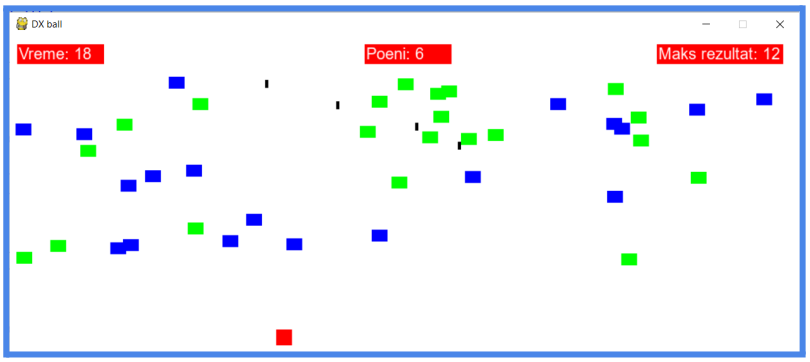
****

Igrica 4

****

OPIS IGRE:



Igrač upravljanjem miša pomera po horizontal crveni kvadrat iz kojeg može da puca na levi taster miša kako bi sklanjao blokove koji su se generisali na sceni (1000x400). Cilj je osvojiti što više poena za vreme od 30 sec, pri čemu blokovi imaju različit broj poena i generisani su na slučajne pozicije. Igrač ima neogrničen broj metaka. U igri se ispisuju vreme, poeni i maksimalan reultat.

OPIS IMPLEMENTACIJI IGRE:

- Dva različita načina za zaustavljanje igre (istek vremena, i zatvaranje prozora)
- Kolizija metka i bloka - ranije smo koliziju radili preko uporedjivanja koordinata jer smo likove pravili preko draw()
- Moguće je detektovati koliziju dva objekta u pygameu i programski, ukoliko oni nasledjuju klasu Sprite
- Sprite kao klasa uglavnom postoji u svim grafičkim bibliotekama jer predstavlja objekat koji ima sirinu visinu, i neko ponašanje, npr pomeranje, skok, itd
- pygame.sprite.Sprite je mesto gdje je definisana ta klasa sa visible sprajtovima
- Ideja je da metke napravimo kao klasu koja je izvedena iz Sprite klase
- Slično i za blokove
- Prednosti ovog pristupa, pored same kolizije jesu da ćemo moći napraviti samu animaciju na lakši način, npr napravićemo u toj klasi Metak metoda update() koja će da se poziva u svakom frejmu i metak ukoliko je ispaljen će se kretati
- Što se blokova tiče, prednost je i u tome što na nivou klase možemo da definišemo polja u toj klasi kao što su poeni i boja, te tako može da postoji nekoliko vrsta objekata te klase sa različitim brojem poena
- Dodatno, može da se doda bilo kakvo ponašanje kao metoda ili neki property kao polje klase (npr, polje zvuk, koje cuva zvuk koji se pušta za taj specifično uništen blok)

PODSEĆANJE / UPOZNAVANJE NA OOP (ovde samo rečenicu 2 reći ako učenik nije upoznat sa tim, narednih časova će se detaljnije pričati o tome, jer se u ovom prvom delu ne koriste ovi elementi realizacije):

- Generalno
- OOP u pythonu - sintaksa

IMPLEMENTACIJI IGRE (delovi):

- Deo 1 - setap igre
    - **Osnovne postavke pygame prozora**

    - **Neophodne promenljive za boje, funkcije, odbrojavanje vremena**
    - **Glavna while petlja za iscrtavanje i mesto gde ce se kasnije dodavati game logika**
    - **Dodati razlicito zatvaranje prozora (kada se klikne na X prozora, treba skoroz da se zatvori, a kada istekne vreme, igra se samo zaustavi)**
- Deo 2
    - Dodavanje igraca,
    - Dodavanje poena, max rezultata
    - Klasa za igraca - i on mora da bude sprite
- Deo 3
    - Klasa za metak - i on mora da bude sprite
    - Klik misa
    - Ispaljivanje metaka
- Deo 4 - Dodavanje blokova
    - Klasa za blokove
    - Dodatne liste za pracenje koji su blokovi uklonjeni a koji ne

