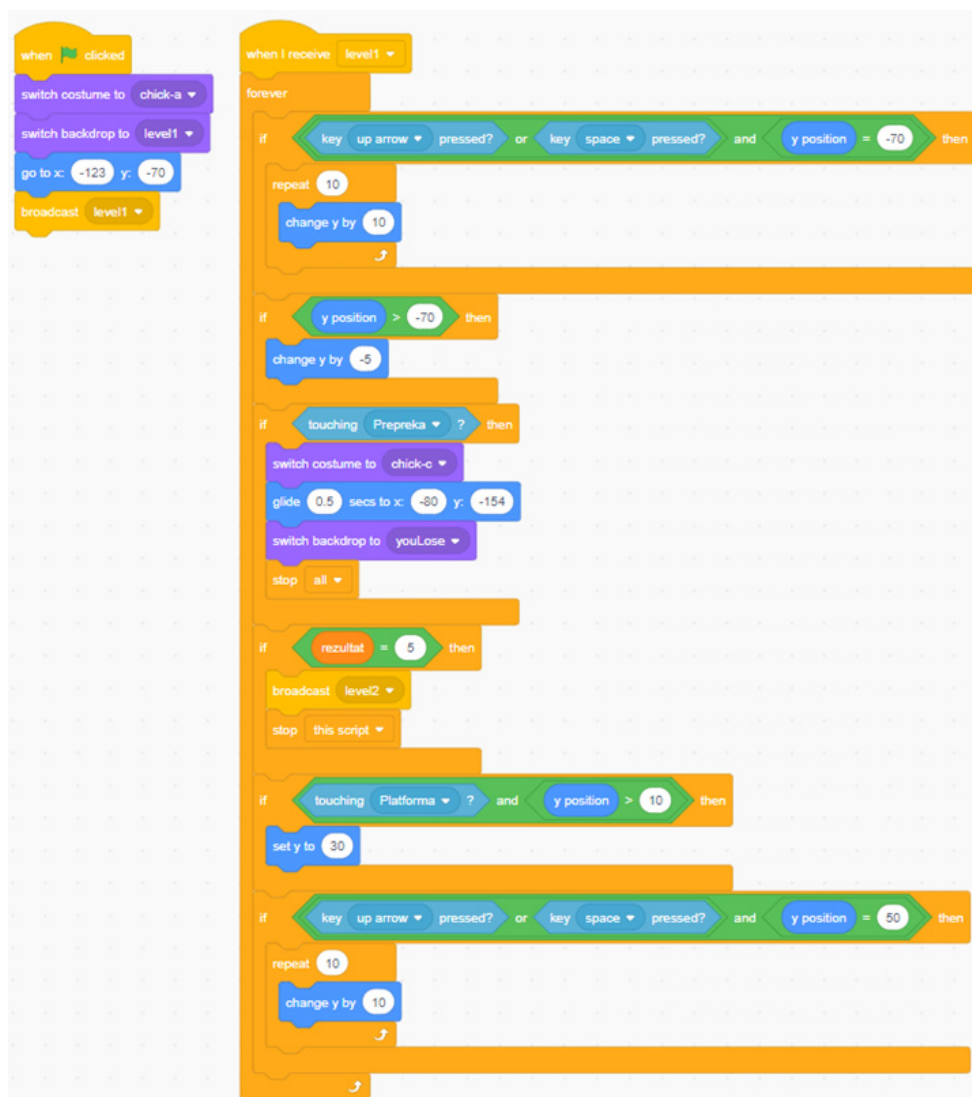
Zaustaviti igru u momentu kada pile dodirne loptu. Potrebno je tada prikazati pozadinu "IZGUBILI STE".

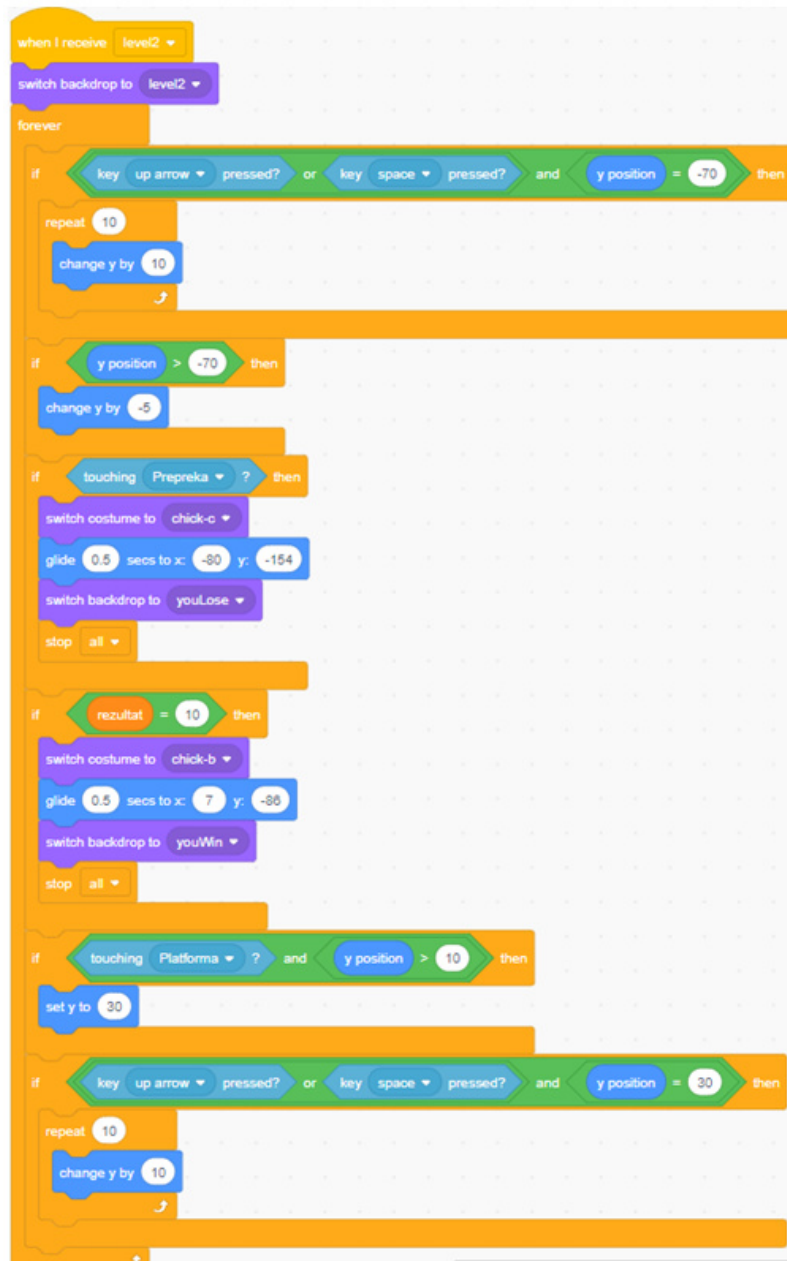
Prikazati pozadinu "POBIJEDILI STE" i zaustaviti igru nakon što osvojimo 10 poena.

Omogućiti dobijanje nagradnih poena upotrebom novog lika koji se kreće iznad visine pileta. Potrebno je skočiti i dodirnuti navedeni lik te na taj način osvojiti dodatni poen. U momentu dodira, lik nestaje sa ploče a poeni se povećavaju. Inače, lik treba da prođe pored pileta i da nestane (kao i lopta).

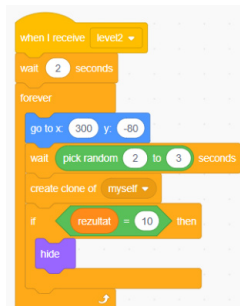
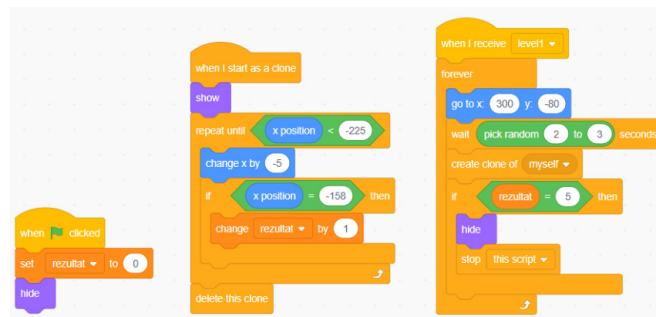


**Rješenje:** Kod za pile:

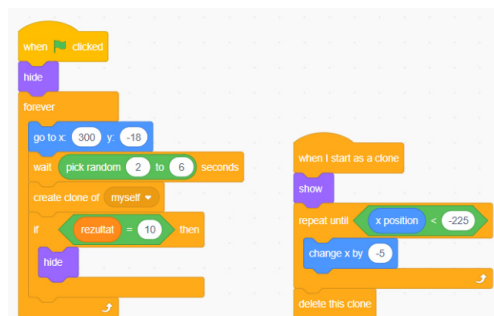




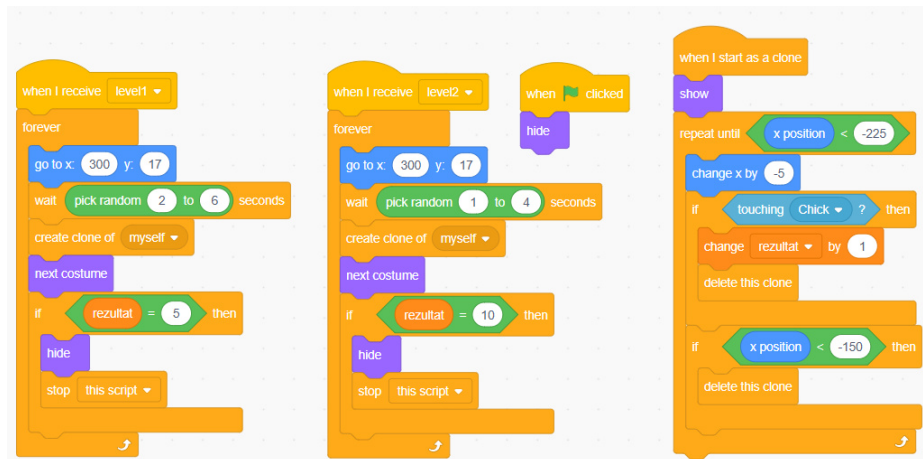
Kod za prepreku:



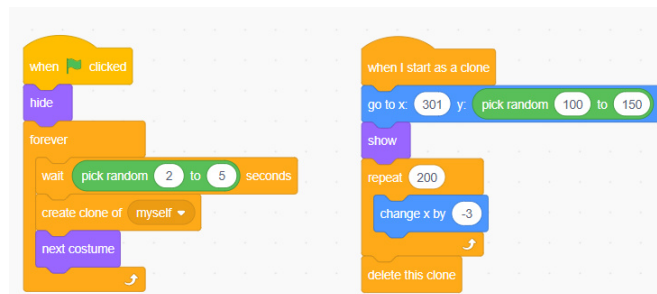
Kod za platformu:



Kod za bonus:



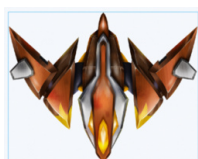
Kod za oblake:





## 12 Pametni anti algoritmi

Neprijateljskim letjelicama (8 njih) oblika



smo napravili odgovarajuća oružja (po 3 svakoj navedenoj letjelici), tj. oružja u formi energetskih krugova



Sjetimo se da smo pomenute letjelice zvali: Gobo13, Gobo14, Gobo16-19, Gobo21, Gobo22, dok smo odgovarajuća oružja zvali: Lightning13A, Lightning13B, Lightning13C (za Gobo13), ..., Lightning22A, Lightning22B, Lightning22C (za Gobo22), redom.

Pitanje aktivnosti navedenih oružja smo regulisali upotrebom prekidača (24 prekidača ukupno). Prekidačima, tj. promjenjivim smo dali imena: L23, L24, ..., L46 (ovi prekidači odgovaraju redom upravo imenovanim oružjima). Inicijalno smo na kodu za podloge navedenim prekidačima dodijelili vrijednosti 0. Ta vrijednost 0 nam je značila da je odgovarajuće oružje aktivno. Konkretnije, činjenica da je L23 jednako 0, nam je značila da je oružje Lightning13A aktivno. U slučaju dodira naše Letjelice M14 sa oružjem Lightning13A (kod za M14), vrijednost promjenjive L23 smo postavljali da je jednaka 1. Ta informacija se onda koristila u kodu za oružje Lightning13A, kao signal da oružje treba da bude sklonjeno sa ploče (tada se L23 ponovo postavljalo da bude jednako 0 da bi oružje bilo spremno za novu upotrebu). Ovdje smo se podsjetili algoritma rada navedenih oružja (energetskih krugova), gdje smo posebno obratili pažnju na to kada se navedena oružja sklanjaju sa ploče. Dakle, pri dodiru naše letjelice M14 sa nekim od navedenih oružja, to oružje se upotrebom odgovarajućeg opisanog prekidača (čija vrijednost je tada jednaka 1), sklanja sa ploče. Postavlja se pitanje: da li je

to jedina situacija kada pomenuto oružje treba da bude sklonjeno sa ploče ? Naime, mi možemo napraviti protu - oružje, koje će biti u stanju da eliminiše ovdje opisana oružja (energetske krugove). U slučaju dodira takvog novog protu - oružja sa energetskim krugovima, energetski krugovi takođe treba da budu sklonjeni sa ploče. Poenta je da koncept rada prekidača L23 - L46 možemo koristiti i u ovoj dodatnoj situaciji. Dakle, rad sa prekidačima L23 - L46 će biti dvostran: prvo, u slučaju dodira letjelice M14 sa nekim od energetskih krugova (ovaj dio smo već uradili), i drugo, pri dodiru odgovarajućeg protu - oružja sa energetskim krugovima. Zbog toga je već izgrađena infrastruktura prekidača jako bitna i u nastavku.

Posvetimo se sada izradi novog oružja (koje smo iznad zvali protu - oružje). Ovo oružje će biti novo oružje za letjelicu M14, čiji će cilj biti taj da eliminiše neprijateljske energetske krugove (zbog toga smo ga i zvali protu - oružje).

Za ovo oružje ćemo koristiti dva kostima



Jasno, drugi kostim je nastao rotacijom prvog (da bismo njihovom alternacijom dobili animaciju okretanja - analogno kao kod energetskih krugova).

Kao i do sada, idemo desno - dole na Choose a Sprite, pa na Upload Sprite, te upload - ujemmo Sprite po imenu Kugla1 (to je ustvari prvi od pomenuta dva kostima)

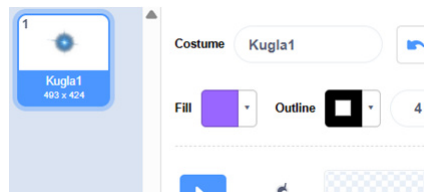


Kostim je po automatizmu u vektorskom obliku pa ne moramo dodatno brinuti o mogućnosti njegovog oštećenja. Ime Sprite - u kojeg smo upravo upload - ovali stavimo da bude Kugla (ovo je dakle naziv našeg novog oružja)





Ime kostima ostavimo da bude Kugla1



Ponovo, kao i da sada, idemo lijevo - dole na Choose a Costume, pa na Upload Costume, te upload - ujem drugi kostim: Kugla2



Ime kostima ostavimo kakvo jeste (Kugla2).

Veličinu Sprite - a Kugla (Size) stavimo da bude jednaka 20. Sačuvamo izmjene.

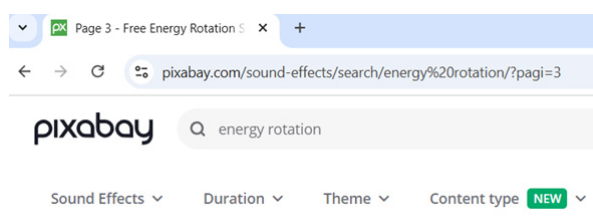
Za oružje Kugla ćemo koristiti dva zvuka. Jedan od njih će da bude zvuk koji će da prati kretanje Kugle po ploči (u potrazi za najbližim neprijateljskim energetske krugom). Zvuk koji ćemo koristiti u ove svrhe će biti zvuk



tank-turret-rotate-14879 (1).wav



Naime, ovaj zvuk je ustvari zvuk okretanja kupole tenka, ali nam vrlo adekvatno dočarava kretanje koje mi želimo da napravimo. Za odabir nekog drugog zvuka može se koristiti stranica koju smo i do sada upotrebljavali

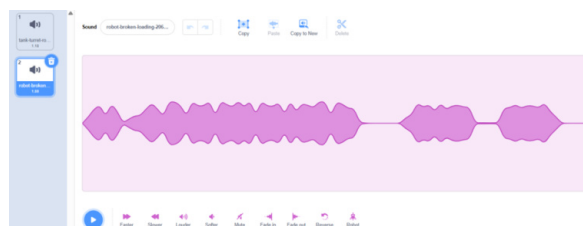


Oružje koje pravimo, tj. oružje Kugla će da bude klasično oružje tipa "seek and destroy". Napraviti ćemo ga tako da se aktivira pritiskom na neku tipku na tastaturi. Neka to bude slovo N. Ukoliko na ploči nema neprijateljskih energetske krugova, onda pritisak na tipku N ne treba da izazove bilo kakvu reakciju (ne treba da aktivira oružje Kugla). Naime, ako na ploči nema neprijateljskih krugova, onda nema potrebe da Kugla bude aktivirana. Ukoliko na ploči ima bar jedan energetski krug, onda pritisak na N treba da aktivira Kuglu. Kugla koja je aktivirana, treba da se pojavi kraj letjelice M14, te da počne da se kreće prema najbližem energetskom krugu (od letjelice M14) s ciljem njegove eliminacije. Dakle, težište oružja je da se traži najbliži energetski krug (letjelici M14), i da se taj krug uništi. Zašto najbliži? Zato što je onaj krug koji je najbliži letjelici M14 ujedno i najveća prijetnja za M14. Dakle, ako smo aktivirali Kuglu pritiskom na N (to je slučaj ako na ploči ima bar jedan energetski krug), onda Kugla treba da počne da se kreće direktno prema najbližem energetskom krugu (od letjelice M14). Pri tome računanje udaljenosti energetskih krugova (od letjelice M14) treba da se odvija u realnom vremenu, tj. treba konstantno da se update - uje, da bi se na osnovu svježih informacija eventualno korigovala putanja Kugle. Jasno je dakle da ćemo na nekoj lokaciji, tj. najvjerojatnije u nekoj onda listi, morati čuvati informacije o udaljenostima letjelice M14 od 24 neprijateljskih energetskih krugova. Te informacije, kako rekosmo, će morati da budu update - ovane u realnom vremenu, tj. konstantno u cilju stalnog kretanja Kugle prema najbližoj meti od M14. Jasno je da će onda u ovakvoj konstalaciji odnosa Kugla eventualno i da promijeni svoj pravac kretanja (ako zaključiti da je neki drugi energetski krug postao bliži letjelici M14 od onog prema kome se Kugla inicijalno kretala). Sjetimo se da smo energetske krugove programirali tako da idu prema dnu ekrana uz blagi naklon prema letjelici M14. U konačnici, svi ovi krugovi se sklanjaju sa ploče. Može se desiti tako situacija da je na ploči

bilo energetskih krugova, te da je shodno tome pritiskom na tipku N aktivirana Kugla, ali da su u međuvremenu energetski krugovi sklonjeni sa ploče. Postavlja se pitanje šta sa Kuglom raditi u takvim situacijama kada je ona već aktivirana, a na ploči nema niti jedan energetski krug. Jasno je da onda treba da kuglu sklonimo sa ploče na neki način. Način koji ćemo mi upotrijebiti je taj da ćemo napraviti animaciju zbunjenosti kugle, kao robota, koja ne zna na koju stranu da krene, te koja nakon određene neuspjele kalkulacije jednostavno nestaje. Događaj kalkulacije ćemo popratiti adekvatnom animacijom uz proizvođenje odgovarajućeg zvuka. Zvuk koji ćemo tada koristiti je sljedeći zvuk



robot-broken-loading-206293.wav

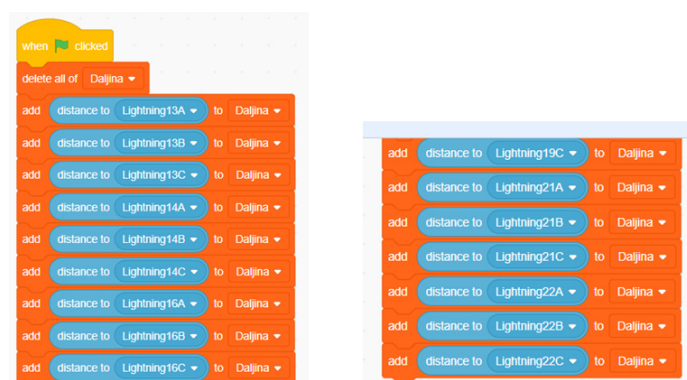


Riječ je o zvuku, kako i sam njegov naziv govori, pokvarenog robota.

Potrebno je dakle da upload - ujem prvi od ova dva opisana zvuka na Sprite Kugla. Uradimo to i sačuvajmo izmjene.

Prvo što u nastavku moramo uraditi jeste da definišemo listu udaljenosti letjelice M14 od 24 neprijateljska energetska kruga. Ovu listu ćemo zvati Daljina. Definišimo onda listu Daljina

U ovu listu ćemo smjestiti udaljenosti letjelice M14 do neprijateljskih energetskih krugova. Kako energetskih krugova ima 24 ukupno, to će onda lista Daljina imati 24 elementa. Kako smo već napomenuli, listu ćemo konstantno osvježavati. Takođe, pošto se u listi Daljina nalaze udaljenosti od letjelice M14 do neprijateljskih energetskih krugova, to ćemo kod za listu Daljina morati unijeti na lokaciju koda za M14. Idemo onda na M14, pa na Code, te unosimo sljedeći kod



Jasno je da iako ne prikazujemo, ovdje dodajemo udaljenosti do svih energetskih krugova u listu Daljina. Zatim, na istoj lokaciji dodajemo kod za osvježenje informacija o udaljenostima



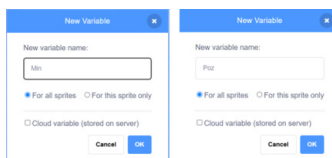
Dva navedena kod uklopimo u jedan, odnosno kod sa forever petljom dodamo na

kraj prvog koda. Imamo



Šta radi, objedinjeni kod ? Prvo, počistimo listu Daljina, i u nju dodamo udaljenosti (letjelice M14) do svih neprijateljskih energetskih krugova (24 njih). Zatim, vršimo konstantno osvježanje podataka. Naime, forever petljom, svake 0.1 sekunde osvježavamo podatke u listi Daljina, tako što stari podatak, npr. podatak na lokaciji 1 te liste mijenjamo svježim podatkom, tj. trenutnom (novom) udaljenošću letjelice M14 do energetskog kruga Lightning13A. Analogno za ostalih 23 podataka liste Daljina. Na ovaj način u listi Daljina uvijek imamo aktuelne udaljenosti M14 od energetskih krugova. Sačuvamo izmjene.

Sljedeće što nam je potrebno jeste da znamo koji od energetskih krugova je najbliži letjelici M14. Potrebno je dakle da nađemo minimum između navedenih 24 udaljenosti, te da upamtimo na kojoj poziciji se taj minimum dostiže. Kako to uraditi ? Razmišljamo klasično matematički. Proglasimo da je prvi energetski krug najbliži M14 (da je to Min udaljenosti) i pamtimo njegovu poziciju Poz (broj 1 je onda ta pozicija u ovom momentu). Zatim ispitujemo da li je drugi krug bliži M14 od prvog, tj. da li je udaljenost M14 od drugog kruga manja od Min. Ako to jeste slučaj, onda proglašavamo udaljenost drugog kruga za Min i pamtimo njegovu poziciju Poz (to je onda pozicija 2). Zatim posmatramo treći krug, itd. Ovo radimo za sve energetske krugove, tj. 24 njih, i to konstantno, tj. cijelo vrijeme, odnosno u realnom vremenu osvježavamo podatke da bi u svakom momentu znali koji od energetskih krugova je najbliži M14 (šta je dakle Poz). Vidimo da će nam za implementaciju ovog dijela koda trebati dvije promjenjive Min i Poz, pa definišimo te promjenjive



I dalje smo jasno na kodu letjelice M14, pa unosimo sljedeći kod (za stalno traženje minimuma udaljenosti Min letjelice M14 do energetske krugova, i pozicije na kojoj se taj minimum dostiže Poz)



Drugi kod prolazi listom Daljina od itema 2 do itema 24.

Drugi kod uklapamo u prvi kod, tako što ga zakačimo nakon naredbe: set Poz to

1.

Tako, uklopljeni kod ima oblik



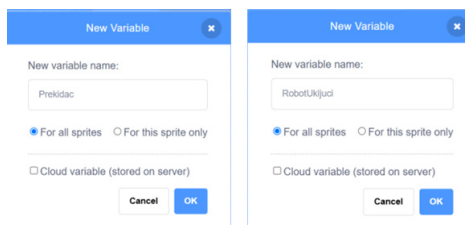
Kako smo već iznad opisali, upravo uneseni kod traži minimum Min udaljenosti letjelice M14 do 24 neprijateljska energetska kruga, te pamti poziciju Poz na kojoj se

taj minimum dostiže. Za naše potrebe važnija je informacija Poz. Naime, kada znamo Poz, mi znamo o kojem energetsom krugu je riječ (ako je npr. Poz jednako 3, onda je najbliži energetski krug letjelici M14 ustvari treći energetski krug, tj. energetski krug Lightning13C). Tako, kada znamo Poz, onda znamo odmah prema kojem energetsom krugu ćemo usmjeriti naše oružje Kugla. Sačuvamo izmjene.

Sada smo u situaciji da možemo da pristupimo kodiranju oružja Kugla.

Prvo, definisat ćemo određene promjenjive koje ćemo koristiti u nastavku.

Definišimo promjenjive: Prekidac i RobotUkljuci



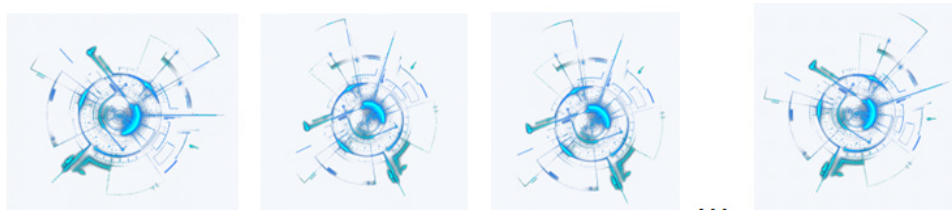
Ove promjenjive, kao i promjenjive Poz i Min, ali i lista Daljina, nema potrebe da budu aktivne na ekranu, pa kvakice pored njihovih imena možemo ukinuti.

Promjenjivu Prekidac ćemo koristiti da naznačimo kada je oružje Kugla slobodno za upotrebu, a kada je zauzeto. Tako, inicijalno ćemo staviti da je Prekidac jednako 1, što će značiti da imamo pravo na ispaljenje Kugle. Čim ispalimo Kuglu, vrijednost promjenjive Prekidac ćemo stavljati da bude jednaka 0 (da ne bi došlo do dodatnog ispaljenja Kugle), što će značiti da je Kugla zauzeta za upotrebu, tj. da nije dostupna za upotrebu. U slučaju da Kugla pogodi neki od energetskih krugova, vrijednost promjenjive Prekidac ćemo ponovo postavljati na vrijednost 1, što će značiti da je Kugla ponovo dostupna za upotrebu. Takođe, ako je Kugla ispaljena (tada je Prekidac jednako 0), a desi se da se svi energetski krugovi sklone sa ploče, tada smo rekli da nakon animacije sklanjamo Kuglu sa ekrana, i onda je jasno da ponovo moramo postaviti Prekidac da bude jednak 1, da bi Kugla ponovo mogla biti spremna za upotrebu.

Promjenjivu RobotUkljuci ćemo inicijalno postaviti da bude jednaka 0. Ukoliko se desi da smo Kuglu ispalili (znači da je prije njenog ispaljenja na ploči bilo energetskih krugova), a da su se nakon ispaljenja Kugle svi energetski krugovi sklonili sa ploče (dakle Kugla je ostala sama na ploči), onda vrijednost promjenjive RobotUkljuci postavljamo da bude jednaka 1. Ova vrijednost 1 daje informaciju da treba da se izvrši animacija zbunjenosti Kugle (pomenuli smo iznad), u smislu provođenja nekakve kalkulacije s njene strane, tj. neuspjele potrage za energetskim krugovima,

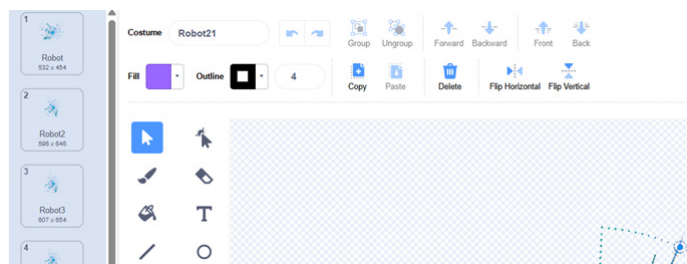
a onda i njeno sklanjanje sa ploče. Jasno, nakon sklanjanja Kugle sa ploče u ovakvoj situaciji, vrijednost promjenjive RobotUkljuci se ponovo postavlja da bude jednaka 0, da bi se ova promjenjiva mogla koristiti za dalju upotrebu.

Za upravo pomenutu animaciju zbunjenosti Kugle, koristit ćemo sljedeće kostime koje ćemo zvati: Robot, Robot2, Robot3, ..., Robot21



Nije teško za primijetiti da su ovi kostimi nastali od prvog, ili jednom grubom rotacijom, ili nizom uzastopnih ali blagih rotacija. Iz strukture koda vezanog za animaciju koju želimo postići ovim kostimima će postati jasno koja je svrha navedenih kostima.

