

Gimnazija "Veljko Petrović"

Sombor

**Maturski rad iz programiranja
JEDNOSTAVNA IGRICA**

Mentor:

Duško Obradović

Učenik:

Sara Milosavljević, IV-7

Sombor, jun 2024. god.

Predgovor

Razvoj video igara je oblast koja spaja kreativnost i tehničko znanje, omogućavajući stvaranje interaktivnih iskustava koja zabavljaju i angažuju igrače širom sveta. Ideja za ovaj maturski rad proistekla je iz želje da se bolje razumeju osnovni principi razvoja igara, kao i da se steknu praktična znanja u programiranju i upotrebi grafičkih alata.

Flappy Bird je jedna od najpoznatijih i najjednostavnijih igara koja je postigla globalnu popularnost zbog svoje izazovne mehanike i jednostavnog dizajna. Izrada kopije ove igre pružila je savršen okvir za istraživanje ključnih aspekata razvoja igara, kao što su upravljanje događajima, detekcija kolizije, animacija i optimizacija performansi.

SADRŽAJ

	Strana
PREDGOVOR	2
SADRŽAJ	
1. UVOD	4
1.1. Tema i cilj rada.....	4
1.2. Kratak opis igre Flappy Bird.....	4
1.3. Razlog izbora Pygame biblioteke	4
2. FLAPPY BIRD I PYGAME	5
2.1. Istorija igara i Flappy Bird-a.....	5
2.2. Pygame i njegove mogućnosti.....	6
3. PLANIRANJE I DIZAJN IGRE	7
3.1. Koncept igre.....	7
3.2. Specifikacija zahteva i funkcionalnosti.....	7
3.3. Potrebne grafike.....	7
3.4. Tehnologije i alati.....	8
4. IMPLEMENTACIJA	9
4.1. Organizacija projekta.....	9
4.2. Detaljna analiza ključnih funkcionalnosti.....	9
4.3. Definicija klasa.....	9
4.4. Prikaz rezultata i Game Over ekran.....	14
5. REZULTATI I DISKUSIJA	16
ZAKLJUČAK	17
Literatura	18

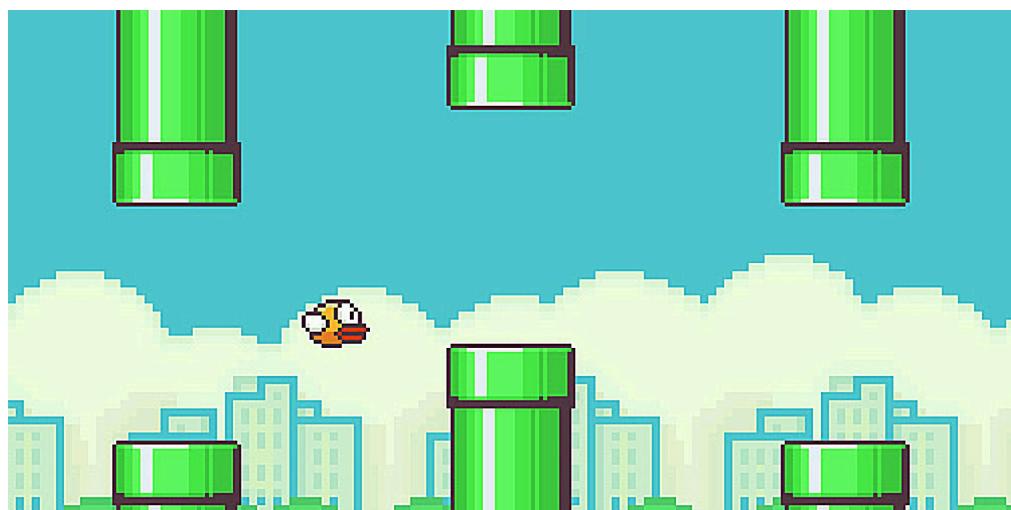
1. UVOD

1.1. Tema i cilj rada

Tema ovog maturskog rada je razvoj jednostavne igrice, koja je kopija popularne igre Flappy Bird, korišćenjem Pygame biblioteke u programskom jeziku Python. Cilj rada je demonstracija osnovnih principa razvoja igara, kao i savladavanje osnovnih tehnika objektno orijentisanog programiranja i grafike.