IMPLEMENTACIJA - DEO 1:

```
# korak 1 - osnovni kod za pravljenje prozora
import pygame as pg
import pygamebg as pgbg
# pravljenje prozora
sirina = 1000
visina = 400
prozor = pgbg.open_window(sirina, visina, "DX ball")
pgbg.wait_loop()

#korak 2 - dodavanje promenljivih koje nam trebaju odmah nakon importa

import pygame as pg
import pygamebg as pgbg

# Boje koje ce nam trebati
BLACK = (0, 0, 0)
WHITE = (255, 255, 255)
RED = (255, 0, 0)

# pravljenje prozora
sirina = 1000
#korak 3 - dodavanje promenljive za fontove, nakon napravljenog prozora

prozor = pgbg.open_window(sirina, visina, "DX ball")

# fontovi za ispis
fontVreme = pg.font.SysFont("Arial", 20)

pgbg.wait_loop()
```

```
#korak 4 - dodavanje pomoćnih funkcija odmah nakon importa

import pygame as pg
import pygamebg as pgbg

#pomocne funkcije za ispis informacija u igri
def nacrtajVreme(vreme):
    pg.draw.rect(prozor, RED, (10, 10, 110, 25))
    slikaVreme=fontVreme.render("VREME: {}".format(vreme), True, WHITE)
    prozor.blit(slikaVreme, (12, 10))

#korak 5 - dodati promenljive za fps i glavnu while petlju igre nakon
promeljivih za fontove

fontMaksRezultat = pg.font.SysFont("Arial", 20)
# fleg za detekciju kraja igre
krajIgre = False
# sat za fps
sat = pg.time.Clock()
while not krajIgre:
    # hvatanje, detekcija i obrada dogadjaja
    for event in pg.event.get():
        if event.type == pg.QUIT:
            krajIgre = True
    # logika igre - to be added!

    # iscrtavanje prozora
    prozor.fill(WHITE)
    pg.display.update()
    sat.tick(60)

pgbg.wait_loop()

#korak 6 - dodati pozive navedenih funkcija i napraviti promenljive

…
sat = pg.time.Clock()
poeni = 0
#odbrojavanje vremena
maxVreme = 30
     …
    prozor.fill(WHITE)

    nacrtajVreme(maxVreme)

    pg.display.update()
```

```
#korak 7 - dodati događaj na svakih 1000 milisekundi za odbrojavanje vremena
…
maxVreme = 30
timeEvent = pg.USEREVENT + 1
pg.time.set_timer(timeEvent, 1000)
while not krajIgre:
…
                krajIgre = True
            elif event.type == timeEvent:
                maxVreme -= 1
                if maxVreme == 0:
                    krajIgre = True
        # logika igre - to be added!
        …
#korak 7 - dodati kod za zatvaranje prozora kako bi se razlikovao kraj igre kada je isteklo vreme (prozor ostaje da stoji samo korisnik nece moci vise da puca) i kada je zaista kliknuto na X (skroz treba da se zatvori prozor)

pg.time.set_timer(timeEvent, 1000)
#kod za kraj
kod = 0
…
            if event.type == pg.QUIT:
                kod = -1
                krajIgre = True
                …
                if maxVreme == 0:
                    kod = -2
                    krajIgre = True
    …
    sat.tick(60)
# provera kada se skroz gasi prozor a kada samo zaustavlja igra
if kod == -1:
    pg.quit()
else:
    pgbg.wait_loop()
```

DEO 2 - IMPLEMENTACIJA:

```
# korak 1 - dodavanje dodatnih fontova

# fontovi za ispis
fontVreme = pg.font.SysFont("Arial", 20)
fontPoeni = pg.font.SysFont("Arial", 20)
fontMaksRezultat = pg.font.SysFont("Arial", 20)
…
```

#korak 2 - dodavanje pomoćnih funkcija odmah nakon one za vreme

```
    slikaVreme=fontVreme.render("VREME: {}".format(vreme), True, WHITE)
    prozor.blit(slikaVreme, (12, 10))
def nacrtajPoene(poeni):
    slikaPoeni=fontPoeni.render("POENI: {}".format(poeni), True, WHITE)
    pg.draw.rect(prozor, RED, (450, 10, 110, 25))
    prozor.blit(slikaPoeni, (452, 10))
def nacrtajMaksRezultat():
    rez = procitajMaxRez()
    slikaMaksRezultat = fontMaksRezultat.render("MAX REZULTAT:
{}".format(rez), True, WHITE)
    pg.draw.rect(prozor, RED, (800, 10, 185, 25))
    prozor.blit(slikaMaksRezultat, (802, 10))
def procitajMaxRez():
    file = open("score.txt", "r")
    rez = file.readline()
    file.close()
    return rez
def upisiNoviMaxRez(p):
    file = open("score.txt", "w")
    file.write(str(p))
    file.close()
# Boje koje ce nam trebati
BLACK = (0, 0, 0)
```

