SVEUČILIŠTE U ZAGREBU
FAKULTET ORGANIZACIJE I INFORMATIKE
VARAŽDIN

Nina Krsnik

APLIKACIJA ZA IGRANJE DRUSTVENE IGRE SNAKE PUTEM MREŽE

PROJEKT

TEORIJA BAZA PODATAKA

Varaždin, 2025.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Nina Krsnik

Matični broj: 0016155048

Studij: Baze podataka i baze znanja

APLIKACIJA ZA IGRANJE DRUSTVENE IGRE SNAKE PUTEM MREŽE

PROJEKT

Mentor:

Prof. dr. sc. Markus Schatten

Varaždin, veljača 2025.

Nina Krsnik

Izjava o izvornosti

Izjavljujem da je ovaj projekt izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autorica potvrdila prihvaćanjem odredbi u sustavu FOI Radovi

Sažetak

Ovaj seminarski rad bavi se razvojem mrežne aplikacije za pohranu i upravljanje rezultatima igrača u igri "Snake". Aplikacija omogućava korisnicima spremanje vlastitih rezultata i prikaz najbolje rangiranih igrača. Implementirana je pomoću Python socket programiranja za komunikaciju između klijenta i servera, dok se podaci o korisnicima i njihovim rezultatima pohranjuju u objektno-orijentiranu bazu podataka ZODB. Server održava korisničke profile, prati njihove najbolje rezultate i omogućava dohvaćanje top 5 najboljih rezultata. Ovaj rad opisuje tehničku implementaciju sustava kao što je korištenje serijalizacije podataka s pickle modulom, osiguranje konzistentnosti baze podataka i obrada zahtjeva klijenata putem servera. Cilj rada je prikazati kako se može izraditi jednostavna i efikasna aplikacija za vođenje ljestvice igrača, koristeći mrežnu komunikaciju i objektno-orijentiranu bazu podataka.

Ključne riječi: mrežna igra, snake, Python, ZODB, objektno-orijentirana baza podataka, socket komunikacija, pickle, pohrana podataka, praćenje rezultata

Sadržaj

1. Uvod

Pohrana i upravljanje rezultatima igrača u videoigrama postala je standardna funkcionalnost u modernim igrama. Svaka igra koja uključuje natjecateljski aspekt mora imati sustav koji omogućuje bilježenje, ažuriranje i prikaz najboljih rezultata igrača. U ovom seminarskom radu istražit će se razvoj aplikacije koja omogućava spremanje rezultata igrača u igri Snake te prikaz ljestvice najboljih pet igrača.

Aplikacija je osmišljena kao mrežni sustav koji omogućuje komunikaciju između klijentskih uređaja i centralnog servera. Klijenti šalju svoje rezultate na server, gdje se oni pohranjuju, uspoređuju i ažuriraju. Implementacija uključuje socket programiranje u Pythonu, gdje server obrađuje zahtjeve više klijenata i koristi ZODB objektno-orijentiranu bazu podataka za sigurno spremanje korisničkih podataka i rezultata.

Osim osnovnih funkcionalnosti spremanja i dohvaćanja rezultata, aplikacija koristi pickle modul za serijalizaciju podataka kako bi omogućila sigurno slanje objekata preko mreže. Kroz ovaj rad bit će prikazano kako je aplikacija razvijena, koji su tehnički izazovi prepoznati tijekom implementacije te prednosti korištenja objektno-orijentirane baze podataka za upravljanje podacima igrača.

Motivacija za odabir ove teme proizlazi iz znatiželje za implementacijom poznate društeven igre na jedan drugi način, a to je povezivanje nje s mrežom i korištenjem objektno-orijentirane baze podataka putem ZODB za spremanje rezultata korisnika.