Kao i do sada, idemo desno - dole na Choose a Sprite, pa na Upload Sprite, te upload - ujemmo Sprite Robot. Ime mu ostavimo da bude Robot, zatim Size, tj. veličinu mu postavimo da bude jednaka 60, i ime kostima ostavimo da bude Robot. Ovaj Sprite je već u vektorskom obliku pa eventualnim njegovim pomjeranjem po ploči ne dovodimo u pitanje kvalitet slike. Sada, idemo lijevo - dole, na Choose a Costume, pa na Upload Costume, i onda upload - ujemmo preostale kostime: Robot2, Robot3, ..., Robot21. Dakle, upload - ujemmo ih dok smo pozicionirani na Sprite Robot

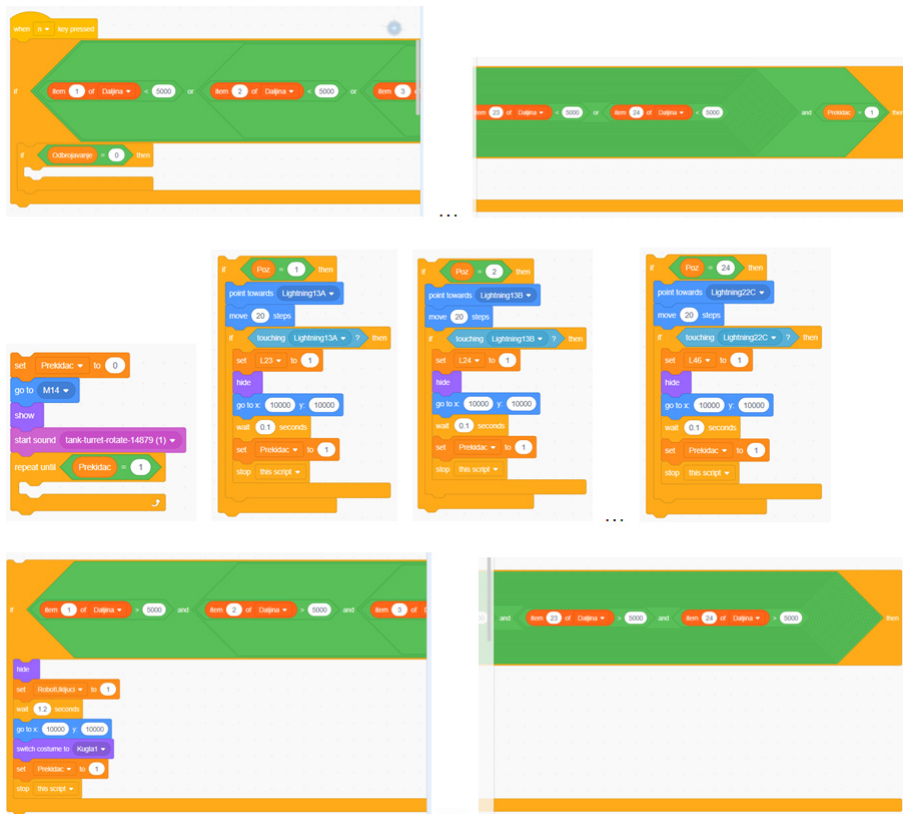
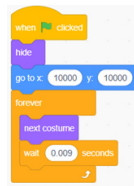


I dalje, dok smo pozicionirani na Sprite Robot, idemo na Sounds, te upload - ujemmo zvuk koji smo opisali iznad, tj. zvuk robot-broken-loading-206293.



Sačuvamo izmjene.

Pozicionirajmo se sada ponovo na Sprite Kugla. Idemo na Code. Možemo dakle pristupiti pravljjenju koda ovog Sprite - a. Unosimo sljedeći kod (kodove) koje onda kasnije eventualno uklapamo



Ovdje imamo četiri reda kodova. Tako ćemo ih i zvati: kodovi prvog reda, kodovi drugog reda, kodovi trećeg reda, i kodovi četvrtog reda.

Prvi kod je jasan. Inicijalno, Kugla treba da je sakrivena do momenta kada treba da postane vidljiva. Kao i ranije objekte aplikacije koju pravimo, tako i Kuglu držimo na lokaciji rezervisanoj za nju, tj. na lokaciji (10000,10000). Bez obzira da li je Kugla vidljiva ili ne, zauvijek rotiramo dva njena kostima u razmaku od 0.009 sekundi što stvara utisak animacije rotacije.

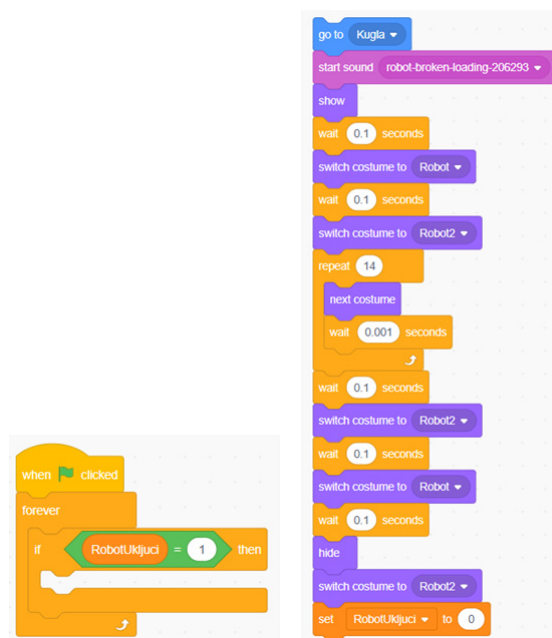
Kodovi drugog reda, ili, preciznije kod drugog reda je dat na dvije slike. Naime, uslov koji je dat ovim kodom je predug, pa smo kroz dvije navedene slike naznačili šta je njegov početak a šta kraj. Rekli smo da Kuglu aktiviramo pritiskom na tipku N. Bez obzira na pritisak na tipku N, da bi uopšte imalo smisla aktivirati kuglu, mora biti zadovoljeno da je na ploči prisutan bar jedan neprijateljski energetska krug. Dakle, bar jedan od tih krugova ne smije biti daleko od letjelice M14. Drugim riječima mora item 1 liste Daljina biti manji od 5000 ili item 2 liste Daljina biti manji od 5000 ili ... ili item 24 liste Daljina biti manji od 5000. Takođe, rekli smo da će aktivacija Kugle biti moguća ako Kuglu uopšte imamo na raspolaganju, tj. ako Kugla nije već zauzeta. Da je Kugla slobodna za upotrebu označili smo činjenicom da je Prekidac jednako 1. Ako je Prekidac jednako 0, Kugla je zauzeta. Tako, ako je uz dugi gornji uslov zadovoljen i uslov da je Prekidac jednako 1, onda ima smisla aktivacija Kugle. Znači aktivacija Kugle ima smisla ako na ploči imamo bar jedan energetska krug i ako je Kugla uopšte dostupna. Jasno, mora biti zadovoljen i već standardni uslov koji cijelo vrijeme zadovoljavamo, a to je uslov da je Odbrojanje jednako 0, što znači da se ne nalazimo u situaciji da ili nismo ušli u igru ili da su svi neprijatelji eliminisani sa ploče.

Ako smo zadovoljili uslove koda drugog reda, tj. ako ima smisla upotrijebiti Kuglu, onda možemo preći na kodove trećeg reda. Znači, upotrebljavamo Kuglu, pa da je ne bi u isti moment mogli upotrijebiti ponovo, postavljamo Prekidac da bude jednako 0. Jasno, šaljemo je do letjelice M14, prikažemo je, startujemo zvuk kupole tenka, i otpočinjemo potragu za neprijateljskim energetskim krugovima, koja će jasno da traje sve dok ne pogodimo neki od njih ili dok niti jedan od njih ne bude prisutan na ploči. Svaki od ova dva scenarija se završava time što se Kugla sklanja sa ploče i priprema za novu upotrebu. Priprema za novu upotrebu znači da se Prekidac postavlja ponovo na vrijednost 1. Zbog toga petlja repeat traje sve dok uslov da je Prekidac jednako 1 ne bude zadovoljen. Šta je to što vrtimo u upravo opisanoj petlji ? To su preostali kodovi trećeg reda u kojima figurišu uslovi:  $Poz = 1$ ,  $Poz = 2$ , ...,  $Poz =$

24. Ove kodove nadovezujemo jedan na drugi: kod sa  $Poz = 2$  nadovezujemo na kod prije njega, kod sa  $Poz = 3$  na kod prije njega, itd., kod sa  $Poz = 24$  nadovezujemo na kod prije njega, tj. na kod u kojem figuriše  $Poz = 23$ . Fokusirajmo se sada na pojašnjenje rada koda u kojem se nalazi uslov  $Poz = 1$  (analogno bi razmotrili preostale kodove trećeg reda). Dakle, ako je  $Poz = 1$ , to znači da je prvi energetski krug, tj. *Lightning13A*, najbliži letjelici *M14*. Drugim riječima, to znači da *Kuglu* treba da usmjerimo prema *Lightning13A*. Pomjerimo je za 20 koraka (i svaki eventualni sljedeći put je pomjeramo za 20 koraka). Ako prilikom pomjeranja dodirnemo *Lightning13A* (prvi energetski krug), onda *Lightning13A* treba da sklonimo sa ploče (to kako odranije znamo radimo stavljanjem odgovarajućeg prekidača *L23* da bude jednak 1). Isto tako treba da skrijemo *Kuglu*, i pošaljemo je na njenu rezervisanu lokaciju (10000,10000). Pošto je *Kugla* sklonjena, to znači da treba da je spremna za novu upotrebu, pa *Prekidac* postavljamo da bude jednak 1. Konačno, zaustavimo skriptu. Sjetimo se da smo ranije isticali kako korištenje naredbe za zaustavljanje skripte nije produktivno u smislu nemogućnosti ponovne aktivacije te iste skripte. Ipak, važno je napomenuti da je to zaista tako ako inicijalna komanda počinje naredbom: *When* (zelena zastavica) *clicked*. Dakle, ako takvu strukturu zaustavimo kao skriptu onda je nećemo moći ponovo pokrenuti. Ipak, strukture koje počinju naredbom: *When n key pressed*, i sličnim naredbama (naredbe koje reaguju na dodir neke tipke na tastaturi), nemaju problem sa zaustavljanjem. Tako, i ako ih zaustavimo kao skripte, ponovnim pritiskom na tipku tastature koja ih aktivira, aktiviramo tu strukturu.

Kodovi četvrtog reda, tj. kod četvrtog reda je dat na dvije slike. Riječ je o jednom kodu koji je zbog dužine uslova morao biti postavljen na dvije slike. Ovaj kod se bavi situacijom u kojoj je *Kugla* lansirana na ploču, a nakon toga se našla u okolnostima da na ploči nema niti jednog energetskog kruga. Činjenicu da na ploči nema niti jedan energetski krug opisujemo uslovom da su svi energetski krugovi daleko od letjelice *M14*, tj. uslovom da su njihove udaljenosti od *M14* veće od 5000. Znači, ako su sve ove udaljenosti (njih 24 ukupno) veće od 5000, šta radimo ? Prvo, sakrijemo *Kuglu*. Zatim, želimo da animiramo zbunjenost *Kugle* (odnosno objekta koji je zamjenjuje), u smislu da on kao traži stranu na koju da krene jer oko njega nema niti jednog energetskog kruga. Rekli smo da će navedena animacija zbunjenosti *Kugle* (odnosno objekta koji je mijenja) biti aktivirana pomoću promjenjive *RobotUkljuci* koja će inicijalno biti postavljena na vrijednost 0, a koja će, u slučaju da se *Kugla* nađe u upravo opisanoj situaciji, biti postavljena na vrijednost 1. Ova vrijednost 1 će ustvari da šalje informaciju *Sprite* - u *Robot*, da pokrene odgovarajuću opisanu animaciju (koju gradimo u nastavku). Dakle, pošto se *Kugla* našla u opisanoj situaciji, to vrijednost promjenjive *RobotUkljuci* postavljamo da bude jednaka 1. U tom momentu će *Sprite* - om

Konačno, idemo na Sprite Robot, pa na Code, i unosimo sljedeće kodove



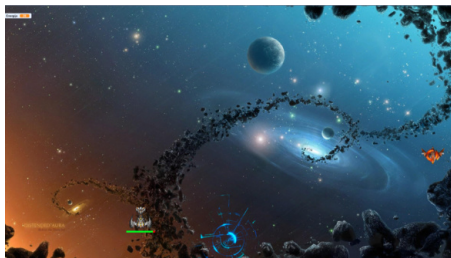
U prvom kodu sakrivamo Sprite Robot jer ne želimo da bude vidljiv do momenta kada to zaista želimo da se desi i kada nam to zaista treba. Postavimo se na kostim Robot2 (uočiti da je ovaj kostim znatno zakrivljen ulijevo u odnosu na kostim Robot). Želimo da navedene kostime na određeni način rotiramo (na moment drsko na moment blago) da stvorimo utisak portage za nečim, tj. za nekim od energetske krugova. Kako smo i rekli, inicijalno, vrijednost promjenjive RobotUkljuci postavljamo da bude jednaka 0.

U drugom kodu forever petljom lovimo moment kada se desi da je RobotUkljuci jednako 1. Već smo istakli da činjenica da je RobotUkljuci jednako 1 označava informaciju Sprite - u Robot da otpočne animaciju zbunjenosti Kugle (odnosno objekta koji je mijenja - to je sam objekat Robot).

Šta to onda želimo da se animira ako je RobotUkljuci jednako 1 ? Želimo da se animira ono što je dato trećim kodom. Pošaljemo Sprite Robot do Kugle (u kostimu Robot2 kako smo već naredili). Otpočnemo zvuk pokvarenog robota. Prikažemo Sprite Robot. Sačekamo 0.1 sekundu, i onda drsko promijenimo kostim sa Robot2 na

kostim Robot koji je zakrivljen prilično udesno. Zatim ponovo sačekamo 0.1 sekundu i isto tako drsko se vratimo sa kostima Robot na kostim od kojeg smo počeli, tj. na kostim Robot2. Sada želimo da od kostima Robot2 blago se zarotiramo udesno (pomoću 14 narednih kostima). Koristimo repeat petlju (sa 14 iteracija) u kojoj mijenjamo 14 narednih kostima uz vrlo kratko vrijeme čekanja od 0.001 sekundi. Iskoristili smo dakle kostime zaključno sa kostimom Robot16 (kostime Robot17 - Robot21 ovdje nećemo ni upotrijebiti zbog tempiranosti vremena, ali ih možemo sačuvati kao objekte u slučaju eventualne želje njihove primjene na neki sasvim drugi animirani način). Nakon repeat petlje i čekanja od 0.1 sekunde, ponovo se drsko vratimo na kostim Robot2. Sačekamo 0.1 sekundu i onda se opet drsko prebacimo na kostim Robot. Do ovog momenta provedeni postupak bi trebao da stvori željeni vid animacije (probati animaciju i uvjeriti se u to na licu mjesta). Preostaje da sačekamo 0.1 sekundu, sakrijemo Sprite Robot, prebacimo se na kostim Robot2 (da Sprite Robot bude spreman za narednu upotrebu). Konačno, vrijednost promjenjive RobotUkljuci postavimo da bude jednaka 0 (da bude spremna za ponovnu upotrebu). Treći kod uklopimo u drugi kod.

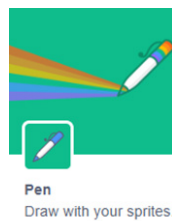
Sačuvamo izmjene i probamo igru



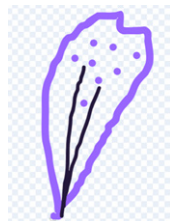
## 12.1 Riješeni zadaci

◇ **Zadatak 12.1.1** U ovom zadatku potrebno je koristiti ekstenziju Pen za crtanje cvijeta.

Aktivirati ekstenziju Pen putem opcije Add Extension



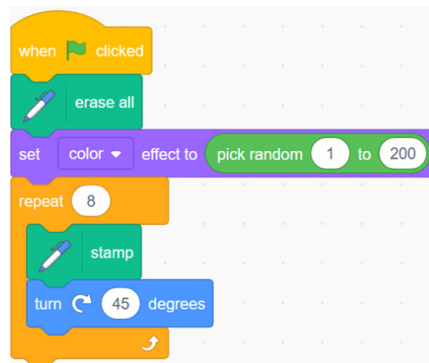
Kreirati Sprite "Latica" pomoću opcije Paint koji predstavlja laticu cvijeta (nacrtati proizvoljno). Voditi računa da se dno latice nalazi na sredini ekrana



Koristeći blokove "erase all" i "stamp" kreirati cvijet koji mijenja boju pri svakom pokretanju programa (opciono, omogućiti korisniku da unese broj latica koje će biti iscrtane)



**Rješenje:**



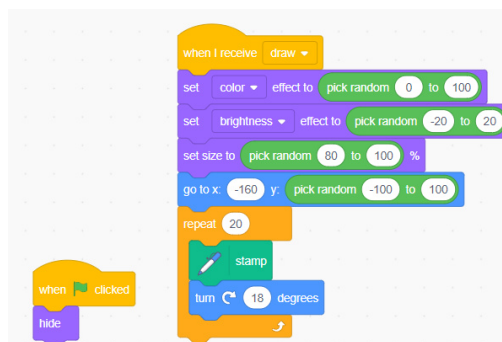
◇ **Zadatak 12.1.2** Kreirati četiri Sprite-a koji će predstavljati četiri različite latice cvijeta. Potrebno je napraviti program koji će prilikom svakog klika na pozornicu (stage) crtati četiri različita cvijeta. Voditi računa o sljedećem:

podesiti efekte boje, osvjjetljenja i veličine na nasumične vrijednosti,

Sprite-ove rasporediti ravnomjerno po ekranu tako da  $x$  bude fiksno, a  $y$  birati nasumično,

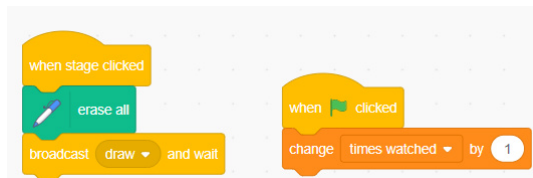
dodati varijablu "times watched" pomoću koje ćemo pratiti broj klikova na stage.

**Rješenje:** Kod za Sprite (isti za sve cvjetove):



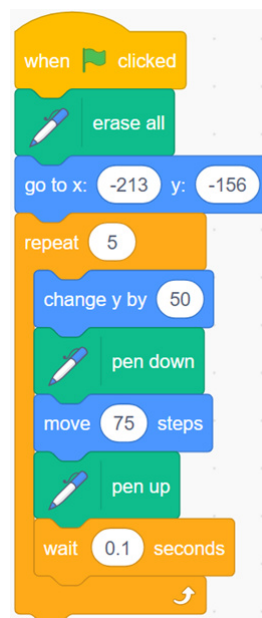


Kod za stage:

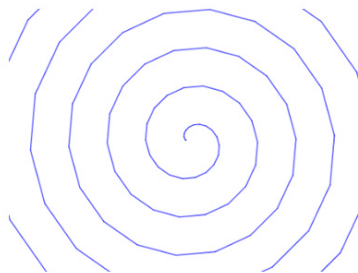


◇ **Zadatak 12.1.3** Korištenjem petlje i blokova "pen down" i "pen up" iscrtati 5 stepenica, od kojih se svaka sastoji od jedne horizontalne linije dužine 75 koraka. Nakon crtanja svake stepenice Sprite treba da se pomjera vertikalno za 50 koraka. U zadatku koristiti Sprite "Pencil" iz biblioteke, smanjiti mu veličinu, te postaviti vrh olovke na sredinu ekrana za uređivanje.

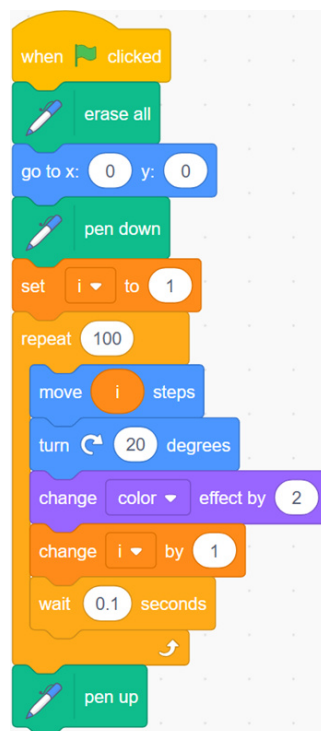
**Rješenje:**



◇ **Zadatak 12.1.4** Koristeći ekstenziju "Pen" nacrtati spiralu kao na slici



**Rješenje:**



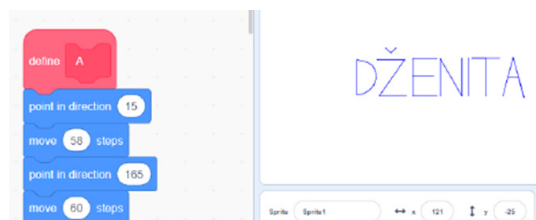
## 12.2 Zadaci za vježbu

♣ **Zadatak 12.2.1** Napraviti program koji iscrtava kućicu sa slike



Program treba da sadrži procedure za iscrtavanje: vrata, prozora, krova, itd. Potrebno je iskombinovati kreirane procedure da se omogući iscrtavanje navedene kućice.

♣ **Zadatak 12.2.2** Kreirati proceduru za ispis pojedinačnog slova vlastitog imena. Nakon pokretanja, program treba da ispiše naše ime na ekranu upotrebom kreiranih procedura (blokova). Primjer ispisa imena je dat na slici



♣ **Zadatak 12.2.3** Korištenjem ekstenzije "Pen" i vlastite definisane funkcije (My Blocks), nacrtati apstraktni uzorak sastavljen od više krugova različitih veličina i boja koji se preklapaju i rotiraju oko centra ekrana.



## 13 Ažuriranje sadržaja

Kako smo već vidjeli, naše oružje Kugla



uspješno eliminiše sa ploče neprijateljska oružja Lightning13A - Lightning22C (energetske krugove)



Ipak, taj proces odvija se tiho. Naime, prilikom dodira Kugle sa energetskim krugovima, oba oružja samo nestanu sa ploče. Ne desi se nikakav dodatni događaj koji bi potvrdio da su dva navedena oružja poništila jedno drugo



Sada ćemo doraditi pomenuti dodir, u smislu stvaranja popratnog efekta, neke vrste eksplozije, koja će dodatno stvoriti utisak bliskog susreta dva objekta.

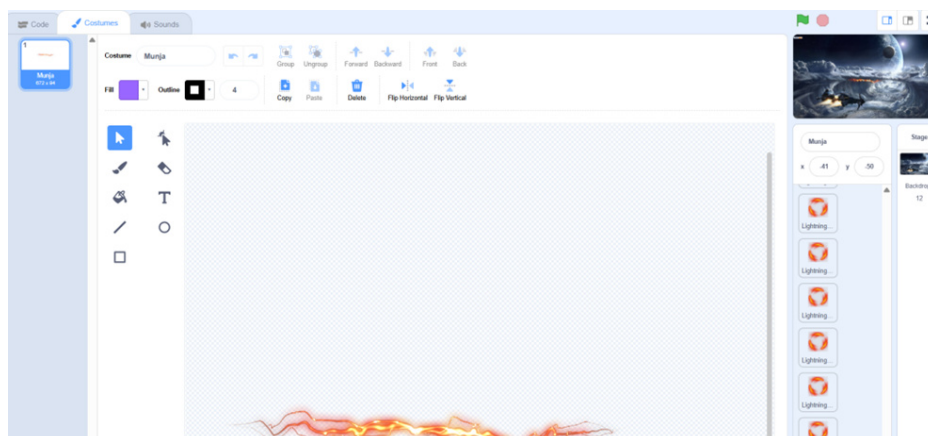
Koristit ćemo Sprite po imenu Munja



Osim toga, koristit ćemo zvuk kojeg ćemo zvati Ok



Klasično dakle, idemo desno - dole na Choose a Sprite, pa na Upload Sprite, te upload - ujemmo Sprite Munja (sa lokacije na koju smo ga smjestili: obično je to folder Downloads). Ovaj Sprite je već u vektorskom obliku pa ne moramo voditi računa o eventualnim neželjenim modifikacijama kvaliteta slike prilikom njegovog kretanja nakon pokretanja programa. Ime kostima ćemo ostaviti da bude Munja. Naziv Sprite - a smo već rekli da će biti Munja, a njegovu veličinu (Size) stavimo da bude jednaka 80



Očekivano sada, dok smo na Sprite - u Munja, idemo na Sounds, zatim dole - lijevo na Choose a Sound, pa na Upload Sound, te upload - ujemmo pomenuti zvuk Ok.

Sada smo u poziciji da pristupimo kodiranju željenog efekta. Naime, želja nam je da se prilikom dodira Kugle sa bilo kojim od neprijateljskih energetske krugova pojavi eksplozija Munja, da se ona na momentat zadrži (bljesne), da se pri tome proizvede zvuk Ok, te da oba objekta, i Kugla i posmatrani energetski krug nestanu sa ploče. Isto tako, želimo da se upravo opisana eksplozija dogodi i prilikom eventualnog dodira naše letjelice M14 sa nekim od energetske krugova. Važno je da se sada sjetimo upotrebe prekidača L23 - L46. Ove prekidače smo koristili za energetske krugove Lightning13A - Lightning22C, redom. Prekidače smo inicijalno postavili da budu jednaki 0. Onda kada se dogodi da neki energetski krug dodirne našu letjelicu M14, njegov prekidač smo postavljali na vrijednost 1. Isto tako, u momentima kada se dogodi da Kugla dodirne neki energetski krug, odgovarajući prekidač smo postavljali da ima vrijednost 1. Ova vrijednost 1 o kojoj govorimo je korištena kao informacija kodu odgovarajućeg energetske kruga da samog sebe skloni sa ploče. U ovom smislu ćemo pomenute prekidače koristiti i u nastavku. Naime, ako se desi da je vrijednost nekog od prekidača jednaka 1, onda će to značiti da je ili odgovarajući energetski krug dodirnuo našu letjelicu M14, ili da je taj energetski krug dodirnula Kugla. U oba slučaja treba da se pojavi eksplozija Munja kojom se bavimo ovdje. Nakon ove uvodne motivacije, a dok se i dalje nalazimo na Sprite - u Munja, idemo na Code, te unosimo sljedeći kod (kodove)



Kodove koji sadrže prekidače L23 - L46 nadovežemo u jedan: onaj sa L24 nadovežemo na onaj sa L23, itd., kod sa L46 nadovežemo na kod sa L45. Taj objedinjeni kod uklopimo u forever petlju drugog koda.

Prvi kod je jasan, Munja ne treba da bude vidljiva na ekranu u startu. Dakle, ona treba da bude vidljiva onda kada to mi želimo.

Drugi kod sadrži forever petlju koja treba kao i do sada da "lovi" pravi, tj. željeni događaj (momenat). Šta je to što ova petlja treba da "hvata" ? To je momenat kada neki od prekidača L23 - L46 postane jednak 1. To je dakle priča kodova koji dolaze. Pošto su kodovi koji dolaze u biti međusobno jednaki, dovoljno je da objasnimo rad koda koji sadrži prekidač L23.

Posmatrajmo kod koji sadrži prekidač L23. Ako se desi da je prekidač L23 jednak 1, onda to znači ili da je energetski krug Lightning13A dodirnuo našu letjelicu M14, ili da je naše oružje Kugla dodirnuo energetski krug Lightning13A. Koji god scenario od opisana dva se desi, mi treba da animiramo u uvodnom dijelu opisanu eksploziju. Dakle, u tom momentu, treba da pošaljemo Sprite Munja na odgovarajuću lokaciju, a to je lokacija energetskog kruga Lightning13A. Munja treba da postane vidljiva, treba da se čuje zvuk Ok. Pošto Munja treba da se na datoj lokaciji zadrži samo na moment (treba samo da bljesne), zadržimo je vidljivom samo neko kratko vrijeme, npr. 0.1 sekundu. Nakon toga je sakrijemo.

Sačuvamo izmjene i probamo igru.



Vratimo se sada na Sprite energije naše letjelice M14



Tokom izgradnje igre, dalo se vidjeti da je izgled ovog Sprite - a veizuelno prevelik



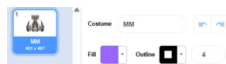
Zbog toga, smanjit ćemo veličinu (Size) ovog Sprite - a sa 100 na razumnih 50



Sačuvamo izmjene.

Ono čemu sada želimo da se posvetimo jeste ranjavanje naše letjelice M14. Up-ravo smo opisali važnost animacije dodira oružja Kugla sa neprijateljskim energetskim krugovima. Naime, sada je doživljaj igre u tom smislu znatno potpuniji. Znamo da prilikom dodira neprijateljskih oružja sa letjelicom M14 imamo napravljeno to da naša letjelica gubi energiju (zeleni dio energije se mijenja crvenim dijelom). Ipak, pri takvom jednom dodiru nemamo nikakvu dodatnu animaciju koja bi poboljšala doživljaj dodira, niti ugrađen neki adekvatan zvuk. Napomenimo i to da za sada nemamo razrađen niti dodir naše letjelice M14 sa nekom od neprijateljskih letjelica Gobo1 - Gobo50. U nastavku se bavimo ovim pitanjima.

