SREDNJA ELEKTROTEHNIČKA ŠKOLA
SARAJEVO

MATURSKI RAD

TEMA: Tetris

Mentor :
Delić Adnan dipl. el. ing.

Učenik:
Bašović Arman

Sarajevo, april 2022.

Sadržaj

1. Tetris

1.1 Šta je Tetris?

Tetris je logička video igra koju je kreirao sovjetski softverski inženjer Aleksej Pajitnov 1984. godine. Objavilo ju je nekoliko kompanija za više platformi, najistaknutije tokom rasprave oko prisvajanja prava kasnih 1980-ih. Nakon značajnog perioda objavljivanja od strane Nintenda, prava su vraćena Pajitnovu 1996. godine, koji je zajedno sa Henkom Rodžersom osnovao kompaniju Tetris kako bi upravljao licenciranjem.

U Tetrisu, igrači završavaju linije pomicanjem komada različitog oblika (tetromina), koji se spuštaju na polje za igru. Završene linije nestaju i igraču daju bodove, a igrač može nastaviti popunjavati prazna mjesta. Igra se završava kada je polje za igru popunjeno. Što duže igrač može odlagati ovaj ishod, to će njegov rezultat biti veći. U igrama za više igrača, igrači moraju trajati duže od svojih protivnika; u određenim verzijama, igrači mogu nanijeti kazne protivnicima ispunjavanjem značajnog broja linija. Neke verzije dodaju varijacije pravila, kao što su trodimenzionalni prikazi ili sistem za rezervisanje komada.

1.2 Princip rada Tetrisa

Tetris se prvenstveno sastoji od polja za igru u kojem se komadi različitih geometrijskih oblika, zvani "tetromino", spuštaju s vrha polja. Tokom ovog spuštanja, igrač može pomicati figure bočno i rotirati ih sve dok se ne dodiruju dno polja ili padaju na komad koji je bio postavljen prije njega. Igrač ne može ni usporiti padajuće komade niti ih zaustaviti, ali ih može ubrzati u većini verzija. Cilj igre je koristiti figure za stvaranje što više horizontalnih linija blokova. Kada je linija završena, ona nestaje, a blokovi postavljeni iznad pada za jedan rang. Upotpunjavanje linija daje bodove, a akumuliranje određenog broja poena pomiče igrača na viši nivo, što povećava broj bodova datih po završenoj liniji.

U većini verzija, brzina padajućih komada raste sa svakim nivoom, ostavljajući igraču manje vremena za razmišljanje. Igrač može očistiti više linija odjednom, što može zaraditi bonus bodove u nekim verzijama. Moguće je završiti do četiri linije istovremeno uz korištenje tetromina u obliku slova I; ovaj potez se zove "Tetris" i osnova je naslova igre. Ako igrač ne može učiniti da blokovi nestanu dovoljno

brzo, polje će početi da se popunjava, a kada figure stignu do vrha polja. polje i spriječi dolazak dodatnih komada, igra se završava. Na kraju svake igre, igrač dobija rezultat na osnovu broja završenih linija. Igra se nikada ne završava pobjedom igrača; igrač samo može sakupljati poene za svoj konačni rezultat prije neizbježnog gubitka.

Od 1996. godine, kompanija Tetris ima interno definisane specifikacije i smjernice kojih se izdavači moraju pridržavati da bi dobili licencu za Tetris. Sadržaj ovih smjernica uspostavlja takve elemente kao što su korespondencija dugmadi i radnji, veličina terena za igru, sistem rotacije i dr.

1.3 Razvoj Tetrisa kroz historiju

1.3.1 Koncepcija

1984, dok je pokušavao da rekonstruiše omiljenu slagalicu iz detinjstva sa pentominom, Pajitnov je zamislio igru koja se sastoji od spuštanja nasumičnih delova koje bi igrač okretao da popuni redove. Pajitnov je smatrao da bi igra bila nepotrebno komplikovana sa dvanaest različitih varijacija oblika, pa je koncept sveo na tetromine, od kojih postoji sedam varijanti.
Pajitnov je igricu nazvao Tetris, riječ nastala od kombinacije reči "tetra" (što znači "četiri") i njegovog omiljenog sporta, "tenis". Pošto Elektronika 60 nije imala grafički interfejs, Pajitnov je modelovao polje i delove koristeći razmake i zagrade. Shvativši da su završene linije brzo ispunile ekran, Pajitnov je odlučio da ih izbriše, stvarajući ključni deo igre Tetris. Ova rana verzija Tetrisa nije imala sistem bodovanja i nivoe, ali je njen kvalitet koji izaziva zavisnost razlikovao od ostalih slagalica koje je Pajitnov napravio.

1.3.2 Sticanje prava od strane Mirrorsoft i Spectrum HoloByte

Pajitnov je želeo da izveze Tetris, ali nije znao o poslovnom svetu. Zamolio je svog pretpostavljenog Viktora Brjabrina, koji je poznavao svijet izvan Sovjetskog Saveza, da mu pomogne u objavljivanju Tetrisa. Pajitnov je ponudio da prenese prava igre na Akademiju, i bio je oduševljen što je ovim ugovorom dobio neobaveznu naknadu od Brjabrina.
Brjabrin je 1986. godine poslao kopiju Tetrisa mađarskom izdavaču igrica Novotrade. Odatle su kopije igre počele da kruže preko disketa širom Mađarske i sve do Poljske. Robert Stein, internacionala, prodavač softvera za londonsku firmu Andromeda Software, vidio je komercijalni potencijal igre tokom posjete Mađarskoj u junu 1986. Nakon ravnodušnog odgovora Akademije Stein je kontaktirao Pajitnova i Brjabrina putem faksa kako bi dobio licencna prava. Stein se obratio izdavačima na sajmu potrošačke elektronike 1987. u Las Vegasu.