2. Teorijski uvod

Baze podataka služe kao ključna komponenta u modernim računalnim sustavima jer omogućuju pohranu, organizaciju i pristup podacima na učinkovit način. Različite vrste baza podataka koriste se ovisno o specifičnim zahtjevima aplikacija. U kontekstu videoigara pohrana i upravljanje podacima igrača, poput rezultata, statusa ili postavki postali su standardna funkcionalnost koja je nužna za osiguranje kontinuiranog iskustva korisnika. [1]

U ovom projektu, korištenje objektno-orijentirane baze podataka (ZODB) povezano je s razvojem aplikacije koja omogućuje igranje društvene igre Snake putem mreže, s ciljem pohrane rezultata igrača i njihova prikaza na ljestvici najboljih. ZODB omogućuje pohranu podataka u obliku objekata što je izuzetno korisno u igrama gdje su podaci vezani uz specifične entitete poput igrača, rezultata i postavki igre. Ovaj pristup omogućuje direktno mapiranje objekata u igri na bazu podataka bez potrebe za kompliciranim upitima ili transformacijama.

2.1. Objektno orijentirane baze podataka

Objektno-orijentirane baze podataka (OODB) su vrste baza podataka koje koriste objektno-orijentirani pristup za pohranu podataka. Umjesto da pohranjuju podatke u tablicama, kao što to rade tradicionalne relacijske baze podataka, objektno-orijentirane baze podataka pohranjuju podatke u obliku objekata. Ovi objekti su instancije klasa, što znači da oni mogu sadržavati podatke (atribute) i metode (funkcije ili operacije koje se mogu izvršavati na podacima) unutar istog entiteta. [1]

Glavne značajke objektno-orijentiranih baza podataka su svakako objekti koji predstavljaju osnovne entitete.U OODB-u podaci se modeliraju kao objekti, što znači da su u bazi podataka pohranjeni isti objekti koji se koriste u aplikaciji. To omogućuje izravnu interakciju između objekata u aplikaciji i objekata u bazi podataka. Zatim, klasifikacija i nasljeđivanje. Korištenje klasa i nasljeđivanja omogućuje strukturiranje podataka na način koji je sličan načinima na koje se podaci organiziraju u objektno-orijentiranim programskim jezicima kao što su Python, Java, C++. To omogućava lakše održavanje i proširivanje sustava.Veze između objekata, poput veza između instanci raznih klasa, bez potrebe za složenim SQL upitima te podrška za složene podatke što omogućuje pohranu složenih podataka kao što su slike, zvukovi i druge vrste binarnih podataka, koji se mogu lako pohraniti u objektima. [1]

2.2. ZODB

ZODB (Zope Object Database) je specifična objektno-orijentirana baza podataka za Python. ZODB omogućuje pohranu podataka u obliku objekata, što znači da aplikacija može raditi s objektima i koristiti ih bez potrebe za transformacijom u relacijske tablice ili druge formate. U ZODB-u objekti se izravno pohranjuju u bazu podataka, što znači da objekti koji su stvoreni i korišteni u aplikaciji mogu biti odmah pohranjeni u bazu bez potrebe za dodatnim mapiranjem ili pretvaranjem podataka u relacijski format, a to se radi naredbom db.commit(). ZODB koristi

sustav transakcija koji omogućuje sigurno čuvanje podataka i brz pristup. Kad god se podaci promijene, ZODB pohranjuje samo promijenjene dijelove objekta, čime optimizira performanse i smanjuje potrebu za pohranjivanjem cijelih objekata svaki put. ZODB omogućuje aplikaciji da stvori objekte koji su trajni (tj. pohranjeni u bazi podataka), čak i nakon što aplikacija završi s radom. Kada se aplikacija ponovno pokrene, objekti će biti dostupni i mogu se koristiti ponovo, jer su trajno pohranjeni u bazi podataka što je primijenjeno i u igri "Snake". S obzirom da ZODB koristi transakcijski model, to znači da se sve promjene u bazi podataka, uključujući dodavanje, ažuriranje ili brisanje objekata, dolaze kao dio transakcije. Ako se nešto dogodi tijekom transakcije, može se poništiti (rollback), osiguravajući konzistentnost podataka. Pohranjeni objekti mogu biti izravno povezani s aplikacijama koje koriste Python, što znači da aplikacija može jednostavno pristupiti podacima, izmijeniti ih i pohraniti ih natrag u bazu podataka bez potrebe za dodatnim mapiranjima ili transformacijama podataka. [2] [3]

