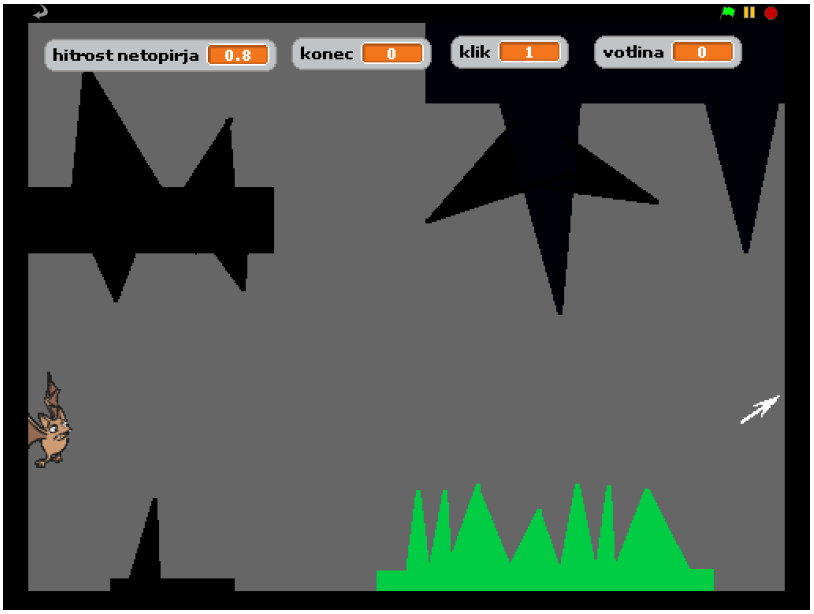
3 IGRICA V BYOB

3.1 BESEDILO NALOGE IGRICA V BYOB

Za nalogo sem si zadala narediti igro, kjer igralec upravlja lik netopirja. Igralec lik upravlja s pomočjo miškinega kazalca. Lik sledi miškinemu kazalcu, kar povzroči rahlo zakasnitev upravljanja. Če igralec dovoli, da se lik dotakne miškinega kazalca, se liku povečuje hitrost. Z višjo hitrostjo pa se poveča težavnost igre. Lik lahko vodimo v vse smeri. Potuje po votlinah, ki so različnih oblik. Cilj je preleteti celotno votlino in prispeti do konca le-te, ne da bi se pri tem dotaknili tal, stropa votline ali ovir. Konec votline je označen s puščico. Ko lik prepotuje votlino in se dotakne puščice, se premakne v naslednjo. Vsaka naslednja je težavnejša od prejšnje. Prostor, ki ga mora igralec preleteti se oža, hitrost lika pa povečuje s časom, iz votline v votlino, kot tudi, če se lik dotakne miškinega kazalca. Ko igralec prepotuje vse votline se igra konča.

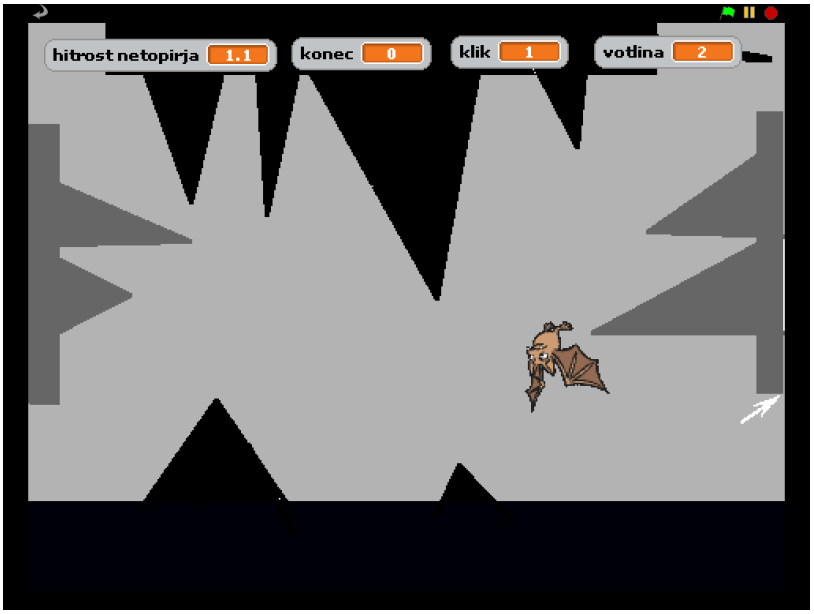
3.2 ZASLONSKE SLIKE IGRE TER OPIS LE TEH