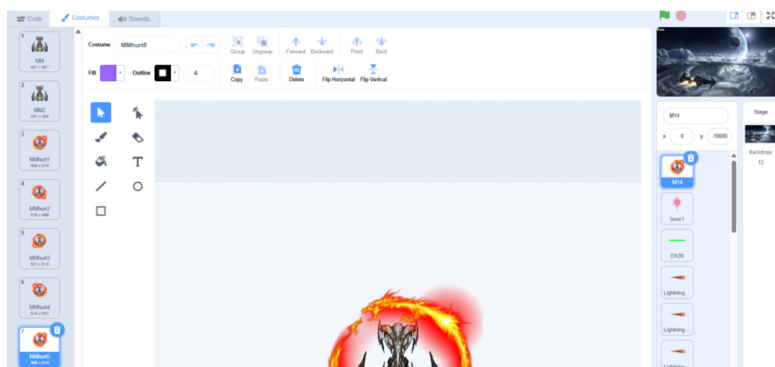
Kliknemo na Sprite naše letjelice M14. Za sada imamo jedan jedini kostim po imenu MM veličine 20



Dok smo na ovoj lokaciji idemo dole - lijevo na Choose a Costume, pa na Upload Costume, te upload - ujemmo sljedeće kostime (prvi od njih je isti kao postojeći kostim MM)



Imamo situaciju kao na slici



Pošto smo kostim MM upload - ovali, a već smo imali kostim pod tim imenom, to je upload - ovan kostim preimenovan u MM2. Nama nije potrebno da imamo dva ista kostima MM i MM2. Potreban nam je samo jedan od njih. Zbog toga ćemo prvi od njih (to je stari kostim MM) obrisati. Kostim koji ostaje, tj. MM2 ćemo sada preimenovati da se zove MM. Razlog zbog kojeg smo uradili upravo opisano je jednostavan. Naime, kostimi koje smo upravo upload - ovali su nastali od jednog početnog kostima letjelice, na kojeg smo onda dodavali narandžastu optraku sa likom zmaja. Svi ti kostimi u svojoj osnovi imaju letjelicu čiju veličinu nismo mijenjali. Da

smo ostavili stari kostim MM a dodali samo kostime MMhurt1 - Mmhurt5, moglo se desiti da letjelica unutar kostima MM, te letjelica koja se pojavljuje unutar kostima MMhurt1 - Mmhurt5, ne budu iste veličine. Tako, na opisani način smo osigurali da letjelica unutar svakog od kostima MM, MMhurt1 - MMhurt5 bude iste veličine.

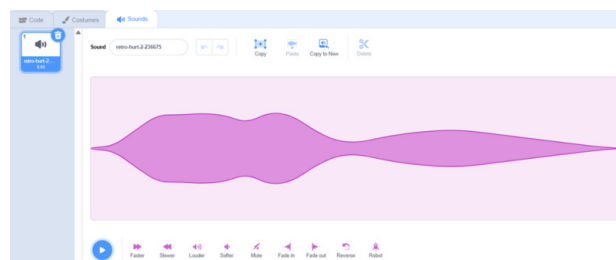
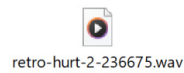
Svi upload - ovi kostimi su u vektorskom obliku pa nemamo problem sa eventualnim neželjenim izmjenama kvaliteta slike prilikom njenog kretanja. Informativno, jasno je da su kostimi MMhurt1 - MMhurt5 nastali od kostima MM, tako što smo na njega dodavali opraku sa likom zmaja



Naime, kostim MM je bilo dovoljno kopirati, pa na dobijeni kostim zaljepiti opraku zmaja. Zatim bi kopirali novo - nastali kostim, pa na njemu izvršili rotaciju oprake. Takav kostim bi ponovo kopirali pa na njemu izvršili dodatnu rotaciju oprake. Postupak bi nastavili dok ne dobijemo sve željene kostime. Naravno, pri ovakvom postupku, da ne bi došlo do odsjecanja dijela slike, potrebno je svaki prethodni kostim (onaj koji je već završen), pretvoriti u vektorski oblik, tj. čitav kostim pretvoriti prvo u Bitmap format a onda opet u vektorski format, da bi došlo do objedinjavanja lika letjelice sa likom oprake zmaja. Zainteresovani čitalac može provesti upravo opisani postupak samostalno ukoliko na taj način želi da produbi svoje znanje, tj. rutinu (ili u smislu vježbe bar), a eventualno ne želi da koristi već podešene opisane Sprite - ove.

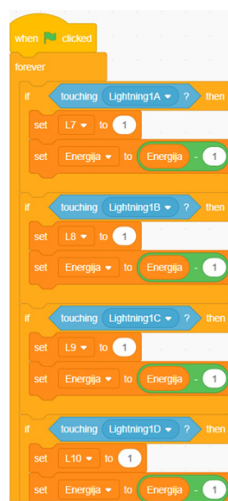
Veličinu Sprite - a M14 (naše letjelice) sada postavimo na 30 i sačuvamo izmjene.

Iz oblika kostima MMhurt1 - MMhurt5 možemo vidjeti šta se sa ovim kostimima ustvari želi postići. Kostim MMhurt1 i kostim MMhurt5 su ustvari jedan te isti kostim. Kostimi koji se nalaze između ova dva kostima (uključujući i njih) čine jednu potpunu rotaciju (od 360 stepeni) oprake lika zmaja. Naime, želimo da se prilikom dodira nekog od neprijateljskih oružja sa letjelicom M14, ili prilikom dodira letjelice M14 sa nekim od neprijateljskih letjelica Gobo1 - Gobo50, odigra rotacija pomenute oprake. Preciznije, želimo da se pri takvom jednom događaju lik zmaja zavrti oko naše letjelice, animirajući na taj način ranjavanje. Pri tome jasno treba da se čuje odgovarajući zvuk. Zvuk koji ćemo koristiti je sljedeći zvuk



Tako, dok smo i dalje pozicionirani na letjelicu M14, idemo na Sounds, te upload - ujem upravo pomenuti zvuk retro-hurt2. Sačuvamo izmjene.

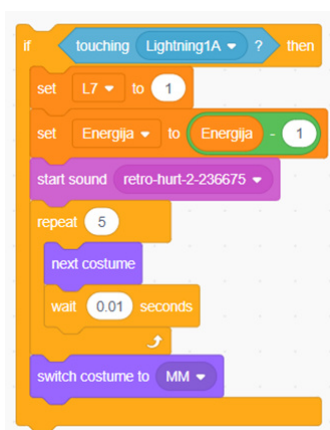
Sada smo u mogućnosti da pristupimo kodiranju, tj. u situaciji smo da možemo da doradimo postojeće kodove i eventualno dodamo nove. Dok smo još uvijek pozicionirani na Sprite M14, idemo na Code. Uočimo kod koji se odnosi na dodir letjelice M14 sa neprijateljskim oružjima



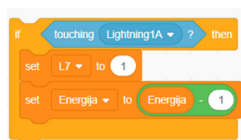
Svi elementi ovog koda koji se nalaze u sklopu naredbe forever počinju naredbom If. Uzmimo jedan takav element, npr. element



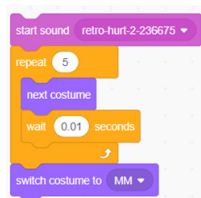
Ovaj kod sada modifikujemo u kod



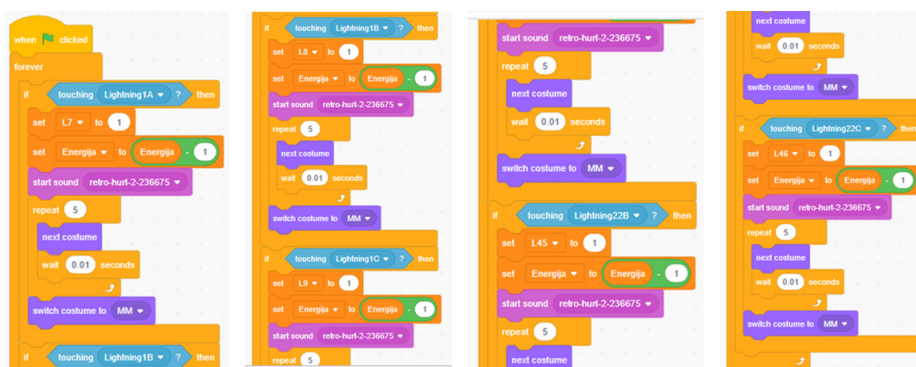
Naime, ako letjelica M14 dodirne neprijateljsko oružje Lightning1A, treba kao i do sada da ga skloni sa ploče postavljanjem prekidača L7 na vrijednost 1, te da umanji vrijednost energije letjelice M14 za 1 (ovaj dio koda ostaje isti, tj. nepromijenjen). Ono što dodatno dodajemo je animacija ranjavanja M14. Prvo, puštamo zvuk ranjavanja retro-hurt2. Zatim, jednostavno, koristimo petlju od 5 iteracija, u kojoj mijenjamo kostime (MMhurt1 - MMhurt5) uz vrlo kratko vrijeme čekanja od 0.01 sekundi. Ovo nam govori da će rotacija lika zmaja oko naše letjelice da se odigra relativno brzo (nije dobro da se odigra ni presporo niti prebrzo). Jasno, nakon prolaska kroz navedenu petlju, tj. prolaska kroz dodatne kostime MMhurt1 - MMhurt 5, potrebno je da stvari vratimo na početak, tj. da postavimo kostim naše letjelice M14 na verziju MM. Vidimo da smo faktički unutar postojećeg koda



umetnuli kod

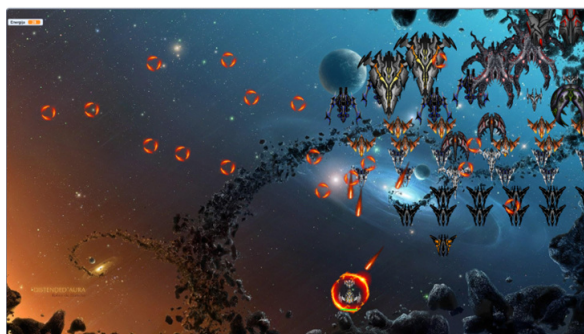


i to iza naredbe: set Energija. Ovo što smo upravo uradili treba da uradimo za sve If naredbe koje se nalaze unutar navedene forever petlje. Dakle, kopirat ćemo posljednji kod i zalijepiti ga unutar svih If naredbi koje figurišu unutar forever petlje koda kojeg modifikujemo, i to tako da se dodatni kod postavi iza odgovarajućih naredbi tipa: set Energija. Uradimo to. Početak i kraj ovako modifikovanog koda (za sada) imaju oblik, redom

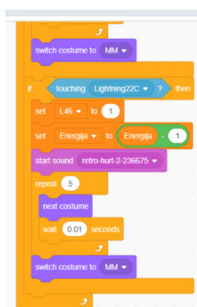


Sačuvamo izmjene i probamo igru. Vidimo da se željeni vizuelno - zvučni efekat

zaista postiže prilikom dodira letjelice M14 sa nekim od neprijateljskih oružja



Ono što ostaje da uradimo jeste dodir naše letjelice M14 sa neprijateljskim letjelicama Gobo1 - Gobo50. Uradit ćemo kod za dodir M14 i Gobo1. Ostali kodovi će da budu analogni. Dakle, dodir M14 i Gobo1 treba da polučiti isti rezultat kao i dodir M14 sa nekim od neprijateljskih oružja. Drugim riječima, potrebno je da se prilikom jednog takvog dodira animira ranjavanje M14, te da se umani energija M14 za 1. Naime, nema potrebe da pravimo diferencijaciju dodira M14 sa različitim Gobo - ima. Možemo smatrati da dodir M14 sa bilo kojim od Gobo - a, letjelici M14 odzuiima 1 podiok energije. Čisto da primijetimo da ovo nije slučaj prilikom dodira M14 sa neprijateljskim oružjima. Naime, neka od njih nam oduzimju 1, neka 2, a neka 3 podioka energije. Dodatno, kodirat ćemo tako da dodir letjelice M14 sa nekim od Gobo - a ranjava M14 ali ne i dotičnog Gobo - a. I dalje smo pozicionirani na kod za M14, i to najbolje da smo pozicininirani na dno prethodno modifikovanog koda, tj. da se nalazimo iza koda



Neovisno od ovog koda, tj. van njega, odnosno fizički ispod njega unosimo kod (za dodir M14 i Gobo1)



Tako, ako letjelica M14 dodirne Gobo1, umanjimo Energiju za 1, započnemo zvuk retro-hurt2, izvedemo animaciju ranjavanja petljom od 5 iteracija, i postavimo kostim M14 na početni kostim MM. Ovakav kod moramo da imamo za sve Gobo - e (dovoljno je u prethodnom kodu mijenjati Gobo - e). Dakle, potrebni su nam kodovi

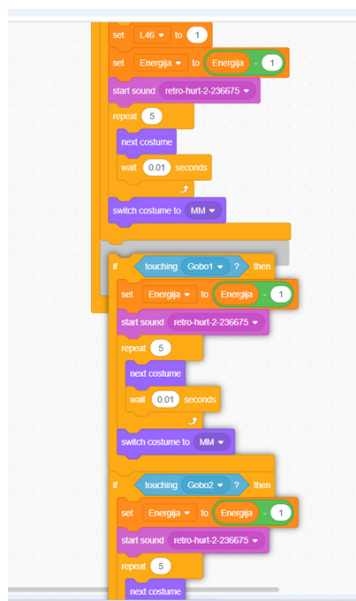


Ove kodove nadovežemo jedan na drugi, tako da kod za Gobo2 nadovežemo na kod za Gobo1, kod za Gobo3 nadovežemo na kod za Gobo2, itd., kod za Gobo50 nadovežemo na kod za Gobo49. Tako objedinjeni kod uklopimo u kod iza kojeg smo se pozicionirali, tj. u kod

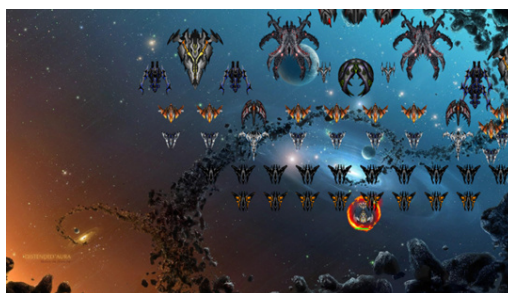




i to tako da novi kod ubacimo na kraj posljednjeg koda, i to iza završetka posljednje Ff naredbe a prije završetka forever petlje (vidjeti sljedeću sliku)



Kvakicu pored promjenjive Energija možemo ukinuti, jer nam više nema prevelikog značaja da znamo koliko ta energija zaista iznosi (nema potrebe da vrijednost ove promjenjive bude prisutna na ploči jer možemo vizuelno da pratimo kako se Energija smanjuje eventualnim dodirima M14 sa neprijateljskim oružjima i letjelicama). Sačuvamo izmjene i probamo igru

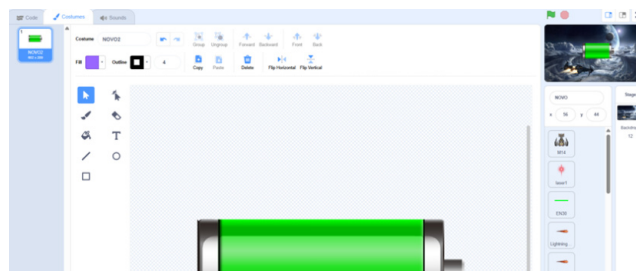


Vidimo da dodir M14 sa nekom od neprijateljskih letjelica rezultira efektom kojeg smo i željeli.

Uradimo u ovom dijelu i energiju po imenu NOVO koja se povremeno pojavljuje na vrhu ekrana (kao bočica sa energijom), i koja se pomjera polagano prema dnu ekrana. U slučaju njenog dodira sa letjelicom M14, Energija se poveća za 1 podiok (u slučaju da je Energija bila manja od 30). Ako je Energija jednaka 30, onda dodir M14 sa NOVO ne rezultira bilo kakvim promjenama Energije. Bilo da je Energija manja ili jednaka 30, dodir M14 sa NOVO treba da rezultira sklanjanjem NOVO sa ekrana. Trebat će nam sljedeći Sprite (NOVO2)

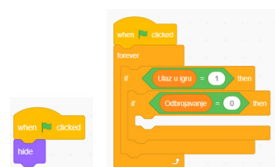


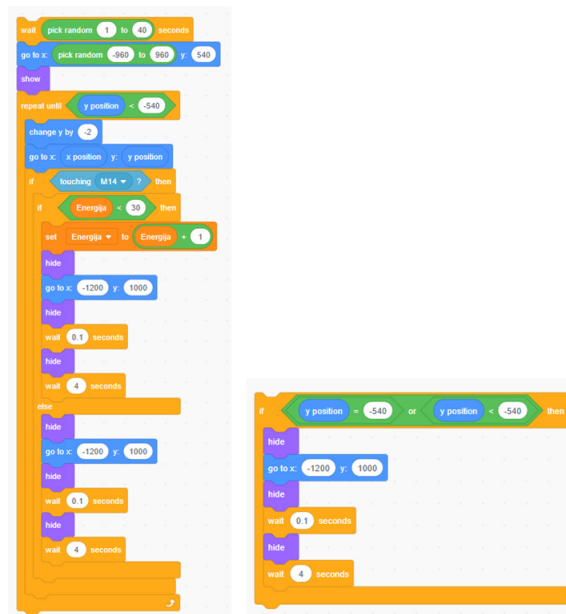
Upload - ujemmo ovaj Sprite (kao i Sprite - ove koje smo upload - ovali do sada)



Ime mu promijenimo u NOVO, a naziv kostima ostavimo da bude NOVO2. Osim toga, veličinu Sprite - a (Size) stavimo da je 10. Sačuvamo izmjene.

Dok smo pozicionirani na Sprite NOVO, idemo na Code, te unosimo sljedeće kodove





Prvi kod, klasično, govori o tome da Sprite NOVO (energija) ne treba da bude vidljiv na samome startu, nego da treba da postane vidljiv onda kada mi to odlučimo.

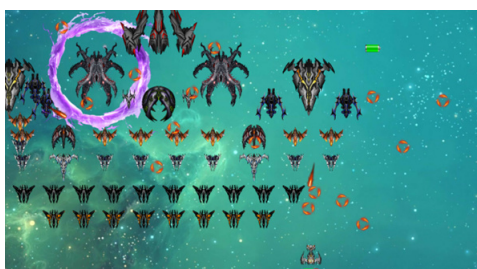
Drugi kod predstavlja glavnu petlju koda. Jasno, koristimo forever petlju za "lov" na događaje koji nam odgovaraju. Kao i do sada, ima smisla da se bočica NOVO pojavljuje nakon što smo kliknuli na dugme START (dakle nakon što je promjenjiva Ulaz u igru poprimila vrijednost 1), ali i dodatno, onda kada nismo u situaciji da uopšte nismo ušli u igru, i onda kada nismo u situaciji da na ploči nema neprijatelja (to je uslov da je Odbrojanje jednako 0). Ako su navedena dva uslova zadovoljena, bočica NOVO ima smisla da se pojavi na ekranu. Pojavljivanje sprovodimo u kodovima koji slijede.

Posmatrajmo sada treći kod. Ako su zadovoljena pomenuta dva uslova iz prethodnog dijela, onda ima smisla da se bočica NOVO pojavi na ekranu. Prvo, uvodimo neko vrijeme čekanja. Nema smisla da se bočica sa energijom stalno pojavljuje na ekranu. Znači, biramo neko proizvoljno vrijeme između 1 i 40 sekundi, i to vrijeme čekamo. Nakon toga se pozicioniramo na neku proizvoljnu lokaciju na vrhu ekrana. Govorimo o proizvoljnoj lokaciji po  $x$  osi (znači vrijednosti između -960 i 960), dok je lokacija po  $y$  osi fiksna, iznosi 540, što predstavlja sami vrh ekrana.

Prikažemo bočicu nakon toga. Jasno, moramo onda da bočicu pomjeramo prema dnu ekrana. Treba da je pomjeramo prema dnu, tj. dok ne dodirnemo dno ili dok bočicu ne dodirnemo našom letjelicom M14. Kako god, ciljamo da bočicu pomjeramo prema dole dok ne dostignemo poziciju da se po  $y$  osi nalazimo niže od vrijednosti -540 (što je dno ekrana). Zbog toga koristimo petlju repeat (sve dok  $y$  koordinatom ne dostignemo niže od -540). U svakoj iteraciji bočicu pomjeramo po  $y$  osi za -2 koraka, pri čemu  $x$  koordinatu ne mijenjamo. Ako na ovom putu bočica ne dodirne letjelicu M14, jasno je da će se ona onda spustiti na dno ekrana. Šta će se nakon toga desiti vidjet ćemo u četvrtom kodu. Zanima nas šta se dešava ako bočica na svome putu dodirne letjelicu M14. Ako bočica na svojoj putanji prema dnu ekrana dodirne letjelicu M14, onda se možemo pitati da li je naša Energija manja od 30, ili je pak jednaka 30. Tako, ako bočica dodirne M14, a Energija je manja od 30, onda Energiju povećamo za 1 po-diok, sakrijemo bočicu, premjestimo je na za nju rezervisanu lokaciju (-1200,1000), pri čemu dodatno sakrivanje obavimo nekoliko puta (da bi se osigurali), uz kratkotrajna zadržavanja od 0.1 i 4 sekunde. S druge strane, ako je bočica dodirnula letjelicu M14, a pri tome je Energija jednaka 30, onda sa Energijom ne treba da se desi bilo šta dodatno (ostavimo je na miru), a bočicu sklonimo sa ekrana na isti način kao u prethodnom slučaju. Na ovaj način smo opisali sastav trećeg koda.

Konačno, četvrti kod se odnosi na situaciju kada bočica na svome putu prema dnu ekrana uopšte ne dodirne letjelicu M14, nego zaista se uspije spustiti po  $y$  osi na manje ili jednako -540. Tada, jasno, treba da sklonimo bočicu sa ekrana, što radimo na isti način kao u prethodnom slučaju: sakrijemo bočicu, premjestimo je na njenu lokaciju (-1200,1000), i sakrivanje obavimo više puta uz povremena zadržavanja od 0.1 i 4 sekunde.

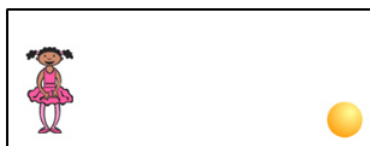
Početak četvrtog koda nadovežemo na kraj trećeg koda, i tako objedinjeni kod uklopimo u uslov: `if Odbrojanje = 0`, drugog koda. Na taj način je kod za Sprite NOVO kompletiran. Sačuvamo izmjene i probamo igru



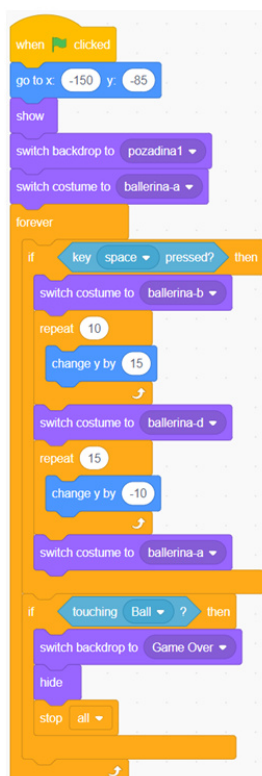
Nije teško za uočiti da uneseni kod radi upravo ono što smo i željeli da on radi. Bočica sa energijom se pojavljuje povremeno na vrhu ekrana i putuje polagano prema dnu. Ukoliko nam je Energija oštećena a dodirnemo bočicu, energija će porasti. S druge strane, ako je naša Energija na maksimumu, onda pri dodiru sa bočicom NOVO, stanje ostaje nepromijenjeno. Prilikom dodira M14 sa bočicom NOVO, bez obzira da li je naša Energija manja ili jednaka 30, bočica NOVO će biti sklonjena sa ekrana. Ukoliko nikako ne dodirnemo bočicu letjelicom M14, bočica će doći do dna ekrana i nestati. Ovdje bi eventualno zainteresovani čitalac mogao razmisliti o tome kako da prilikom dodira M14 sa bočicom NOVO napravi dodatni vizuelni ili audio efekat liječenja M14 energijom sadržanom u bočici NOVO. To bi mogao da bude neki popratni zvuk ili eventualno neki dodatni kostim za Sprite M14 koji bi na određeni način sugerisali da se liječenje zaista i dogodilo.

### 13.1 Riješeni zadaci

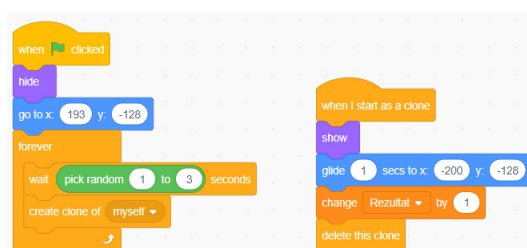
◇ **Zadatak 13.1.1** Na sceni su 2 lika, balerina (lijevo) i lopta (desno). Potrebno je animirati loptu tako da se, u nasumičnim intervalima, kotrlja prema balerini (koristiti kloniranje). Balerina treba da preskoči loptu pritiskom na tipku space (koristiti različite kostime za stanje mirovanja i kretanja). Ukoliko balerina udari loptu, igra se zaustavlja i prikazuje se Game Over pozadina.



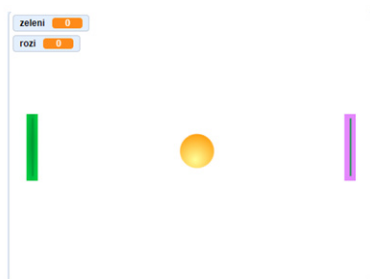
**Rješenje:** Kod za balerinu:



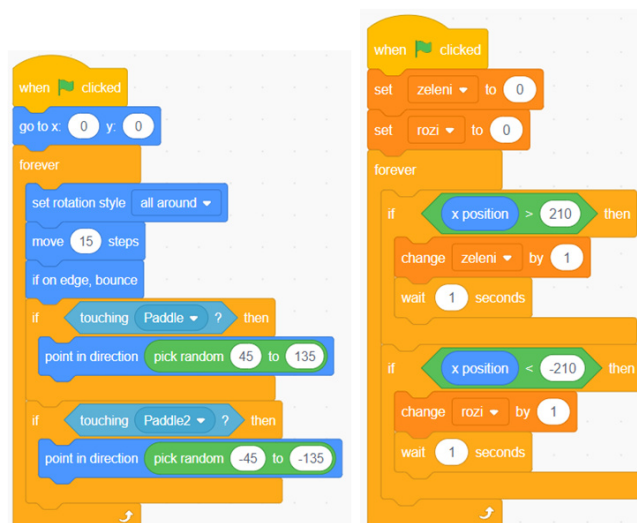
Kod za loptu:

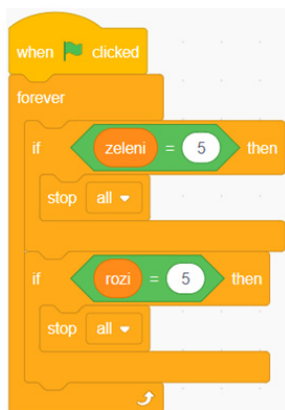


◇ **Zadatak 13.1.2** Potrebno je kreirati klasičnu Pong igricu. Na sceni se nalazi lopta koja se neprestano kreće i odbija od gornje i donje ivice ekrana, dok se s lijeve i desne strane nalaze reketi koje igrači pomjeraju pomoću tastature. Kada lopta dotakne reket, odbija se nazad. Ukoliko prođe mimo reketa, protivnički igrač dobija poen. Na ekranu se prikazuje rezultat obje strane. Omogućiti kontrolu oba reketa pomoću različitih tipki na tastaturi. Cilj igre je da igrač koji prvi osvoji određeni broj poena (npr. 5) bude proglašen pobjednikom. Uz to, treba da bude prikazana odgovarajuća poruka

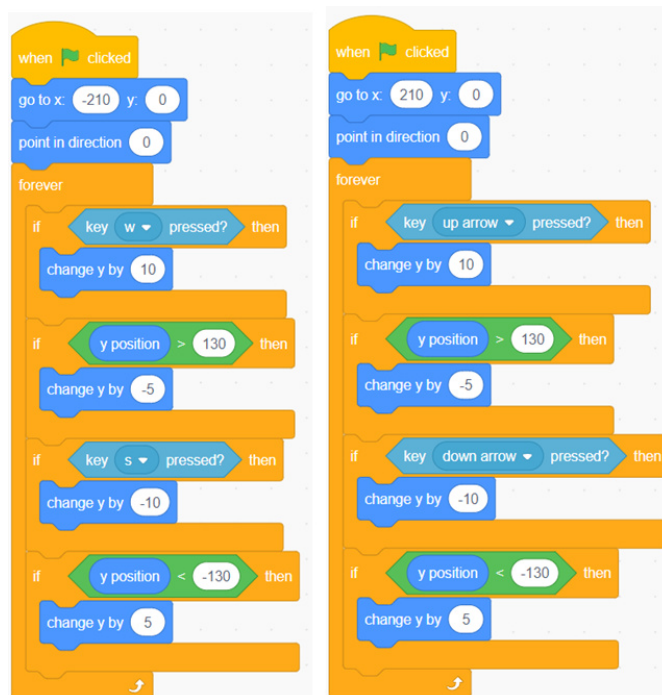


**Rješenje:** Kod za loptu:



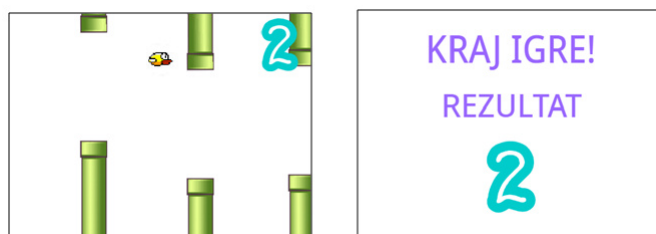


Kod za zeleni (lijevo) i rozi (desno) reket:

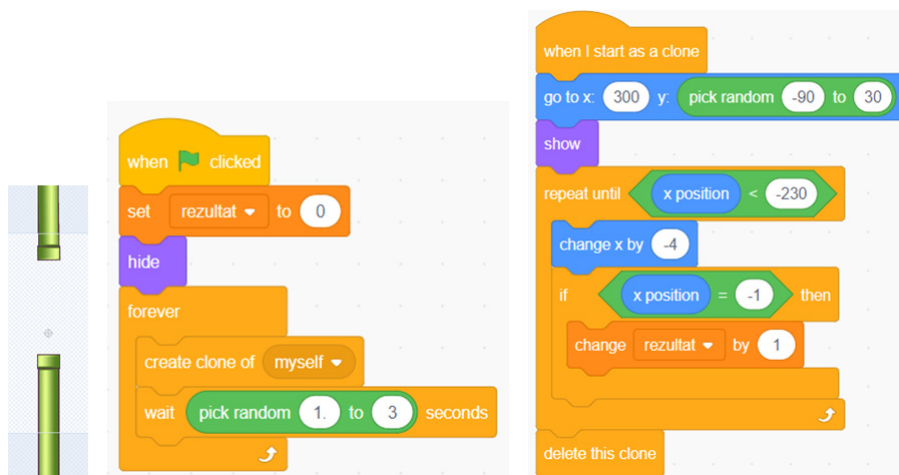




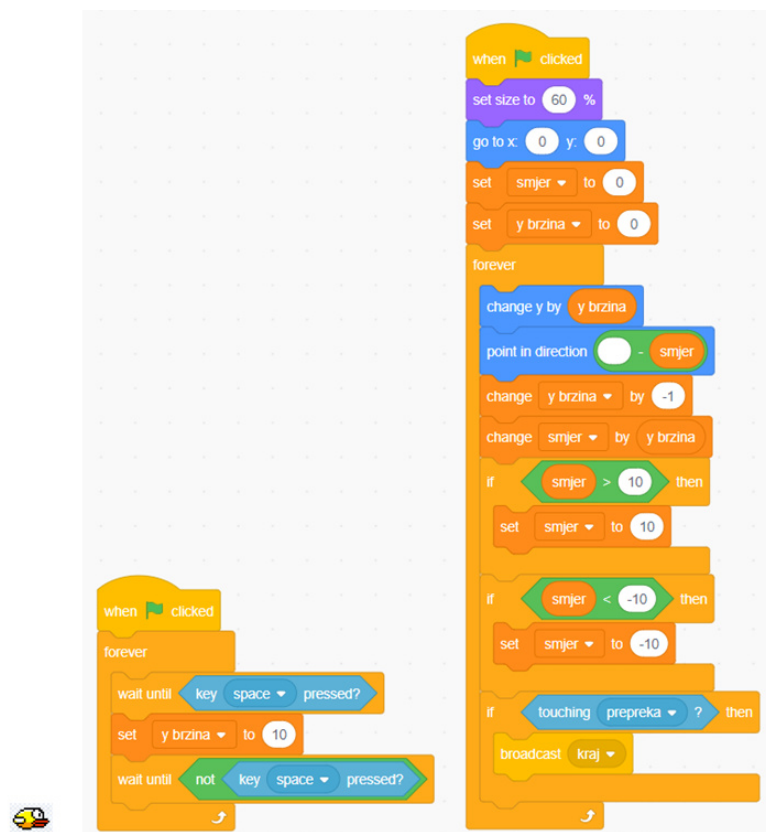
◇ **Zadatak 13.1.3** Kreirati jednostavnu verziju igrice Flappy Bird. Na sceni se nalazi lik ptice koja neprestano pada zbog sile gravitacije. Igrač pritiskom na tipku space omogućava ptici da leti prema gore, dok inače ptica pada naniže. Na sceni treba generisati prepreke u obliku stubova koji se kreću zdesna prema lijevo, između kojih je prostor kroz koji ptica mora proći. Ako ptica udari u prepreku ili dodirne tlo, igra se zaustavlja i prikazuje se poruka o kraju igre. Za svaki uspješno savladani prolaz kroz prepreke igrač dobija jedan poen, a trenutni rezultat se prikazuje tokom igre. Za prikaz rezultata koristiti posebne Sprite-ove (slika ispod)



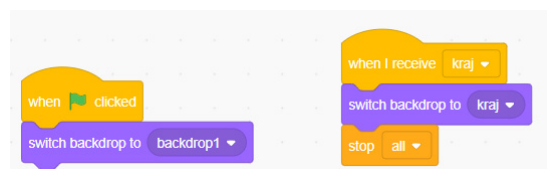
**Rješenje:** Kod za prepreku:



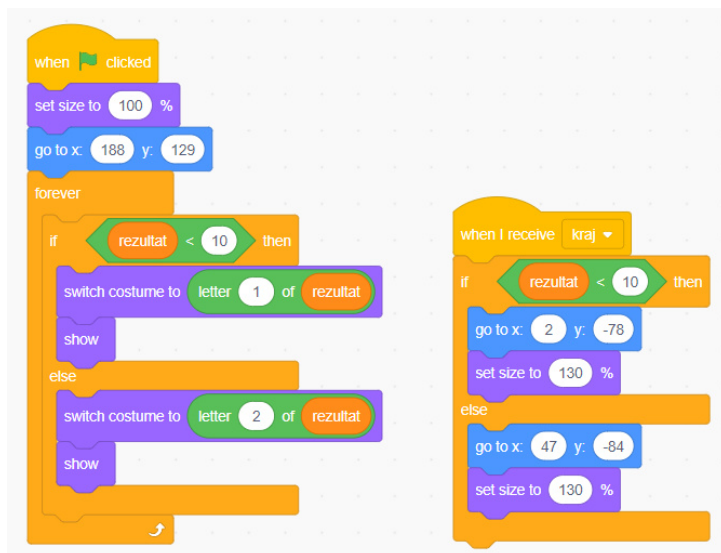
Kod za pticu:



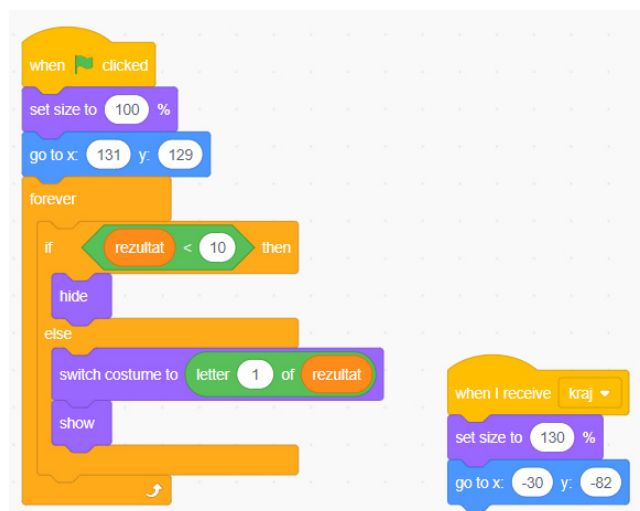
Kod za pozadinu:



Kod za prikaz jedinica rezultata:



Kod za prikaz desetica rezultata:



### 13.2 Zadaci za vježbu

♣ **Zadatak 13.2.1** Kreirati igricu tipa Memory Game (igra pamćenja). Na sceni se nalazi 12 prekrivenih kartica (6 parova). Kada igrač klikne na dvije kartice, one se privremeno otkrivaju. Ako su iste, ostaju otvorene. Ako nisu, ponovo se zatvaraju nakon kraćeg vremena. Cilj igre je pronaći sve parove u što kraćem vremenu ili sa najmanje pokušaja. Prikazivati broj pokušaja na ekranu. Dodati efekat čestitke u momentu kada su svi parovi pronađeni.

♣ **Zadatak 13.2.2** Na sceni se nalazi reket koji prati pokrete kursora i lopta koja se konstantno kreće i odbija od ivica ekrana i reketa. Potrebno je implementirati program tako da se reket pomjera prema poziciji kursora (u oba smjera: vodoravno i okomito), dok se lopta odbija prilikom dodira sa zidovima ekrana i reketom. Ako lopta dodirne donju ivicu ekrana (pozicionira se ispod reketa), igra se zaustavlja i prikazuje se poruka da je igra završena. Cilj igre je zadržati loptu u pokretu što je duže moguće, odnosno spriječiti je da padne.

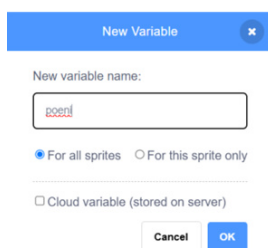
♣ **Zadatak 13.2.3** Proširiti program iz prethodnog zadatka tako da se u toku igre prati broj uspješnih odbijanja lopte od reketa i prikazuje kao rezultat na ekranu. Nakon svakih 5 uspješnih odbijanja, lopta treba da poveća svoju brzinu kretanja, čime se igra postepeno otežava. Takođe, nakon svakih 10 uspješnih odbijanja, na sceni treba da se pojavi nova lopta (koristiti kloniranje), pa igrač mora da istovremeno zadrži više loptica u igri. Kada bilo koja lopta dodirne donju ivicu scene, igra se završava i prikazuje se ukupan broj odbijanja kao rezultat. Cilj igre je ostvariti što veći broj uspješnih odbijanja.

## 14 Suplementi

Ono što svaka igra oblika Galaksije zasigurno ima su poeni, odnosno pomno razrađen sistem bodovanja, tj. nagrađivanja eliminacije neprijateljskih letjelica. Tako, eliminacija određene letjelice donosi korisniku toliko i toliko poena, dok eliminacija neke druge vrste letjelice daje obično neki drugi broj poena. Razrađeni su obično i sistemi nagrađivanja takvi da prelaskom određene granice poena, korisnik dobija dodatni život, i sl.

Želja nam je sada da u naše razmatranje uvedemo i poene.

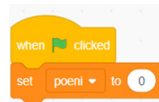
Prvo, definisat ćemo jednu promjenjivu po imenu "poeni"



Neka za sada ova promjenjiva bude vidljiva na ekranu, tj. neka pored njenog imena za sada bude aktivna kvakica.

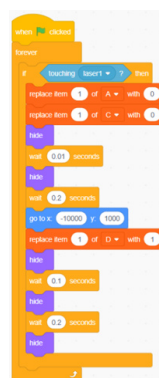
Trenutno, nećemo razrađivati sistem po kome ćemo eliminaciju različitih neprijateljskih letjelica različito i nagrađivati. Naime, ako razmislimo, za sada se eliminacija neprijateljskih letjelica ne razlikuje. Dovoljan je jedan pogodak oružjem "laser1" pa da bilo koja od neprijateljskih letjelica bude eliminisana. Drugim riječima, prilikom eliminacije neprijateljskih letjelica, ne ulažemo dodatni napor za bilo koju od njih. Nemamo razrađen sistem energija neprijateljskih letjelica ili neki drugi sistem koji bi otežavao eliminaciju pojedinih od njih. Zbog toga, za sada eliminaciju neke od neprijateljskih letjelica smatramo ravnopravnom sa eliminacijom bilo koje druge neprijateljske letjelice. Možemo onda smatrati da eliminacija bio koje od 50 neprijateljskih letjelica povećava broj poena za 1 (jasno je da su svi za sada pomenuti parametri u potpunosti varijabilni i da se mogu prilagoditi željama i viziji korisnika). Znači, očekivano, prilikom pokretanja igre, treba da postavimo promjenjivu "poeni" na vrijednost 0. Ova promjenjiva je dostupna svim dijelovima programa, a na određeni način je neutralna jer je vezana sa svih 50 neprijateljskih brodova. Ima smisla da je kao i ostale

neutralne objekte inicijaliziramo na neutralnoj lokaciji, tj. na lokaciji za kod podloga. Idemo na Backdrops, pa na Code, i unesemo sljedeći kod

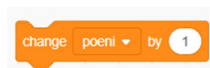


Kada treba da mijenjamo sadržaj promjenjive "poeni", tj. kada treba da povećavamo vrijednost sadržanu u promjenjivoj "poeni" za 1 ? Onda, kada bilo koja od 50 neprijateljskih letjelica bude pogođena oružjem "laser1". Naime, od oružja koja smo do sada izgradili "na našoj strani", tj. od oružja vezanih za našu letjelicu M14, jedino oružje koje je u stanju da eliminiše neprijateljske letjelice je oružje "laser1". Dakle ima smisla rezonovati na način da onda kada neka od 50 neprijateljskih letjelica dodirne "laser1", sadržaj promjenjive "poeni" uvećamo za 1.

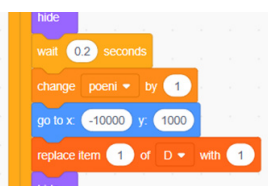
Vodeći se upravo opisanim rezonom, pogledajmo sadržaj koda neke od neprijateljskih letjelica. Dovoljno je da posmatramo prvu od njih, tj. da se pozicioniramo na Code za Gobo1. Svaka od pomenutih 50 letjelica ima za sada dva bitna koda. Jedan, kojeg smo zvali "lijevi" koji je duži, i kraći "desni" kod. Sjetimo se da je "lijevi" tj. duži kod bio zadužen za inicijalni raspored letjelice na ploču, a zatim za regulaciju njenog kretanja, dok je "desni", tj. kraći kod bio zadužen za vizuelno uklanjanje letjelice sa ploče onda kada ona dodirne "laser1". Odmah nam je jasno da treba da se pozicioniramo na ovo mjesto, tj. da posmatramo "desni", tj. kraći kod. Riječ je o kodu



Tako, ako Gobo1 dodirne "laser1", postavljamo item 1 lista: A, C da bude jednak 0, dok item 1 liste D postavljamo da bude jednak 0. Sve ovo, naravno, radimo uz upotrebu naredbe "hide" kojom krijemo Gobo1, i to više puta (radi sigurnosti) uz duža i kraća zadržavanja (objasnili smo raniju situaciju po kojoj jednostruka upotreba naredbe "hide" nekada nije dovoljna da se sakrivanje objekta zaista i desi). Kada item 1 liste A postavljamo na vrijednost 0, to nam je značilo da smatramo da je Gobo1 eliminisan sa ploče. Činjenicu da je item 1 liste C jednak 0 smo koristili kao informaciju da "laser1", ma gdje se u tom momentu nalazio na vidljivom dijelu ploče, treba da sklonimo sa ploče. Konačno, postavljanje itema 1 liste D na vrijednost 1 smo koristili kao informaciju da je Gobo1 spreman za ponovno inicijalno raspoređivanje na ploču (nakon kojeg smo item 1 liste D postavljali na vrijednost 0). Ovdje smo se ukratko podsjetili rada ovog "kratkog" koda. Po pitanju promjenjive "poeni", nama je ovaj kod dovoljan jer se on odigrava onda kada Gobo1 dodirne "laser1". Naime, taj događaj nam odgovara. Drugim riječima, ako Gobo1 zaista dodirne "laser1", onda to znači da je Gobo1 mrtav, i da treba da vrijednost promjenjive "poeni" povećamo za 1. Naravno, ovaj rezon vrijedi i za ostale Gobo - e. Dovoljno je onda da u posmatrani "kratki" kod za Gobo1 ubacimo naredbu kojom povećavamo vrijednost promjenjive "poeni" za 1. Tako, u posmatrani "kratki" kod ubacujemo (na bilo koju lokaciju) naredbu



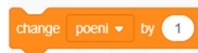
Odgovarajući dio dobijenog koda sada ima oblik (jedna od mogućih varijanti)



Vidimo da smo naredbu za povećanje vrijednosti promjenjive "poeni" umetnuli prije naredbe "go to" koju smo koristili za transfer letjelice Gobo1 na za nju unaprijed rezervisanu lokaciju (-10000,1000).

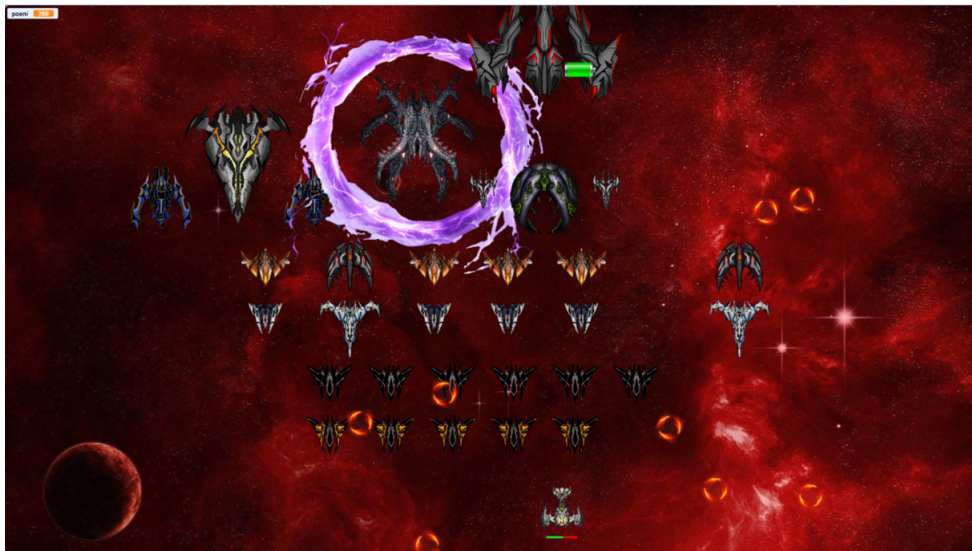
Navedeni rezon možemo koristiti i za preostalih 49 neprijateljskih letjelica.

Ubacimo onda navedenu naredbu



na analogno mjesto u "kratki" kod svake od preostalih 49 neprijateljskih letjelica. Dovoljno je da kopiramo postojeću korištenu naredbu za Gobo1, i zalijepimo je na preostalih 49 Gobo - a.

Sačuvamo izmjene i probamo igru



Vidimo da promjenjiva "poeni" zaista služi svojoj svrsi. Eliminacijom neprijatelja povećava se njena vrijednost za 1, rezultat se update - uje u realnom vremenu i prikazuje u gornjem lijevom vrhu ekrana.

Neko bi bio zadovoljan sa urađenim, ali ne i mi. Naime, jasno je svakome ko poznaje odkruženje Eticsques da je prikaz promjenjive "poeni" u gornjem lijevom uglu ekrana ustvari samo prikaz ugrađene promjenjive, tj. da je način na koji je ona prikazana ustvari onaj jedini mogući prikaz koji nam nudi okruženje Eticsques. Drugim riječima, jasno je svakome ko koristi navedeno okruženje da se u ovakvoj konstalaciji odnosa tvorac nije pretjerano potrudio da izgradi valstiti izgled promjenjive "poeni", već da je samo iskoristio i prikazao postojeće objekte. Zbog svega navedenog, naša



želja je da prikaz poena uradimo na vlastiti originalan način. Ne želimo da za prikaz promjenjive "poeni" ali i za prikaz sličnih objekata koristimo ugrađene elemente programa Eticques koji su prikazani nekim sitnim slovima i narandžastom bojom (osim ako tako želimo). Želja je dakle, kao i svakog drugog programera koji drži do svoje originalnosti, da u skladu sa mogućnostima i potrebama izgradimo vlastiti prikaz u ovom slučaju poena, tj. promjenjive "poeni". Mi ćemo u ovoj izgradnji svakako koristiti sadržaj promjenjive "poeni", jer ona zaista broji poene, ali ćemo taj rezultat prikazati na originalan način.

Trebat će nam nekoliko novih Sprite - ova koje ćemo sami izgraditi.

Prvo treba nam Sprite po imenu "poeni". Njegov izgled je



Za izgradnju ovog Sprite - a (slovo po slovo) koristili smo transparentnu slovaricu "metal"



Dakle, kao i prilikom izgradnje dugmeta START (slovo po slovo), sada i ovdje, koristeći isti font (transparentnu slovaricu) "metal" gradimo riječ POENI. Jasno je da to radimo slovo po slovo, tj. da upload - ujmemo datu slovaricu kao neki Sprite po nekom imenu, te da iz tog Sprite - a isjecamo slova "P", "O", itd, i onda ih lijepimo na neki drugi Sprite, sve dok ne dobijemo željenu riječ "POENI". Jasno je da pri tom procesu pazimo na format slike, tako da krajnji format bude u vektorskom obliku. Drugim riječima, kao i do sada vršimo isjecanje, lijepljenje, te konačno objedinjavanje zalijepljenih slova u jednu cjelovitu riječ "POENI". Ukoliko se sjetimo procesa rada u slučaju riječi START, onda znamo da je za navedeno objedinjavanje dovoljno koristiti prijelaz na format Bitmap a odmah nakon toga prijelaz na vektorski oblik slike sastavl-

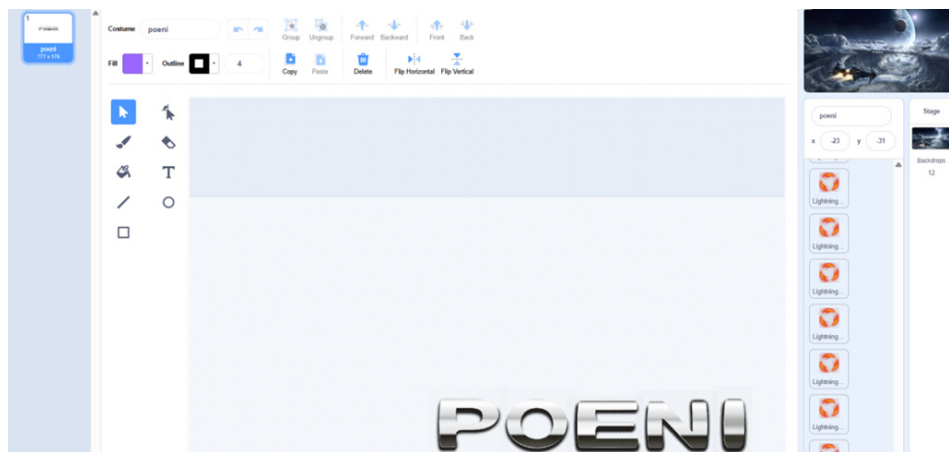
jene od slova "P", "O", "E", "N", "I". Ukratko smo se podsjetili principa rada prilikom formiranja riječi koje želimo da budu prikazane na ploči u datom momentu i to datim fontom. Podrazumijevamo jasno da smo navedenim tehnikama već ovladali (u slučaju da je potrebno, korisnik se može podsjetiti principa našeg prethodnog analognog rada prilikom izgradnje riječi "START" ali i drugih objekata koji su zahtijevali analogan rezon).

Ukoliko raspoložemo gotovim, izgrađenim objektima, onda nastavljamo na očekivani način.

U donjem desnom uglu idemo na Choose a Sprite, pa na Upload Sprite, te upload - ujem Sprite "poeni" sa lokacije na koju smo ga pohranili



Ime kostima ostavimo da bude "poeni". Isto tako, ime Sprite - a ostavimo da bude "poeni". Veličinu ovog Sprite - a (Size) postavimo da bude jednaka 18

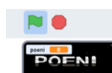


Riječ "POENI" treba da se pojavi nakon klika na dugme START, i to na lokaciji koju mi proglasimo pogodnom. Tako, dok smo na ovome Sprite - u, idemo na Code,

pa unesemo sljedeći kod



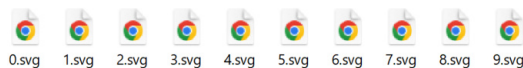
Jasno, u momentu pokretanja igre, nema potrebe da Sprite "poeni" bude vidljiv na ekranu. Zbog toga ga inicijalno sakrijemo. Učinit ćemo ga vidljivim nakon klika na dugme START. Znamo da nakon klika na dugme START vrijednost promjenjive "Ulaz u igru" postaje jednaka 1. Tako, forever petljom "lovimo" pogodan događaj, tj. događaj klika na dugme START, odnosno, prevedeno na jezik naredbi, događaj kada je "Ulaz u igru" jednako 1. Tada šaljemo Sprite "poeni" na lokaciju (-870,500), što je gornji lijevi ugao, i prikazemo ga. Nakon pokretanja igre, i klika na dugme START, imamo situaciju kao na slici



Na ovaj način smo riješili fiksni dio, tj. dio koji se odnosi na riječ POENI (vidljivost promjenjive "poeni" koja se nalazi iznad ćemo riješiti kasnije - ona će biti sklonjena sa ekrana). Sačuvamo izmjene.

Ono što ostaje da uradimo jeste glavni i najteži dio posla ovog dijela rada. Naime, želimo da iza upravo ispisane riječi POENI, budu zaista ispisani poeni, tj. vrijednost sadržana u promjenjivoj "poeni" (broj sadržan u narandžastom polju iznad), i to fontom kojeg mi odaberemo i proglasimo podobnim.

Trebaju nam kostimi po imenu: "0", "1", "2", ..., "9"



Njihov izgled je



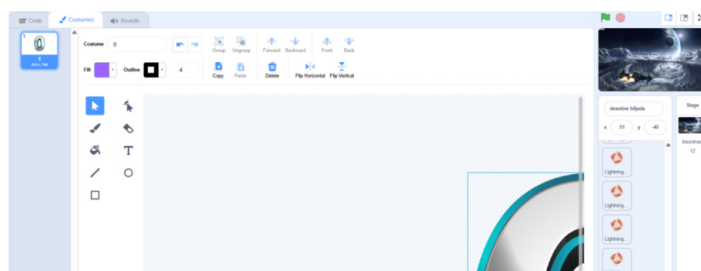
Kao i do sada, ove transparentne Sprite - ove (slike) nalazimo na stranici

<https://www.pngwing.com>

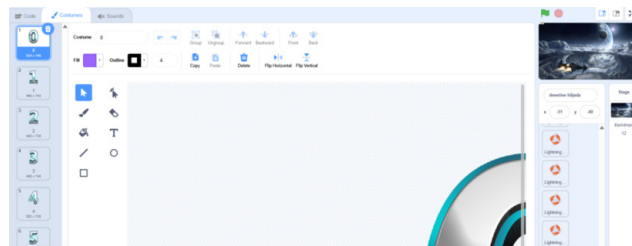
Naime, u pretrazi na navedenoj stranici dovoljno je tražiti objekat tipa "numbers" ili slično. Brojevi mogu ali i ne moraju biti dio određenih slovarica. Češće ih nalazimo kao samostalne transparentne objekte, te ih je s te strane lakše i koristiti (zaobilazimo potrebu za njihovim isjecanjem i lijepljenjem). Primijetimo da navedeni Sprite - ovi brojeva nisu istog tipa kao slova koja smo prethodno koristili (ne moraju ni da budu istog tipa). Ipak, oblikom i izgledom se uklapaju u sveukupni ambijent igre koju pravimo.

Idemo sada da napravimo novi Sprite.

Kao i do sada, idemo u donjem desnom uglu na Choose a Sprite, pa na Upload Sprite, te upload - ujem sliku po imenu "0" (sa lokacije na koju smo je smjestili). Slika je u vektorskom obliku (kao i prethodna slika "poeni"), pa ne moramo voditi računa o tome da će doći do neželjenih komplikacija i modifikacija slike prilikom njenog eventualnog kretanja i upotrebe po ploči. Ovom Sprite - u dajemo ime "desetine hiljada", ime kostima ostavljamo da je "0" (bez navodnika naravno), dok veličinu Sprite - a (Size), stavljamo da bude jednaka 3



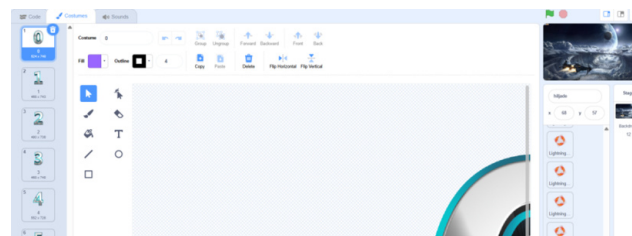
Sada, dok smo i dalje pozicionirani na ovaj Sprite, idemo dole lijevo na Choose a Costume, pa na Upload Costume, te upload - ujemo kostime ostalih brojeva, tj. upload - ujem slike: "1", "2", ..., "9". Sve nazive ostavimo iste, tj. nazive kostima: "1", "2", ..., "9", ostavljamo nepromijenjenim. Pozicioniramo se na kostim "0" i sačuvamo izmjene



Jasno je da ako želimo da upotrijebimo desetine hiljada kao mjernu jedinicu, da će nam onda trebati i hiljade, stotice, desetice i jedinice. Zbog toga ćemo, radi jednostavnosti, već napravljeni Sprite "desetine hiljada", duplicirati. Idemo na Sprite "desetine hiljada", pa na duplicate



Na novo - dobijenom Sprite - u, ponuđeno ime "desetine hiljada2" promijenimo u "hiljade". Ostale elemente Sprite - a, tj. nazive kostima i veličinu ostavimo nepromijenjenim



Trebaju nam sada i stotice, desetice i jedinice.

Ponovimo postupak.

Sprite "hiljade" dupliciramo, i novo - dobijenom Sprite - u promijenimo ime u "stotice", dok ostale elemente ostavimo iste. Zatim, Sprite "stotice" dupliciramo i dobijenom Sprite - u promijenimo ime u "desetice" (ostale elemente Sprite - a ostavljamo onakvim kakvi su i bili), te konačno, Sprite "desetice" dupliciramo, promijenimo naziv dobijenog Sprite - a u "jedinice", dok ostale elemente Sprite - a ne diramo. Sačuvamo izmjene.