Prednosti ZODB-a:

Jednostavnost: Pohrana podataka u obliku objekata smanjuje složenost koda jer nema potrebe za pisanjem SQL upita ili mapiranjem podataka na tablice. Integracija s Pythonom: ZODB je napisan u Pythonu i izvorno je dizajniran za rad u Python aplikacijama, pa je integracija jednostavna. Fleksibilnost: ZODB omogućuje pohranu složenih struktura podataka i može se koristiti za širok raspon aplikacija, od web aplikacija do desktop programa. Optimalne performanse: ZODB optimizira pohranu tako da samo mijenja podatke koji su se stvarno promijenili, čime smanjuje potrebu za prekomjernom pohranom podataka. [3]

Nedostaci ZODB-a:

Skalabilnost: Iako ZODB dobro funkcionira u manjim aplikacijama i za aplikacije koje ne zahtijevaju visoku skalabilnost, može biti manje efikasan u vrlo velikim sustavima s velikim brojem korisnika ili podataka. Kompatibilnost: ZODB nije toliko široko korišten kao tradicionalne relacijske baze podataka, što može otežati njegovu integraciju u ekosisteme koji koriste druge baze podataka. Podrška i zajednica: Iako je ZODB aktivan projekt, nije toliko popularan kao druge baze podataka, što može značiti manju zajednicu za podršku i manje resursa za razvoj. ZODB je, dakle, koristan za aplikacije koje se oslanjaju na pohranu složenih objekata i gdje je jednostavnost integracije s aplikacijom prioritet. [3]

2.3. Python PyGame

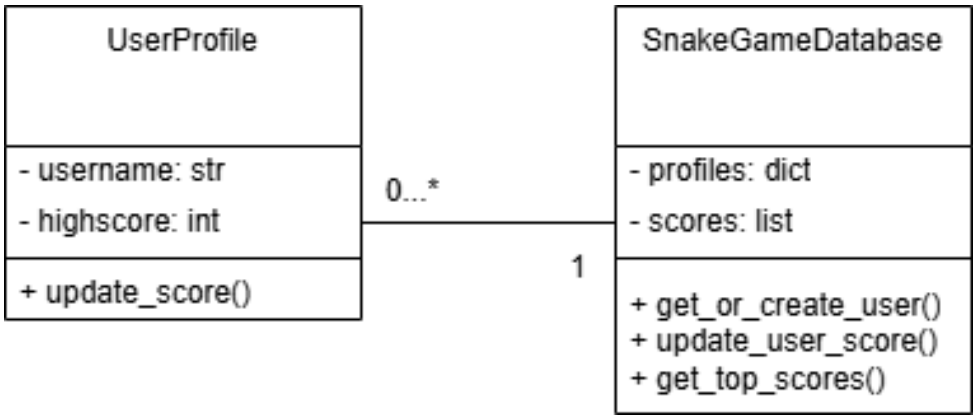
Pygame je biblioteka za razvoj igara u Pythonu koja pruža jednostavan način za stvaranje igara i interaktivnih aplikacija. Pygame omogućuje prikazivanje 2D grafike pomoću slika i oblika. Grafički objekti mogu biti prikazani na ekranu i manipulirani (pomicanje, rotacija, skaliranje) u stvarnom vremenu. To uključuje osnovne funkcije poput crtanja linija, pravokutnika, krugova, a također podržava uvoz i prikaz slika u popularnim formatima (npr. PNG, JPEG). Pygame nudi podršku za uvoz i reprodukciju zvučnih efekata i glazbe, što je ključno za stvaranje interaktivnih iskustava u igrama. Pomoću modula za zvuk moguće je reproducirati zvučne datoteke, upravljati njihovim volumenom i implementirati pozadinsku glazbu ili zvučne efekte u realnom vremenu. Omogućuje jednostavnu interakciju s korisnikom putem tipkovnice i miša.

