PODATKOVNE BAZE

Seminarska naloga

Luka Šveigl, 63200301

1. NALOGA (DDL)

Iz tabele x_world, opisane z relacijsko shemo

```
x_world(id, x, y, tid, vid, village, pid, player,
                                    aid, alliance, population)
```

ustvarite (CREATE TABLE) in napolnite tabele z naslednjimi relacijskimi shemami in pomeni:

- `pleme(tid, tribe)`
  - šifra in ime plemena (imena vstavite ročno – glej opis datoteke Xworld.sql v teh navodilih)
- `aliansa(aid, alliance)`
  - šifra in ime alianse
- `igralec(pid, player, #tid, #aid)`
  - šifra in ime igralca, njegovo pleme in njegova aliansa
- `naselje(vid, village, x, y, population, #pid)`
  - šifra vasi, ime vasi, x in y koordinati, populacija, šifra igralca lastnika vasi

Iz tabele aliansa odstranite vrstico z vrednostjo aid = 0 ter v tabeli igralec zamenjajte vse vrednosti aid = 0 z NULL. Pri vseh tabelah tudi pravilno določite primarne in tuje ključe.

Tabeli naselje dodajte tudi omejitve (CHECK), tako da sprejme le pravilne vnose koordinat in populacije.

1.1 KREIRANJE TABELE PLEME

```
CREATE TABLE IF NOT EXISTS pleme (
    tid INT PRIMARY KEY,
    tribe VARCHAR(20) NOT NULL UNIQUE
);
```

1.2 KREIRANJE TABELE ALIANSA

```
CREATE TABLE IF NOT EXISTS aliansa (
    aid INT PRIMARY KEY,
    alliance VARCHAR(120) NOT NULL UNIQUE
);
```

1.3 KREIRANJE TABELE IGRALEC

```
CREATE TABLE IF NOT EXISTS igralec (
    pid INT PRIMARY KEY,
    player VARCHAR(120) NOT NULL UNIQUE,
    tid INT NOT NULL,
    aid INT,
    FOREIGN KEY (tid) REFERENCES pleme (tid)
        ON UPDATE RESTRICT      #Dodano, da ponesreči ne brišemo zapisov iz drugih tabel
        ON DELETE RESTRICT,     #kot bi to storili z uporabo ON DELETE CASCADE
    FOREIGN KEY (aid) REFERENCES aliansa (aid)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT
);
```

1.4 KREIRANJE TABELE NASELJE

```
CREATE TABLE IF NOT EXISTS naselje (
    vid INT PRIMARY KEY,
    village VARCHAR(120) NOT NULL UNIQUE,
    x INT NOT NULL,
    y INT NOT NULL,
    population INT NOT NULL,
    pid INT NOT NULL,
    FOREIGN KEY (pid) REFERENCES igralec (pid)
        ON UPDATE RESTRICT
        ON DELETE RESTRICT,
    CHECK (x BETWEEN -250 AND 250),
    CHECK (y BETWEEN -250 AND 250),
    CHECK (population >= 0)
);
```

1.5 POLNJENJE TABEL

```
INSERT INTO pleme VALUES (1, "Rimljani");
INSERT INTO pleme VALUES (2, "Tevtoni");
INSERT INTO pleme VALUES (3, "Galci");
INSERT INTO pleme VALUES (4, "Narava");
INSERT INTO pleme VALUES (5, "Natarji");
INSERT INTO pleme VALUES (6, "Huni");
INSERT INTO pleme VALUES (7, "Egipcani");

INSERT INTO aliansa (aid, alliance)
    SELECT DISTINCT xw.aid, xw.alliance
    FROM x_world xw;

INSERT INTO igralec (pid, player, tid, aid)
    SELECT DISTINCT xw.pid, xw.player, xw.tid, xw.aid
    FROM x_world xw;

INSERT INTO naselje (vid, village, x, y, population, pid)
    SELECT xw.vid, xw.village, xw.x, xw.y, xw.population, xw.pid
    FROM x_world xw;
```

1.6 ODSTRANITEV VREDNOSTI aid = 0

```
# Prenastavimo vrednost aid igralca
UPDATE igralec SET aid = NULL
    WHERE aid = 0;

# Pobrišemo prazno alianso
DELETE FROM aliansa WHERE aid = 0;
```

2. NALOGA (DML)

Nad dobljenimi tabelami iz naloge 1 v jeziku SQL napišite poizvedbe, s pomočjo katerih boste lahko odgovorili na naslednja vprašanja.

a) Izpišite šifro in ime igralca z največjim naseljem ter šifro, ime in velikost tega naselja.