Pozicioniramo se na Sprite "desetine hiljada", idemo na Code, pa unesemo sljedeći kod (kodove)



Kodove koji sadrže naredbe: if letter 1 of poeni = 1, if letter 1 of poeni = 2, ..., if letter 1 of poeni = 9, nadovežemo jedan na drugi, onako kako slijede (drugi na prvi, treći na drugi, ..., deveti na osmi), i dobijeni kod uklopimo u drugi kod, tj. unutar naredbe: if length of poeni = 5. Na ovaj način je kod Sprite - a "desetine hiljada" kompletiran. Objasnimo još šta ovaj kod radi.

Prvi kod je jasan, jer jasno je da Sprite "desetine hiljada" ne treba da bude vidljiv prilikom pokretanja igre. Dakle, ovaj Sprite je na početku sakriven (činimo ga vidljivim onda kada mi to smatramo da je potrebno).

Drugi kod se odigrava onda kada sa brojem poena dostignemo vrijednosti koje se izražavaju u desetinama hiljada. Drugim riječima, ako sa poenima dostignemo vrijednosti između 10000 i 99999, ovaj kod će da se izvrši. Ukoliko sa brojem poena budemo strogo ispod 10000, navedeni kod se neće izvršiti. Forever petlju kao i do sada koristimo za "hvatanje" događaja koji nam odgovaraju. Jasno je da izvršavanje koda ima smisla tek onda kada smo kliknuli na dugme START, tj. kada je vrijednost promjenjive "Ulaz u igru" postala jednaka 1. Osim toga, kako rekosmo, kod ima smisla da se izvršava samo onda kada se sa poenima nalazimo između vrijednosti (uključivo) 10000 i 99999, tj. onda kada je dužina vrijednosti sadržane u promjenljivoj "poeni"

jednaka 5. Ukoliko su zadovoljeni navedeni uslovi, tj. ukoliko je vrijednost sadržana u promjenljivoj "poeni" zaista neki petocifren broj, šta onda treba da ispišemo ? Treba da ispišemo prvu cifru tog petocifrenog broja (govorimo o mjernoj jedinici "desetine hiljada"). Radimo dakle ono što slijedi u preostalim kodovima.

Tako, ako je prvi znak stringa "poeni" (prva cifra broja sadržanog u promjenljivoj "poeni") jednak 1, ispisujemo jedinicu. To znači da se prebacujemo na kostim "1" (koji i sadrži znak 1), postavljamo ga na lokaciju neposredno desno od riječi POENI (to je lokacija (-770,500)), i prikazujemo ga. Jasno, ukoliko je prvi znak stringa "poeni" jednak 7, onda se prebacujemo na kostim "7", postavljamo ga na lokaciju (-770,500), i prikazujemo ga.

Napomenimo da osim ako sa brojem poena ne dostignemo vrijednost između 10000 i 99999, rezultat izvršavanja ovog koda (koda datog na Sprite - u "desetine hiljada") nećemo biti u mogućnosti da vidimo. Sačuvamo izmjene i probamo igru.

Postavlja se sada pitanje kako iskodirati Sprite "hiljade".

Potrebno nam je da ukoliko broj poena premaši 999, tj. ako bude veći ili jednak 1000, da nam onda kod za Sprite "hiljade" zaista i ispisuje na odgovarajućem mjestu broj hiljada sadržanih u vrijednosti koja se nalazi pohranjena u promjenljivoj "poeni". Tako, ako se u promjenljivoj "poeni" nađe vrijednost 5421, onda kod za "hiljade" treba da ispiše na lokaciju (-770,500) broj 5. S druge strane, ako se u promjenljivoj "poeni" zadesi vrijednost 37698, onda kod za "desetine hiljada" treba da nam na lokaciji (-770,500) ispiše broj desetina hiljada (to je broj 3), dok kod za "hiljade" onda treba da na lokaciji desno od (-770,500) ispiše broj hiljada, tj. broj 7. Kao pomjerenu lokaciju desno od (-770,500), koristit ćemo lokaciju (-745,500). Vidimo da prilikom implementacije koda za "hiljade", moramo voditi računa o tome da li broj sadržan u promjenljivoj "poeni" ima 4 cifre ili 5 cifara. Ako taj broj ima 4 cifre, onda njegovu prvu cifru ispisujemo na lokaciju (-770,500). Ako taj broj ima 5 cifara, onda njegovu drugu cifru ispisujemo na lokaciju (-745,500).

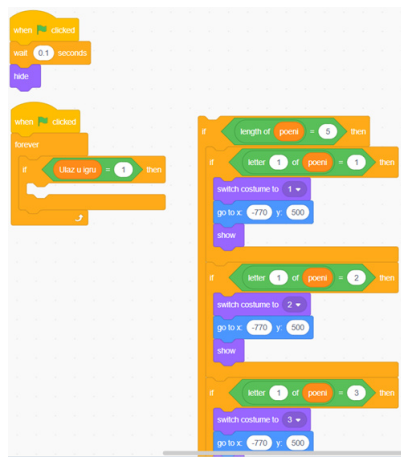
Na ovaj način smo dali pseudo kod za Sprite "hiljade", tj. opisali smo riječima šta taj kod zaista treba da radi.

Pristupimo sada implementaciji opisanog koda.

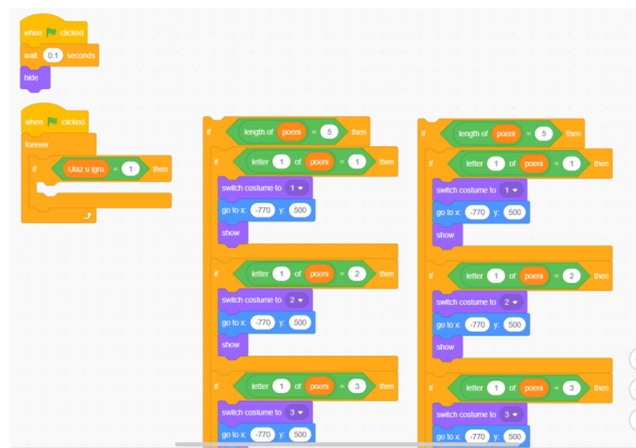
Uvijek, ako je moguće, treba pokušati iskoristiti neki postojeći kod, tj. pokušati olakšati sebi rad u smislu da ne moramo ponavljati određene radnje. Za kod koji se odnosi na Sprite "hiljade" pokušat ćemo iskoristiti kod koji imamo na Sprite - u "desetine hiljada".

Idemo na Sprite "desetine hiljada", i njegov kod (kodove) prenesemo (kopiramo) na Sprite "hiljade".

Postavimo se sada na Code za Sprite "hiljade", i iz velikog koda izvučemo van skup naredbi sadržanih u if uslovu: if Ulaz u igru = 1. Imamo situaciju kao na slici

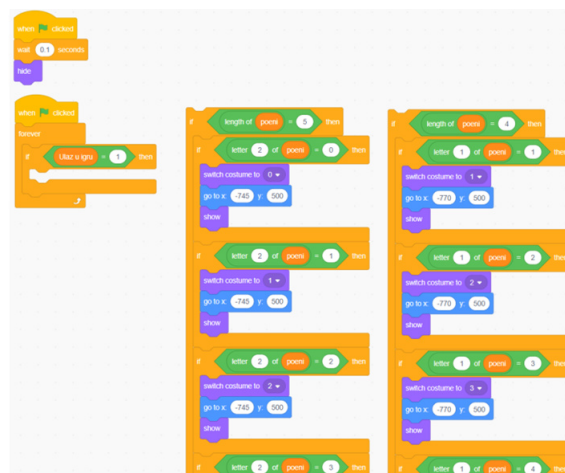


Desni kod umnožimo, tj. dupliramo i postavimo desno od postojećeg koda. Dobijamo



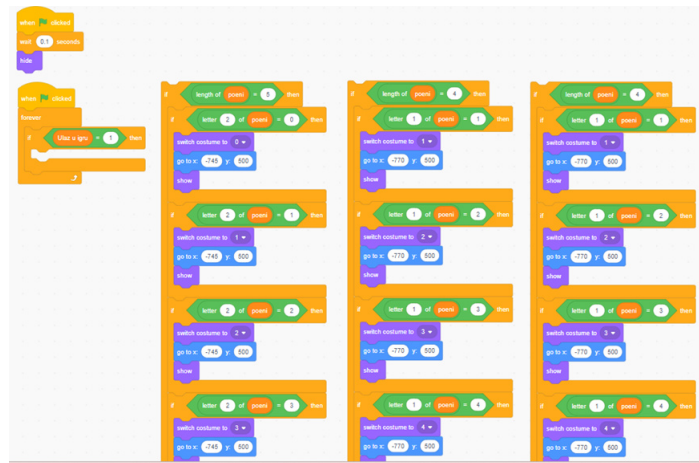


Drugi "dugi" kod će da se odnosi na situaciju kada je dužina stringa "poeni" jednaka 4, tj. na situaciju kada se u promjenjivoj "poeni" bude nalazio četverocifren broj. Pošto nas zanima ispis hiljada (a desetina hiljada sada nemamo), to na prvu lokaciju (-770,500) treba da ispišemo prvi znak stringa "poeni", tj. prvu po redu cifru vrijednosti sadržane u promjenjivoj "poeni". Znači riječ je o broju tipa 4859, gdje nemamo desetina hiljada, pa prva cifra ovakvog broja ne može biti jednaka 0. Tako, u drugom "dugom" kodu, dovoljno je uslov: `if length of poeni = 5` zamijeniti uslovom: `if length of poeni = 4`. Ostale naredbe ostavljamo nepromijenjenim. Ovdje kako smo vidjeli, lokaciju (-770,500) kolokvijalno zovemo "prvom" lokacijom, dok ćemo lokaciju (-745,500) zvati "drugom" lokacijom. Lokacije koje se budu pojavljivale u nastavku ćemo zvati "trećom", "četvrtom" i "petom". Nakon opisanih izmjena u prvom i drugom "dugom" kodu, treba da imamo situaciju kao na slici

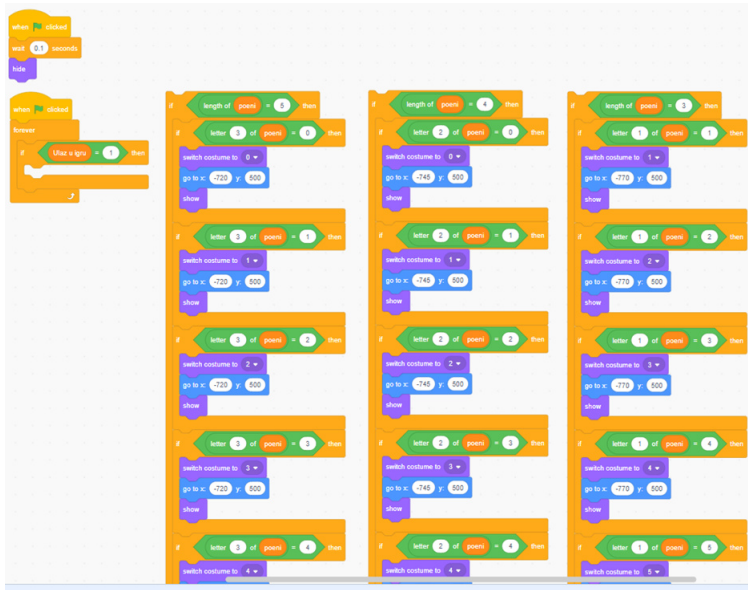


Drugi "dugi" kod nadovežemo na prvi "dugi" kod, i tako objedinjeni kod uklopimo u if uslov prvog koda. Dobijeni kod je željeni kod za Sprite "hiljade". Ovdje, kao i u slučaju Sprite - a "desetine hiljada", napominjemo da rezultat izvršavanja ovog koda, tj. koda za Sprite "hiljade", neće biti vidljiv osim ako sa brojem poena ne dostignemo vrijednost veću ili jednaku 1000. Sačuvamo izmjene i probamo igru. Vezano za pomenute lokacije, pod prvom lokacijom ćemo podrazumijevati lokaciju (-770,500), pod drugom lokaciju (-745,500), pod trećom (-720,500), pod četvrtom (-695,500), i pod petom lokaciju (-670,500).

Jasno je da ćemo sada za pravljenje koda Sprite - a "stotice" iskoristiti upravo napravljeni kod za Sprite "hiljade". Postavimo se na Sprite "hiljade" i njegov kod (kodove) prenesemo na Sprite "stotice". Postavimo se na kod za "stotice", i iz dugog koda izvučemo van kod koji se nalazi unutar uslova: if Ulaz u igru = 1. Izvučeni kod postavimo desno. Kako se izvučeni kod sastoji iz dva if bloka, razdvojimo ih, postavimo jedan pored drugog (donji desno od gornjeg). Zatim, desni kod uduplamo te ga postavimo desno od kopiranog koda. Imamo situaciju kao na slici



Rezonujemo analogno kao prilikom modifikacije koda za Sprite "desetine hiljada" u kod za Sprite "hiljade". Sada nas zanimaju stotice. Ako u promjenljivoj "poeni" bude sadržan petocifren broj, onda stotice (treću cifru) ispisujemo na treću lokaciju (-720,500), pri čemu broj stotica može biti jednak i 0. Ako u promjenljivoj poeni bude sadržan četverocifren broj, onda stotice (drugu cifru) ispisujemo na drugu lokaciju (-745,500), pri čemu broj stotica može biti jednak i 0. Napokon, ako u promjenljivoj "poeni" bude sadržan trocifren broj, onda stotice (prvu cifru) ispisujemo na prvu



Vidimo da onda kada je broj poena jednak 104 (ili 105 ili 132 ili ...), iza slike POENI stoji broj 1, što je korektno. Primijetimo da se na sredini ekrana vide i dvije nule, zbog toga što još uvijek nismo napravili kodove za Sprite - ove "desetice" i "jedinice".

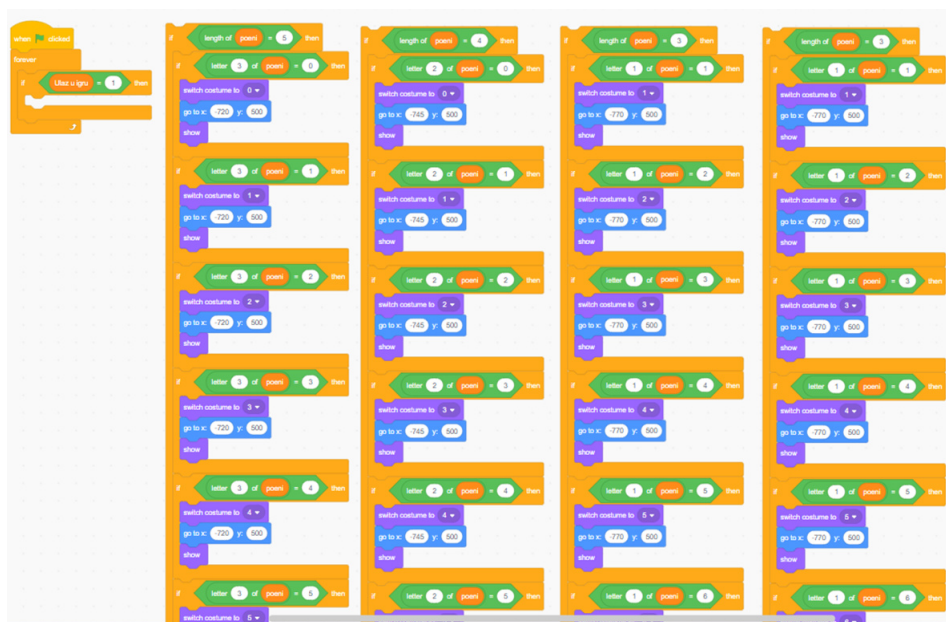
Sada je jasno kako napraviti kod za Sprite "desetice".

Postavimo se na kod za "stotice" i prenesemo ga na Sprite "desetice".

Postavimo se na Sprite "desetice".

Iz velikog koda izvučemo van kod sadržan u if uslovu: if Ulaz u igru = 1, i postavimo ga desno.

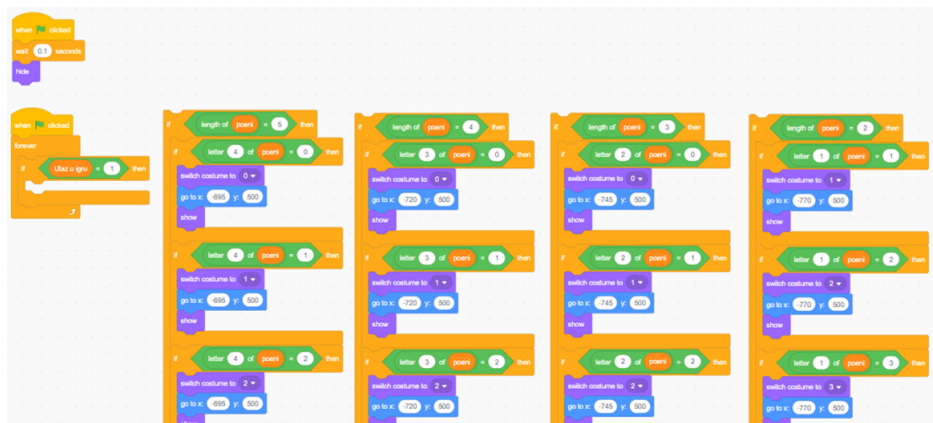
Izvučeni kod je sastavljen od 3 if bloka. Razdvojimo blokove tako da onaj sa if uslovom: if length of poeni = 4 stavimo desno, a onaj sa if uslovom: if length of poeni = 3 stavimo desno od njega. Kod koji se nalazi krajnje desno dupliciramo, i postavimo ga ponovo desno. Imamo situaciju kao na slici



Ovaj kod modifikujemo na način da on bude u stanju da ispisuje broj desetica. Modifikaciju provodimo analogno prethodnim modifikacijama.

Modifikovani kod treba da ima sljedeći oblik

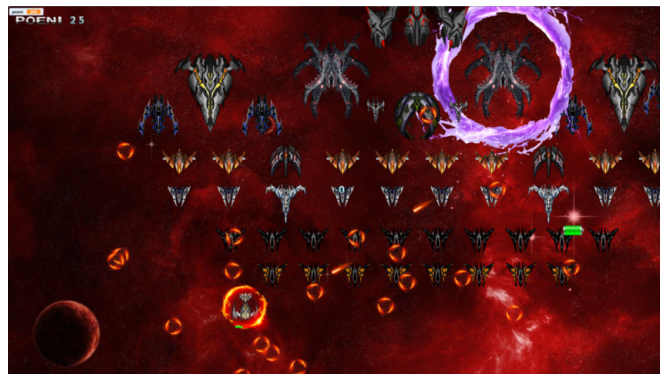
Dobijene kodove (desno od drugog) nadovežemo jedan na drugi (kao i u prethod-



nim slučajevima), te tako objedinjeni kod ubacimo unutar if uslova drugog koda. Sačuvamo izmjene i probamo igru



Tako, ako je broj poena postao jednak 18 (ili dovoljno 10), onda imamo ispis desno od slike POENI u obliku broja 1. Na sredini ekrana se nalazi još jedna nula jer još uvijek nismo napravili kod za Sprite "jedinice". Ako igramo dalje, dobijamo situaciju kao na slici



Ovdje je broj poena dostigao vrijednost 250, pa desno od slike POENI imamo ispisano 25. Naime, kako smo već rekli, još uvijek nismo napravili kod za Sprite "jedinice" pa je trenutni ispis korektan. I ovdje iz istih razloga na sredini ekrana još uvijek imamo prikazanu jednu nulu.

Konačno, možemo da pristupimo izradi koda za Sprite "jedinice".

Postavimo se na kod za Sprite "desetice" i penesemo ga na Sprite "jedinice".

Prvi kod



je isti kao i do sada pa mu ne posvećujemo posebnu pažnju.

Iz velikog koda izvučemo van kod sadržan u if uslovu: if Ulaz u igru = 1, i postavimo ga desno.

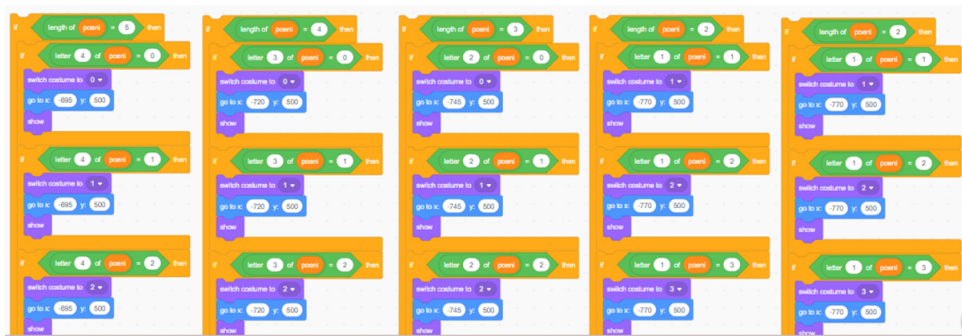
Kod koji ostaje lijevo je kod



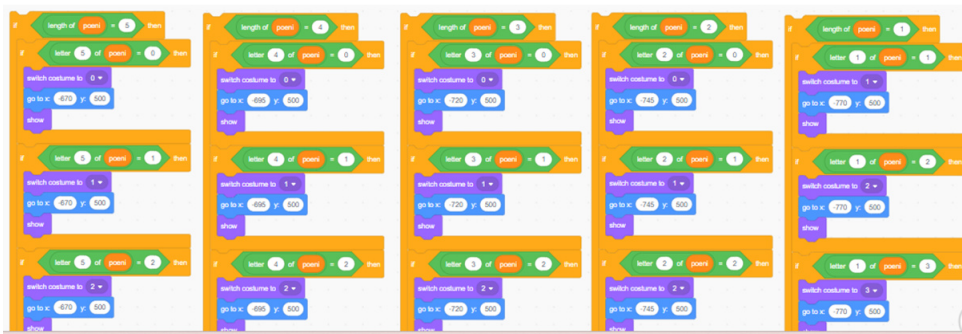


tako da ni njemu ne posvećujemo dodatnu pažnju.

Izvučeni kod je sastavljen od 4 if bloka. Razdvojimo blokove tako da onaj sa if uslovom: if length of poeni = 4 stavimo desno, onaj sa if uslovom: if length of poeni = 3 stavimo desno od njega, te onaj sa if uslovom: if length of poeni = 2, stavimo ponovo desno. Kod koji se nalazi krajnje desno dupliciramo, i postavimo ga ponovo desno. Imamo situaciju kao na slici



Kodove modifikujemo tako da kod koji generalno dobijemo na Sprite - u "jedinice" bude u stanju da ispisuje jedinice. U skladu sa prethodnim modifikacijama, modifikovani prethodni kod treba da ima sljedeći oblik



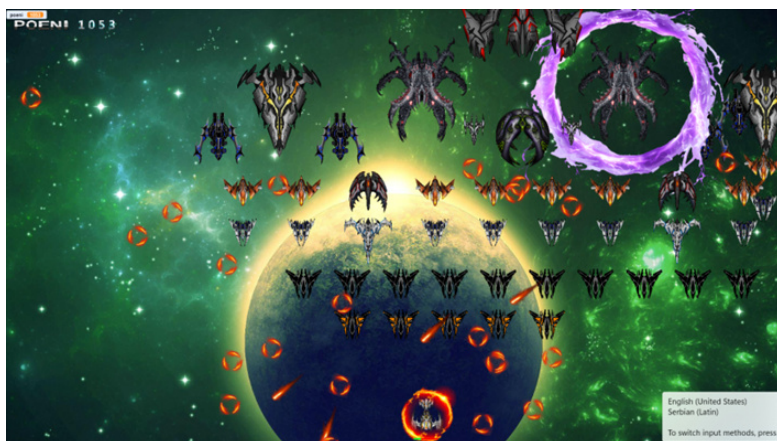
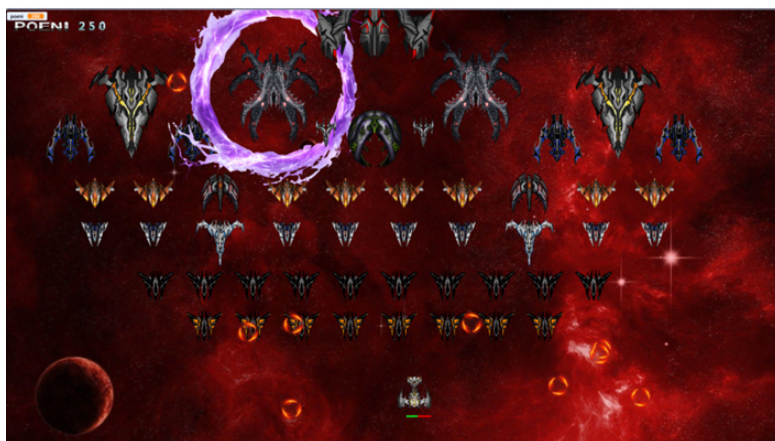
Ove kodove u skladu sa već uvedenom praksom nadovežemo jedan na drugi: peti na četvrti, pa dobijeni kod na treći, itd. Objedinjeni kod ubacimo u if uslov: if Ulaz u igru = 1 ranijeg koda



Dobijeni kod je tako kod za Sprite "jedinice".  
Sačuvamo izmjene i probamo igru







Iz navedenih primjera, od jedinica, preko desetica, stotica, hiljada, itd, možemo vidjeti da ovdje izvedeni kodovi za manipulaciju poenima zaista obavljaju svoj posao onako kako smo željeli.

Vidimo da potreba za promjenjivom "poeni" da bude prisutna na ekranu cijelo vrijeme više ne vrijedi. Možemo tako ukinuti kvakicu pored njenog imena, i sačuvati izmjene.

## 14.1 Riješeni zadaci

◇ **Zadatak 14.1.1** Napraviti jednostavnu platformsku igricu sa sljedećim elementima:

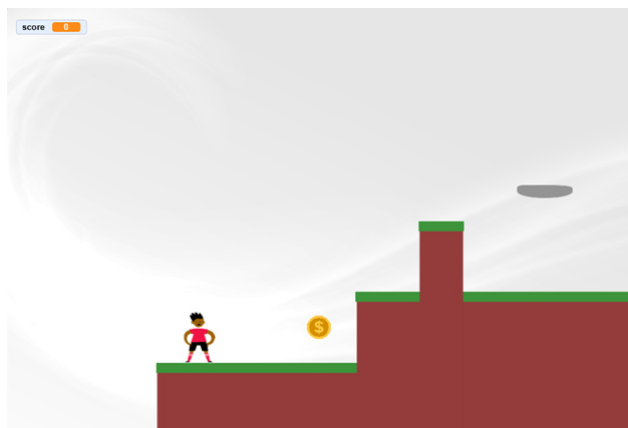
Ben (glavni lik) hoda lijevo (desno), skače, pada pod dejstvom gravitacije i ne može prolaziti kroz platforme,

platforme se stalno pojavljuju ispred igrača dok hoda (beskrajni nivo) a one koje ostaju iza se brišu,

kamera prati Bena pomijeranjem svijeta tako da Ben ostaje u vidnom polju,

ako Ben padne sa platforme na dno ekrana igra se resetuje,

novčići se povremeno pojavljuju iznad novih platformi te ih Ben skuplja i osvaja bodove



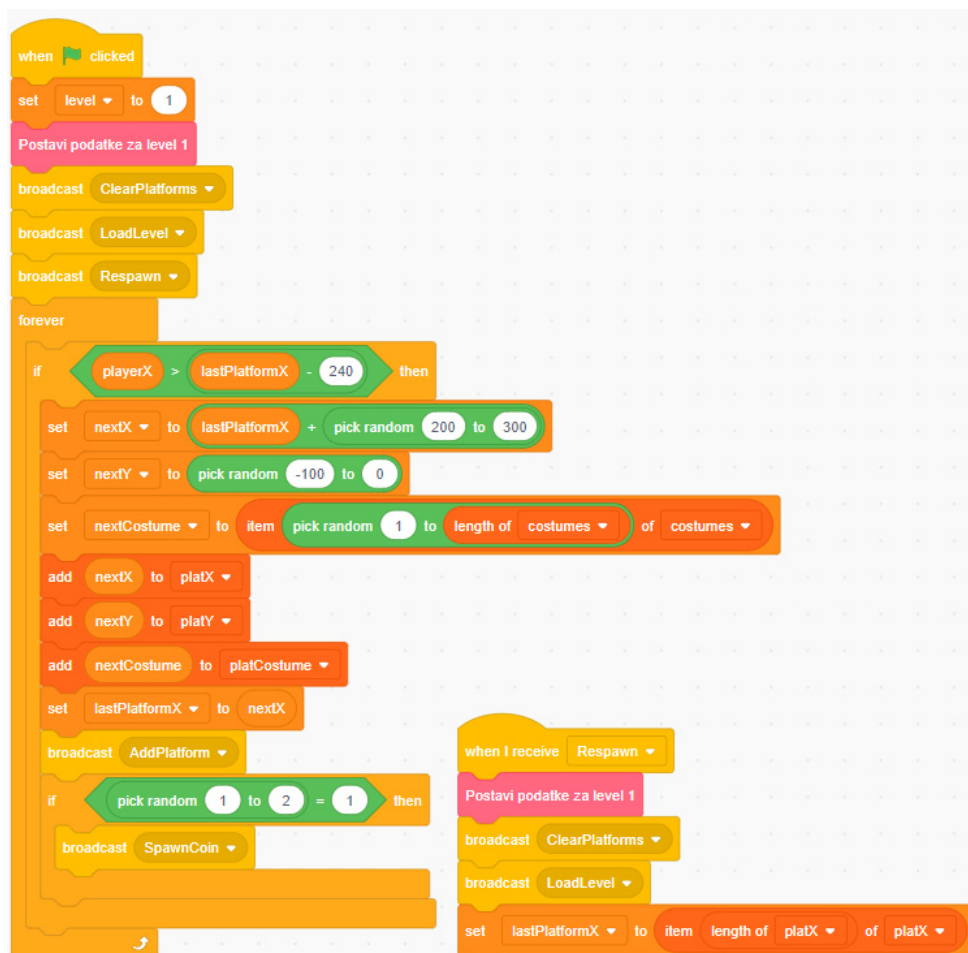
**Rješenje:** Dimenzije ekrana su postavljene na  $900 \times 600$ .