KOMENTAR: igralec na prvi sliki dobi navodila za začetek igre.



KOMENTAR: primer votline skozi katero igralec upravlja lik. V prvi votlini je začetna hitrost 0.8. Konec je na 0, saj se še nismo dototaknili črne ovire. Klik je na 1, saj smo kliknili, ko smo začeli z igro. Votlina pa je 0, prva votlina.

KOMENTAR: z zaslonske slike je razvidno, da se je liku povečala hitrost in da ni vse ovira, kot se zdi na prvi pogled. Klik je še vedno na 1 in bo vedno, dokler ne zadanemo v oviro ali pa uspešno končamo igre. Votlina pa se je povečala, saj smo v novi votlini. Lik je na poti do bele puščice, kar mu omogoča prehod v novo votlino.



KOMENTAR: tu je razvidno, da lahko lik potuje v vse smeri. Liku se je povečala hitrost in votlina.

KOMENTAR: ko pridemo skozi vse votline, ne da bi se dotaknili ovir, smo zmagali. Netopirček nam čestita za uspešen konec igre.

3.3 OPIS IDEJ

Prva ideja – izgled lika in ozadje igre

V BYOB-u sem izbrala že obstoječi lik netopirja in ga pomanšala na 30% prvotne velikosti. To sem naredila z grednikom »set size to«. Ker natopirji živijo v votlinah, sem dobila idejo za ozadje igre, ozadja so različne votline, ki sem jih v Paint-u izdelala sama, tako da sem za začetek naredila lažje votline in jih oteževala. V votlinah je vedno manj prostora, več je ovir in vedno bolj skupaj so.

Netopirja sem zmanjšala šele, ko sem naredila vse votline, da se je lahko gibal po njih, ne da bi trčil v ovire, ki jih imajo votline.

Druga ideja – zasnova gibanja lika in upravljanje

Nato sem natopirju sprogramirala program, s katerim ga lahko upravljamo. Idejo sem dobila s prosojnic. Uporabila sem gradnik s pogojnim stavek if in else. V if stavku sem dodala operator »not«, ki sem mu dodala še zaznamovalni ukaz »touching mouse-pointer«. Se pravi, če se ne dotika miškinega kazalca, se zgodi naslednje. Lik sledi miškemu kazalcu, to sem naredila s pomočjo ukaza za premikanje. V isti ukazni paleti sem kasneje našla tudi gradnik »muve steps«, v katero sem s pomočjo ukaza za spremenljivke, naredila novo »hitrost netopirja«. Spremenljivki sem z gradnikom »change hitrost netopirja by« dodala hitrost netopirju. To sledi v peti ideji. V stavku else se zgodi podobno. S tem sem dosegla, da se zdaj netopirju povečuje hitrost tako s časom, kot če se ga dotaknemo z miškinim kazalcem. Igralec zdaj netopirja premika s pomočjo miškinega kazalca. Netopir sledi miškinemu kazalcu, ki ga igralec lahko premika v vse smeri.
Še predno sem dogradila igrico, se je netopir ustavil, če se je ustavil igralec. Se pravi, če igralec ni premaknil miškinega kazalca, se je netopir ustavil takoj ko je bil na miškinem kazalcu.