Gary Carlston, suosnivač Broderbunda, pronašao je kopiju i donio je u Kaliforniju. Uprkos entuzijazmu među svojim zaposlenicima, Broderbund je ostao skeptičan zbog sovjetskog porijekla igre. Isto tako, suosnivač Mastertronic-a Martin Alper izjavio je da "nijedan sovjetski proizvod nikada neće raditi u zapadnom svijetu". Stein je na kraju potpisao dva sporazuma: prodao je evropska prava izdavaču Mirrorsoft, i američka prava na Spectrum HoloByte. Ipak, on je prodao prava na dvije kompanije za £3,000 i tantijeme od 7,5 do 15% na prodaju.

Tetris je portiran na platforme uključujući Amigu, Atari ST, ZX Spectrum, Commodore 64 i Amstrad CPC. U to vreme nije spominjao Pajitnova i dolazi sa najavom „Proizvedeno u Sjedinjenim Američkim Državama, dizajnirano u inostranstvu“. Tetris je bio komercijalni uspjeh u Evropi i Sjedinjenim Državama: Mirrorsoft je prodao desetine hiljada primjeraka za dva mjeseca, a Spectrum HoloByte je prodao preko 100.000 jedinica u roku od godinu dana.

1.3.3 Sticanje prava od strane Nintenda i pravna bitka

1988. Spectrum HoloByte je prodao japanska prava na svoje kompjuterske igre i arkadne mašine Henku Rogersu iz Bullet-Proof Software-a, koji je tražio igre za japansko tržište. Mirrorsoft je prodao svoja japanska prava Atari Games podružnici Tengen-u. , koji je zatim prodao japanska prava na igranje arkada kompaniji Sega i prava konzole BPS-u, koji je objavio verzije za japanske računare, uključujući Nintendo Family Computer (Famicom), poznat izvan Japana kao Nintendo Entertainment System i MSX2. U ovom trenutku, skoro desetak kompanija je vjerovalo da imaju prava na Tetris, a Stein je zadržao prava na verzije kućnih računara. Iste godine, Nintendo se pripremao za lansiranje svoje prve prijenosne konzole, Game Boy.

Nintendo je privukao Tetris svojom jednostavnošću i postignutim uspjehom na Famicom-u. Rogers, koji je bio blizak tadašnjem Nintendovom predsjedniku Hirošiju Jamaučiju, nastojao je da dobije ručna prava. Nakon toga. neuspjeli pregovori sa Atarijem, Rogers je kontaktirao Steina u novembru 1988. Stein je pristao da potpiše ugovor, ali je objasnio da mora konsultovati Elorga prije nego što se vrati pregovorima sa Rogersom.

U martu 1989. Nintendo je poslao prekid i odustajanje Atari Games-u u vezi sa proizvodnjom NES verzije Tetrisa. Atari Games je kontaktirao Mirrorsoft i uvjeravao se da i dalje zadržavaju prava. Nintendo je, međutim, zadržao svoju poziciju. Kao odgovor, vlasnik Mirrorsofta Robert Maksvel izvršio je pritisak na lidera Sovjetskog Saveza Mihaila Gorbačova da poništi ugovor između Elorga i Nintenda. Uprkos pretnjama Belikovu, Elorg je odbio da popusti i istakao finansijske prednosti njihovog ugovora u odnosu na one potpisane. sa Steinom i Mirrorsoftom.

Dana 15. juna 1989. Nintendo i Atari Games su započeli pravnu bitku na sudovima u San Francisku. Atari Games je pokušao da dokaže da je NES računar, na šta ukazuje i njegov japanski naziv

"Famicom", skraćenica od "Family Computer". U ovom slučaju, početna licenca bi ovlastila Atari Games da pusti igru. Sudija Fern M. Smith je izjavio da Mirrorsoft i Spectrum HoloByte nikada nisu dobili izričito odobrenje za marketing na konzolama, i 21. juna 1989. presudio je u korist Nintenda, dajući im preliminarnu zabranu protiv Atari Gamesa u procesu. sljedećeg dana, Atari Games je povukao svoju NES verziju iz prodaje, a hiljade kertridža je ostalo neprodato u skladištima kompanije

1.3.4 Komercijalni uspeh i sticanje prava od strane Pajitnova

Kroz pravnu istoriju licence, Pajitnov je stekao reputaciju na Zapadu. Redovno su ga pozivali novinari i izdavači, preko kojih je otkrio da se njegova igra prodala u milionskim tiražima, od čega nije zaradio. Ipak, ostao je skroman i ponosan na igru, koju je smatrao "elektronskim ambasadorom dobrohotnosti". Godine 1991, Pajitnov i Pokhilko su emigrirali u Sjedinjene Države. Pajitnov se preselio u Sijetl, gde je proizvodio igre za Spectrum HoloByte.

U aprilu 1996, kao što je dogovoreno sa Akademijom deset godina ranije i nakon dogovora sa Rogersom, prava na Tetris vraćena su Pajitnovu. Pajitnov i Rogers su osnovali kompaniju Tetris u junu 1996. kako bi upravljali pravima na svim platformama, budući da su prethodni ugovori istekli. Pajitnov sada prima honorar za svaku Tetris igru i derivat prodatu širom svijeta. Godine 2002. Pajitnov i Rogers su osnovali Tetris Holding nakon kupovine preostalih prava na igru od Elorga, sada privatnog subjekta nakon raspada Sovjetskog Saveza. .

Kompanija Tetris sada posjeduje sva prava na brend Tetris i uglavnom je odgovorna za uklanjanje nelicenciranih klonova sa tržišta; kompanija redovno poziva Apple Inc. i Google da uklone ilegalne verzije iz njihove mobilne aplikacije trgovine.