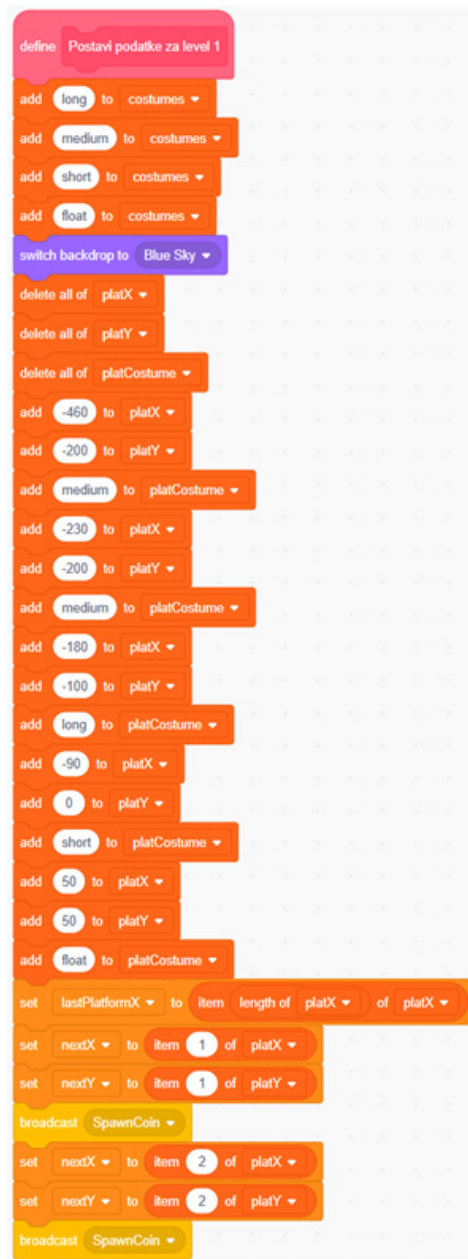
Kod za Stage (Pozornicu):

Prvo je potrebno napraviti sljedeće varijable i liste:

platX, platY, platCostume - liste globalnih koordinata i kostima platformi (npr. long/medium/short/float),

costumes - lista naziva kostima,  
 lastPlatformX -  $x$  koordinata kraja platforme koja je trenutno pozicionirana najviše udesno,  
 nextX, nextY, nextCostume - privremene vrijednosti za novu platformu,  
 sceneX - pomak kamere (svijeta),  
 playerX, playerY - globalne koordinate Bena (radi lakšeg praćenja),  
 level - broj nivoa (korisno za kasniju nadogradnju igrice),  
 score - broj prikupljenih novčića



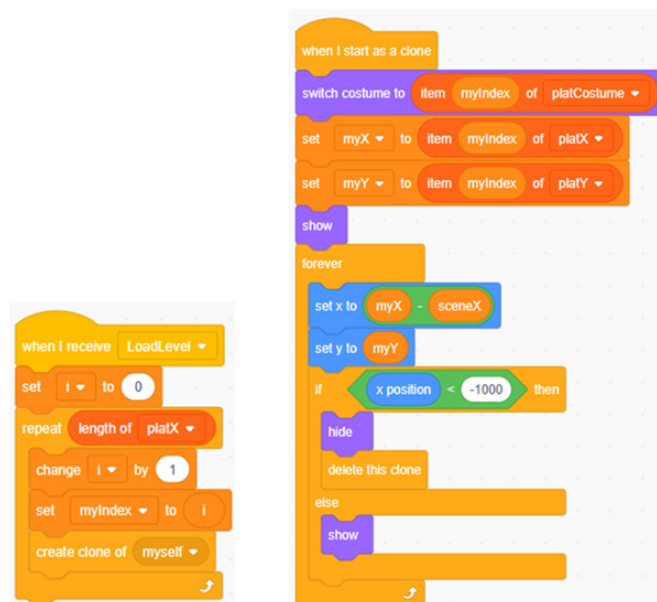
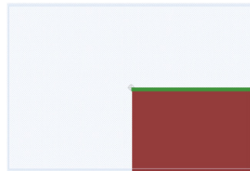


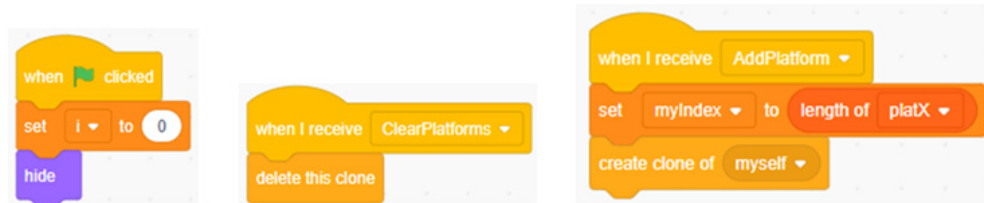
Kod za Sprite Platforma:

Prvo je potrebno kreirati varijable samo za ovaj Sprite (for this sprite only):

myIndex - indeks platforme u listama,  
myX, myY - koordinate platforme.

Platforma treba da ima 4 kostima: long, medium, short i float. Kreirati kostime koristeći Paint opciju i to tako da je gornji lijevi ugao platforme smješten u centar ekrana. Na slici je prikazan kostim long



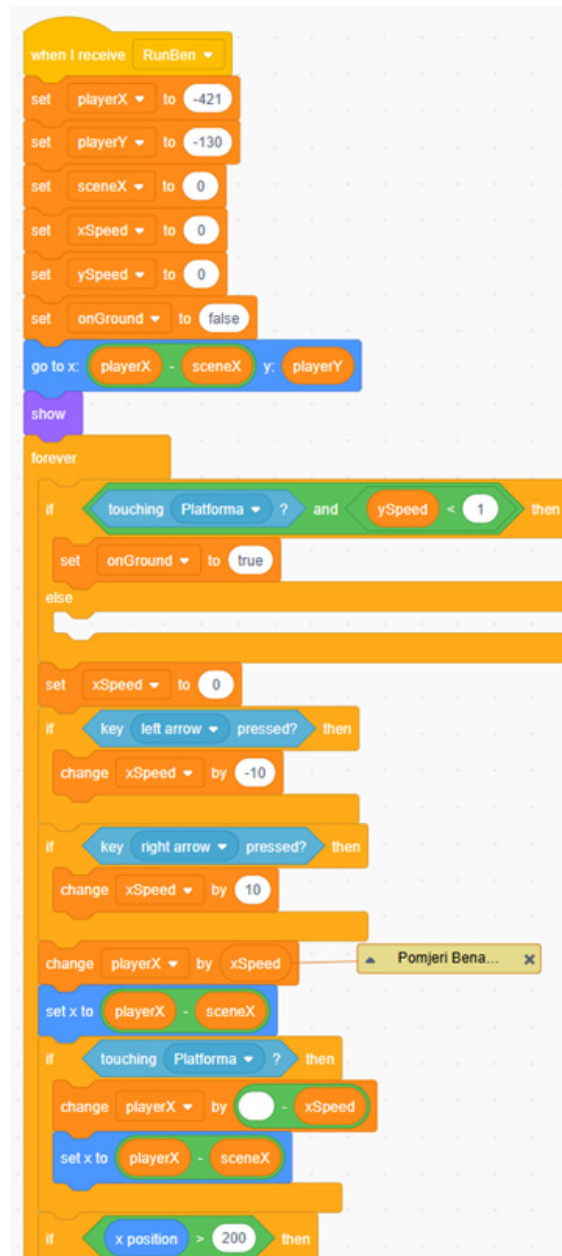


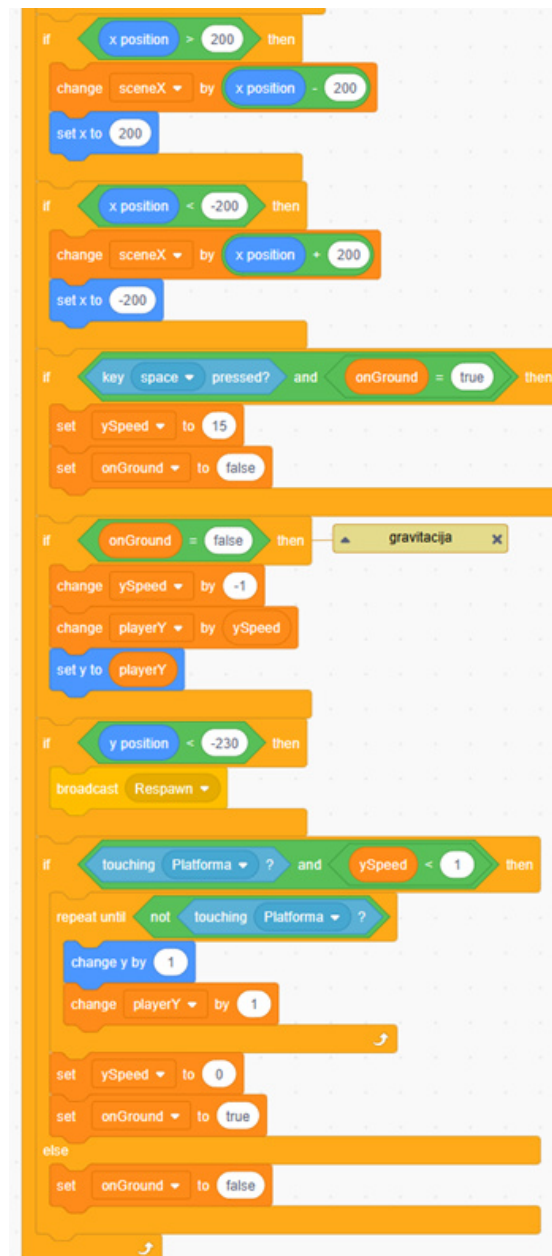
Kod za Sprite Ben (glavni lik):

Prvo je potrebno kreirati varijable samo za ovaj Sprite (for this sprite only):

xSpeed, ySpeed - horizontalna i vertikalna brzina Bena,  
onGround - logička varijabla (true/false), označava da li stoji na platformi



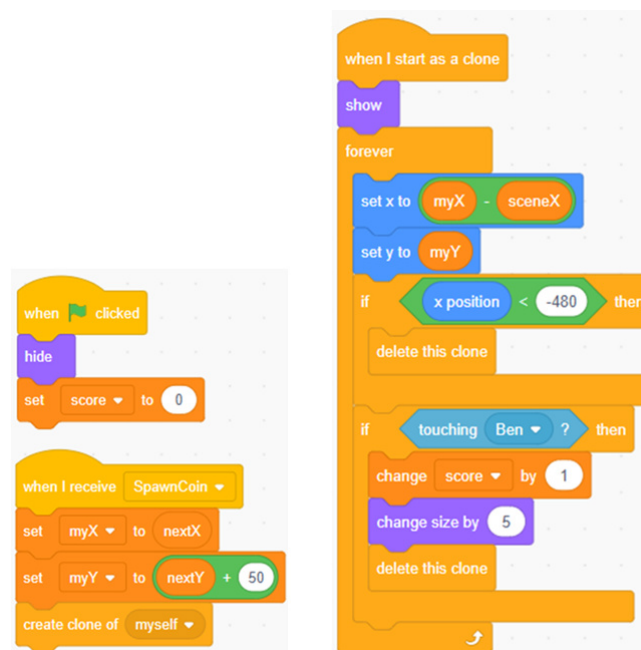






Kod za Sprite Coin:

I za ovaj Sprite su nam potrebne lokalne varijable myX i myY



## 14.2 Zadaci za vježbu

♣ **Zadatak 14.2.1** Unaprijediti igricu iz Zadatka 14.1.1 dodajući neprijatelje koji se kreću po platformi i koje treba preskakati. Kreirati Sprite "Enemy" koji se klonira iznad novih platformi i "patrolira" u razmaku leftBound i rightBound. Ako Ben dodirne neprijatelja sa strane izvršiti Respawn. Ako ga dodirne odozgo dok pada (npr. ySpeed < 0), neprijatelj treba da nestane a Ben da dobije bod(ove).

♣ **Zadatak 14.2.2** U Zadatku 14.1.1 dodati novu vrstu platforme koja se pomjera gore - dole ili lijevo - desno. Napraviti varijable (for this sprite only) npr. dx, dy, minY/maxY ili minX/maxX za klon platforme i u odgovarajućoj forever petlji pomi-

jerati myX/myY unutar granica (dolaskom do ruba promijeniti smjer). Ben treba prirodno da se prizemljuje na njih.

♣ **Zadatak 14.2.3** Kreirati u Zadatku 14.1.1 Sprite "PowerUp" (u obliku npr. zvjezdice) koji se povremeno "spawna" iznad platformi. U momentu kada Ben pokupi PowerUp, on 5 sekundi dobija povećani "jumpPower" (npr. sa 15 na 22). Potrebne varijable su: globalna "powerTimer" ili lokalna (za Bena) "superJumpActive", uz odbrojanje.

♣ **Zadatak 14.2.4** U Zadatku 14.1.1 dodati Sprite "Checkpoint" (zastavica). Onda kada ga (je) Ben dodirne, zapamtiti checkpointX i checkpointY (global). Na Respawn, umjesto početka levela, vratiti Bena na posljednji checkpoint i ne resetovati platforme, odnosno, resetovati ih samo ako nije aktiviran niti jedan checkpoint.

♣ **Zadatak 14.2.5** U Zadatku 14.1.1 dodati Sprite "Hazard" (npr. trnje ili lava) koji se spawn-a na nekim platformama. Ako Ben dodirne hazard, odmah je potrebno izvršiti njegov Respawn (na ovaj način dodajemo potencijalnu koliziju mimo standardnih platformi i forsiramo igrača na pažljivije kretanje).

♣ **Zadatak 14.2.6** Dodati u Zadatak 14.1.1 Sprite "BackgroundLayer" (ili više njih) koji se kreću sporije od sceneX (npr. postaviti  $x$  na  $(-sceneX * 0.3)$ ), da dobijemo utisak dubine. Ovaj aspekt ne utiče na gameplay, ali podiže kvalitet prezentacije.

♣ **Zadatak 14.2.7** U Zadatku 14.1.1 iskoristiti level varijablu na način da nakon određenog playerX ili određenog broja prikupljenih novčića, broadcast NextLevel → Stage očisti sve i pozove "Postavi podatke za level 2" (uz drugačiji raspored/visine), a igra nastavi sa radom kao i prije (beskonačno generisanje).

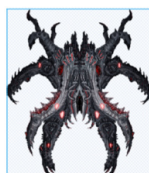
## 15 TurboWarp Packager i exe fajlovi

Pozabavimo se sada bojenjem nekih od objekata koje smo uključili u igru. Ovdje prvenstveno mislimo na neprijateljske letjelice a eventualno i na druge objekte: oružja, energije, itd. Naime, nekada može biti potrebno uskladiti nijanse letjelica sa podlogom nivoa na kojem se nalazimo. Može to biti i samo stvar ukusa. Vidjet ćemo da neki objekti jesu a drugi isto tako nisu zahvalni za bojenje.

Posmatrajmo npr. prvi nivo igre. Startna ploča ima oblik



Jasno nam je da su kreatori, tj. grafički dizajneri neprijateljskih letjelica koje koristimo, na neke od letjelica uložili mnogo više truda u odnosu na neke druge od njih. Jasno je da su neke letjelice urađene mnogo detaljnije od nekih drugih. Bojenje takvih objekata je onda otprilike direktno proporcionalno uloženom trudu, u smislu da je mnogo lakše podesiti boju objekta koji je jednostavnije urađen. Neki od objekata tipa Gobo4 i Gobo6



su doslovno filigranski izvedeni. Osjećaj za bojenje jednog ovakvog objekta izgleda vrlo nezahvalan. Puno je sitnih grafičkih detalja koji bi zahtijevali bojenje, puno je nijansi koje su prisutne na ovakvim objektima. Dakle, mnogo uloženog truda za izradu jednog ovakvog objekta implicira mnogo dodatnog potencijalnog truda za njegovo bojenje. Postavlja se pitanje kako bojiti jedan ovakav objekat. Jasno je da svaki objekat možemo pokušati bojiti ručno, tačku po tačku. Takođe je jasno da nas ovakav pristup prilikom razmatranja grafički složenijih objekata neće daleko dovesti. U takvim situacijama je ipak lakše pokušati napraviti određeni kod koji će bojenje obaviti za nas. Prilikom jednog takvog automatskog bojenja bit ćemo u situaciji da vidimo da li je objekat koji posmatramo uopšte pogodan za bojenje.

Vidjet ćemo na konkretnim primjerima da neki od objekata sa kojima manipuliramo uopšte nisu pogodni za bojenje, tj. da njihovo bojenje ne polučuje neki konkretni rezultat. Takva osobina objekata ne zavisi od nas nego od njihovih kreatora koji su objekte i izgradili na taj način da ona forma boje koju su oni postavili bude za posmatrani objekat ujedno i najzahvalnija. Ipak, na nama je, u slučaju da to želimo, da objekte sa kojima imamo dodira pokušamo obojiti. Riječ bojenje ne moramo shvatiti literalno nego u smislu variranja nijanse boje koja je pristutna na objektu. Na osnovu testiranja koje ćemo provesti bit ćemo u poziciji da za neki od objekata kažemo da je zahvalan za bojenje, za neki drugi ćemo pak reći da je najbolje da bude ostavljen onakav kakav jeste, tj. sa svojim originalnim nijansama boje. Sa bojenjem generalno ne treba pretjerati. Naime, ne moraju svi objekti sa kojima radimo biti u nekakvoj istoj nijansi sa podlogom npr, ili sa nekim drugim objektom po kojem se ravnamo. U takvim situacijama može doći do pojave monotonije. S druge strane, ako neki objekat inicijalno odudara svojom bojom od ambijenta u kojem se nalazi, onda je na nama da pokušamo da nađemo najbolji način da taj objekat vizuelno uklopimo u okruženje u kojem se on nalazi.

Često, nije poenta ni u samome bojenju. Naime, neki od objekata svojom konstrukcijom odudaraju od okruženja u koje smo ga stavili. Teško je zamisliti da Sprite jednoroga ima bilo šta da traži u kosmičkoj ratnoj igri (ili u nekom kontekstu ipak ima). Želimo da kažemo da mnogo toga zavisi od kreatora objekata, autora igre, priče, itd.

U našem slučaju, dobra okolnost je ta što od 50 neprijateljskih letjelica sa kojima raspolažemo, mnoge od njih se ponavljaju. Tako, dovoljno je da onda obojimo jednu od njih a na ostale koje su istog tipa, zalijepimo nijansu boje koju smo našli za adekvatnu za prvu od njih. Neprijateljske letjelice sa kojima raspolažemo su:



Vidimo da raspolažemo sa ukupno 12 različitih objekata.

Dovoljno je onda pokušati poraditi na njihovom bojenju. Naime, dovoljno je da od svakog od navedenih tipova, uzmemo samo prvi primjerak, i pozabavimo se njegovim

bojenjem.

Riječ je o letjelicama: Gobo1, Gobo2, Gobo4, Gobo5, Gobo10, Gobo11, Gobo13, Gobo15, Gobo23, Gobo25, Gobo33 i Gobo43.

Da bi mogli obaviti željeno razmatranje, trebat će nam jedna nova promjenjiva. Dat ćemo joj ime BOJA. Definišimo onda ovu promjenjivu i ostavimo je aktivnom na ekranu

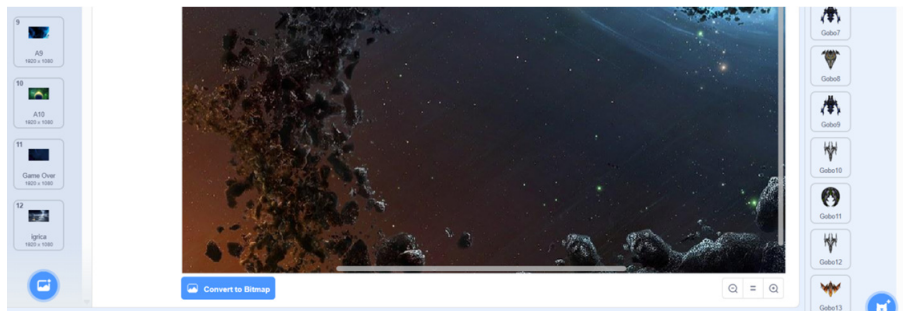


Ako odemo na podloge, onda je lahko vidjeti i podsjetiti se da osim inicijalne ulazne podloge "igrica", imamo 10 podloga koje koristimo kao podloge nivoa koji slijede. Ove podloge smo zvali: A1, A2, ..., A10. Napokon, imamo podlogu koju smo rezervisali za slučaj da energija naše letjelice M14 bude svedena na nulu, a tu podlogu smo zvali (i dalje zovemo) "Game Over". Nas trenutno zanimaju podloge: A1 - A10. Naime, ideja je ta da namjeravamo da mijenjamo boje neprijateljskih letjelica (tamo gdje zaista to želimo), u zavisnosti od aktuelne podloge (dakle neke od pomenutih podloga: A1 - A10).

Tako, s obzirom da imamo 10 podloga od interesa: A1 - A10, i 50 neprijateljskih letjelica, napraviti ćemo univerzalni kod kojeg ćemo moći koristiti, a ne moramo ga koristiti ako ne želimo, koji će da bude jedinstven za sve letjelice, bit će pristutan na kodu svake od 50 letjelica, i na određeni način će predstavljati formu, tj. tabelu za popunjavanje odgovarajućim podacima koji će da regulišu boju posmatrane letjelice na nivou na kojem se nalazimo.

Prije nego pristupimo izradi pomenutog koda, potrebno ja da napravimo odgovarajuću pripremu. Prvo, idemo na pozadine (Backdrops), pa na Backdrops. Na lijevoj strani, na listi pozadina, pozadina koja se fizički nalazi na prvom mjestu je pozadina "igrica". Slijede je pozadine: A1 - A10, te konačno pozadina "Game Over". Svaka pozadina ima svoj pridruženi redni broj. Tako, u ovakvoj postavci, pozadina "igrica" ima redni broj 1, pozadina A1 redni broj 2, itd, pozadina A10 redni broj 11, te pozadina "Game Over" redni broj 12. Ovakav raspored nam ne odgovara, jer u nastavku želimo da pozadina A1 ima redni broj 1, itd, da pozadina A10 ima redni broj 10. Zbog toga ćemo uraditi preraspored pozadina, tako što ćemo pozadinu "igrica" fizički premjestiti

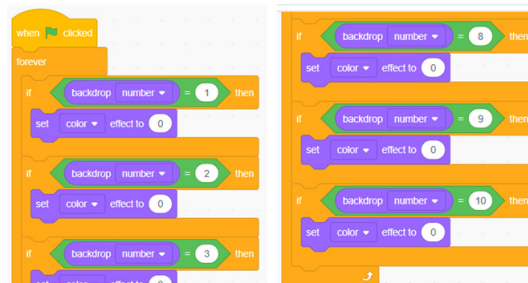
"prema dole" tako da se ona nađe iza pozadine "Game Over". Uradimo to. Imamo situaciju kao na slici



Sačuvamo izmjene.

Bez dodatnog komplikovanja, pristupamo kodiranju.

Idemo na Gobo1, pa na Code, te unosimo sljedeći kod



Na ovim slikama smo prikazali početak i kraj koda. Dakle, unutar forever petlje imamo 10 if uslova.

Objasnimo šta radi ovaj kod.

Prvo, ovaj kod smo unijeli u prostor za kodove letjelice Gobo1. Zbog toga ovaj kod treba da se odnosi na letjelicu Gobo1. Pošto je u kodu jasno riječ o bojama, to se onda ovaj kod na određeni način odnosi na boju letjelice Gobo1.

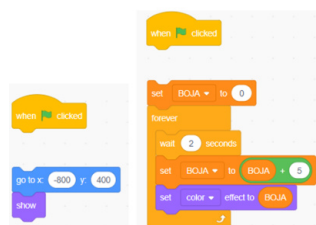
Pošto imamo za sada 10 podloga, to u navedenom kodu imamo 10 if uslova.

Forever petlju koristimo kao i do sada za "lov" na ciljani događaj.

Možemo se fokusirati na prvi if uslov: ako je broj pozadine jednak 1, onda postavljamo efekat za boju na vrijednost 0. Možemo se pitati: efekat za boju čega? Riječ je o efektu za boju objekta na kojem smo pozicionirani, a to je kako znamo letjelica Gobo1. Na drugi način formulirano: ako se nalazimo na prvoj podlozi, tj. podlozi koju smo nazvali A1, onda postavljamo efekat boje za Gobo1 da ima vrijednost 0. Ovo ne znači ništa drugo do to, da (za sada), želimo da boju za Gobo1 ostavimo nepromijenjenom. Ukoliko želimo da promijenimo boju za Gobo1, onda bismo umjesto vrijednosti 0 pisali neku drugu vrijednost, npr: 5, 55, 125, i sl. Tako, za sada, na startu, pošto ne znamo koju boju želimo da imamo za Gobo1 na podlogama A1 - A10, sve vrijednosti za efekat boje u prethodnom kodu smo postavili na vrijednost 0. Naime, kako smo gore istakli, navedeni kod je ustvari forma (tabela) u koju možemo (a i ne moramo) unijeti određene vrijednosti za efekat boje. Ako za neku od podloga promijenimo vrijednost za efekat boje, onda će se boja Gobo - a za tu podlogu shodno i promijeniti. Inače, tamo gdje ostavimo da je vrijednost efekta boje jednaka 0, Gobo će da zadrži svoje standardne fabričke nijanse boje. O tome kako doći do odgovarajućih nijansi za pojedine Gobo - e, a opet za pojedine podloge, vidjet ćemo u nastavku. Za sada nam je jasno da upravo uneseni kod treba da imaju svi Gobo - i bez izuzetka, pa ćemo posljednji kod za Gobo1, kopirati i zalijepiti na sve ostale Gobo - e. Uradimo to i sačuvajmo izmjene (kodove možemo fizički postaviti "ispod" ranijih kodova koji se nalaze na Gobo - ima).

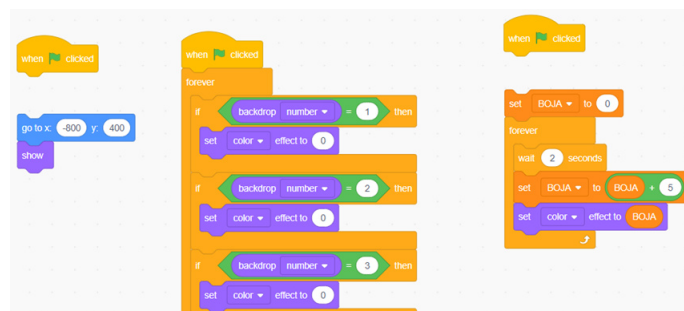
Sada smo u poziciji da pristupimo nijansiranju boja Gobo - a u zavisnosti od podloge koju razmatramo. Priča je analogna za sve podloge, pa će biti dovoljno da se fokusiramo na prvu podlogu, tj. na podlogu A1. Osim toga, kako smo već istakli iznad, dovoljno je da se bavimo Gobo - ima: Gobo1, Gobo2, Gobo4, Gobo5, Gobo10, Gobo11, Gobo13, Gobo15, Gobo23, Gobo25, Gobo33, Gobo43 jer su svi ostali Gobo - i kopije upravo pobrojanih Gobo - a.

Na upravo pobrojane Gobo - e ćemo unijeti kod koji slijedi. Dovoljno je da ga unesemo na Code za Gobo1, a onda da ga zalijepimo na Gobo2, Gobo4, Gobo5, Gobo10, itd, do Gobo 43. Riječ je o sljedećem kodu (kodovima)





Unesene kodove možemo rasporediti tako da zajedno sa prethodno unijetim kodom čine na neki način cjelinu koja će da reguliše boje za Gobo1, tako da njihov stvarni raspored može biti sljedeći



Primijetimo ovdje da smo naredbe o pokretanju programa pomoću zelene zastavice odvojili od tijela koda u slučaju prvog i trećeg koda.

Naime, ovo nije slučajno. I prvi i treći kod će da budu kodovi pomoćnog karaktera koji neće cijelo vrijeme biti aktivni i uključeni, nego ćemo ih mi, u zavisnosti od naših potreba, povremeno aktivirati (dok ne dobijemo nijansu boje koju želimo), a onda ponovo isključiti (rasklopiti). Naime, svi kodovi kod kojih je zaglavlje odvojeno od tijela petlje su neaktivni kodovi. Oni se ne izvršavaju, i kao takvi ne predstavljaju nikakav problem niti smetnju izvršavanju ostalih kodova. Praksa posjedovanja ovakvih kodova je dobra, jer često se dešava da korisnik treba da u pripremi ima neki kod koji će možda koristiti a možda i ne, a kojeg u slučaju potrebe može vrlo jednostavno aktivirati.

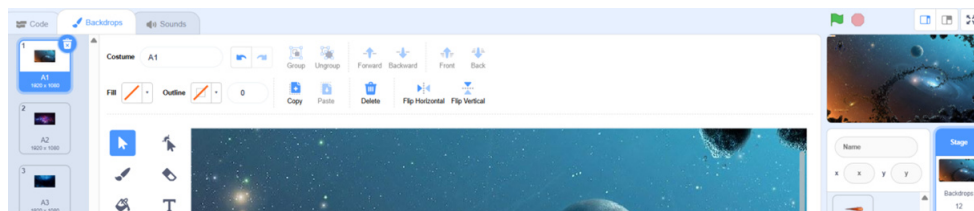
Kodove koje smo upravo unijeli (dva rasklopljena koda) kopiramo i zalijepimo na osnovne Gobo - e (takođe u rasklopljenom obliku): Gobo2, Gobo4, Gobo5, Gobo10, Gobo11, Gobo13, Gobo15, Gobo23, Gobo25, Gobo33, Gobo43. Možemo rasklopljene kodove na momenat sklopiti, pa ih zalijepiti na navedene Gobo - e, a ona ih opet tamo rasklopiti radi jednostavnosti. Uradimo upravo rečeno i sačuvajmo izmjene.