1.2. Kratak opis igre Flappy Bird

Flappy Bird je jednostavna igra u kojoj igrač kontroliše pticu koja se kreće kroz niz prepreka. Cilj igre je proći što više prepreka bez sudaranja. Igra se odlikuje jednostavnom mehanikom i visokom težinom.



1.3. Razlog izbora Pygame biblioteke

Pygame je popularna biblioteka za izradu 2D igara u Pythonu. Laka je za korišćenje, pruža bogat skup funkcionalnosti za rad sa grafikama, zvukom i upravljanje događajima, što je čini idealnim izborom za početnike i male projekte.

2. FLAPPY BIRD I PYGAME

2.1. Istorija igara i Flappy Bird-a

Razvoj video igara započeo je sredinom 20. veka, sa prvim eksperimentima koji su se odigrali na tadašnjim računarima i osciloskopima. Prva zabeležena video igra, "Tennis for Two", stvorena je 1958. godine, dok je prva komercijalno uspešna igra, "Pong", lansirana 1972. godine od strane Atari-a.

Sa napretkom tehnologije i rastom dostupnosti mobilnih uređaja, mobilni gejming je postao ključni segment industrije video igara. Mobilne igre su posebno privlačne zbog svoje pristupačnosti i jednostavnosti, omogućavajući igračima da uživaju u igri bilo gde i bilo kada. Ovaj trend je doveo do eksplozije jednostavnih, ali zaraznih igara koje su brzo postale globalni fenomen.

Flappy Bird, kao predstavnik modernih jednostavnih igara, izuzetno je popularan primer igre koja koristi minimalistički pristup dizajnu i mehanici. Objavljena 2013. godine, igra se istakla svojom jednostavnom, ali izazovnom mehanikom koja je zahtevala visoku preciznost i koordinaciju od strane igrača.



2.2. Pygame i njegove mogućnosti

Pygame je biblioteka za programiranje multimedijalnih aplikacija (posebno video igara) u programskom jeziku Python. Razvijena je kao besplatna i otvorenog koda, omogućavajući programerima svih nivoa da kreiraju 2D igre sa relativno jednostavnom sintaksom i bogatim skupom funkcionalnosti.

Pygame se temelji na biblioteci SDL (Simple DirectMedia Layer) i pruža funkcije za rad sa grafikama, zvukom, događajima (kao što su pritisci tastera i pomeranje miša), kao i za upravljanje vremenskim intervalima i animacijama.

Osnovne komponente Pygame biblioteke uključuju:

- **Pygame.display:** Modul za rad sa prozorima i ekranima.
- **Pygame.draw:** Funkcije za crtanje osnovnih oblika poput linija, krugova i pravougaonika.
- **Pygame.image:** Funkcije za učitavanje i manipulaciju slikama.
- **Pygame.mixer:** Modul za rad sa zvukom, omogućavajući učitavanje i reprodukciju zvučnih datoteka.
- **Pygame.sprite:** Sistem za rad sa animiranim objektima, omogućavajući jednostavno upravljanje grupama objekata.

Zahvaljujući ovim mogućnostima, Pygame je postao popularan alat za edukaciju i razvoj jednostavnih igara, omogućavajući korisnicima da brzo kreiraju interaktivne aplikacije i eksperimentišu sa različitim mehanikama igre

3. PLANIRANJE I DIZAJN IGRE

3.1. Koncept igre

Koncept igre je jednostavan: igrač kontroliše pticu koja mora da prolazi kroz niz vertikalnih prepreka (cevi) koje se nasumično generišu. Ptica se kreće horizontalno, a igrač mora da pritisne taster kako bi ptica poletela prema gore, dok gravitacija vuče pticu prema dole. Igra se završava kada ptica udari u prepreku ili padne na tlo.