#korak 3 - dodavanje poziva tih funkcija

```
    prozor.fill(WHITE)
    nacrtajVreme(maxVreme)
    nacrtajPoene(poeni)
    nacrtajMaksRezultat()
    pg.display.update()
```

#korak 4 - dodavanje fajl score.txt u isti folder kao i ovaj py fajl i dodati upis u taj fajl (npr broj 12)

```
12
```

#korak 5 - dodavanje promenljivih za boje koje nam trebaju odmah nakon ostalih boja koje smo naveli, boje nam trebaju za igraca i blokove

```
WHITE = (255, 255, 255)
RED = (255, 0, 0)
BLUE = (0, 0, 255)
GREEN = (0, 255, 0)
YELLOW = (255, 255, 0)
# pravljenje prozora
sirina = 1000
```

```
#korak 6 - klasa za igraca

class Igrac(pg.sprite.Sprite):

    def __init__(self):
        super().__init__()
        self.image = pg.Surface([20, 20])
        self.image.fill(RED)
        self.rect = self.image.get_rect()

    def update(self):
        pozicija = pg.mouse.get_pos()
        self.rect.x = pozicija[0]
```

OBJASNJENJE KLASE PLAY

- Klasa / fabrika, objekti klase / proizvodi fabrike, koji se prave po nekom pravilu ili standardu / konstruktror klase
- Zasto postoji ovaj pristup, zasto ne moze sve da bude napisano kroz funkcije? Objasniti nakon sto se pita ucenik sta misli
- Ovde imamo opciju da za igraca i ne pravimo klasu, nego da uvek imamo pozicije, kao sto je to bio slucaj sa prethodnim slicnim igricama
- Na ovaj nacin odvajamo one podatke koje su bitne za ovaj deo igre, sve vezano za igraca ide u ovu klasu, svo programiranje se desava tu, a u glavnim delovima igre, koriste se samo pozivi funkcija koje ovde napravimo da opisuju neku promenu
- Polje klase u ovom slucaju moze biti bilo sta sto u init() funkciji setujemo sa operatorom tackica. Polje nam sluzi da skladistimo neku vrednost koju mozemo koristiti sa spoljasnjim svetom, koju spoljasnji svet moze da procita, ali ne i da promeni. Polja se menjaju unutar klasa.
- Ovde je ideja da igrac bude takav sprajt koji ima pozicije x i y jer ce se one menjati. Ova klasa mogla je da ima samo dva polja x i y, i da se onda samo rectangle crta. Medjutim, na ovaj nacin, pravimo polje rectangle koji je po tipu surface, a samim tim ce se i iscrtavati kroz poziv odredjenih funkcija, pa je jednostavnije
- Da bi se napravio rectangle kao surface, prvo se pravi surface u odredjenim dimenzijama, boji se, a zatim se “konvertuje” u rectangle. Takav rectangle ima polja x i y, kome mozemo da pristupimo (i da ih procitamo i da upisujemo u njih nove vrednosti - na nivou klase)
- Organizovati kod tako da je redosled sledeci
    - Importi
    - Klase
    - Promenljive za boje
    - Dodatne pomocne funkcije van klasa
    - Promenljive za pravljenje prozora
    - Glavni tok programa
- Klasa nasledjuje Sprite klasu (po sintaksi, to se navodi u zagradama nakon imena klase)
    - Objasniti nasledjivanje ako uceniku nije jasno na nekom jednostavnom primeru