Koda:

```
SELECT i.pid, i.player, n.vid, n.village, n.population
FROM igralec i INNER JOIN naselje n USING(pid)
ORDER BY n.population DESC
LIMIT 1;
```

Rezultat:

	pid	player	vid	village	population
▶	3802	Bogatin	3391	Stibera	1243

b) Izpišite šifre in imena alians, ki imajo maksimalno število članov.

Koda:

```
SELECT a.aid, a.alliance
FROM aliansa a
WHERE (SELECT COUNT(i.aid)
        FROM igralec i
        WHERE i.aid = a.aid) = 60; # 60 članov je max
```

Rezultat:

	aid	alliance
▶	27	RS-H1N1™

c) Koliko igralcev ima nadpovprečno veliko naselje?

Koda:

```
SELECT COUNT(DISTINCT i.pid) AS 'St. z nadpovprecjem'
FROM igralec i INNER JOIN naselje n USING(pid)
WHERE n.population > (SELECT AVG(n2.population)
                      FROM naselje n2);
```

Rezultat:

	St. z nadpovprecjem
▶	2074

d) Izpiši podatke o vseh naseljih igralcev brez alianse, urejeno padajoče po x in nato y koordinati.

Koda:

```
SELECT n.*
FROM naselje n INNER JOIN igralec i USING(pid)
WHERE i.aid IS NULL
ORDER BY x DESC, y DESC;
```

Rezultat:

vid	village	x	y	population	pid
35105	04Odin	250	171	544	72
38845	05Thor	249	172	330	72
21410	02Slavs	248	171	787	72
43834	New village	247	168	74	72
26076	New village	247	-244	538	11104
41176	New village	246	170	201	72
37257	New village	246	-243	316	11104
46309	New village	245	176	4	72
44939	New village	245	118	307	28632

e) Katero pleme je najštevilčnejše (glede na skupno populacijo)?

Koda:

```
SELECT p.tribe
FROM pleme p INNER JOIN igralec i USING(tid)
INNER JOIN naselje n USING(pid)
GROUP BY p.tribe
ORDER BY SUM(n.population) DESC
LIMIT 1;
```

Rezultat:

	tribe
▶	Huni

f) Izpišite število alians z nadpovprečnim številom članov.

Koda:

```
SELECT COUNT(DISTINCT a3.aid)  as Nadpovprecne
FROM aliansa a3
WHERE a3.aid IN
    (SELECT a2.aid # ID alians z nadpovprecnim stevilom igralcev
    FROM aliansa a2 INNER JOIN igralec i2 USING(aid)
    GROUP BY a2.aid
    HAVING COUNT(i2.pid) > (SELECT AVG(t1.IgralciAliansa) # Povprecno st igralcev na alianso
                            FROM (
                                 SELECT COUNT(i.pid) as IgralciAliansa # Igralci po aliansah
                                 FROM igralec i INNER JOIN aliansa a USING(aid)
                                 GROUP BY a.aid
                                 ) as t1));
```

Rezultat:

	Nadpovprecne
▶	44

g) Napišite shranjeno funkcijo popObmocja(x, y, razdalja), ki za poljubne koordinate vrne populacijo na območju od vključno [x,y] do [x+razdalja,y+razdalja]. Izpišite rezultat klica popObmocja(40, 40, 10) in popObmocja(40, 40, 20).

Koda:

```
DROP FUNCTION IF EXISTS popObmocja;
DELIMITER //
CREATE FUNCTION  popObmocja(x INTEGER,
                            y INTEGER,
                            razdalja INTEGER) RETURNS INTEGER
BEGIN
    DECLARE pop INTEGER;

    SELECT SUM(n.population) INTO pop
    FROM naselje n
    WHERE n.x BETWEEN x AND x + razdalja
      AND n.y BETWEEN y AND y + razdalja;

    RETURN pop;
END //
DELIMITER ;
SELECT popObmocja(40, 40, 10);
SELECT popObmocja(40, 40, 20);
```

Rezultat:

	popObmocja(40, 40, 10)
▶	12424

	popObmocja(40, 40, 20)
▶	38924

h) Izpišite imena igralcev, ki imajo vsa svoja naselja na območju x, ki je med 150 in 200 in y, ki je med 0 in 100.