3.2. Specifikacija zahteva i funkcionalnosti

Osnovne funkcionalnosti koje igra treba da ima su sledeće:

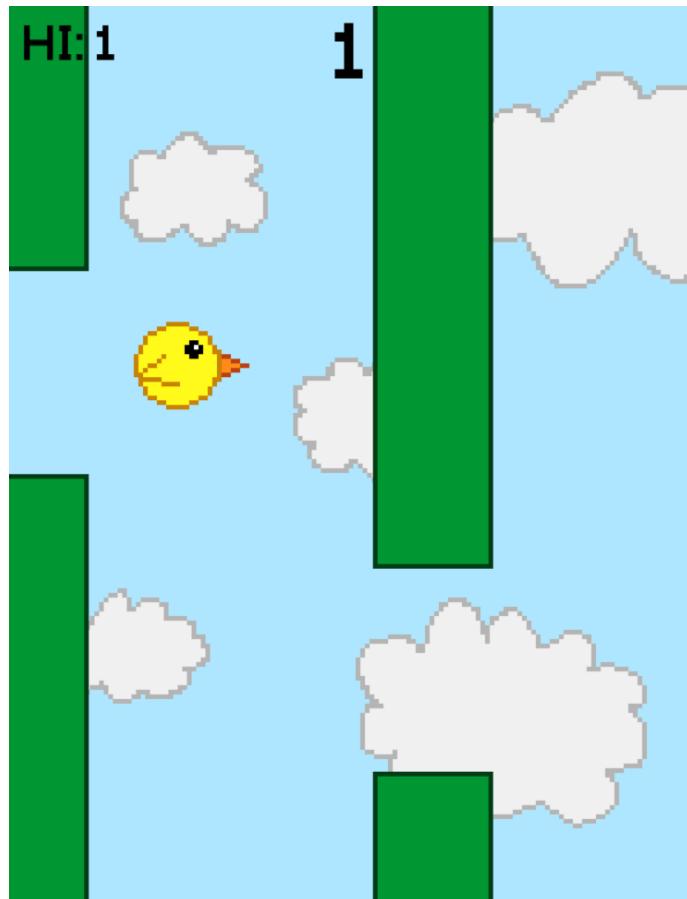
1. **Kontrola ptice:** Pritiskom na space, ptica će da skoči.
2. **Gravitacija:** Gravitacija vuče pticu na dole.
3. **Ubrzano padanje:** Što duže prođe od poslednjeg skoka, to ptica brže pada.
4. **Generisanje prepreka:** Cevke se generišu sa prazninama kroz koje ptica treba da prođe. Visina praznine se generiše nasumično.
5. **Detekcija kolizije:** Igra se završava kada ptica udari u prepreku ili padne na tlo.
6. **Računanje poena:** Svaka uspešno pređena prepreka povećava rezultat igrača.
7. **Prikaz rezultata:** Tokom igre, prikazuje se rezultat igrača.
8. **Pamćenje najboljeg rezultata:** Najbolji rezultat se pamti u eksterni fajl, da bi ostao zapamćen i kad se igra ugasi.
9. **Prikaz najboljeg rezultata:** Najbolji rezultat se učitava iz fajla i prikazuje na početku igre. Tokom igre on se prikazuje iz promenljive.
10. **Ponovno pokretanje igre:** Mogućnost ponovnog pokretanja bez gašenja i paljenja igre

3.3. Potrebne grafike

Grafika igre treba da bude jednostavna i privlačna. Sledеće grafičke elemente je potrebno dizajnirati:

1. **Pozadina:** Jednostavna slika koja predstavlja nebo.
2. **Ptica:** Okrugla, žuta ptica koja je upadljiva u odnosu na nebo.
3. **Prepreke (cevi):** Slike cevi koje se pojavljuju u različitim visinama.
4. **Tekstualni elementi:** Tekst za prikaz rezultata i poruke o završetku igre

5. “Game over” ekran: Poruka koja obaveštava igrača da je igra gotova i daje uputsvo da se igra pokrene opet



3.4. Tehnologije i alati

Za razvoj ove igre koristila sam sledeće tehnologije i alate:

1. **Python**: Programsko okruženje u kojem je igra napisana.
2. **Pygame**: Biblioteka za rad sa grafikom i zvukom u Pythonu.
3. **Aseprite**: Softver za izradu i obradu retro pikselizovane grafike
4. **Visual Studio Code**: Okruženje za pisanje koda
5. **Git i GitHub**: Sistem za kontrolu i čuvanje istorije verzije koda

4. IMPLEMENTACIJA

Izvorni kod možete da nadete na mom GitHub-u: <https://github.com/pluto13x/FlapPy-Bird>

4.1. Organizacija projekta

Projekat je organizovan u više datoteka kako bi se kod lakše održavao i razumeo. Osnovne komponente projekta uključuju sledeće datoteke:

- **main.py**: Glavna datoteka koja pokreće igru i sadrži glavni loop igre.
- **funkcije.py**: Datoteka koja sadrži korisničke funkcije za ispis teksta, detekciju sudara, crtanje oblaka i prikaz game over ekrana.
- **ptica.py**: Datoteka koja definiše klasu **Ptica**, koja predstavlja igrača u igri.
- **stub.py**: Datoteka koja definiše klasu **Stub**, koja predstavlja prepreke u igri.

4.2. Detaljna analiza ključnih funkcionalnosti

Projekat je organizovan u više datoteka kako bi se kod lakše održavao i razumeo. Osnovne komponente projekta uključuju sledeće datote

4.2.1. Kontrola ptice

Promenljiva „cvrle” je objekat klase “Ptica”, više o njoj kasnije.

Ptica se kontroliše pomoću tastera SPACE. Kada je SPACE pritisnut, ptica skače, a gravitacija se postavlja na početnu vrednost. Zvučni efekat skoka (whoosh) se takođe reprodukuje. Kada taster nije pritisnut, ptica pada pod uticajem gravitacije koja se pojačava svaki frejm.

```
if keys[pygame.K_SPACE] and cvrle.gravity == 0:
    #kad tek kreće igrica
    hsp = 5 #brzina kojom ptica skače na gore
    ubrzanje = 0.01 #ubrzanje padanja

    if keys[pygame.K_SPACE] and skaci == 1: #kad skoci
        cvrle.gravity = gr
        skaci = 0 #sprečava da ptica nastavi da skace ako se drzi space
        skokf = 10
        #koliko frameova će da ide na gore pre nego što krene da pada
        pygame.mixer.Sound.play(whoosh)
```

```

    elif keys[pygame.K_SPACE] == False: #kad igrac pusti space
        skaci = 1 #dozvoli da moze da skoci opet

    if (skokf > 0): #skok animacija
        cvrle.update(dt)
        skokf -= 1
    else:
        cvrle.gravity += ubrzanje #ubrzavanje pada

```

4.2.2. Generisanje prepreka

Prepreke, objekta Stub, se generišu sa nasumičnom visinom rupe i pomeraju se s desna na levo, što daje iluziju da kamera prati pticu. Kada prepreka izđe izvan ekrana („nestao” metoda), generiše se nova prepreka na desnoj strani ekrana. Ako ptica prođe kroz prepreku („prosao” metoda), rezultat (rez) se povećava i reproducuje se zvučni efekat (ping). Ako ptica udari u prepreku („sudar” metoda), igra se završava i reproducuje se zvučni efekat sudara (boom)

Dva stuba su smeštena u listu „stubic”. Pošto samo dva stuba mogu da budu na ekranu u isto vreme, kad jedan stub nestane sa ekrana, on se zapravo samo pomeri na desni kraj ekrana i promeni mu se visina rupe. Ovako izbegavamo bespotrebno punjenje memorije.

```

for i in range(0,2):

    if stubic[i].nestao():

        #ako stub nestane sa ekrana, napravi novi

        rand = random.randint(a, screenHeight - a)

        #novi stubic ima nasumicnu visinu rupe

        stubic[i] = Stub(screenWidth + stubic[i].b / 2, rand, a, b,
bojal, boja2)

    if cvrle.prosao(stubic[i]): #broji score

        rez += 1

        stubic[i].gotov = True

        #ako je ptica presla stub, vise ne mora da se gleda kolizija
        #za njega

        pygame.mixer.Sound.play(ping)

    if stubic[i].gotov == False: #kolizija