To uključuje praćenje pritisnutih tipki, pomicanje kursora miša te registriranje klikova. Korištenje tih ulaza omogućuje igračima da kontroliraju likove ili obavljaju radnje u igri. Pygame podržava animacije pomoću modula koji omogućuju kretanje objekata na ekranu u određenim intervalima vremena. Animacije se mogu izvoditi kroz promjene pozicije objekta ili kroz animirane spriteove (slike koje se zamjenjuju u određenom vremenskom okviru). [4]

Osnovna struktura Pygame aplikacije obično uključuje inicijalizaciju koja se izvršava funkcijom pygame.init(). Pygame omogućuje kreiranje grafičkog prozora pomoću funkcije pygame.display.set mode() koja definira veličinu ekrana na kojem se igra prikazuje. Glavna petlja aplikacije poznata kao "game loop" obrađuje sve ulazne događaje (tipkovnica, miš), ažurira stanje igre (pokreće animacije, pomiče objekte) i zatim prikazuje nove slike na ekranu. Unutar event loop-a, aplikacija reagira na događaje kao što su pritisci tipki ili pokreti miša, obavljajući odgovarajuće akcije na temelju tih ulaza. Svaka promjena stanja igre prikazuje se na ekranu putem renderiranja novih slika i objekata. Ovo uključuje crteže objekata, animacije likova, i ostale vizualne elemente igre. Kada igrač odluči zatvoriti igru ili dođe do kraja, aplikacija zatvara prozor i izlazi iz event loop-a pomoću funkcije pygame.quit(). [4]

3. Model baze podataka

Za implementaciju sustava za pohranu rezultata igrača u igri "Snake" koristi se objektno-orijentirana baza podataka ZODB. Pomoću UML dijagrama klasa prikazana su dva entiteta koji su ključni za implementaciju sustava za pohranu podataka u igri "Snake". Prvi entitet je klasa za korisničke profile UserProfile koja ima dva atributa to su korisničko ime igrača username i najbolji rezultat koji je taj igrač postigao highscore. Metodom update score omogućuje se ažuriranje korisnikovog najboljeg rezultata ako postigne veći broj bodova. SnakeGameDatabase je klasa koja omogućuje rad s bazom podataka i upravlja rezultatima igrača. Njezini atributi su profiles gdje se koristi rječnik koji sadrži korisničke profile gdje je ključ korisničko ime, scores koji prikazuje listu od top 5 rezultata. Glavne metode su get or create user koji dohvaća korisnički profil ako on već postoji ili kreira novi, update user score koji ažurira rezultat igrača i top listu, get top scores vaća popis top 5 rezultata. Odnos između SnakeGameDatabase i UserProfile je 1:N što znači da klasa SnakeGameDatabase sadrži više korisničkih profila, ali korisnik ima samo jedan najbolji rezultat u bazi podataka.



Slika 1: UML dijagram klasa (draw.io, vlastiti rad)

4. Implementacija

Za implementaciju projektnog zadatka koriste se tri Python skripte client.py, server.py i storage.py. Klijentska i serverska strana služe za mrežnu komunikaciju i međusobno slanje podataka, dok storage.py pomoću ZODB sprema podatke odnosno rezultate igrača u bazu podataka te na kraju igre ispisuje top 5 rezultata.

client.py

Client.py je skripta koja implementira klijentsku stranu mrežne verzije igre "Snake" pomoću Pygame za grafiku i Socket za mrežnu komunikaciju sa serverom. Inicijalizacija se provodi pygame.init(), a pygame.font.init() omogućuje prikaz teksta. U kodu su pomoću WIDTH i HEIGHT definirane dimenzije ekrana. Na početku skripte poostavljene su i boje igre i tekst. Klasa SnakeGame definira samu igru i ponašanje, tako imamo zmijine koordinate self.snake i self.direction, nasumično generiranje hrane pritom je postavljeno da se hrana ne generira na zmiji i početno postavljanje rezultata na 0 self.score=0.