Tretja ideja – premikanje in koordinatni sistem

Ker netopir sledi miškinemu kazalcu, sem mu morala nastaviti začetne koordinate. To sem naredila s pomočjo gradnikov za premikanje in sicer »set x to« in »set y to«. Do sedaj je začel tam, kjer se je nazadnje dotaknil ovire, stropa ali tal. Koordinati mu omogočata, da netopir vedno začne svojo pot na istem mestu in nadaljuje pot proti miškinemu kazalcu. Koordinate pa je potrebno navesti še enkrat, saj mora lik v vsaki votlini začeti svojo pot na tem mestu, ki ga določata koordinati.

Četrta ideja - različne votline in težavnost

Igra bi bila prekratka, če bi natopir imel samo eno votlino. Zato sem naredila več votlin, ki jih štejem s spremenljivko votlina, da vem kdaj igralec pride do zadnje votline. Votline sem naredila sama v Paint-u. V vsaki votlini začne svojo pot na isti točki in jo nadaljuje tako, da sledi miškinemu kazalcu. Vsaka votlina ima tudi kako oviro, ki se je netopir ne sme dotakniti. Netopir se ne sme dotakniti nobene črne površine, kot so ovire, strop ali tla. To sem sprogramirala v stavku if, in sicer z zaznavnim ukazom »touching color«. Sledijo še gradniki za izgled. Ko se lik dotakne črne ovire, se v okvirčku na odru izpiše »IGRE JE KONEC!!!«. Nato čez 2 sekundi dobi igralec novo obvestilo, in sicer »ZA APONOVNO IGRO PRITISNI SPACE«. Vsaka naslednja votlina je težja, ima več ovir in zato omeji gibanje liku. Igralec mora priti iz začetnega položaja, to je skrajno levo, preko votline do puščice, ki je na skrajni desni strani. Ko se natopir dotakne puščice, ga le ta prestavi v naslednjo votlino. Puščica, ki mu to omogoča je bele barve. To sem sprogramirala v stavku if, ki sem mu s pomočjo zaznavnega ukaza, dodala gradnik »touching bela puščica« .

Peta ideja – težavnost igre

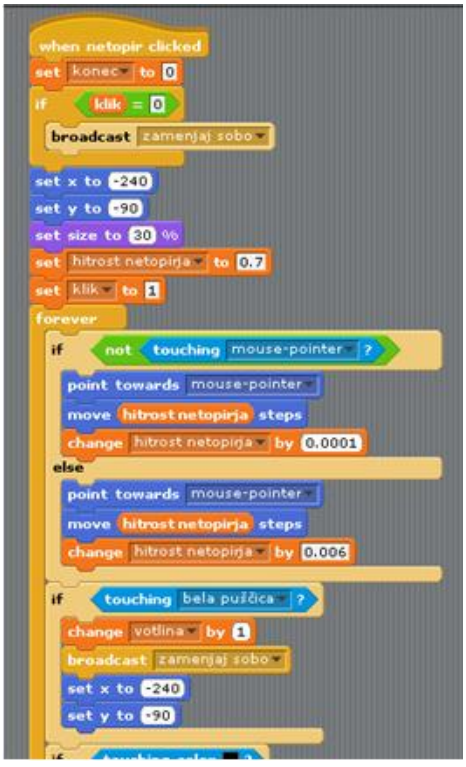
Ker je bila igrica prelahka, sem jo morala malo otežiti. Najprej sem sprogramirala povišanje hitrosti s časom, nato pa, če se netopir dotakne miške. Pred tem se je netopir ustavljal skupaj z miško, kar je

igralcu omogočilo lažji prehod čez ovire in votlino. Hitrost pa se povečuje tudi iz votline v votlino. Ker je netopir hitrejši, je težje obvladljiv. To hitrost sem sprogramirala figuri »bela puščica«. Ko se je dotakne lik, ga le ta prestavi v novo votlino in mu doda hitrost.