Koda:

```
SELECT DISTINCT i.player
FROM igralec i INNER JOIN naselje n USING(pid)
WHERE (SELECT COUNT(n2.vid)     #Pogledamo, ce je st. vseh igralcevih naselji
       FROM naselje n2          #enako st. igralcevih naselji na tem obmocju
       WHERE n2.pid = i.pid) = (SELECT COUNT(n3.vid)
                                FROM naselje n3
                                WHERE n3.pid = i.pid
                                  AND n3.x BETWEEN 150 AND 200
                                  AND n3.y BETWEEN 0 AND 100);
```

Rezultat:

	player
▶	kon2005
	merika
	RedNavy
	Senbonzakura
	Сирак скитник

i) Izpišite šifre, imena in delež celotne populacije alians, ki imajo vsaj 3% vse populacije v igri. Rezultat uredite padajoče po deležu.

Koda:

```
SELECT a.aid,
       a.alliance,
       (SUM(n.population) * 100) /
       (SELECT SUM(n2.population) FROM naselje n2) AS 'Delez'
FROM aliansa a INNER JOIN igralec i USING(aid)
INNER JOIN naselje n USING(pid)
GROUP BY a.aid, a.alliance
HAVING Delez > 3
ORDER BY Delez DESC;
```

Rezultat:

	aid	alliance	Delez
▶	95	RS-SN	5.2346
	27	RS-H1N1™	4.8184
	24	DT RS #2	4.6677
	315	BFM	4.3388
	73	DARK™	4.1391
	185	DLZ SOS	3.8908
	167	FIGHT~TS	3.0917
	138	FIGHT-W	3.0606

j) Igralec “Sirena” želi preimenovati vsa svoja naselja na naslednji način. Uredil jih bo po populaciji, najmočnejše bo “Grad 01”, naslednje “Grad 02” in tako dalje. Nalogo lahko rešite v več korakih (zaporedju poizvedb).

Koda:

```
SET @counter = 0;

UPDATE naselje SET village = CONCAT("Grad ", LPAD(@counter := @counter + 1, 2, 0))
WHERE pid = (SELECT i.pid FROM igralec i WHERE i.player = "Sirena")
ORDER BY population DESC;

SELECT n.*
FROM naselje n INNER JOIN igralec i USING(pid)
WHERE i.player = "Sirena"
ORDER BY n.population DESC;
```

Rezultat:

vid	village	x	y	population	pid
23018	Grad 01	-202	-196	1025	11164
34803	Grad 02	-202	-197	797	11164
39090	Grad 03	-200	-198	778	11164
28624	Grad 04	-203	-195	773	11164
35161	Grad 05	-200	-196	761	11164
37577	Grad 06	-199	-197	760	11164
32198	Grad 07	-201	-195	752	11164
41437	Grad 09	-196	-195	728	11164
40135	Grad 08	-199	-198	728	11164
42493	Grad 10	-196	-196	714	11164
43293	Grad 11	-195	-197	693	11164
44061	Grad 12	-194	-195	629	11164
45015	Grad 13	-194	-196	586	11164

3. NALOGA (DDL)

a) Napišite transakcijo (zaporedje ukazov), ki bo združila člane alians HORDA in CAR v novo imenovano alianso HORDA-CAR.

Koda:

```
#Začnemo transakcijo
START TRANSACTION;

#V tabelo aliansa vstavimo novo alianso "HORDA-CAR"
INSERT INTO aliansa VALUES((SELECT MAX(a.aid) FROM aliansa a) + 1, "HORDA-CAR"); #AID ena vecji kot najvecji

#"Prevežemo" igralce v aliansah "HORDA" in "CAR" v novoustvarjeno alianso
UPDATE igralec SET aid = (SELECT aid FROM aliansa WHERE alliance = "HORDA-CAR")
WHERE aid = (SELECT aid
             FROM aliansa
             WHERE alliance = "HORDA")
   OR aid = (SELECT aid
             FROM aliansa
             WHERE alliance = "CAR");

#Ko smo igralce prevezali, izbrišemo stari aliansi
DELETE FROM aliansa WHERE alliance = "HORDA" OR alliance = "CAR";

COMMIT;
```

Rezultat:

	aid	alliance
▶	882	HORDA-CAR

b) Napišite bazni prožilec, ki bo ob spremembah vrednosti aid v tabeli igralec preveril, če aliansa še lahko sprejme novega člana.