Metodom update ažurira se trenutno stanje igre, a to je pomicanje zmije, provjerava je li se zmija sudarila s vlastitim tijelom ili zidom jer to znači da je igra završila te provjerava je li zmija pojela hranu jer tada se povećava njezina duljina, mijenja score za +1 i generira se nova hrana.

```
def update(self):
    head = self.snake[0]
    new_head = (head[0] + self.direction[0], head[1] + self.direction[1])

    # zid ili zmija kraj igre
    if new_head in self.snake or new_head[0] < 0 or new_head[0] >= WIDTH or new_head
     [1] < 0 or new_head[1] >= HEIGHT:
        return False

    self.snake = [new_head] + self.snake[:-1]

    if new_head == self.food:
        self.snake.append(self.snake[-1])  # zmija se poveća
        self.food = self.generate_food()  # nova hrana
        self.score += 1

    return True
```

Klijent se povezuje sa serverom. POkušava uspostaviti vezu sa serverom pomoću socket.connect() šalje svoje korisničko ime serveru i prima trenutni high score od servera. Ako veza nije uspjela metodom reconnect pokušava ponovno uspostaviti vezu.

```python
# povezivanje sa serverom
def connect_to_server(username):
    try:
        client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        client.connect(('localhost', 5555))
        client.send(username.encode("utf-8"))

        highscore = pickle.loads(client.recv(1024))
        print(f"Your current high score: {highscore}")
        return client
    except socket.error as e:
        print(f"Connection error: {e}")
        return None

def reconnect_to_server(username):
    client = connect_to_server(username)
    if not client:
        print("Unable to reconnect to server.")
    return client
```

Slika 2: Povezivanje sa serverom (snimka zaslona, vlastiti rad)

Kraj igre prikazan je u show game over screen gdje se na kraju igre otvara ekran s obavijesti o završetku igre s ispisom trenutnog rezultata korisnika i liste od top 5 rezultata. Igra se može ponovo pokrenuti pritiskom tipke Enter ili izaći iz nje pritiskom tipke Q.

U glavnoj funkciji main pokreće se ekran igre, traži se unos korisničkog imena, povezuje se sa serverom, sadrži petlju igre koja omogućuje tijek igre pritisak tipke, ažuriranje rezultata i voća i iscrtavanje igre. Ako igrač izgubi šalje se rezultat serveru pomoću pickle dumps i prima se lista rezultata od servera pomoću pickle loads.

server.py

Server.py omogućuje da se klijent poveže i šalje svoje rezulate i prima listu najboljih rezultata. Koristi ZODB za spremanje korisničkih podataka i rezultata. Socket omogućava mrežnu komunikaciju TCP/IP socket, pickle omogućuje serijalizaciju što znači da pretvara Python objekte u bajtove za slanje putem mreže, storage koji importa SnakeGameDatabase klasu za upravljanje bazom podataka korisnika i rezultata iz storage.py. Pomoću db kreira se instanca baze podataka koja se koristi za spremanje rezultata u .fs datoteku. Funkcija handle client obraćuje jednog klijenta koji se poveže na server. Čeka da klijent pošalje svoje korisničko ime i dekodira primljene podatke u string, ako korisnik nije posao korisničko ime veza se prekida. Uneseni korisnik provjera se u bazi i ako ga nema kreira se i dohvaća se najbolji rezultat i šalje se klijentu najbolji trenutni rezultat pickle dumps. Naredbom pickle.loads deserijalizira se primljeni rezultat, zatim se ažurira rezultat korisnika i dohvaća 5 najboljih rezultata.

```
1 def handle_client(client_socket):
2     try:
3         username = client_socket.recv(1024).decode("utf-8")
4         if not username:
5             print("Primljen prazan username, prekidam vezu.")
6             return
```

```
7
8           user = db.get_or_create_user(username)
9           highscore = user.highscore
10
11          client_socket.sendall(pickle.dumps(highscore))
12
13          data = client_socket.recv(1024)
14          if not data:
15              print(f"Korisnik {username} je prekinuo vezu prije slanja rezultata.")
16              return
17
18          score = pickle.loads(data)
19
20          # ažurira se korisnikov rezultat iz zODB
21          db.update_user_score(username, score)
22
23          # 5 update-anih rezultata
24          top_scores = db.get_top_scores()
25
26          # slanje klijentu tih 5 rezultata
27          client_socket.sendall(pickle.dumps(top_scores))
```