U jednom značajnom slučaju iz 2012. godine, Tetris Holding, LLC protiv Xio Interactive, Inc., Tetris Holding i Tetris Company branili su svoja autorska prava protiv klona iOS-a, koji je uspostavio novi stav o procjeni kršenja klonova video igara na osnovu izgleda i osjećaja. U decembru 2005. Electronic Arts je kupio Jamdat, kompaniju specijalizovanu za mobilne igre. Jamdat je prethodno kupio kompaniju koju je osnovao Rogers 2001. godine, a koja je upravljala licencom Tetris na mobilnim platformama. Kao rezultat toga, Electronic Arts je imao 15-godišnju licencu za sva izdanja Tetrisa za mobilne telefone, koja je istekla 21. aprila 2020.

2. Python

2.1 Šta je Python?

Python je programski jezik visokog nivoa opšte namjene. Njegova filozofija dizajna naglašava čitljivost koda uz korištenje značajnog uvlačenja. Njegove jezičke konstrukcije i objektno orijentisani pristup imaju za cilj da pomognu programerima da pišu jasan, logičan kod za male i velike projekte.

Python se dinamički kuca i sakuplja smeće. Podržava više paradigmi programiranja, uključujući strukturirano (posebno proceduralno), objektno orijentirano i funkcionalno programiranje. Često se opisuje kao jezik "sa baterijama" zbog svoje opsežne standardne biblioteke. Guido van Rossum je počeo da radi na Pythonu kasnih 1980-ih kao nasljednika ABC programskog jezika i prvi ga je objavio 1991. kao Python 0.9.0. Python 2.0 je objavljen 2000. godine i uveo je nove funkcije kao što su razumijevanje liste, sakupljanje smeća za otkrivanje ciklusa, brojanje referenci i podrška za Unicode.

Python 3.0, objavljen 2008. godine, bio je velika revizija koja nije u potpunosti kompatibilna s prethodnim verzijama. Python 2 je ukinut s verzijom 2.7.18 2020. godine. Python se konstantno rangira kao jedan od najpopularnijih programskih jezika

2.2 Historija Python-a

Kasnih 1980-ih, Guido van Rossum je osmislio Python iz Centrum Wiskunde & CWI u Holandiji kao nasljednika ABC programskog jezika, koji je inspirisan SETL-om, sposobnog za rukovanje izuzetcima i povezivanje sa operativni sistem Amoeba. Njegova implementacija počela je u decembru 1989. godine. Van Rossum je snosio isključivu odgovornost za projekat, kao vodeći programer, sve do 12. jula 2018., kada je najavio svoj "trajni odmor" od odgovornosti kao Pythonov "doživotni dobronamerni diktator", titulu koju mu je zajednica Python dodijelila kako bi odražavala njegov dugoročna posvećenost kao glavni donosilac odluka u projektu. U januaru 2019. aktivni programeri Python jezgra izabrali su petočlano Upravno vijeće koje će voditi projekat.

Python 2.0 je objavljen 16. oktobra 2000. godine, sa mnogo važnih novih karakteristika.
Python 3.0, objavljen 3. decembra 2008, sa mnogim njegovim glavnim karakteristikama prebačenim na Python 2.6.xi 2.7.x. Izdanja Pythona 3 uključuju uslužni program 2to3, koji automatizira prevođenje

Python 2 koda u Python 3.

Kraj života Pythona 2.7 je prvobitno bio određen za 2015. godinu, a zatim je odgođen za 2020. zbog zabrinutosti da se veliki dio postojećeg koda ne može lako proslijediti na Python 3. Za njega neće biti objavljene dodatne sigurnosne zakrpe ili druga poboljšanja.

Sa prestankom životnog vijeka Pythona 2, podržani su samo Python 3.6.x i novije verzije.

Python 3.9.2 i 3.8.8 su ubrzani jer su sve verzije Pythona (uključujući 2.7) imale sigurnosne probleme koji su doveli do mogućeg daljinskog izvršavanja koda i trovanja web keša.

2.3 Funkcije unutar Python-a

2.3.1 Šta su funkcije?

Funkcije su blokovi organiziranog, ponovno iskoristivnog, koda koji ima zadatak obavljanje nekog složenijeg zadatka. Funkcije pružaju aplikacijama modularnost jer se funkcije mogu više puta iskoristiti unutar istog programa. Postoje ugrađene funkcije poput raw_input(), te korisničke funkcije.

2.3.2 Definisanje funkcija

Svaka funkcija u programskom jeziku Python mora započinjati ključnom riječi def nakon čega slijedi naziv funkcije te zatvorene zagrade tj. (). Funkcije mogu primati vrijednosti. stoga ukoliko funkcija prima neke vrijednosti one se navode unutar zagrada. Nakon zagrada obavezno slijedi znak dvotočja. Prva izjava nije nužna, te može poslužiti kao dokumentacija. Svaka sljedeća izjava mora biti ispravno uvučena te ići jedna ispod druge. Funkcija završava u trenutku kada razina identacije završi na nuli.

```
def ime_funkcije ( parametri ):
  “Dokumentacija funkcije”
        izjava 1
        izjava 2
        ...
        return [ izraz ]
```

2.3.3 Pozivanje funkcije

Definicijom je funkciji dodijeljen naziv, parametri koje prima, kod koji se izvršava te povratni izraz. Kako bi se ta funkcija izvršila nužno ju je pozvati, a poziva se na način da se upiše naziv funkcije te da se navedu potrebni parametri. Na slici 2.1. možemo vidjeti primjer funkcijskog poziva.