```

```

    if sudar(cvrle, stubic[i]):

        gameover = True

        pygame.mixer.Sound.play(boom)

        if rez > high:

            #da li je napravljen novi najbolji rezultat

            high = rez

        stubic[i].draw(screen)

        stubic[i].pomeri(hsp)      Generisanje prepreka

```

4.2.3. Sudar

Detekcija sudara između ptice i prepreka je ključna za igru. Ako dođe do sudara, igra se završava. U prethodnom kodu ste mogli da vidite da se funkcija „sudar”, koja vraća promenljivu tipa boolean, ispituje za svaki stub koji je trenutno na ekranu.

Funkcija „sudar” proverava da li je došlo do sudara između ptice (tica) i prepreke (cev). Pticein hitbox je u obliku upisanog kvadrata u okruglu pticu, a za stubove se gleda samo leva strana i donji/gornji deo, zato što je nemoguće da će se ptica ikad sudariti za desnim delom.

```

def sudar(tica, cev):
    res = False
    r = tica.radius * sqrt2 / 2
    #Pticin hitbox je kao kvadrat oko kog je okrugla ptica opisana
    c = pygame.Vector2(tica.x, tica.y) #centar ptice
    vrh = pygame.Rect(cev.x - cev.b / 2, 0, cev.b, cev.a1) #gornja cev
    pod = pygame.Rect(cev.x - cev.b / 2, cev.y + cev.a / 2, cev.b, \
                      cev.a2) #donja cev
    if c.x + r >= vrh.left and c.y - r <= vrh.bottom: #gornja kolizija
        res = True
    if c.x + r >= pod.left and c.y + r >= pod.top: #donja kolizija
        res = True
    return res

```

4.3. Definicija klasa

Klasa Ptica je definisana sledećim atributima: pozicija (x, y), brzina skoka (vsp), gravitacija (gravity) i poluprečnik (radius). Ima metode za crtanje (draw), ažuriranje pozicije (update),

primenu gravitacije (gravitacija), detekciju pada (pao) i proveru prolaska kroz prepreku (prosao).

```
def draw(self, screen):
    image.convert_alpha()

    screen.blit(image, (self.x - 19 * scale / 2, self.y - 19 * \
                        scale / 2))

def update(self, dt):
    # Update pozicije po dugmicima

    self.y -= self.vsp * dt

    self.center[1] = self.y

    if self.y >= screenHeight - self.radius:
        self.y = screenHeight - self.radius

        self.center[1] = self.y

    elif self.y <= self.radius:
        self.y = self.radius

        self.center[1] = self.y

def gravitacija(self, dt):
    self.y += self.gravity * dt

    self.center[1] = self.y

def pao(self):
    return (bool)(self.y + self.radius > screenHeight)

def prosao(self, stu):
```

```
        return (bool)(self.x == stu.x + self.radius * 2)
```