Funkcija start server pokreće server i čeka dolazne veze klijenta, kreira TCP server socket server.bind(('localhost', 5555)) Radi na lokalnoj mreži. Čeka da se klijetn spoji accept i kada se spoji pokreće se client socket.

storage.py

Storage.py koristi ZODB za spremanje korisničkih profila i njihovih rezultata. Skripta omogućuje dodavanje korisnika, ažuriranje rezultata i dohvaćanje top 5 rezultata. FileStorage, DB omogućuje rad s ZODB bazom koristi se datoteka za spremanje podataka. Transaction se koristi za spremanje promjena odnosno commit u ZODB. Definira se klasa UserProfile koja ima username i highscore koji je na početku igre 0. Kada se ostvari veći rezultat ažurira se trenutni score i uzima se novi ostvareni rezultat.

```
1 class UserProfile:
2     def __init__(self, username):
3         self.username = username
4         self.highscore = 0
5
6     def update_score(self, score):
7         if score > self.highscore:
8             self.highscore = score
9         client_socket.sendall(pickle.dumps(top_scores))
```

Klasa SnakeGameDatabase omogućuje pohranu i dohvaćanje podataka iz ZODB, ažurira korisničke rezultate i dohvaća top 5 igrača i njihove rezultate, otvara ZODB bazu, inicijalizira korijenski objekt self.root. Pozivom initialize database postavlja početne strukture baze gdje kreira profiles i scores i sprema promjene u bazu pomoću transaction.commit(). Ako se korisnik ne može dohvatiti znači da ne postoji i kreira se novi UserProfile i vraća se kroisnički objekt. Nakon što se dohvati korisnik i krene igra ažurira se njegov rezultat i dodaje se na listu najboljih rezultata, a te promjene se spremaju u ZODB pomoću transaction.commit(). U update scores

briše se stari rezultat kroisnika iz liste i dodaje se novi, sortira se lista tako da je najbolji rezultat na vrhu i čuva se prvih 5 rezultata. Vraća se lista formata "Korisnik- Rezultat" za top 5 igrača. Na kraju se funkcijom close zatvara veza s bazom.

```
class SnakeGameDatabase:
    def __init__(self, db_file="snake_game_data.fs"):
        self.storage = FileStorage.FileStorage(db_file)
        self.db = DB(self.storage)
        self.connection = self.db.open()
        self.root = self.connection.root
        self.initialize_database()

    def initialize_database(self):
        if not hasattr(self.root, 'profiles'):
            self.root.profiles = {}
        if not hasattr(self.root, 'scores'):
            self.root.scores = []

        transaction.commit()

    def get_or_create_user(self, username):
        if username not in self.root.profiles:
            self.root.profiles[username] = UserProfile(username)
        return self.root.profiles[username]

    def update_user_score(self, username, score):
        user = self.get_or_create_user(username)
        user.update_score(score)
        self.update_scores(user)
        transaction.commit()

    def update_scores(self, user):
        # izbaci vec prisutan ili replicirajuci rezultat
        self.root.scores = [entry for entry in self.root.scores if entry[0] != user.
         username]

        # dodaj azuriran rezultat usera
        self.root.scores.append((user.username, user.highscore))

        # uzlazno sortiranje rezultata
        self.root.scores = sorted(self.root.scores, key=lambda x: x[1], reverse=True
         )

        # zadrzava samo 5 rezultata
        if len(self.root.scores) > 5:
            self.root.scores = self.root.scores[:5]

    def get_top_scores(self):
        # vraca 5 rezultata
        return [f"{user} - {score}" for user, score in self.root.scores]

    def close(self):
        self.connection.close()
```