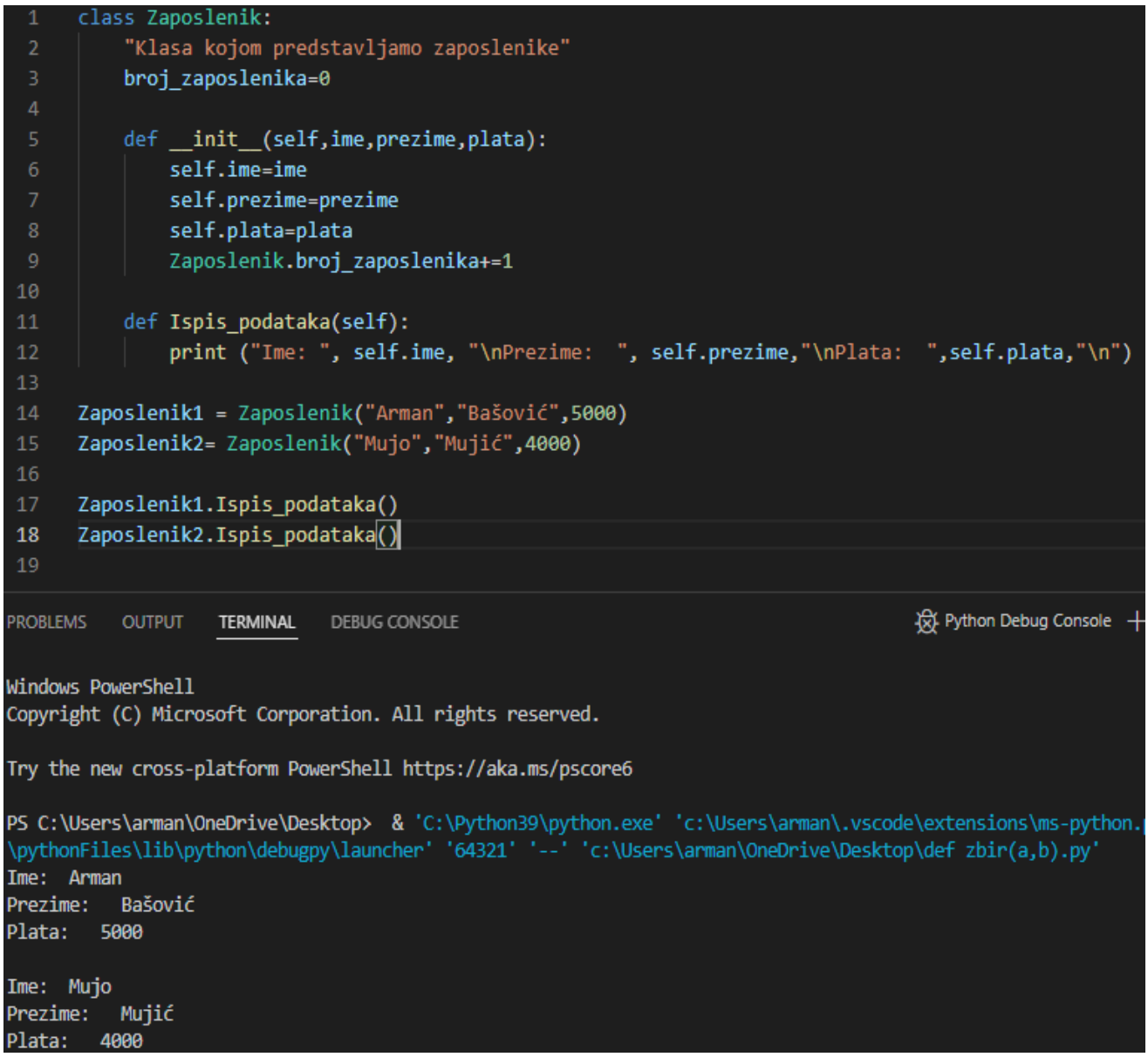
```
1   def zbir(a,b):
2       "Funkcija sabira dva broja"
3       return a + b
4
5   print (zbir(5,2))
6   print (zbir(50,30))
7   print (zbir(30,-20))
```

```
PROBLEMS    OUTPUT    TERMINAL    DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\arman\OneDrive\Desktop>  & 'C:\Python39\python.exe'
7
80
10
PS C:\Users\arman\OneDrive\Desktop>
```

Slika 2.1.

2.4 Objekti i klase

U objektno orijentiranim programskim jezicima klasom se nazivaju dijelovi programskog koda koji služe kao predlošci za stvaranje korisničkih tipova podataka. Objekt je instanca određene klase čije je ponašanje i inicijalno stanje definirano unutar klase. Na slici 2.2. prikazan je programski kod stvaranja klase i objekta, te rezultat izvršavanja.



Slika 2.2.

Varijabla broj_zaposlenika je klasna varijabla čiju vrijednost dijele svi objekti klase Zaposlenik. Prva metoda __init__() je posebna metoda koja se naziva konstruktor ili inicijalizacijska metoda koja se poziva prilikom stvaranja novog objekta. Sve ostale metode unutar klase pišu se poput normalnih funkcija. Kako bi se stvorila instanca klase, tj. objekt, nužno je koristiti njegov naziv uz listu argumenata, kao što je u primjeru vidljivo: Zaposlenik(“Mujo”, “Mujić”, 4000).

3. Pygame

Pygame je višeplatformski skup Python modula dizajniranih za pisanje video igrica. Uključuje kompjutersku grafiku i biblioteke zvukova dizajnirane da se koriste sa programskim jezikom Python.

Pygame je prvobitno napisao Pete Shinners da zamijeni PySDL nakon što je njegov razvoj zastao. To je projekat zajednice od 2000. godinei objavljen je pod GNU Lesser General Public License (koja "omogućava distribuciju Pygamea sa otvorenim kodom i komercijalnim softverom").

Pygame koristi biblioteku Simple DirectMedia Layer (SDL),[a] sa namjerom da omogući razvoj kompjuterskih igara u realnom vremenu bez mehanike niskog nivoa programskog jezika C i njegovih derivata. Ovo se zasniva na pretpostavci da se najskuplje funkcije unutar igara mogu apstrahovati iz logike igre, što omogućava korištenje programskog jezika visokog nivoa, kao što je Python, za strukturiranje igre. Ostale karakteristike koje SDL ima uključuju vektorsku matematiku, detekciju sudara, upravljanje grafikom 2D sprite scene, MIDI podršku, kameru, manipulaciju nizom piksela, transformacije, filtriranje, naprednu podršku za fontove slobodnog tipa i crtanje. Aplikacije koje koriste Pygame mogu raditi na Android telefonima i tabletima uz korištenje podskupa Pygame za Android.