Klasa Stub definiše prepreke sa njihovim pozicijama (x, y), gde x označava horizontalan položaj, a y sredinu praznine kroz koju ptica može da prođe. Dimenzija a označava visinu rupe, a b širinu cevi. Boja cevi se čuva u promenljivi color, a boja spoljašnjih linija u promenljivi outline. Definisana je metodama za crtanje (draw), pomeranje (pomeri) i proveru izlaska iz ekrana (nestao). Stubovi se crtaju u samom kodu i nemaju svoj sprajt.

```
def draw(self, screen):
    width = 4

    #region gornji

    rect1 = pygame.Rect(self.x - self.b / 2, 0, self.b, self.a1)

    pygame.draw.rect(screen, self.color, rect1)

    pygame.draw.line(screen, self.outline, (self.x - self.b / 2, 0), \
                    (self.x - self.b / 2, self.y - self.a / 2 + width / 2), width)

    pygame.draw.line(screen, self.outline, (self.x + self.b / 2, 0), \
                    (self.x + self.b / 2, self.y - self.a / 2 + width / 2), width)

    pygame.draw.line(screen, self.outline, (self.x - self.b / 2, \
                                           self.y - self.a / 2), (self.x + self.b / 2, self.y - self.a / 2), width)

    #endregion

    #region donji

    rect2 = pygame.Rect(self.x - self.b / 2, self.y + self.a / 2, \
                        self.b, self.a2)

    pygame.draw.rect(screen, self.color, rect2)

    pygame.draw.line(screen, self.outline, (self.x - self.b / 2, \
                                           screenHeight), (self.x - self.b / 2, self.y + self.a / 2 - 1), width)

    pygame.draw.line(screen, self.outline, (self.x + self.b / 2, \
                                           screenHeight), (self.x + self.b / 2, self.y + self.a / 2 - 1), width)
```

```

    pygame.draw.line(screen, self.outline, (self.x - self.b / 2, self.y \
+ self.a / 2), (self.x + self.b / 2, self.y + self.a / 2), width)

#endregion

def pomeri(self, hsp):

    self.x -= hsp

def nestao(self):

    return (bool)(self.x + self.b / 2 < 0)

```

4.4. Prikaz rezultata i Game Over ekran

Funkcije za prikaz rezultata i game over ekrana su definisane u datoteci funkcije.py. Font je uzet iz poznate video igre Minecraft:

```

def ispis(sc, boja, screen, screenWidth):

    font = pygame.font.Font('assets/Minecraft.ttf', 64)

    text = font.render(str(sc), True, boja)

    textRect = text.get_rect(center=(screenWidth // 2, 50))

    screen.blit(text, textRect)


def highscoreIspis(hi, boja, screen):

    font = pygame.font.Font('assets/Minecraft.ttf', 40)

    text = font.render("HI: ", True, boja)

    textRect = text.get_rect(center=(50, 40))

    screen.blit(text, textRect)

    text = font.render(str(hi), True, boja)

    textRect = text.get_rect(midleft = (80, 40))

```

```
    screen.blit(text, textRect)

def gameOver(screen, screenWidth, screenHeight, boja):

    font = pygame.font.Font('assets/Minecraft.ttf', 90)

    text = font.render("GAME OVER", True, boja)

    textRect = text.get_rect(center=(screenWidth / 2, screenHeight / 2
- 25))

    screen.blit(text, textRect)

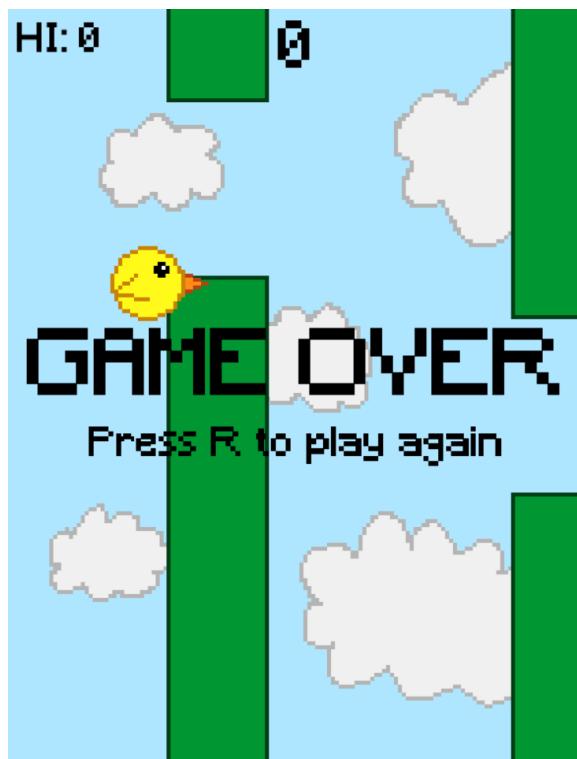
    font = pygame.font.Font('assets/Minecraft.ttf', 40)

    text = font.render("Press R to play again", True, boja)

    textRect = text.get_rect(center=(screenWidth / 2, screenHeight / 2
+ 50))

    screen.blit(text, textRect)

    return 0, 0
```