    - Mi ovde koristimo Sprite klasu kao baznu zbog detekcije kolizije izmedju svih objekata u igrici
- Init funkcija / mora da bude u svakoj klasi, konstruktor klase, tj funkcija koja govori kako ce se praviti objekti ove klase
- Sve ostale funkcije u klasi su definicija ponasanja objekta koji se izvede iz klase (npr klasa pas koja ima metodu lajati(), a u glavnom porgramu mozemo napraviti objekat klase pas, npr Lesi, i pozvati funkciju Lesi.lajati())

```
#korak 7 - pravljenje jednog objekta ove klasa i setovanje i azuriranje pozicije, objekat se pravi pre while loopa

#kod za kraj
kod = 0
igrac = Igrac()
igrac.rect.y = 370
while not krajIgre:

#korak 8 - da bi se nacrtao igrac, kao i blokovi i meci, sve cemo drzati u jednoj zajednickoj grupi, koja ce da bude zaduzena za njihov prikaz. Jednom funkcijom cemo prikazivati sve njih zajedno

fontMaksRezultat = pg.font.SysFont("Arial", 20)
# Grupa za sve sprajtove u igri
sviSprajtovi = pg.sprite.Group()

#korak 9 - dodavanje igraca u grupu

igrac = Igrac()
sviSprajtovi.add(igrac)
igrac.rect.y = 370

#korak 10 - azuriranje pozicija svih sprajtova u grupi (za sada samo igraceva). Ako ovo nije podeseno, pomeranjem misa, igrac se nece mrdati. Update() poziva update() od svih svojih sprajtova u grupi

    # logika igre - to be added!
    sviSprajtovi.update()
    # iscrtavanje prozora

#korak 11 - iscrtavanje svih sprajtova u grupi (za sada samo igraceva). Ako ovo nije podeseno, igrac se nece iscrtavati. Draw funkcija poziva blit za sve sprajtove u ovoj grupi sprajtova

    nacrtajMaksRezultat()
    sviSprajtovi.draw(prozor)
    # update za fps
    pg.display.update()
```

DEO 3 IMPLEMENTACIJA:

```
# korak 1 - dodavanje klase za metak

class Metak(pg.sprite.Sprite):
    def __init__(self):
        super().__init__()

        self.image = pg.Surface([4, 10])
        self.image.fill(BLACK)
        self.rect = self.image.get_rect()

    def update(self):
        self.rect.y -= 3
```

OBJASNJENJE:

- Klasa nasledjuje Sprite, u konstruktoru pozivamo konstruktor parenta i dodatno dodajemo sta zelimo da se specijalno desava u ovoj klasi
- Pravimo sliku preko surface, gde definisemo velicinu koju filujemo da je crna (proveriti da smo u promenljivima dodali crnu boju)
- Pravimo rectengle na osnovu napravljene slike
- Zasto ne pravimo samo sa draw.Rect()? Zato sto preko Sprite klase imamo detekciju dodira koja nam treba radi sklanjanja blokova

```
#korak 2 - pravljenje metka pri kliku misa

while not krajIgre:
    # hvatanje, detekcija i obrada dogadjaja
    for event in pg.event.get():
        if event.type == pg.QUIT:
            kod = -1
            krajIgre = True
        elif event.type == pg.MOUSEBUTTONDOWN:
            # napravi metak
            metak = Metak()
            # pozicioniranje metka gde je i igrac
            metak.rect.x = igrac.rect.x
            metak.rect.y = igrac.rect.y
            # dodajemo metak u listu svih sprajtova zbog iscrtavanja i
detekcije kolizije kasnije
            sviSprajtovi.add(metak)
        elif event.type == timeEvent:
            maxVreme -= 1
            if maxVreme == 0:
                kod = -2
                krajIgre = True
```

OBJASNJENJE:

- Meci se kreiraju pri kliku i pri tome se pozicioniraju u levi cosak kod igraca. Promenom x pozicije, podesiti ga da bude npr na srediti igraca
- Pri kreiranju, dodaju se u listu koja je napravljena na proslom casu, a kreira se pre ovog whilea, tako da je bezbedno da se na ovom mestu dodaju

```
#korak 3 - pravljenje liste za sve metke (sad vec treba da radi pucanje metaka nakon ovog koraka!!!!!)

1-Pravimo listu kod liste za sve sprajtove:

# Grupa za sve sprajtove u igri
sviSprajtovi = pg.sprite.Group()
listaSvihMetaka = pg.sprite.Group()
2-Dodajemo metak u listu
            …
            metak.rect.y = igrac.rect.y
            # dodajemo metak u listu svih sprajtova zbog iscrtavanja i detekcije kolizije kasnije
            sviSprajtovi.add(matak)
            listaSvihMetaka.add(metak)
        elif event.type == timeEvent:
…
```