4. Programski kod

Način na koji ćemo predstavljati naše komade će biti korištenjem višedimenzionalnih lista. Svaka lista će imati više podlista koje predstavljaju sve moguće rotacije svakog oblika. To će znatno olakšati rotaciju i vizualno predstavljanje naših oblika kada ih na kraju nacrtamo na ekran.

4.1 Oblici

Na slici 4.1 smo odredili širinu i visinu prozora, igrice i svakog pojedinacnog bloka unutar prostora za igranje. Tetris grid uvijek mora imati omjer visine i širine 2:1 tačnije mora imati 20 blokova visine i 10 širine. Zatim imamo gornji_lijevi_x i gornji_lijevi_y to nam predstavlja poziciju unutar prozora odakle čemo početi crtati blokove i odakle započinjemo našu igru. Svaki oblik ima više od jednu rotaciju. Kao što vidimo na slici oblike smo realizovali putem „ 0 “, a „ . “ predstavljaju šupljine oko njih koje ne popunjavaju. Zatim sve vrijednosti tih oblika ćemo postaviti u jedan niz oblici i dodjeliti im boje.

```python
import pygame
import random

pygame.font.init()

s_x = 800
s_y = 700
širina_igre = 300
visina_igre = 600
veličina_bloka = 30

gornji_lijevi_x = (s_x - širina_igre) // 2
gornji_lijevi_y = s_y - visina_igre


S = [['.....',
      '.....',
      '..00.',
      '.00..',
      '.....'],
     ['.....',
      '..0..',
      '..00.',
      '...0.',
      '.....']]

Z = [['.....',···

I = [['..0..',···

O = [['.....',···

J = [['.....',
      '.0...',
      '.000.',
      '.....',
      '.....'],
     ['.....',
      '..00.',
      '..0..',
      '..0..',
      '.....'],
     ['.....',
      '.....',
      '.000.',
      '...0.',
      '.....'],
     ['.....',
      '..0..',
      '..0..',
      '.00..',
      '.....']]

L = [['.....',···

T = [['.....',···

oblici = [S, Z, I, O, J, L, T]
oblici_boje = [(0, 255, 0), (255, 0, 0), (0, 255, 255), (255, 255, 0), (255, 165, 0), (0, 0, 255), (128, 0, 128)]
```

Slika 4.1

4.2 Klasa figura

Budući da ćemo kreirati više oblika, ima smisla kreirati klasu komada koja može pohraniti neke informacije o svakom obliku.

```
class Figura(object):
    def __init__(self, x, y, oblik):
        self.x = x
        self.y = y
        self.oblik = oblik
        self.boja = oblici_boje[oblici.index(oblik)]
        self.rotacija = 0
```

Slika 4.2

4.3 Kreiranje grid-a

Način na koji ćemo pratiti dijelove u igri je korištenje strukture podataka mreže. Napravićemo višedimenzionalnu listu koja sadrži 20 lista od 10 elemenata (redova i kolona). Svaki element na listi će biti torka koja predstavlja boju komada na toj trenutnoj poziciji. Ovo će nam omogućiti da prilično lako nacrtamo sve kvadrate u boji jer možemo jednostavno proći kroz višedimenzionalnu listu. Parametar zaključane pozicije će sadržati rečnik parova vrednosti ključeva gde je svaki ključ pozicija komada koji je već pao, a svaka vrednost je njegova boja. Proći ćemo kroz ove zaključane pozicije i modificirati našu praznu mrežu da prikažemo ove dijelove.

```
def napraviti_grid(fiskna_pozicija={}):
    grid = [[(0,0,0) for _ in range(10)] for _ in range(20)]

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            if (j, i) in fiskna_pozicija:
                c = fiskna_pozicija[(j,i)]
                grid[i][j] = c
    return grid
```

Slika 4.3

4.4 Okretanje oblika

U ovom trenutku svaki naš komad je predstavljen višedimenzionalnom listom. Treba nam nešto što može prevesti ovu listu u oblik koji kompjuter može razumjeti. U idealnom slučaju s obzirom na format oblika, želimo ga pretvoriti u listu pozicija koje onda možemo vratiti. To je ono što će okretanje_oblika() učiniti za nas.

```
def okretanje_oblika(oblik):
    pozicije = []
    format = oblik.oblik[oblik.rotacija % len(oblik.oblik)]

    for i, linija in enumerate(format):
        red = list(linija)
        for j, kolona in enumerate(red):
            if kolona == '0':
                pozicije.append((oblik.x + j, oblik.y + i))

    for i, pos in enumerate(pozicije):
        pozicije[i] = (pos[0] - 2, pos[1] - 4)

    return pozicije
```

Slika 4.4

4.5 Provjera kretnje

Kada se krećemo i rotiramo svoj oblik, moramo biti sigurni da se pomiče u valjani prostor. Koristit ćemo funkciju valid_space() da ovo provjerimo. Ova funkcija će imati dva parametra: mrežu i oblik. Provjerit ćemo mrežu kako bismo se uvjerili da trenutna pozicija u koju pokušavamo preći nije zauzeta. To možemo učiniti tako da vidimo da li bilo koja od pozicija u mreži u koju se oblik pokušava pomaknuti ima boju. Ako imaju drugu boju osim crne, to znači da su zauzeti, inače su slobodni.

```
def provjera_kretnje(oblik, grid):
    dozvoljena_pozicija = [[(j, i) for j in range(10) if grid[i][j] == (0,0,0)] for i in range(20)]
    dozvoljena_pozicija = [j for sub in dozvoljena_pozicija for j in sub]

    formatted = okretanje_oblika(oblik)

    for pos in formatted:
        if pos not in dozvoljena_pozicija:
            if pos[1] > -1:
                return False
    return True
```