Šesta ideja – začetek igre in igraj ponovno

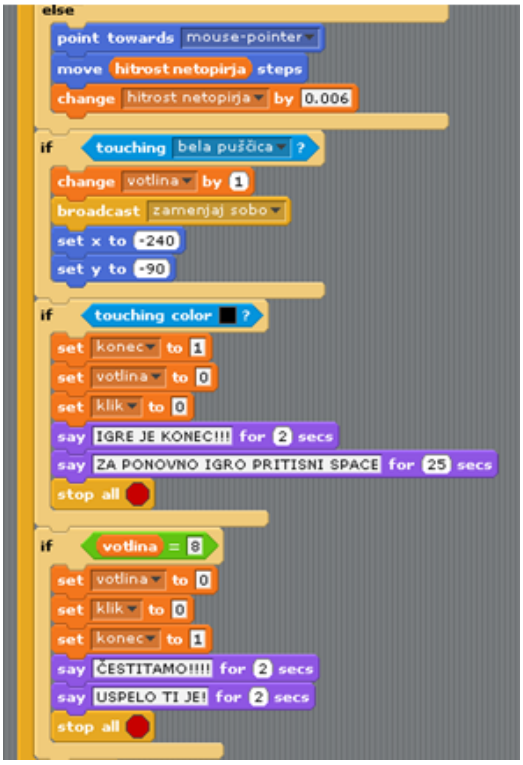
Za začetek igre igralec kilkne na manšega netopirja, kar igralcu omogoči pričetek igre, navodila dobi na začetni strani. To sem storila z ukazom za upravljanje oz. z začetnim gradnikom »when netopir clicked«. Ko klikne na netopirja spremeni spremenljivko klik na 1, kar nam pove, da se je igra že začela in da klik ne pomeni več začetka igre. Pred tem se je igralec s klikom na netopirja lahko premaknil v novo votlino, ne da bi prišel do bele puščice. Sedaj pa se s klikom premakne na začetek votline, v kateri se nahaja. Ko se lik dotakne katerekoli ovire črne barve, je igre konec. Tukaj sem si pomagala s spremenljivko konec, ki nam pove, da se je igralec zaletel v črno oviro. Za ponovno igro pa igralec pritisne gumb space. To sem sprogramirala na »staga« in sicer z gradnikom »when space key pressed«. Gumb ga vrne na začetno stran, kjer lahko ponovno začne igro. Na začetno stran, pa igralec pride tudi, ko uspešno zaključi igro. To naredimo z novim stavek if. Na oder se izpišeta še dva oblačka, ki sem ju naredila z gradnikoma za izgled »say for 2 sec«. Prvi oblaček je »ČESTITAMO!!!«, drugi pa »USPRRELO TI JE!«.

3.4 KODA – komentirane zaslonske slike ustreznih skript objektov

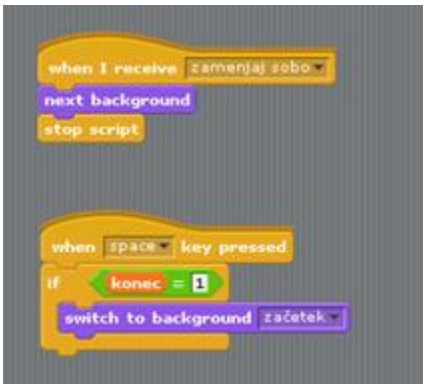


KOMENTAR: to je skripta lika netopirja. Začne se, ko igralec z miškinim kazalcem klikne na lik, takoj za tem se zamenja ozadje. To se zgodi samo takrat, ko je vrednost spremenljivke klik enaka 0.
Pred tem, sem spremenljivko konec nastavila na 0, kar pomeni začetek igre (vrednost 1 je konec igre). Skripta liku določa pozicijo po koordinati x in y, velikost, način premikanja – lik sledi miškinemu kazalcu, hitrost lika – lik ima na začetku hitrost nastavljeno na 0.7, povečuje se mu s časom, če se ga

dotakne miškin kazalec in iz votline v votlino (vidimo v nadaljevanju). Če se lik dotakne bele puščice se znajde v naslednji votlini in na poziciji določeni s koordinatama. S premikanjem po votlinah, se povečuje spremenljivka votlina. Z namenom, da se ugotovi, kdaj smo v zadnji votlini.