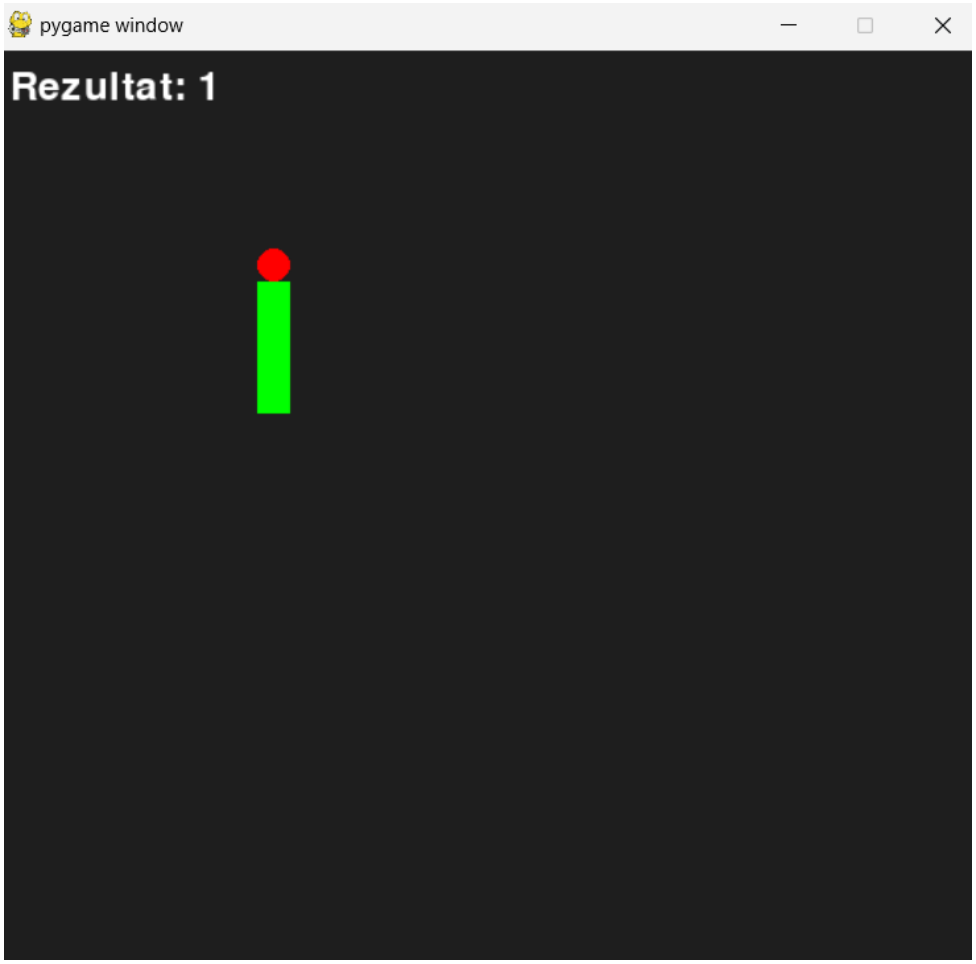
```
48            self.db.close()
```

5. Primjeri korištenja

Na početku igra se pokreće tako da se u jednom terminalu pokrene python server.py a u drugom python client.py. Na ovaj način pokreću se obje skripte i komunikacija preko socketa moći će se uspješno provesti. Na serverskoj strani ispisuje se Server listening on port 5555... što znači da se server uspješno pokrenuo na localhostu koji je postavljen na 5555. Nakon što pokrenemo klijentovu stranu u temrinalu se ispisuje poruka pygame 2.6.1 (SDL 2.28.4, Python 3.13.2) Hello from the pygame community. https://www.pygame.org/contribute.html koja označava da je pygame pokrenut i automatski se otvara prozor s igricom na čijem zaslonu se pojavljuje prostor za unos korisnikova imena.