Slika 4.5

4.6 Provjera ako je igrač izgubio

Da bismo završili igru moramo stalno provjeravati da li je korisnik izgubio igru. izgubiti() funkcija će to učiniti umjesto nas. Jednostavno ćemo provjeriti da li se neka pozicija na datoj listi nalazi iznad ekrana. Ako jeste, došli smo do vrha i zbog toga izgubili utakmicu.

```
def izgubiti(pozicije):
    for pos in pozicije:
        x, y = pos
        if y < 1:
            return True

    return False
```

Slika 4.6

4.7 Dobijanje nasumičnog oblika

Budući da ćemo nasumično ispuštati oblike niz ekran, moramo generirati nasumični oblik. Ovo će biti urađeno u funkciji preuzmi_oblik().

```
def preuzmi_oblik():
    return Figura(5, 0, random.choice(oblici))
```

Slika 4.7

4.8 Prikazivanje tekstova

Ova funkcija će nam pomoći u prikazivanju teksta na sredini ekrana i koristit će se kada kreiramo glavni meni i završni ekran za našu igru.

```
def glavni_text(površina, text, size, boja):
    font = pygame.font.SysFont("comicsans", size, bold=True)
    label = font.render(text, 1, boja)

    površina.blit(label, (gornji_lijevi_x + širina_igre /2 - (label.get_width()/2), gornji_lijevi_y + visina_igre/2 - label.get_height()/2))
```

Slika 4.8

4.9 Prikazivanje grid-a

Ova funkcija će jednostavno nacrtati sive linije mreže u području za igru tako da možemo vidjeti u kojem se kvadratu nalaze naše figure.

```
def prikazi_grid(površina, grid):
    sx = gornji_lijevi_x
    sy = gornji_lijevi_y

    for i in range(len(grid)):
        pygame.draw.line(površina, (128,128,128), (sx, sy + i*veličina_bloka), (sx+širina_igre, sy+ i*veličina_bloka))
        for j in range(len(grid[i])):
            pygame.draw.line(površina, (128, 128, 128), (sx + j*veličina_bloka, sy),(sx + j*veličina_bloka, sy + visina_igre))
```

Slika 4.9

4.10 Čišćenje redova

Kao što svi znamo u Tetrisu, kada su sve pozicije u nizu popunjene, taj red se briše i svaki red oko njega se pomera naniže. Funkcija očistiti_redove() će biti mjesto gdje ćemo to učiniti. U suštini ono što ćemo uraditi je provjeriti da li je red pun. Ako jeste, izbrisat ćemo taj red. Ovo će pomjeriti svaki red za jedan naniže, ali će nam ostaviti jedan red manje nego prije. Da bismo to kompenzirali, dodaćemo jedan red na vrh naše mreže. Vraća vrijednost inc kako bi ubuduće mogli pratiti rezultat igrača.

```
def očistiti_redove(grid, locked):

    inc = 0
    for i in range(len(grid)-1, -1, -1):
        red = grid[i]
        if (0,0,0) not in red:
            inc += 1
            ind = i
            for j in range(len(red)):
                try:
                    del locked[(j,i)]
                except:
                    continue

    if inc > 0:
        for key in sorted(list(locked), key=lambda x: x[1])[::-1]:
            x, y = key
            if y < ind:
                newKey = (x, y + inc)
                locked[newKey] = locked.pop(key)

    return inc
```

Slika 4.10

4.11 Prikaz sljedećeg oblika

Ova funkcija će prikazati sljedeći padajući oblik na desnoj strani ekrana.

```
def prikazati_sljedeci_oblik(oblik, površina):
    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Sljedeći oblik', 1, (255,255,255))

    sx = gornji_lijevi_x + širina_igre + 50
    sy = gornji_lijevi_y + visina_igre/2 - 80
    format = oblik.oblik[oblik.rotacija % len(oblik.oblik)]

    for i, linija in enumerate(format):
        red = list(linija)
        for j, kolona in enumerate(red):
            if kolona == '0':
                pygame.draw.rect(površina, oblik.boja, (sx + j*veličina_bloka, sy + i*veličina_bloka, veličina_bloka, veličina_bloka), 0)

    površina.blit(label, (sx + 10, sy - 50))
```

Slika 4.11

4.12 Određivanje high score

Implementiraćemo vrlo jednostavan sistem bodovanja koji jednostavno daje igraču 10 poena svaki put kada očisti red. Kasnije ćemo sačuvati ovaj rezultat i prikazati i najbolji rezultat za igrača. To znači da moramo definisati skor = 0 i last_score = max_score() na vrhu naše main() funkcije. Zatim, kada pozivamo očistiti_redove() iz funkcije main(), jednostavno ćemo uraditi score += očistiti_redove () * 10 tako da dodaje vraćenu vrijednost našem rezultatu. Max_score ostaje u teksutalnom dokumentu scores.txt koji se nalazi na istom mjestu kao naš python program.

```
def ažuriranje_rezultata(nscore):
    score = max_score()

    with open('scores.txt', 'w') as f:
        if int(score) > nscore:
            f.write(str(score))
        else:
            f.write(str(nscore))


def max_score():
    with open('scores.txt', 'r') as f:
        lines = f.readlines()
        score = lines[0].strip()

    return score
```

Slika 4.12

4.13 Prikaži prozor

Prikazat ćemo trenutni i najbolji rezultat unutar funkcije draw_window(). To znači da moramo nekako proći rezultate tamo.

```
def prikaži_prozor(površina, grid, score=0, prethodni_high_score = 0):
    površina.fill((0, 0, 0))

    pygame.font.init()
    font = pygame.font.SysFont('comicsans', 60)
    label = font.render('Tetris', 1, (255, 255, 255))

    površina.blit(label, (gornji_lijevi_x + širina_igre / 2 - (label.get_width() / 2), 30))


    font = pygame.font.SysFont('comicsans', 30)
    label = font.render('Score: ' + str(score), 1, (255,255,255))

    sx = gornji_lijevi_x + širina_igre + 50
    sy = gornji_lijevi_y + visina_igre/2 - 100

    površina.blit(label, (sx + 20, sy + 160))

    label = font.render('High Score: ' + prethodni_high_score, 1, (255,255,255))

    sx = gornji_lijevi_x - 200
    sy = gornji_lijevi_y + 200

    površina.blit(label, (sx - 40 , sy + 160))

    for i in range(len(grid)):
        for j in range(len(grid[i])):
            pygame.draw.rect(površina, grid[i][j], (gornji_lijevi_x + j*veličina_bloka, gornji_lijevi_y + i*veličina_bloka, veličina_bloka, veličina_bloka), 0)

    pygame.draw.rect(površina, (255, 0, 0), (gornji_lijevi_x, gornji_lijevi_y, širina_igre, visina_igre), 5)

    prikazi_grid(površina, grid)
```