OBJASNJENJE:

- Meci se cuvaju u listi svih metaka da bismo to mogli da koristimo za kolizuju, da odatle foreachujemo svaki i vidimo koji je u koliziji sa nekim sprajtom.

```
#korak 4 - Brisemo metke koji su dosli na vrh

# logika igre - to be added!
    sviSprajtovi.update()

    for metak in listaSvihMetaka:

        # brisemo metke koji su dosli do vrha
        if metak.rect.y < -10:
            listaSvihMetaka.remove(metak)
            sviSprajtovi.remove(metak)

    # iscrtavanje prozora
```

```
#korak 5 - dodajemo sprajt pre while petlje koji ce da bude primer za koliziju sprajtova i dodajemo ga u listu sprajtova svih, i u listu sprajtova za enemije
```

```
igrac = Igrac()
sviSprajtovi.add(igrac)

igrac.rect.y = 370

igrac2 = Igrac()
igrac2.rect.x = 500
igrac2.rect.y = 200
sviSprajtovi.add(igrac2)
listaSvihEnemija = pg.sprite.Group()
listaSvihEnemija.add(igrac2)

while not krajIgre:

#korak 6 - poziv funkcije za koliziju
for metak in listaSvihMetaka:

        lista = pg.sprite.spritecollide(metak, listaSvihEnemija, True)

        for el in lista:
            listaSvihMetaka.remove(metak)
            sviSprajtovi.remove(metak)

        # brisemo metke koji su dosli do vrha
        if metak.rect.y < -10:
            listaSvihMetaka.remove(metak)
            sviSprajtovi.remove(metak)
```

OBJASNJENJE:

- Ova verzija se radi sa fiksnim Sprajtom, da se demonstriraju funkcije koje trebaju za rad sa kolizijom, pa ce se sledeci cas ovaj Sprajt izbrisati a dodati blokovi
- Funkcija **spritecollide(sprite, group, dokill, collided** = **None) -> Sprite_list** radi na osnovu uporedjivanja rect propertija u sprajtu. Ima parametre koji podrazumevaju da se prosledjuje **sprajt** koji se proverava da li se dodiruje sa grupom sprajtova **group**, dokill je boolean, koji ako je True, automatski izbacuje iz grupe element koji je upravo kolajdovan
- U ovom slucaju taj player2 se mrda kako mrdamo i playera1, sto je ok samo za demonstraciju kolizije.

DEO 4 IMPLEMENTACIJA:

```
# korak 1 - dodavanje klase za blok bez distinkcije za razlicite poene

class Blok(pg.sprite.Sprite):

    def __init__(self):
        super().__init__()
```

```
        self.image = pg.Surface([20, 15])
        self.image.fill(BLUE)

        self.rect = self.image.get_rect()
```

```
#korak 2 - brisemo sprajt za igraca 2 i pravimo blok blokova

import pygame as pg
import pygamebg as pgbg
import random


igrac = Igrac()
sviSprajtovi.add(igrac)
igrac.rect.y = 370

~~igrac2 = Igrac()~~
~~igrac2.rect.x = 500~~
~~igrac2.rect.y = 200~~
~~sviSprajtovi.add(igrac2)~~
listaSvihEnemija = pg.sprite.Group()
~~listaSvihEnemija.add(igrac2)~~

for i in range(50):
    blok = Block()

    # pravimo random lokaciju
    blok.rect.x = random.randint(0, 1000)
    blok.rect.y = random.randint(50, 300)

    # dodajemo u liste
    listaSvihEnemija.add(blok)
    sviSprajtovi.add(blok)


while not krajIgre:
```

OBJASNJENJE:

- Ovde bi trebalo da radi da se skidaju plavi blokovi mecima
- Neophodno je bilo da se izbace linije koda kojima se pravio sprajt igrac2 i da se zamene dodavanjem 50 novih blokova

```
#korak 3 - Dodavanje poena

        for el in lista:
            listaSvihMetaka.remove(metak)
            poeni += 1
            sviSprajtovi.remove(metak)
```

OBJASNJENJE:

- Na mestu gde se detektuje kolizija, tu se dodaju poeni
- Promenljiva poeni je vec ranije setovana, pre while petlje
- Ovaj red ce se menjati, kada napravimo vise vrsta blokova

```
#korak 4 - Dodavanje vise vrsta blokova koji se razlikuju po boji i po poenima koje donose

class Block(pg.sprite.Sprite):
    def __init__(self, bojaBloka, poeniBloka):
        super().__init__()

        self.image = pg.Surface([20, 15])
        self.image.fill(bojaBloka)

        self.rect = self.image.get_rect()

        self.boja = bojaBloka
        self.poeni = poeniBloka
```

OBJASNJENJE:

- Pre adaptacije ove klase na nove funkcionalnosti, proveriti da li ucenik ima ideju kako da napravi. Ako krene da odmah pravi celu novu klasu, sa ovim poljima, navesti na to da se razmisli da li je to neophodno, da li moze da se iskoristi vec postojeca klasa. Da li izvesti iz nje novu ili ne? (ako postoji nesto od funkcionalnosti sto je spec samo za jednu vrsu blokova, onda moze nasledjivanje, pa da u baznoj ostane samo ono sto je za sve blokove zajednicko)
- Polje boja i ne mora da se setuje, jer ne znamo za sta jos moze da se koristi (jer se vec iskoristi za fill funkciju rectanglea), ali setovacemo da imamo i to polje
- Kada se adaptira klasa, ispitati ucenika kako se menja nacin pravljenja objekata te klase

```
#korak 5 - Pravljenje vise vrsta blokova u for naredbi u kome se pravili blokovi. Pravimo mali broj negativnih blokova, a ostalo su blokovi sa 1 i 2 poena.

listaSvihEnemija = pg.sprite.Group()

for i in range(50):

    if i%5 == 0:
        blok = Block(RED, -1)
    elif i%2 == 0:
        blok = Block(GREEN, 1)
```

```
    else:
        blok = Block(BLUE, 2)

    # Set a random location for the block
    blok.rect.x = random.randint(0, 1000)
    blok.rect.y = random.randint(50, 300)

    # Add the block to the list of objects
    listaSvihEnemija.add(blok)
    sviSprajtovi.add(blok)
```

OBJASNJENJE:

- Verovatnoca raspodele koji blokovi ce biti češći može biti proizvoljna.

```
#korak 6 - ažuriranje poena iz bloka

for metak in listaSvihMetaka:

        lista = pg.sprite.spritecollide(metak, listaSvihEnemija, True)

        for el in lista:
            listaSvihMetaka.remove(metak)
            poeni += el.poeni
            sviSprajtovi.remove(metak)

        # brisemo metke koji su dosli do vrha
        if metak.rect.y < -10:
            listaSvihMetaka.remove(metak)
            sviSprajtovi.remove(metak)
```

```
#korak 7 - smisljanje ostalih funkcionalnosti i implementacija
```

IDEJA:

- Može da postoji blok koga treba 2 puta pogoditi da bi se sklonio sa podloge.
- Izvede se novi tip bloka iz klase (ideja je da vidi kako bi se radilo to izvodjenje)

```
class DblBlok(Block):
    def __init__(self):
        super().__init__(GRAY, 5)
        self.brojPogodaka = 2
```

- Podesi se kreiranje tih novih blokova
- Verovatnoca opet proizvoljna

```
    if i%5 == 0:
        blok = Block(RED, -1)
    elif i%2 == 0:
        blok = Block(GREEN, 1)
    elif i % 3 == 0:
        blok = DblBlok()
    else:
        blok = Block(BLUE, 2)
```

- Detekcija i obrada tog dogadjaja

```
for el in lista:
            if el.boja != GRAY:
                listaSvihMetaka.remove(metak)
                poeni += el.poeni
                sviSprajtovi.remove(metak)
            else:
                # boja je gray, to znaci je dbl
                # provera koliko je metaka pogodjeno
                if el.brojPogodaka == 2:
                    el.brojPogodaka -= 1
                else:
                    poeni += el.poeni
                listaSvihMetaka.remove(metak)
                sviSprajtovi.remove(metak)
```

- Ovo ce da radi lepo, i trebace 2 metka da se dobije 5 poena na sivom bloku, ali ne valja sto odmah nestane blok nakon prvog metka
- To je zbog collide funkcije u kojoj je stavljeno True. Kada se prebaci na False, ne brise se pogodjeni sprajt, tako da sivi ostaje, ali se ne brise ni jedan pogodjen sprajt
- To moze da se ostavi na uceniku da istrazi kako bi se moglo uraditi.