Če pa se lik dotakne črne barve, je igre konec, kar ugotovimo z vrednostjo spremenljivke konec (vrednost je 1). Ostali spremenljivki votlina in klik pa nastavimo na 0 (resetiramo). Igralec dobi obvestilo, da je igre konec in da za ponovno igro pritisne gumb space.
Če igralec uspešno konča igro, mu čestitamo. Vrednosti spremenljivk votline in klika pa nastavimo na 0, vrednost konec pa na 1.



KOMENTAR: to je zaslonska slika za ozadja oz. votline. Ko se lik dotakne bele puščice, se zamenja ozadje – lik prestavi v naslednjo votlino.
Če igralec pritisne space, se vrne na začetek igre oz. na naslovno stran. To se zgodi samo, kadar je vrednost spremenljivke konec enaka 1.

KOMENTAR: to je skripta za belo puščico, katere se mora igralec dotakniti, da pride v naslednjo votlino, hkrati se hitrost liku poviša.

3.5 POVEZAVA do datoteke s kodo

BYOB IGRA.ypr

3.6 IDEJE ZA NADGRADNJO

Igrico lahko nadgradimo že s tem, da ji dodamo več votlin in jo s tem podaljšamo. Preuredimo lahko tudi, da se lik premika s pomočjo tipk in ne miškinega kazalca. Igri lahko dodamo nov lik, v katerega se lik ne sme zaleteti ali stvari, ki jih mora lik pobirati. Možno je odpraviti tudi manjše napake.

4 KVADRATI (8.)

4.1 OPREDELITEV PROBLEMA

4.1.1 Opis problema

Podane so celoštevilske koordinate n točk, ki ležijo v ravnini. Vsak par teh točk določa pravokotnik, ki ima s koordinatnima osema vzporedne stranice in ena točka leži v njegovem spodnjem levem oglišču, druga pa v zgornjem desnem oglišču. Napiši program, ki učinkovito poišče število takih pravokotnikov, ki so kvadrati. Štej tudi izrojene kvadrate, pri katerih spodnje levo in zgornje desno oglišče sovpadata.
Število točk n je največ 100. Območje, na katerem ležijo točke v ravnini, je znotraj kvadrata:
$-10\,000 \leq x \leq 10\,000$, $-10\,000 \leq y\ 10\,000$

4.1.2 Podatki

Vhodni podatki
Vhodni podateki so celoštevilske koordinate, ki ležijo v ravnini. Vsak par teh točk določa pravokotnik, ki ima s koordinatnima osema vzporedne stranice in ena točka leži v njegovem spodnjem levem ogljišče, druga pa v zgornjem desnem ogljišču.

Izhodni podatki
Izhodni podatek je število takih pravokotnikov, ki so kvadrati. Štejemo tudi izrojene kvadrate, pri katerih spodnje levo in zgornje desno ogljišče sovpadata.

4.1.3 Izrojeni kvadrat

4.1.4 Primeri pričakovanih podatkov in rezultatov

VHOD

x = [1, 2, 5]
y = [1, 2, 4]

IZHOD
5

4.1.5 Nabori podatkov

KLASIČNI PODATKI

VHOD
x = [1]
y = [1]
IZHOD
1

IZHOD

VHOD
x = [7, 2, 5]
y = [7, 2, 3]
IZHOD
5

VHOD
x = [10, 21, 5]
y = [2, 22, 7]
IZHOD
5

ZLOBNI PODATKI

VHOD
x = []
y = []
IZHOD
0

KOMENTAR:
Če ni podatka, ni kvadratov.

VHOD
x = [13, 24, 5]
y = [2, 4]
IZHOD
raise Exception ('Dolžini nista enaki.')
Exception: Dolžini nista enaki.