5. REZULTATI I DISKUSIJA

Projekat je uspešno postigao cilj stvaranja jednostavne verzije Flappy Bird igre koristeći Pygame biblioteku. Igra omogućava kretanje ptice, generisanje prepreka, detekciju sudara, bodovanje, i ponovni start nakon sudara. Implementirani vizuelni i zvučni efekti dodatno poboljšavaju korisničko iskustvo. Kroz proces razvoja, prevazideni su izazovi poput precizne detekcije sudara i balansa između gravitacije i skakanja. Igra je stabilna i funkcionalna, ali postoje mogućnosti za dalje unapređenje, kao što su raznovrsnije prepreke i naprednija grafika.

Zaključak

Kroz realizaciju ovog projekta, uspešno smo kreirali jednostavnu, ali funkcionalnu verziju popularne igre Flappy Bird. Projekat je omogućio praktično iskustvo u razvoju igara koristeći Pygame, uključujući upravljanje kretanjem, detekciju sudara, i implementaciju vizuelnih i zvučnih efekata. Postignuti rezultati su zadovoljavajući, ali postoji prostor za unapređenje i proširenje funkcionalnosti. Ovaj projekat predstavlja solidnu osnovu za dalje učenje i razvoj složenijih igara.

Literatura

Izvorni kod: <https://github.com/pluto13x/FlapPy-Bird>

- [1] Flappy Bird: <https://flappybird.io/>
- [2] Maravić, M. (2022). *Totalna istorija video-igara*, Akademija Umetnosti Novi Sad
- [3] Pygame 2.6.0 dokumentacija: <https://www.pygame.org/docs/ref/pygame.html>
- [4] Python 3.12.3 dokumentacija: <https://docs.python.org/3/>
- [5] Sweigart, A. (2012). *Making Games with Python & Pygame*
- [6] Vuković, D. (2018). *Programiranje -klase i objekti-*

Датум предаје матурског рада: _____

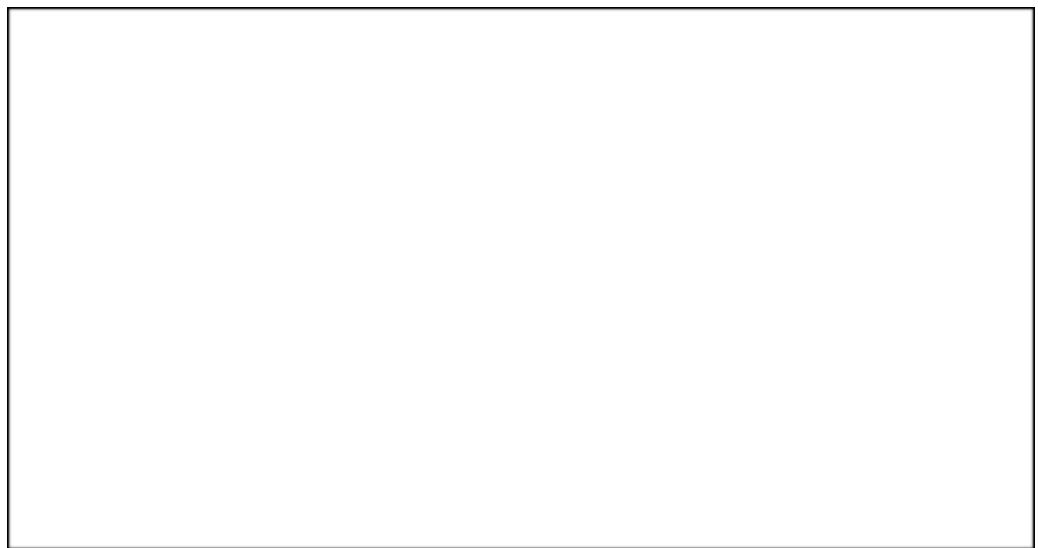
Комисија:

Председник _____

Испитивач _____

Члан _____

Коментар:



Датум одбране: _____

Оцена _____ (____)