Slika 4.13

4.14 Glavni program

U svakoj igri imamo nešto što se zove petlja igre ili glavna petlja. To je ono što će stalno raditi i provjeravati da li se događaji dešavaju. Naša petlja igre će ići unutar funkcije main(). U ovoj funkciji ćemo početi tako što ćemo definirati neke varijable, a zatim prijeći u while petlju.

```
def main(win):
    prethodni_high_score = max_score()
    fiksne_pozicije = {}
    grid = napraviti_grid(fiksne_pozicije)

    promjeni_oblik = False
    run = True
    trenutni_oblik = preuzmi_oblik()
    sljedeći_oblik = preuzmi_oblik()
    clock = pygame.time.Clock()
    vrijeme_pada = 0
    brzina_pada = 0.27
    vrijeme_levela = 0
    score = 0
```

Slika 4.14

Unutar while petlje ćemo provjeriti događaje pritiska na tipke i vidjeti da li korisnik želi izaći iz igre. Kada korisnik pritisne tipku sa strelicom nagore, oblik će se rotirati. To možemo učiniti jednostavnim povećanjem našeg atributa rotacije oblika da bude sljedeći oblik na listi koju smo postavili na početku programa.

Kada korisnik pritisne lijevu ili desnu tipku sa strelicom, mi ćemo se kretati u skladu s tim, mijenjajući x vrijednost našeg komada. Kada korisnik pritisne tipku sa strelicom prema dolje, mi ćemo se pomaknuti za jedan kvadratić dopuštajući korisniku da poveća brzinu kojom oblik pada. Konačno, ukoliko pritisne space oblik će automatski se spustiti na dno table.

```
for event in pygame.event.get():
    if event.type == pygame.QUIT:
        run = False
        pygame.display.quit()

    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT:
            trenutni_oblik.x -= 1
            if not(provjera_kretnje(trenutni_oblik, grid)):
                trenutni_oblik.x += 1
        if event.key == pygame.K_RIGHT:
            trenutni_oblik.x += 1
            if not(provjera_kretnje(trenutni_oblik, grid)):
                trenutni_oblik.x -= 1
        if event.key == pygame.K_DOWN:
            trenutni_oblik.y += 1
            if not(provjera_kretnje(trenutni_oblik, grid)):
                trenutni_oblik.y -= 1
        if event.key == pygame.K_UP:
            trenutni_oblik.rotacija += 1
            if not(provjera_kretnje(trenutni_oblik, grid)):
                trenutni_oblik.rotacija -= 1
        if event.key == pygame.K_SPACE:
           while provjera_kretnje(trenutni_oblik, grid):
            trenutni_oblik.y += 1
           trenutni_oblik.y -= 1
           print(okretanje_oblika(trenutni_oblik))
```

Slika 4.15

Sada ćemo dodati neki kod u petlju igre koji će pomicati naše figure niz ekran u određenom vremenskom intervalu. Kako vrijeme odmiče vjerovatno bismo željeli da otežamo igru. Da bismo to učinili, jednostavno ćemo povećati brzinu kojom dijelovi padaju niz ekran.

```
grid = napraviti_grid(fiksne_pozicije)
vrijeme_pada += clock.get_rawtime()
vrijeme_levela += clock.get_rawtime()
clock.tick()

if vrijeme_levela/1000 > 5:
    vrijeme_levela = 0
    if vrijeme_levela > 0.12:
        vrijeme_levela -= 0.005

if vrijeme_pada/1000 > brzina_pada:
    vrijeme_pada = 0
    trenutni_oblik.y += 1
    if not(provjera_kretnje(trenutni_oblik, grid)) and trenutni_oblik.y > 0:
        trenutni_oblik.y -= 1
        promjeni_oblik = True
```

Slika 4.16

Varijablu promjeni_oblik smo deklarisali unutar glavne funkcije. Ukoliko ona bude ispunjena, tačnije njegova vrijednost bude true program će uzeti sljedeći oblik koji smo deklarisali prethodno i postaviti ga na trenutni i generisati novi sljedeći oblik i postaviti varijablu promjeni_oblik nazad na false. Također svaki put kada se oblik promjeni program će pokrenuti funkciju očistiti_redove i ako je ispunjena dodat će 10 bodova za svaki očišćen red.

```
if promjeni_oblik:
    for pos in oblik_pos:
        p = (pos[0], pos[1])
        fiksne_pozicije[p] = trenutni_oblik.boja
    trenutni_oblik = sljedeći_oblik
    sljedeći_oblik = preuzmi_oblik()
    promjeni_oblik = False
    score += očistiti_redove(grid, fiksne_pozicije) * 10
```

Slika 4.17

Pozivamo dvije funkcije koje nam prikazuju sve na ekranu kao što je igrica, grid, rezultat,trenutni i sljedeći oblik

```
prikaži_prozor(win, grid, score, prethodni_high_score)
prikazati_sljedeci_oblik(sljedeći_oblik, win)
pygame.display.update()
```