Šta rade rasklopljeni kodovi ?

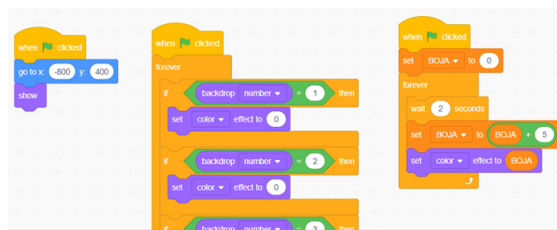
Objasnimo to na konkretnom primjeru.

Dovoljno je da ponovo posmatramo Gobo1.

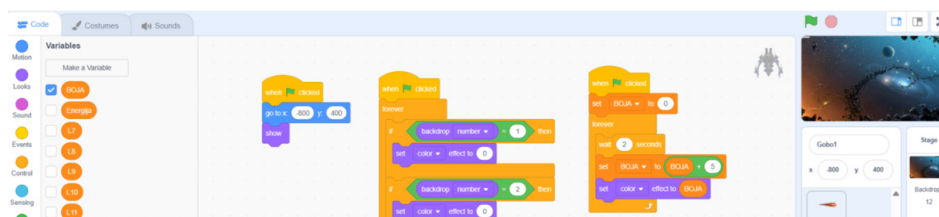
Prvo, bez pokretanja igre, idemo na podloge (Backdrops), pa na Backdrops, te se pozicioniramo na podlogu A1 (želimo da ova podloga bude prikazana na ploči)



Zatim se pozicioniramo na Gobo1, pa na Code, te sklopimo ranije posmatrana dva rasklopljena koda. Treba dakle da imamo situaciju kao na slici

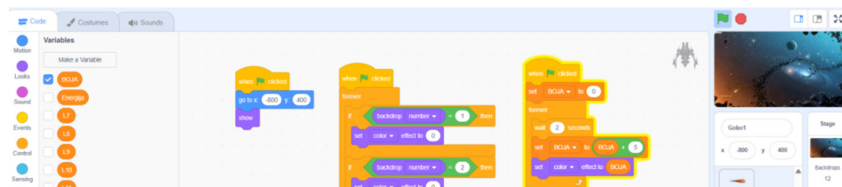


I dalje ne pokrećući igru, kliknemo na zelenu zastavicu prvog od posljednja tri koda. Želimo na taj način da izvršimo samo taj kod (neovisno od ostalih kodova). Ako uradimo kako smo upravo rekli, vidjet ćemo kratki blink kliknutog koda, te pojavu letjelice Gobo1 u gornjem lijevom uglu, preciznije na poziciji (-800,400)

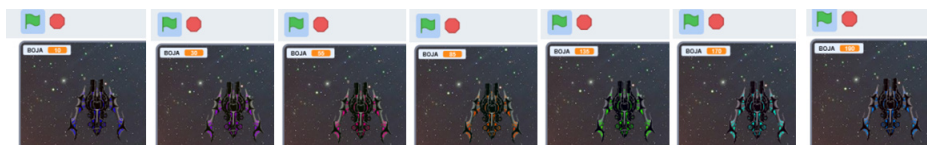


Iznudili smo na taj način (neovisno od svih ostalih kodova) pojavu letjelice Gobo1 na lokaciji (-800,400). Naime, želja nam je da eventualno mijenjamo boju ove (ali i ostalih letjelica), pa je jasno potrebno da je učinimo sebi vidljivom da mi mogli dalje sa njom da manipuliramo. Za sada boja Gobo1 je originalna, tj. nepromijenjena.

Kliknimo sada na zelenu zastavicu trećeg koda (prethodna slika)



Želimo dakle da i ovaj kod izvršimo bez da je igra pokrenuta, tj. bez da se svi dosadašnji kodovi izvršavaju. Uradimo to i maksimizirajmo ploču igre. Svjedoci smo dešavanjima prikazanim na sljedećim slikama (zaustavimo izvršavanje koda nakon izvjesnog vremena klikom na crveno dugme)



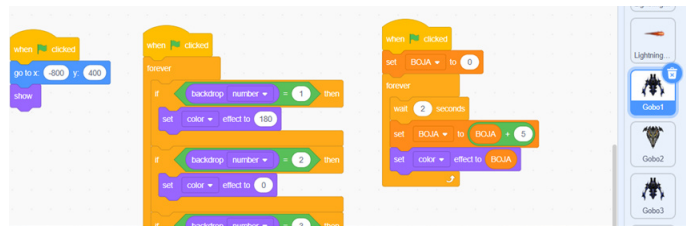
Sada nam je jasno šta zapravo i radi treći kod.

Vrijednost promjenjive BOJA inicijalno stavljamo da bude jednaka 0. Zatim, for-ever petljom mijenjamo tu vrijednost povećavajući je svakom iteracijom za 5 (uz vrijeme čekanja od dvije sekunde - dovoljno vremena da se bez žurbe uoči nijansa boje koja nam odgovara), te mijenjamo efekat boje (za Gobo1) sa prethodne vrijednosti na vrijednost BOJA. Na ovaj način, svakom iteracijom, dobijamo novu nijansu boje, tj. boje nastale promjenom za 5 podioka, uz vrijeme čekanja od dvije sekunde. Kako je promjenjiva BOJA prikazana na ploči, a isto tako i Gobo1, to običnom opservacijom u odnosu na Gobo1 uočavamo koja njegova nijansa nam odgovara, za tu nijansu pamtimo broj prikazan u varijabli BOJA, te taj broj evidentiramo u tabelu pripremljenu ranije (srednji kod - prva if naredba).

Na ovaj način smo istovremeno opisali i šta radi treći kod, ali i dali metod za evidentiranje željenih nijansi boja neprijateljskih letjelica.

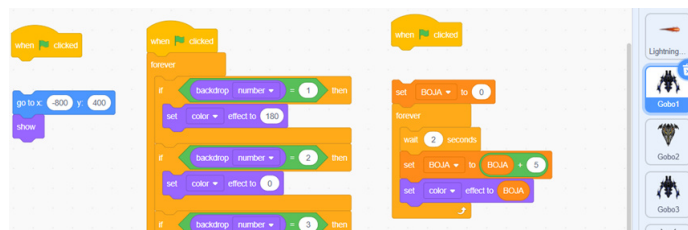
Odgovarajuća nijansa za Gobo1 za podlogu A1 je ona koja odgovara vrijednosti 180 promjenjive BOJA. Evidentiramo ovu vrijednost u tabelu datu drugim kodom kao

na slici



Pošto su letjelice: Gobo3, Gobo7, Gobo9 istog oblika kao Gobo1, to onda i u njihove tabele na odgovarajuće mjesto evidentiramo vrijednost 180.

Sada, pozicioniramo se na Gobo1, na njegov kod, te upravo korišteni prvi i treći kod rasklopimo (jer nam više ne treba)



Pokrenemo igru, zaustavimo je, i sačuvamo izmjene.

Ako ponovo pokrenemo igru vidjet ćemo da su na prvom nivou (pozadina A1), letjelice Gobo1, Gobo3, Gobo7 i Gobo 9 zaista onakve boje kakvu smo upravo odrediti. Ako izigramo prvi nivo, onda ćemo isto tako vidjeti da na drugom nivou (pozadina A2) upravo pobrojane letjelice imaju svoju originalnu nijansu boje (jer tu originalnu nijansu na tom nivou još uvijek nismo mijenjali - odgovarajuće vrijednosti u tabeli, tj. srednjem kodu su jednake 0).

Sada je jasno kako nastavljamo sa radom.

Idemo na podloge (Backdrops) pa na Backdrops, te se pozicioniramo na podlogu A1.

Postavimo se zatim na sljedeći tip letjelice. To je Gobo2.

Idemo na kod, i rasklopljene kodove uklopimo (prvi i treći kod).

Kliknemo na zelenu zastavicu prvog uklopljenog koda da dobijemo pozicioniranje Gobo2 na (-800,400) na ploči.

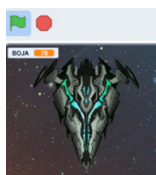
Zatim, kliknemo na zelenu zastavicu trećeg uklopljenog koda da pokrenemo traženje boje, pa maksimiziramo ekran ploče.

Pratimo promjenu boje direktnom opservacijom Gobo2.

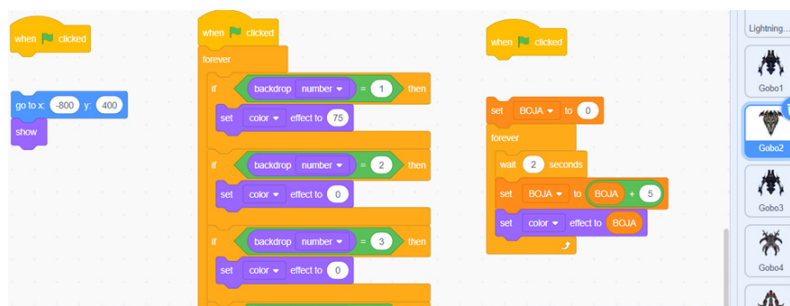
U momentu kada zaključimo da je boja za Gobo2 adekvatna, upamtimo vrijednost promjenjive BOJA u polju iznad, i zaustavimo igru klikom na crveni krug.

Pomenutu vrijednost upišemo na odgovarajuće mjesto u tabeli drugog koda (prva if naredba).

Nije teško provjeriti da je vrijednost promjenjive BOJA koja odgovara Gobo2 i podlozi A1 ustvari vrijednost 70 ili 75 (neka to bude vrijednost 75)



Tako, nakon upisivanja vrijednosti 75 u tabelu, rasklopimo ponovo kodove koje smo prethodno sklopili. Treba da imamo situaciju kao na slici



Pošto je Gobo8 istog tipa kao Gobo2, to onda u tabelu za Gobo8 upišemo takođe vrijednost 75 na odgovarajuće mjesto.

Odgovarajuće vrijednosti koje bi na opisani način dobili za ostale Gobo - e ćemo navesti:

| Gobo                                            | Vrijednost |
|-------------------------------------------------|------------|
| Gobo4, Gobo6                                    | 190        |
| Gobo5                                           | 100        |
| Gobo10, Gobo12                                  | 0          |
| Gobo11                                          | 60         |
| Gobo13, Gobo14, Gobo16 – Gobo19, Gobo21, Gobo22 | 95         |
| Gobo15, Gobo20                                  | 90         |
| Gobo23, Gobo24, Gobo26 – Gobo29, Gobo31, Gobo32 | 180        |
| Gobo25, Gobo30                                  | 180        |
| Gobo33 – Gobo42                                 | 0          |
| Gobo43 – Gobo50                                 | 90         |

Napravimo navedene izmjene.

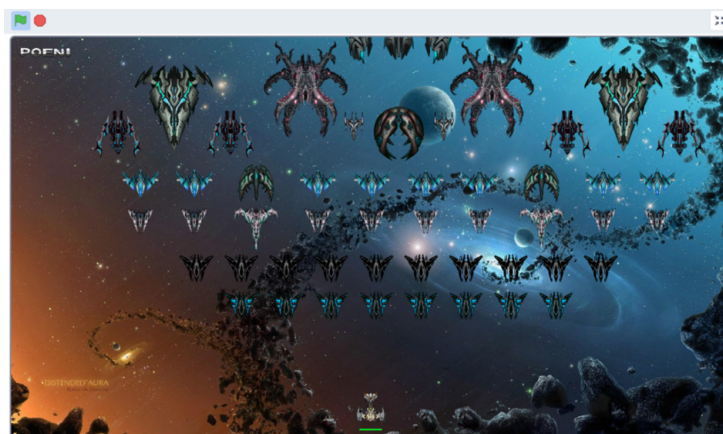
Na kraju, ne trebamo zaboraviti rasklopiti kodove koje smo prilikom pretrage sklopili (prvi i treći kod).

Pošto nam promjenjiva BOJA više nije potreba na ekranu, ukinemo je, tj. ukinemo kvakicu pored njenog imena.

Pokrenemo pa zaustavimo igru.

Sačuvamo izmjene.

U odnosu na polaznu sliku igre koju smo naveli (bez dodatnog farbanja), sada prvi nivo igre izgleda ovako



Boje su dakle usklađene na određeni način sa podlogom. Nijansa kojoj smo težili je plava. Vidljivo je da su neki brodovi na sebi svojevrsan način imuni na izmjene boje, ili su te izmjene na njima moguće samo djelimično. Kod nekih brodova izmjenama ne dobijamo ništa. Obično je riječ o bijelim brodovima. Kao što smo nijanse usklađivali u odnosu na plavi dio podloge, mogli smo ih porediti i sa smeđim dijelom. Jednostavno, stvar je ukusa.

Ako uporedimo polaznu sliku sa novo - nastalom slikom, možda će nam se učiniti da je polazna varijanta podešena sa više ukusa. Možda ne. Ponovo, stvar je ukusa. Kako god, mi smo u ovome dijelu prezentirali metod kojim ako želimo možemo mijenjati nijanse objekata sa kojima radimo. Ovdje smo se fokusirali na neprijateljske brodove, ali je jasno da smo odgovarajuće kodove mogli postaviti i na sve vrste oružja, na energije, na našu letjelicu M14, itd. Na način kako smo radili sa podlogom A1, isto tako bi radili i sa ostalim podlogama. Dobijene vrijednosti promjenjive BOJA bi upisivali unutar drugog if uslova tabele nijansi boja.

Pozabavimo se sada pravljenjem izvršne, tj. exe verzije našeg dokumenta.

Znamo da je naš dokument, tj. dokument kojeg pravimo u sklopu okruženja Eticques tipa sb3. Tokom izrade igre, tj. pomenutog dokumenta došli smo do verzije 13 posmatrano u odnosu na polaznu verziju. Tako, trenutni naziv našeg dokumenta je GalVer13.sb3



GalVer13.sb3

Možemo zadržati ovaj naziv.

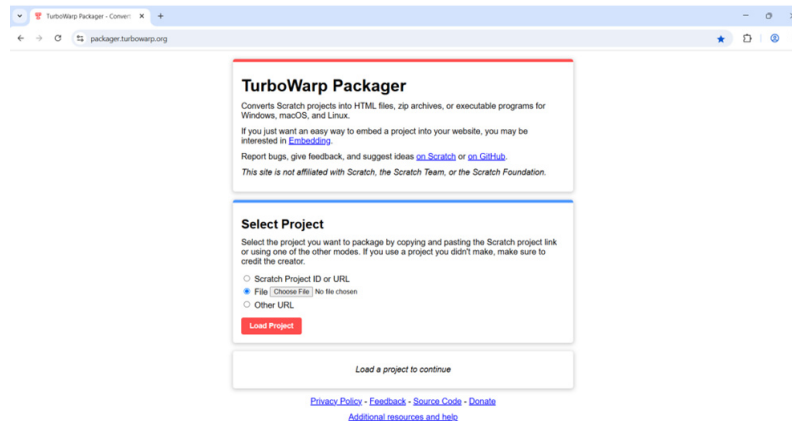
Da bi pravili exe verziju igre GalVer13.sb3, odnosno exe verziju dokumenta općenito kreiranog u Eticques - u, koristimo program TurboWarp Packager

<https://packager.turbowarp.org/>

Možemo upotrijebiti dati link.

Riječ je o okruženju datom na slici

## 15 TURBOWARP PACKAGER I EXE FAJLOVI



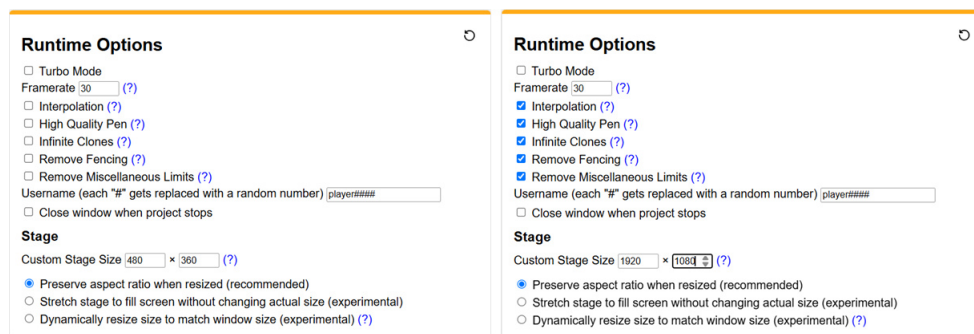
Program TurboWarp Packager (kraće TWP) koristimo za konvertovanje sb3 fajlova u: HTML fajlove, zip arhive, te izvršne programe namijenjene za Windows, macOS i Linux. Nas, očekivano, zanima konvertovanje sb3 fajla u exe file pogodan za korištenje na Windows - u.

Sa slike je jasno da prvo moramo load - ovati, tj. upload - ovati, ili odabrati file koji želimo da konvertujemo. Idemo na Choose File, pa odaberemo naš dokument GalVer13.sb3 sa lokacije na koju smo ga smjestili (pretpostavljamo da smo u nekom od pretraživača već pokrenuli TWP).

Nakon kraćeg load - a, otvaraju se dodatni elementi okruženja TWP.

Kroz elemente od interesa ćemo proći korak po korak.

Prvi prozor možemo popuniti kao na slici





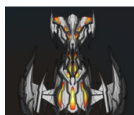
Turbo mod nismo koristili jer bi se u suprotnom u sklopu exe verzije koju pravimo, elementi igre kretali znatno brže od kretanja na koje smo navikli prilikom izrade sb3 fajla u Eticques - u. Ukoliko bilo kome ovakav prijelaz ne predstavlja poteškoću, može se uključiti ovaj mod. Framerate ili FPS (frames per second) ostavljamo na 30. Umjesto 30 se može koristiti bilo koja vrijednost između 1 i 250. Obično se koristi vrijednost 60. Ipak, velika većina projekata ne radi ispravno sa izmijenjenom vrijednošću za FPS. Tako, ako posmatramo npr. jednostavan kod beskonačnog kretanja Sprite - a za po jedan korak



onda će ovaj kod kada radi na 30 FPS (Eticques obično radi na 30 FPS) da se pokrene 30 puta u sekundi. Drugim riječima, Sprite će se kretati 30 koraka u sekundi. Jasno, onda, ako promijenimo vrijednost na 60 FPS, navedeni kod će se izvršiti 60 puta u sekundi, što znači da će u tom slučaju Sprite preći duplo više koraka u sekundi. Za visoko - kvalitetnu olovku, tj. High Quality Pen (HQP) je važno napomenuti da njeno uključivanje može znatno smanjiti performanse igre, pogotovo ako se igra pokreće u Fullscreen varijanti. Naime, tada se broj piksela koji se moraju iscrtati drastično povećava. Takođe, neki projekti zasigurno hoće, dok drugi i neće bolje izgledati sa uključenim HQP - om. U izradi igre (teoretski dio) nismo koristili klonove. Ipak, važno je napomenuti da je limit broja klonova u Eticques - u jednak 300. Ako uključimo opciju Infinite Clones (IC), onda time onemogućavamo postavljeno ograničenje dato brojem 300. Naime, neki projekti pretpostavljaju da će Eticques onemogućiti stvaranje dodatnih kolonova u nekom momentu. Potreban je tako dodatni oprez u slučaju korištenja takvih projekata jer se njihovo izvršavanje može u potpunosti prekinuti onda kada je uključena opcija IC. Opcija Remove Fencing (uklanjanje ograda - ograničenja) omogućava Sprite - ovima kretanje van vidljivog dijela ekrana (ploče), te omogućava korisnicima da Sprite - ove učine velikim odnosno malim onoliko koliko im je to potrebno (sa uključenom ovom opcijom i blokovi onda mogu da rade van okvira vidljivog dijela ekrana). Ukratko, uključivanje RF - a može poboljšati performanse u slučaju nekih projekata, ali isto tako i pokvariti neke druge projekte. Uključivanje opcije Remove Miscellaneous Limits (RML), kako joj samo ime govori, dozvoljava korisniku ukidanje raznih dodatnih ograničenja. Ovdje se misli na neka dodatna

ograničenja (koja nekima i nisu toliko bitna), kao što su: opseg zvučnog efekta (koji nije više ograničen brojem 360 nego je sada granica povećana na 1000), blokovi za postavljanje ili mijenjanje zvučnih efekata ili jačine zvuka (pokreću se sada trenutno, tj. bez prethodnih zaustavljanja u slučaju velikih vrijednosti FPS - a), minimalna veličina olovke je smanjena sa 1 na 0, dok je maksimalna veličina povećana sa 1200 na neograničeno, takođe, ne vrši se više zaokruživanje vrijednosti koordinata  $x$  i  $y$  kursora miša, što može biti jako korisno i precizno prilikom upotrebe fullscreen moda. Tu su i neka dodatna, manje bitna ukidanja ograničenja vezana za upotrebu udaraljki (bubnjeva) prilikom korištenja muzičkih objekata (ukidanje dotadašnjeg ograničenja, tj. mogućnosti istovremenog sviranja maksimalno 30 odgovarajućih - drum nota). Za pretpostaviti je da će se ova i slične liste vremenom proširivati. Postavljamo veličinu rezolucije ekrana na onu koju koristimo na svojim laptopima - računarima. U našem slučaju to je rezolucija 1920 x 1080.

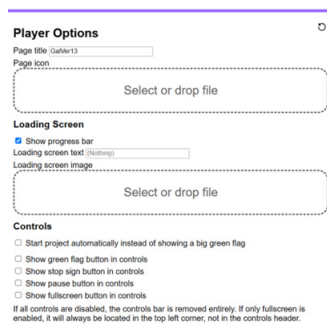
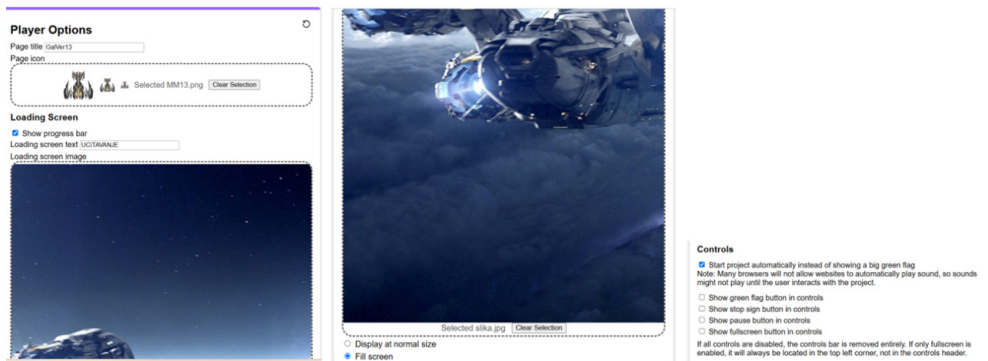
Da bismo nastavili popunjavanje omogućenih prozora u TurboWarp Packager - u, treba nam bar jedna slika i bar jedan transparentni Sprite. Treba nam naime ponovo neka kosmička slika kao i Sprite, tj. slika i Sprite vezani za temu igre. Koristit ćemo sljedeću sliku i Sprite



Zvat ćemo ih "slika" i "MM13".

Slika nam je potrebna kao pozadina koja će da bude aktivna prilikom load - ovanja (učitavanja igre), dok ćemo Sprite koristiti kao ikonu programa (igre) koja treba da bude vidljiva na desktopu.

Sljedeći ekran u TurboWarp Packager - u popunimo kao na slici

Tako, u polje Page icon, upload - ujemmo sliku Spite - a "MM13" (to je polje za ikonu koja će da bude vidljiva na desktopu). S druge strane, u polje Loading screen image upload - ujemmo sliku "slika". Ovdje, jasno, ostavljamo aktivnom kvakicu pored opcije Show progress bar, jer zaista želimo da vidimo postupak učitvanja (kako je to običaj kod većine igara), uz natpis "UCITAVANJE" (tekstovima na maternjem jeziku želimo jasno da stavimo do znanja da je igra proizvod domaćih aktera). Bitno je da stavimo kvakicu i pored opcije Fill screen, jer želimo da slika "slika" bude vidljiva

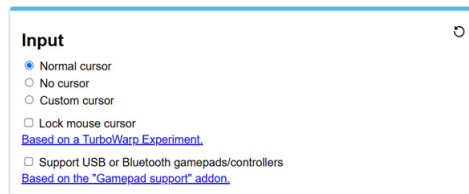
preko cijelog ekrana prilikom učitavanja igre. Dodatno, stavljamo kvakicu pored opcije Start project automatically instead of showing a big green flag. Naime, ako ne bi stavili kvakicu pored ove opcije, onda bi prilikom pokretanja izvršne verzije igre, dobili na sredini ekrana veliku zelenu zastavicu (istu kao onu kojom pokrećemo igru u Eticques - u), čijim bi klikom zaista i pokrenuli rad pomenute izvršne verzije igre. Ipak, pokretanje izvršnih verzija programa alatkama tipa zelenih zastavica je u najmanju ruku neprofesionalno. Drugim riječima, kreatori programa rijetko pristaju da se njihove aplikacije oslanjaju na gotove vizuelne objekte okruženja u kojim ih prave. Najčešće se takvi objekti (osim u slučaju nužde) zamjenjuju vlastitim, originalnim analognim objektima. Tako, sa omogućenom pomenutom opcijom, igra će nakon učitavanja odmah da bude pokrenuta. Primijetimo da na ovom spisku opcija imamo i dodatne opcije:

Show green flag button in controls,  
 Show stop sign button in controls,  
 Show pause button in controls,  
 Show fullscreen button in controls.

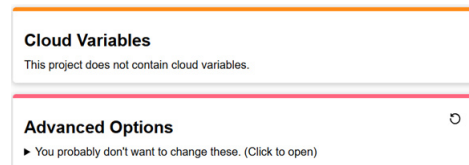
Uključivanje prve tri opcije omogućava pojavu gotovih dugadi za Start, Stop i Pause na izvršnog verziji igre. Ta dugmad su onda prikazana na način kako ih srećemo na raznim čitačima audio i video zapisa širom svijeta, tj. u obliku zelenog trokutića (Start), crvenog kruga (Stop), te dviju žutih vertikalnih linija (Pause). U slučaju njihovog uključivanja, ova dugmad će zaista obavljati funkciju za koju su namijenjena, tj. startat će igru, igra će biti zaustavljena, odnosno pauzirana, redom. Pri tome će dugmad biti raspoređena na vrh vidljivog ekrana čime će posljedično taj ekran onda korisniku da bude djelimično smanjen, a upravo taj ekran u izvršnim varijantama programa obično treba da bude dat kao fullscreen. U skladu sa već rečenim, ali i ranije rečenim, mi ne volimo i ne želimo da koristimo gotove objekte upravo opisanog tipa bez da smo ih mi napravili, tako da ove objekte u izvršnu verziju igre ne uključujemo (zanimljivo bi bilo određeno dodatno razmatranje o tome kako pomenutu dugmad samostalno napraviti).

Elemente na sljedećem ekranu TurboWarp - a ostavljamo nepromijenjenim

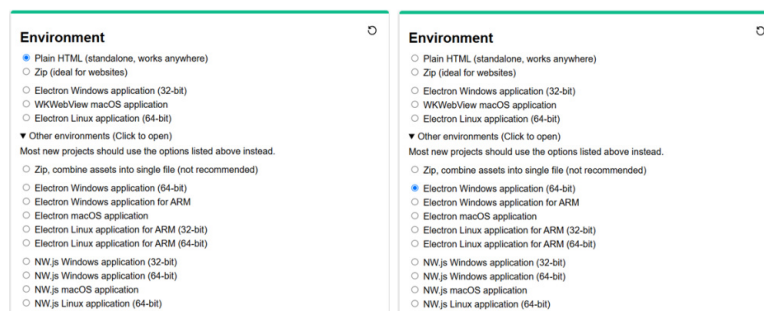




Isto važi i za sljedeći ekran



Ovdje je korisno napomenuti da Advanced opcije (posljednja slika) koristimo ako želimo da na određeni način "izvana", tj. upotrebom npr. JavaScript - a dodamo određene objekte sa kojima inače ne raspolažemo, i koje nam okruženje Eticques nije u stanju inicijalno obezbjediti. Riječ je o raznim objektima koji se naknadno mogu uključiti u Eticques a koji onda mogu biti vidljivi u izvršnoj verziji odgovarajućeg sb3 dokumenta (to su npr. upravo pomenuti objekti tipa: Start, Stop, Pause, ali i razni drugi objekti). Za ovakve dodatke korisnik ipak treba da u određenoj mjeri poznaje rad u JavaScript - u ili rad u sličnim okruženjima. Sljedeći ekran u TWP - u podesimo u skladu sa zahtjevima našeg Windows - a



Sada su nam otključane i dodatne opcije u TWP - u.

Podesimo ih onako kako slijedi

### Application Settings

Package name

Package name should be a unique name used only by your application. The same project should always have the same package name.

Version

Updating the version is completely optional and doesn't affect your project's behavior. It may appear in places like about screens depending on the environment. Should be formatted as X.Y.Z.

Initial window size

When escape key pressed

Window controls

Creating native applications for specific platforms is discouraged. In most cases, Plain HTML or Zip will have numerous advantages:

- Can be run directly from a website on any platform, even phones
- Users are significantly less likely to be suspicious of a virus
- Significantly smaller file size
- Can still be downloaded locally and run offline

If you don't truly need to make a self-contained application for each platform (we understand there are some cases where this is necessary), we recommend you don't.