Slika 3: Unos korisničkog imena (snimka zaslona, vlastiti rad)

Nakon toga kreće igra u kojoj korisnik pomoću strelica na tipkovnici miče zmiju. Cilj je "pojesti" što više voća i skupiti najveći rezultat koji se prikazuje u gornjem lijevom kutu ekrana.



Slika 4: Igranje igre Snake (snimka zaslona, vlastiti rad)

Kada irač izgubi odnosno pojede sam sebe ili prođe rub zaslona tada se otvara zaslon na kojemu se ispiše poruka da je igra završena, prikaže se trenutni rezultat korisnika, prikaže se tekst da igrač moeže nastaviti igru pritiskom tipke Enter ili izaći iz nje pritiskom tipke Q. Na dnu ispisuje se top 5 igrača i njihovi rezultati tako da je najbolji rezultat i igrač koji ga je ostvario na 1. mjestu. Baza podataka se sprema i lista će biti ista i nakon što se igra zatvori.

Slika 5: Kraj igre (snimka zaslona, vlastiti rad)

6. Zaključak

U ovom projektu razvoj mrežne aplikacije za praćenje rezultata igrča u igri pokazuje kako se kombinacijom Python socket programiranja i objektno-orijentirane baze podataka ZODB može osigurati jednostavno i efikasno rješenje za upravljanje podacima korisnika. Implementacijom ove igrice pokazane su prednosti objektno-orijentiranih baza podataka u kontekstu igara kao što su direktna pohrana podataka i dohvaćanje istih u obliku objekata ćime se pojednostavljuje razvoj i održavanje sustava. Tehnologije korištene u projektu pokazale su se adekvatnima za odabranu aplikacijsku domenu. Python socket omogućio je stabilnu komunikaciju između klijenta i servera, pomoću Pygame biblioteke kreirana je 2D igra, dok je ZODB olakšao upravljanje korisničkim podacima i rezultatima i njihovo spremanje u bazu.

Međutim, iako je ZODB jednostavan za korištenje i prikladan za manje projekte, nije optimalan za aplikacije koje zahtiijevaju visoku skalabilnost i podršku za kompleksne upite nad podacima. U budućem radu moguće je unaprijediti sustav za implementaciju dodatnih sigurnosnih mjera, optimizacijom baze podataka i eventualnim prelaskom na skalabilniju bazu podataka.

Ovaj projekt demonstrira kako se s relativno jednostavnim alatima mogu razviti funkcionalni i efikasni sustavi za praćenje rezultata igrača u mrežnoj igrici.

7. Literatura

1 Zadorozhny V. I. (2003.) "Object-Oriented Databases" Preuzeto 13.2..2025. s https://www.sciencedirect.com/science/article/abs/pii/B0122272404001234

2 Okreša Đurić B., Schatten M. Materijali s predmeta Teorije baza podataka, 2024.

3 Krajačić P. (2023.) "Primjer objektno-orijentirane baze podataka u sustavu ZODB" Preuzeto 13.2.2025. s https://zir.nsk.hr/islandora/object/foi:7955

4 McGugan W. (2007.) Beginning Game Development with Python and Pygame [Online] https://books.google.fi/books?hl=hr&lr=&id=Kn8nCgAAQBAJ&oi=fnd&pg=PR14&dq=pygame&ots=eNT_rvDfjD&sig=3YVdZ1byCdt_PaN6CeikFAIKrto&redir_esc=y#v=onepage&q=pygame&f=false

Popis slika