Slika 4.18

Pozivanje funkcije izgubiti sa vrijednostima fiksne_pozicije i ukoliko nema dovoljno blokova u kojem će se oblici kretati izlazimo iz nase while petlje i prikazuje tekst „Izgubio si“ i poziva funkiju ažuriranje_rezultata koja provjerava vrijednost prethodnog high score i ukoliko je trenutni rezultat veći od prethodnog trenutni high score se mijenja.

```
if izgubiti(fiksne_pozicije):
    glavni_text(win, "Izgubio si", 80, (255,255,255))
    pygame.display.update()
    pygame.time.delay(1500)
    run = False
    ažuriranje_rezultata(score)
```

Slika 4.19

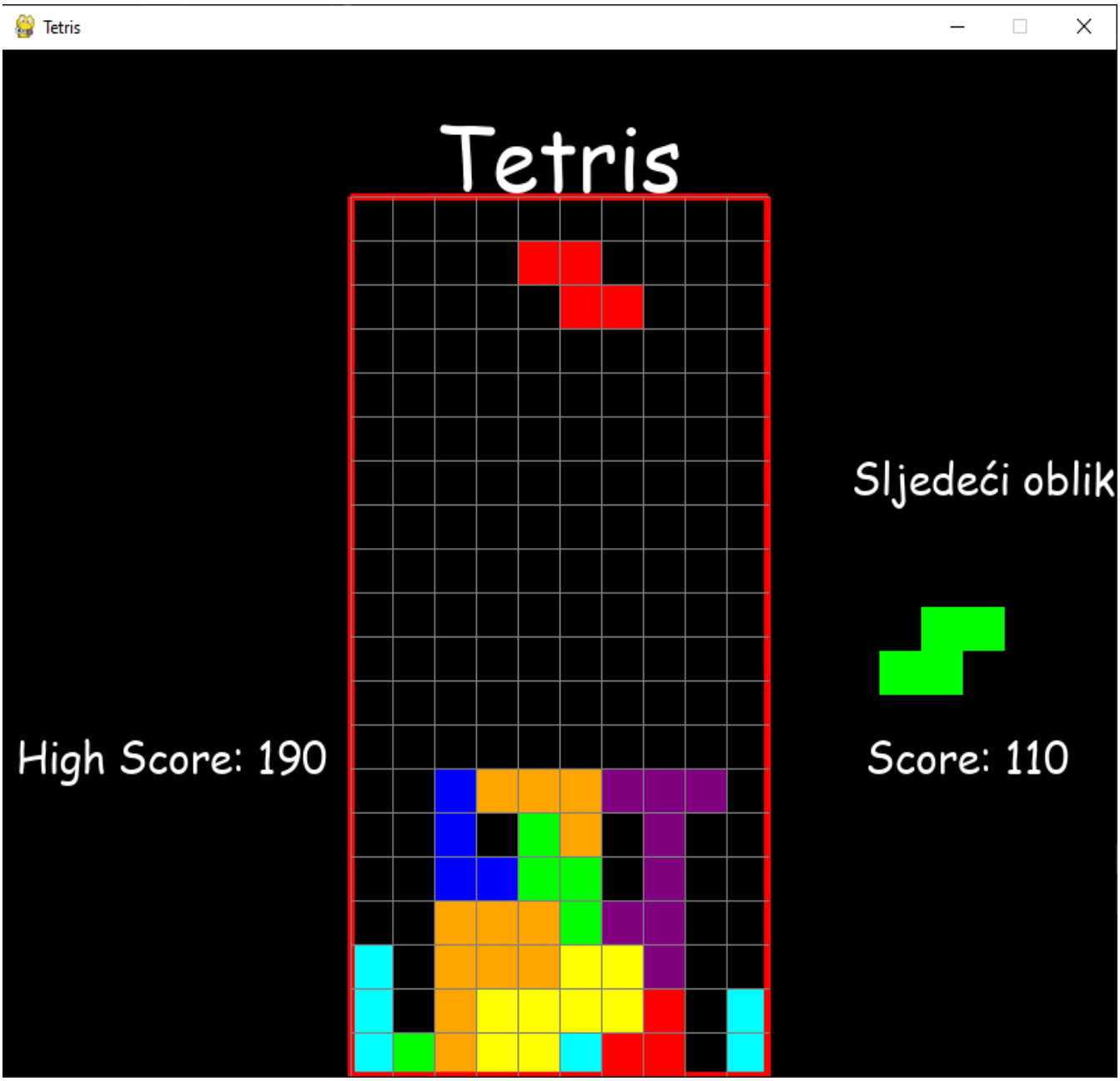
Konačno pokrećemo funkciju main_menu koja postavlja varijbalu run = True i ako izađemo iz igre program postavlja vrijednost run= False ili ako pritisnemo neko drugo dugme pokreće se funkcija main(win) kojom se pokreće čitav program.

```
def main_menu(win):
    run = True
    while run:
        win.fill((0,0,0))
        glavni_text(win, 'Pritisnite dugme da započnete', 40, (255,255,255))
        pygame.display.update()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                run = False
            if event.type == pygame.KEYDOWN:
                main(win)

    pygame.display.quit()


win = pygame.display.set_mode((s_x, s_y))
pygame.display.set_caption('Tetris')
main_menu(win)
```

Slika 4.20

Zaključak

Iako trenutna verzija tetrisa jeste funkcionalna u budućim verzijama može se dodati još mnogo novih funkcija savremenog tetrisa. Prvenstveno grafički dizajn figura i grida se može dodatno preurediti. Zatim savremeni tetris uzima 7-bag randomizer koji uzima različite figure i miješa ih. Ovim igrač dobija svih 7 figura svaki put različitim redoslijedom i najduže što možemo ostati bez jedne određene figure jeste 13 poteza. Također postoji opcija kojom igrač može sačuvati jednu figuru sa strane za buduće upotrebe.



Mišljenje o radu

Potpis mentora:

_____

Ime Prezime dipl.el. ing.

Literatura

https://en.wikipedia.org/wiki/Python_(programming_language)
https://en.wikipedia.org/wiki/Tetris
https://www.pluralsight.com/paths/core-python
https://www.codecademy.com/catalog