### Windows

All Windows applications generated by this site are unsigned, so users will see SmartScreen warnings when they try to run it for the first time. They can bypass these warnings by pressing "More info" then "Run anyways".

To change the icon of the executable file or create an installer program, download and run [Turbowarp Packager Extras](#) and select the output of this website.

### Electron

The Electron environment works by embedding a copy of Chromium (the open source part of Google Chrome) along with your project, which means the app will be very large.

### Application Settings

Package name

Package name should be a unique name used only by your application. The same project should always have the same package name.

Version

Updating the version is completely optional and doesn't affect your project's behavior. It may appear in places like about screens depending on the environment. Should be formatted as X.Y.Z.

Initial window size

When escape key pressed

Window controls

Creating native applications for specific platforms is discouraged. In most cases, Plain HTML or Zip will have numerous advantages:

- Can be run directly from a website on any platform, even phones
- Users are significantly less likely to be suspicious of a virus
- Significantly smaller file size
- Can still be downloaded locally and run offline

If you don't truly need to make a self-contained application for each platform (we understand there are some cases where this is necessary), we recommend you don't.

### Windows

All Windows applications generated by this site are unsigned, so users will see SmartScreen warnings when they try to run it for the first time. They can bypass these warnings by pressing "More info" then "Run anyways".

To change the icon of the executable file or create an installer program, download and run [Turbowarp Packager Extras](#) and select the output of this website.

### Electron

The Electron environment works by embedding a copy of Chromium (the open source part of Google Chrome) along with your project, which means the app will be very large.

Primijetimo samo da pod kraticom Windows (posljednje dvije slike), zadnja rečenica se odnosi na dodjelu ikone izvršnoj verziji naše aplikacije (već smo uradili iznad), odnosno na njenu eventualnu izmjenu, te na generisanje programa za instalaciju igre. Dijelu koji se odnosi na generisanje programa za instalaciju ćemo se vratiti kasnije. Na dnu okruženja TWP - a, imamo prozor u kojem kliknemo na opciju Package

Export Settings Import Settings Reset All Settings

Package Preview

Downloads will appear here

Privacy Policy - Feedback - Source Code - Donate

Additional resources and help

Use system theme

Export Settings Import Settings Reset All Settings

Package Preview

Downloading Electron

Privacy Policy - Feedback - Source Code - Donate

Additional resources and help

Use system theme

Export Settings Import Settings Reset All Settings

Package Preview

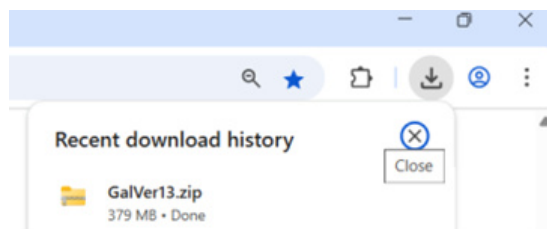
Download GalVer13.zip (397.11MB)

Privacy Policy - Feedback - Source Code - Donate

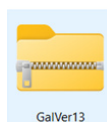
Additional resources and help

Use system theme

Nakon kraćeg zadržavanja, u gornjem desnom dijelu ekrana dobijemo potvrdu o uspješno obavljenom download - u



Idemo na Downloads, i tamo uočimo zip fajl GalVer13



Očekivano, nakon desnog klika na posljednji folder, kliknemo na Extract All, te ne mijenjajući lokaciju koja nam se nudi (C:\Users\Dz.G\Downloads\GalVer13), kliknemo na Extract. Izdvajanje (unzipp) će ubrzo da bude proveden, tako da u istom folderu Downloads, dobijamo novi, raspakovani folder GalVer3



u njemu folder Glaksija



a u njemu odgovarajuće elemente koji ga čine

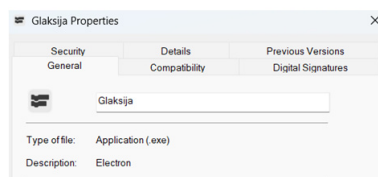
## 15 TURBOWARP PACKAGER I EXE FAJLOVI

|                         |                  |                       |            |
|-------------------------|------------------|-----------------------|------------|
| libGLESv2.dll           | 6/1/2025 9:12 AM | Application extens... | 7,359 KB   |
| licenses                | 6/1/2025 9:12 AM | Microsoft Edge HT...  | 6,658 KB   |
| README                  | 6/1/2025 9:12 AM | Text Document         | 1 KB       |
| resources.pak           | 6/1/2025 9:12 AM | PAK File              | 5,249 KB   |
| snapshot_blob.bin       | 6/1/2025 9:12 AM | BIN File              | 169 KB     |
| v8_context_snapshot.bin | 6/1/2025 9:12 AM | BIN File              | 472 KB     |
| vk_swiftshader.dll      | 6/1/2025 9:12 AM | Application extens... | 5,014 KB   |
| vk_swiftshader_icd.json | 6/1/2025 9:12 AM | JSON File             | 1 KB       |
| vulkan-1.dll            | 6/1/2025 9:12 AM | Application extens... | 894 KB     |
| ffmpeg.dll              | 6/1/2025 9:12 AM | Application extens... | 2,703 KB   |
| Glaksija                | 6/1/2025 9:12 AM | Application           | 153,884 KB |
| icudt.dll               | 6/1/2025 9:12 AM | DAT File              | 10,218 KB  |
| libEGL.dll              | 6/1/2025 9:12 AM | Application extens... | 473 KB     |
| d3dcompiler_47.dll      | 6/1/2025 9:12 AM | Application extens... | 4,777 KB   |
| chrome_100_percent.pak  | 6/1/2025 9:12 AM | PAK File              | 127 KB     |
| chrome_200_percent.pak  | 6/1/2025 9:12 AM | PAK File              | 176 KB     |
| locales                 | 6/1/2025 9:12 AM | File folder           |            |
| resources               | 6/1/2025 9:12 AM | File folder           |            |

Uočimo element po imenu Glaksija

Glaksija 6/1/2025 9:12 AM Application 153,884 KB

S obzirom na njegov produžetak, vidimo da je riječ o aplikaciji. Naime, nakon desnog klika na ovaj dokument, te klika na Properties, uočavamo da je zaista riječ o aplikaciji, tj. da je riječ o izvršnoj odnosno exe (executable) verziji naše sb3 igre



Tako, upravo opisani dokument Glaksija je ustvari dokument koji tražimo. To je izvršna verzija igre koju želimo da dobijemo. Kao i sve aplikacije, i ovu onda pokrećemo klasično dvostrukim klikom. Pratimo šta se dešava





Prvo po redu je učitavanje (ovdje sada dolaze do izražaja elementi koje smo popunjavali u formu TWP - a: podloga za učitavanje, jezik pisanja, tekst, itd). Kako smo već konstatovali, nakon učitavanja dolazi do automatskog pokretanja igre, dakle bez čekanja i upotrebe zelene zastavice



Jasno, nakon klika na dugme START igra biva pokrenuta



Primijećujemo da od samog pokretanja igre operišemo u formatu fullscreen, koji je zbog svoje veličine, preglednosti, mnogo zahvalniji od inicijalnog fullscreen ekrana kojeg nam je obezbjeđivao Eticques. Ono što još moramo da primijetimo jeste to da ikona naše aplikacije još uvijek nije data Sprite - om "MM13". Sadašnja ikona je dakle neka ikona koju nam nudi sam TurboWarp (ikona lijevo na slici ispod)



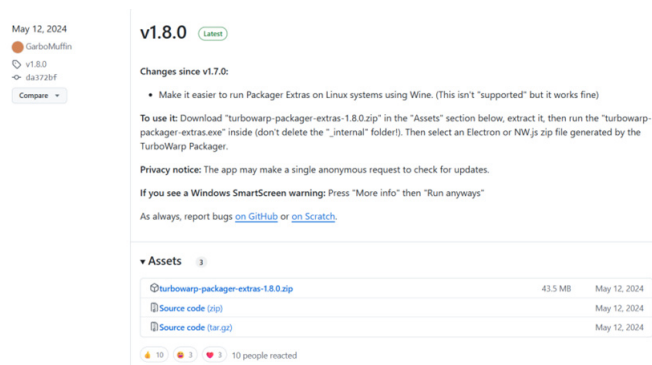
Ipak, sjetimo se da smo prilikom učitavanja neophodnih podataka u TurboWarp takođe učitali sliku "MM13" kao sliku koja će da predstavlja ikonu naše aplikacije. Za sada nismo dobili ono što smo htjeli. Dodatno, istakli smo, razmatrajući prozor Windows u TurboWarpu, da možemo ako želimo generisati program za instalaciju naše igre

#### Windows

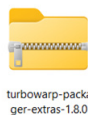
All Windows applications generated by this site are unsigned, so users will see SmartScreen warnings when they try to run it for the first time. They can bypass these warnings by pressing "More info" then "Run anyways".

To change the icon of the executable file or create an installer program, download and run [TurboWarp.Packager.Extras](#) and select the output of this website.

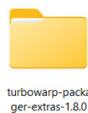
Kako ćemo vidjeti, ova osobina TurboWarpa će ne samo da generiše opisani program za instalaciju, neko će ujedno da izvrši i željeno pridruživanje ikone "MM13" našoj aplikaciji. Dok smo u TurboWarpu, sa svim već load - ovanim elementima, kliknemo na link prikazan na prethodnoj slici



U dnu, idemo na turbowarp-packager-extras-1.8.0.zip. Nakon kraćeg čekanja započne download, tako da vrlo brzo u folderu Downloads možemo uočiti odgovarajući zip fajl



Kao i ranije, ovaj fajl unzipp - ujemo (Extract all) na lokaciju na kojoj se već nalazimo (to je lokacija koja nam se nudi nakon pokretanja unzipp - a). Dobijamo unzip - ovan folder



U njemu otvorimo folder

## 15 TURBOWARP PACKAGER I EXE FAJLOVI

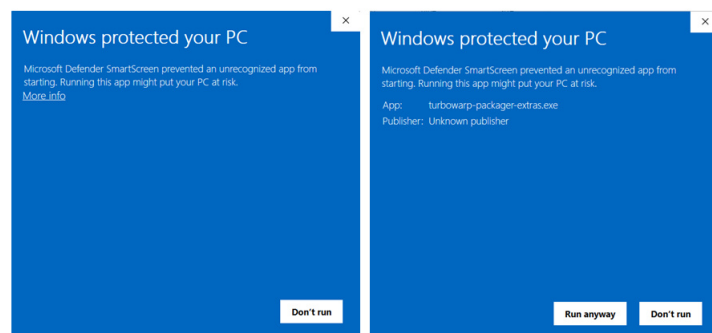
---

turbowarp-packager-extras 6/1/2025 10:08 AM File folder

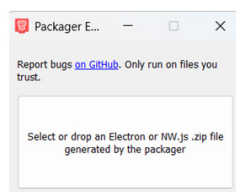
a u njemu pokrenemo aplikaciju

turbowarp-packager-extras 6/1/2025 10:08 AM Application 3,391 KB

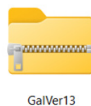
Windows će nam zasigurno javiti da Microsoft Defender SmartScreen onemogućava pokretanje aplikacija koje on smatra neprovjerenim. Ipak, idemo na More info, pa onda na Run anyway



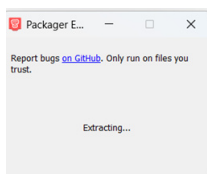
Dobijamo mali prozorčić pokrenute aplikacije koji nas informiše da treba da upload - ujemmo zip fajl tipa Elektron (sjetimo se da smo na prozoru TWP - a birali opciju Electron Windows application (64 - bit)) kojeg nam je generisao TWP



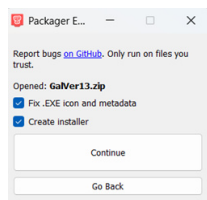
Treba dakle da u posljednji prozor upload - ujem zip fajl (kojeg već imamo u folderu Downloads)



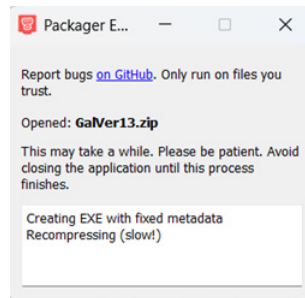
Uradimo to, tj. kliknimo na dio "Select" otvorene aplikacije, i upload - ujm traženi folder. Započinje izdvajanje



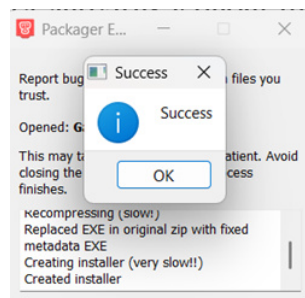
Prilikom izdvajanja moguće je da se aktivira antivirus jednom ili više puta, ali u tom slučaju nema potrebe za blokiranjem bilo čega (u okruženju antivirusa idemo na allow cijelo vrijeme). Dobijamo



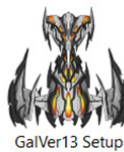
Ovdje sada možemo uočiti mogućnost kreiranja objekata koji nam zaista trebaju: ikone igre, i programa za njenu instalaciju, pa idemo na Continue. Od nas se traži da izaberemo lokaciju na kojoj ćemo sačuvati dokument koji treba da se gradi. Ponovo, možemo ga sačuvati u folderu Downloads. Idemo na Save. Vrijedno je napomenuti da izgradnja posljednjeg objekta može da potraje (što ne treba smatrati nekom greškom ili blokiranjem rada programa). Jednostavno sačekamo



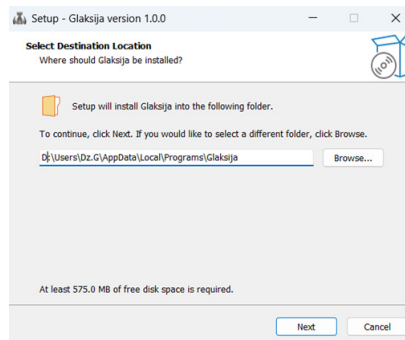
Konačno, dobijamo potvrdu o uspjehu izgradnje



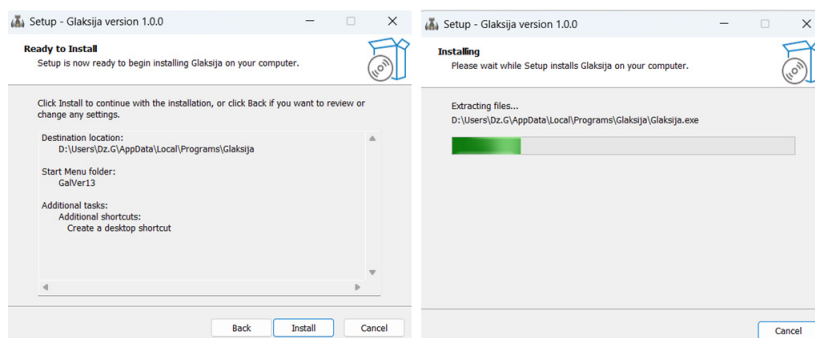
Nakon klika na OK, moguće da se antivirus ponovo aktivira, ali kao i u prethodnom slučaju, idemo na allow. Korištenu aplikaciju (mali prozorčić sa crvenom ikonom) možemo konačno zatvoriti. U folderu Downloads, vidimo da je izgrađen program za instalaciju igre (nastavak mu je Setup). Takođe, vidimo da čak i on sadrži ikonu "MM13"



Pokrenimo onda instalaciju igre (kao probu) dvostrukim klikom na GalVer13 Setup. Dobijamo



Ponuda lokacije instalacije je jasno automatska, s tim što u posmatranom slučaju biramo disk D (umjesto C) radi količine slobodne memorije dostupne na disku D. Idemo na next tri puta, nakon čega instalacija zaista i započne sa radom



Ukinemo kvakicu za automatsko pokretanje igre (kvakicu pored Launch GalVer13 - jer želimo ručno da to uradimo tražeći exe verziju koja bi trebala da sadrži našu ikonu "MM13"), te kliknemo na Finish. Potražimo onda exe verziju naše igre prateći putanju datu na početku instalacije

D:\Users\Dz.G\AppData\Local\Programs\Glaksija

## 15 TURBOWARP PACKAGER I EXE FAJLOVI

Zaista, u posljednjem po redu folderu, tj. folderu Glaksija uočavamo sljedeće elemente

|                         |                   |                       |            |
|-------------------------|-------------------|-----------------------|------------|
| locales                 | 6/1/2025 10:43 AM | File folder           |            |
| resources               | 6/1/2025 10:43 AM | File folder           |            |
| chrome_100_percent.pak  | 6/1/2025 10:24 AM | PAK File              | 127 KB     |
| chrome_200_percent.pak  | 6/1/2025 10:24 AM | PAK File              | 176 KB     |
| d3dcompiler_47.dll      | 6/1/2025 10:24 AM | Application extens... | 4,777 KB   |
| ffmpeg.dll              | 6/1/2025 10:24 AM | Application extens... | 2,703 KB   |
| Glaksija                | 6/1/2025 10:30 AM | Application           | 153,990 KB |
| icudtl.dat              | 6/1/2025 10:24 AM | DAT File              | 10,218 KB  |
| libEGL.dll              | 6/1/2025 10:24 AM | Application extens... | 473 KB     |
| libGLESv2.dll           | 6/1/2025 10:24 AM | Application extens... | 7,359 KB   |
| licenses                | 6/1/2025 10:24 AM | Microsoft Edge HT...  | 6,658 KB   |
| README                  | 6/1/2025 10:24 AM | Text Document         | 1 KB       |
| resources.pak           | 6/1/2025 10:24 AM | PAK File              | 5,249 KB   |
| snapshot_blob.bin       | 6/1/2025 10:24 AM | BIN File              | 169 KB     |
| unins000.dat            | 6/1/2025 10:44 AM | DAT File              | 50 KB      |
| unins000                | 6/1/2025 10:40 AM | Application           | 3,181 KB   |
| v8_context_snapshot.bin | 6/1/2025 10:24 AM | BIN File              | 472 KB     |
| vk_swiftshader.dll      | 6/1/2025 10:24 AM | Application extens... | 5,014 KB   |
| vk_swiftshader_icd.json | 6/1/2025 10:24 AM | JSON File             | 1 KB       |
| vulkan-1.dll            | 6/1/2025 10:24 AM | Application extens... | 894 KB     |

Jedan od njih je i traženi element, tj. izvršna verzija igre popraćena našom ikonom

|                                                                                              |                   |             |            |
|----------------------------------------------------------------------------------------------|-------------------|-------------|------------|
|  Glaksija | 6/1/2025 10:30 AM | Application | 153,990 KB |
|----------------------------------------------------------------------------------------------|-------------------|-------------|------------|

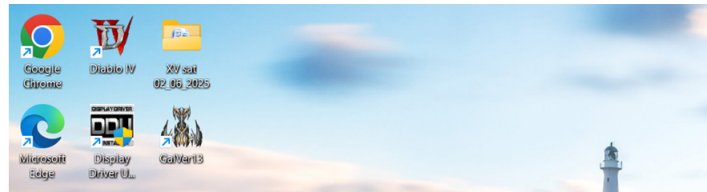
Dvo - klik na ovaj objekat će da pokrene igru na način kako smo je pokretali i ranije.

U istom folderu uočimo i element za eventualnu un - instalaciju igre

|                                                                                              |                   |             |          |
|----------------------------------------------------------------------------------------------|-------------------|-------------|----------|
|  unins000 | 6/1/2025 10:40 AM | Application | 3,181 KB |
|----------------------------------------------------------------------------------------------|-------------------|-------------|----------|

Konačno, uočimo na samome desktopu ikonu igre (GalVer13)





Jasno, kao i u slučaju drugih aplikacija, igru sada možemo pokrenuti i dvo - klikom na ovu ikonu.

Napomenimo da objekte koje smo prvo download - ovali, a onda i unzipp - ovali u folder Downloads, možemo tamo i ostaviti (ako nam ne zauzimaju previše memorije a u slučaju da nam ponovo zatrebaju).

### 15.1 Riješeni zadaci

◇ **Zadatak 15.1.1** Kreirati igricu u kojoj se lik (npr. ptica, balon ili avion) kreće kroz prostor pun prepreka koje se pojavljuju sa desne strane ekrana i pomiču ulijevo.

Obavezni elementi:

- lik koji se kontroliše strelicama ili tipkama (npr. gore/dole),
- prepreke koje se pojavljuju kao klonovi i kreću zdesna nalijevo,
- kloniranje mora biti pravilno korišteno, tj. original mora biti sakriven, dok klonovi imaju ponašanje,
- igra se završava sudarom sa preprekom, pri čemu se prikazuje "Game Over" poruka,
- potrebno je prikazati rezultat (score) - poen se dobija kada igrač uspješno prođe prepreku,
- napraviti početni ekran sa dugmetom "Start".

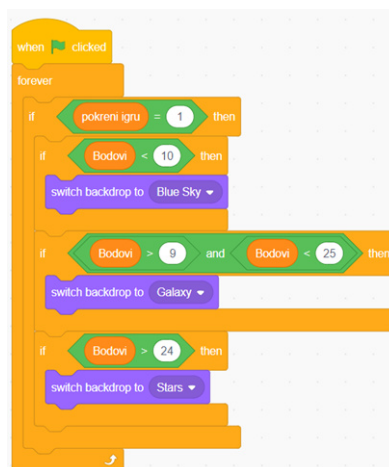
Dodatno:

- obezbijediti pozadine, zvukove, animacije,
- napraviti dugme za ponovni početak (restart),
- omogućiti nasumično generisane prolaze.

**Rješenje:** Kod za Sprite Cat Flying:



Kod za Sprite Start:



Kod za Sprite Dijamant:



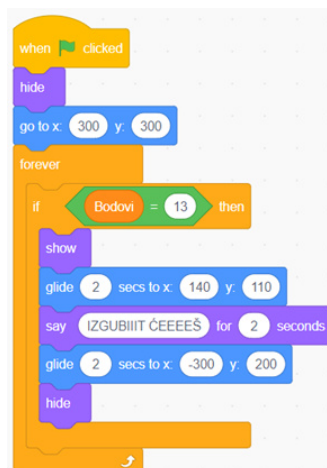
Kod za Sprite Prepreka:



Kod za Sprite Oblak:



Kod za Sprite Ghost:



**15.2 Zadaci za vježbu**

♣ **Zadatak 15.2.1** Kreirati jednostavnu kviz igricu u kojoj igrač odgovara na unaprijed pripremljena pitanja i osvaja bodove.

♣ **Zadatak 15.2.2** Napraviti kviz po uzoru na popularni kviz "Potjera". Igrač igra protiv Lovca na tabli s poljima. Za svaki tačan odgovor igrač i/ili Lovac se pomjeraju, a cilj je da igrač stigne do ciljne linije prije nego ga Lovac sustigne. Kviz može imati početnu fazu u kojoj se odgovara na 5 kratkih pitanja (može se koristiti kviz iz Zadatka 15.2.1). Osvojeni bodovi predstavljaju početni broj bodova u drugoj fazi u kojoj se igra sa Lovcem.





## BIBLIOGRAFIJA

- [1] M. Andujar, L. Jimenez, J. Shah and P. Morreale, Evaluating visual programming environments to teach computing to minority high school students, *The Journal of Computing Sciences in Colleges*, **29** (2013), pp. 140–148.
- [2] M. Armoni, O. Meerbaum-Salant and M. Ben-Ari, From scratch to real programming, *ACM Transactions on Computing Education*, **14** (2015), pp. 1–15.
- [3] M. Bertamini, *Programming Visual Illusions for Everyone*, Springer, 2018.
- [4] K. Binmore, *Game Theory: A Very Short Introduction*, Oxford University Press, 2007.
- [5] R. Branzei, D. Dimitrov and S. Tijs, *Models in Cooperative Game Theory: Crisp, Fuzzy, and Multi-Choice Games*, Springer-Verlag, Berlin Heidelberg, 2005.
- [6] L. Brickman, *Mathematical Introduction to Linear Programming and Game Theory*, Springer-Verlag, 1989.
- [7] M. M. Burnett and M. J. Baker, A classification system for visual programming languages, *J. Vis. Lang. Comput.*, **5** (1994), pp. 287–300.
- [8] J. C. Cheung, G. Ngai, S. C. Chan and W. W. Lau, Filling the gap in programming instruction: A text-enhanced graphical programming environment for junior high, *ACM SIGCSE Bulletin*, **41** (2009), pp. 276–280.
- [9] D. N. Chorafas, *Visual Programming Technology*, McGraw-Hill Pub, 1996.
- [10] P. J. Denning and M. Tedre, Thinking Like a Computer, *American Scientist*, **107** (2019), pp. 187–189.
- [11] T. S. Ferguson, *Game Theory*, University of California, 1991.
- [12] R. M. Friedhoff and M. S. Pearcy, *Visual Computing*, W H Freeman & Co, 2000.
- [13] I. K. Geckil and P. L. Anderson, *Applied Game Theory and Strategic Behavior*, CRC Press, 2010.
- [14] M. Idrees and F. Aslam, A Comprehensive Survey and Analysis of Diverse Visual Programming Languages, *VFAST Transactions on Software Engineering*, **10** (2022), pp. 47–60.
- [15] K. Y. Kaya and K. Cagiltay, Creating and Evaluating a Visual Programming Course Based on Student Experience, *Emerging Research, Practice, and Policy on Computational Thinking*, **1** (2017), pp. 135–151.
- [16] S. Kepple, *Teaching Coding Through Game Creation*, Libraries Unlimited, 2018.
- [17] C. H. Kim, S. J. Kim, S. K. Kim and S. J. Kang, *Real-Time Visual Effects for Game Programming*, Springer, 2015.
- [18] D. Leigh, *Visual Programming*, Crucial, 2003.
- [19] M. Marji, *A Visual Introduction to Programming with Games, Art, Science, and Math*, No Starch Press, 2014.
- [20] M. McShaffy, *Game Coding Complete*, Paraglyph Press, 2003.
- [21] F. Melendez, *Drawing from the Model: Fundamentals of Digital Drawing, 3D Modeling, and Visual Programming in Architectural Design*, Wiley, 2019.

- [22] J. Meyer, *Programming 101: Learn to Code with the Processing Language Using a Visual Approach*, Apress, 2022.
- [23] B. A. Myers, Taxonomies of visual programming and program visualization, *J. Vis. Lang. Comput.*, **1** (1990), pp. 97–123.
- [24] M. Noone and A. Mooney, Visual and textual programming languages: a systematic review of the literature, *J. of Computers in Education*, **5** (2018), pp. 149–174.
- [25] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*, The MIT Press, Cambridge, 1994.
- [26] C. Papadimitrou, *Algorithmic Game Theory*, Cambridge University Press, 2007.
- [27] H. Peters, *Game Theory: A Multi-Leveled Approach*, Springer-Verlag, Berlin Heidelberg, 2008.
- [28] C. Reas and B. Fry, *Processing: A Programming Handbook for Visual Designers and Artists*, The MIT Press, 2007.
- [29] G. Romp, *Game Theory: Introduction and Applications*, Oxford University Press, 1997.
- [30] D. I. Schneider, *Computer Programming Concepts*, Pearson Custom Publishing, 2000.
- [31] N. C. Shu, *Visual Programming*, Van Nostrand Reinhold, 1988.
- [32] J. Slingerland, *Video Game Coding*, North Star Editions, 2019.
- [33] P. D. Straffin, *Game Theory and Strategy*, The Mathematical Association of America, 1993.
- [34] A. Sweigart, *Programming Playground*, No Starch Press, 2021.
- [35] D. Trakosas, C. Tikva and E. Tambouris, Visual programming and computational thinking environments for K-9 education: a systematic literature review, *Int. J. of Learning Technology*, **18** (2023), pp. 94–121.
- [36] K. N. Whitley and A. F. Blackwell, Visual Programming in the Wild: A Survey of LabVIEW Programmers, *Journal of Visual Languages and Computing*, **12** (2001), pp. 435–472.





## IZVODI IZ RECENZIJA

“...Vizuelno programiranje (kroz Etics) pokriva program predmeta Vizuelno programiranje PMF-a UNSA, ali i istoimenog predmeta FF-a UNZE. Udžbenik predstavlja kvalitetan i detaljan materijal dizajniran da otvori vrata svijetu programiranja studentima prve godine studija kompjuterskih nauka i srodnih oblasti. Izdvaja se pristupom koji naglašava pristupačnost, modularno razmišljanje i neposredne vizuelne povratne informacije. Autori vješto koriste Etics, modernu vizuelnu programsku platformu, za generisanje izvršnih verzija projekata. Posebna vrijednost rukopisa leži u pronicljivom poređenju Etics – a sa tradicionalnim, kompajlerskim jezicima poput jezika C++, gdje se detaljno objašnjavaju prednosti i ograničenja svakog pristupa, posebno u kontekstu sintakse, upravljanja memorijom i performansi. Ključna prednost Etics-a je korištenje grafičkog blok interfejsa, koji u potpunosti eliminiše sintaksičke greške, omogućavajući korisnicima da se usredsrijeđe na suštinu programiranja – logiku i strukturu programa, umjesto na jezička pravila i interpunkciju. Jezik knjige je jasan, tehnička terminologija dosljedno korištena i objašnjena, a stilski nivo u potpunosti odgovara univerzitetskoj literaturi. Brojni primjeri (igre, simulacije, kvizovi, vizuelni i zvučni efekti) približavaju apstraktne pojmove kroz motivirajuće i realne scenarije, što udžbenik čini privlačnim početnicima i naprednijim studentima. Autori posvećuju veliku pažnju pedagoškoj dimenziji – svaki koncept je praćen ilustracijama i postupnim objašnjenjima, čime se barijere u učenju značajno smanjuju...”



9 789926 453992