KOMENTAR:
Ker seznama nista enakih dolžin, uporabnik dobi obvestilo.

VHOD
x = [76876,-24,5]
y = [2,4,-46]
IZHOD
raise Exception ('Vrednost x prevelika na mestu ' + str(stevila+1))
Exception: Vrednost x prevelika na mestu 1

KOMENTAR:
Ker je prva vrednost v seznamu x prevelika, uporabnik dobi obvestilo.

4.1.6 Navodila uporabniku

Uporabnik mora za parametra navesti dva seznama. V prvi seznam se shranijo koordinate x-a, v drugi seznam pa se shranijo koordinate y-a. Seznama morata biti enake dolžine. Območje, na katerem ležijo točke v ravnini, je znotraj kvadrata: -10 000 ≤ x ≤ 10 000, -10 000 ≤ y 10 000. Zato mora uporabnik navesti ustrezna x in y.

4.1.7 Zaslonski sliki

```
>>> kvadrati([1,2,3],[5,3])
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    kvadrati([1,2,3],[5,3])
  File "C:\Documents and Settings\Nuta\Desktop\EvaGveniverijaKapl
an\kvadrati\kvadratiiii.py", line 16, in kvadrati
    raise Exception ('Dolžini nista enaki.')
Exception: Dolžini nista enaki.
>>>
```

```
>>> kvadrati([165,24,35,32],[87,41,25, 124])
4
>>> kvadrati([7,2,5],[7,2,3])
5
>>> kvadrati([10,21,5],[2,22,7])
5
>>> kvadrati([1,2],[1,2])
4
>>>
```

4.2 RAZLAGA KODE

Funkcija »kvadrati« ima dva parametra. Parametra sta seznama koordinat x in y.

Najprej se z zanko »for« sprehodimo po obsegu dolžine x in y. S tem preverimo, če se nahajamo znotraj kvadrata -10 000 ≤ x ≤ 10 000, -10 000 ≤ y 10 000. Če se ne nahajamo, uporabnik dobi obvestilo, kateri x oz. y ne ustreza, da ga lahko hitro popravi.
Potem pa s stavkom »if« preverajamo dolžini seznama (preverjama, če sta parametra enakih dolžin), če nista enaka, uporabnik dobi obvestilo, da dolžini nista enaki.
Števec, ki bo štel naše kvadrate nastavimo na 0. Z zanko for se sprehodimo po seznamu x in po seznamu y. Potem s pomočjo stavka »if« primerajamo koordinati x in koordinati y. Če sta razdalji enaki (moramo upoštevati absolutno vrednost), potem gre za kvadrat, saj sta stranici enaki. Povečamo še števec, ki smo ga na začetku nastavili na 0. Funkcija vrne število kvadratov znotaj kvadrata -10 000 ≤ x ≤ 10 000, -10 000 ≤ y 10 000.

4.3 LEPO OBLIKOVANA KODA PROGRAMA in KOMENTARJI

```
def kvadrati(x, y):
    """Funkcija izpiše število kvadratov."""

    for stevila in range(len(x)):
        if x[stevila] > 10000:
            raise Exception ('Vrednost x prevelika na mestu ' +  str(stevila+1))
        if x[stevila] < -10000:
            raise Exception ('Vrednost x premajhna na mestu ' +  str(stevila+1))
    for stevila in range(len(y)):
        if y[stevila] > 10000:
            raise Exception  ('Vrednost y prevelika na mestu ' +  str(stevila+1))
        if y[stevila] < -10000:
            raise Exception  ('Vrednost y premajhna na mestu ' +  str(stevila+1))

    if len(x) != len(y):
        raise Exception ('Dolžini nista enaki.')
    stevec = 0
    for i in range(len(x)):
        for j in range(len(y)):
            if abs(x[i] -x[j]) == abs(y[i] -y[j]):
                stevec = stevec + 1
    return stevec
```