Koda:

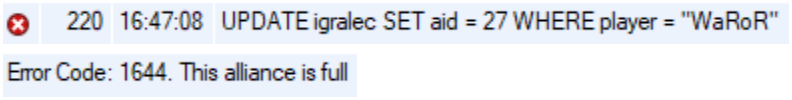
```
DROP TRIGGER IF EXISTS CheckIfAliansaFull_update;

DELIMITER //
CREATE TRIGGER CheckIfAliansaFull_update
BEFORE UPDATE ON igralec
FOR EACH ROW
BEGIN
    DECLARE allianceSize INTEGER;
    DECLARE errorMessage VARCHAR(255);
    SET errorMessage = "This alliance is full"; #Dolocimo sporocilo

    IF OLD.aid != NEW.aid THEN #Pogledamo, ce se aid spremeni
        SET allianceSize = (SELECT COUNT(i.pid) #Dobimo velikost alianse
                            FROM igralec i
                            WHERE i.aid = NEW.aid);

        IF allianceSize + 1 > 60 THEN #Pogledamo, ce je aliansa prevelika
            SIGNAL SQLSTATE '45000' #Ustavimo stavek
            SET MESSAGE_TEXT = errorMessage; #Posljemo sporocilo
        END IF;
    END IF;
END //
DELIMITER ;
#Polna aliansa, testiram delovanje prožilca
UPDATE igralec SET aid = 27 WHERE player = "WaRoR";
```

Rezultat:



4. NALOGA (ODBC)

V programskem jeziku Python napišite program, ki se priključi na podatkovno bazo in za celotno igralno polje izračuna gostoto populacije in gostoto populacije določenega plemena. Gostoto računajte na območjih velikosti 10x10 polj po formulah:

$$Gostota\ populacije = \frac{Skupna\ populacija\ na\ območju}{100}$$

$$Gostota\ plemena = \frac{Skupna\ populacija\ plemena\ na\ območju}{100}$$

Rezultate izračunane gostote (za vsako izmed 50x50=2500 območij) shranite nazaj v svojo bazo v tabeli z imenoma gostotaPopulacije in gostotaPlemena. Za izračun gostote plemena lahko izberete poljubno pleme. Priporočljivo je, da naredite Python funkcijo, ki ima parameter ime plemena.

4.1 OPIS REŠITVE

Pri reševanju te naloge sem se odločil za objektno orientiran pristop, saj mi razredi omogočajo enkapsulacijo kode in potrebnih atributov (spremenljivk), posledica tega pa je boljša berljivost in razdelanost kode.

V sklopu te naloge sem ustvaril razred DataBaseInterface, ki vsebuje atribute **conn** (tipa pyodbc.Connection, vsebuje povezavo na podatkovno bazo), **connection_string** (tipa string, vsebuje podatke, potrebne za povezavo) in **pleme** (tipa string, vsebuje ime plemena, za katerega računamo populacijo). Ta razred vsebuje konstruktor, ki prejme ime podatkovne baze, ki jo želimo uporabiti, "javne" metode **prepare_tabele()** (uporablja se za kreiranje potrebnih tabel), **calculate_gostote(pleme)** (uporablja se za izracun gostot) in tudi **display_gostote()** (požene se opcijsko, omogoča prikaz tabel). Ostale "privatne" metode le bolj podrobno razdelajo samo delovanje in se kličejo znotraj javnih metod avtomatsko.

Program vsebuje 2 skripti, **db_interf.py**, ki vsebuje razred DataBaseInterface ter vso logiko programa in skripto **run_db_interf.py**, ki se uporablja le za zagon programa (kreiranje razreda, povpraševanje po imenu plemena in podatkovne baze, itd.). Sama koda v teh datotekah je tudi podrobno zakomentirana, tako da je delovanje hitro razumljivo.

Sam program izvajamo preko ukazne vrstice z ukazom "*python run_db_interf.py*", skripta pa nas nato vpraša, katero shemo želimo uporabiti in za katero pleme želimo računati gostoto populacije. Da se program izvede v celoti, traja kar nekaj časa (lahko tudi do nekaj minut), saj Python ni najhitrejši jezik, poleg tega pa moramo izračunati gostote za ogromno podatkov (2x 2500).

4.2 PSEVDOKODA

```
1 vprašaj uporabnika po shemi -> shrani v spremenljivko database
2 vprašaj uporabnika po plemenu -> shrani v spremenljivko pleme
3 če pleme ni veljavno, ponovi korak 2
4 kreiraj objekt razreda DataBaseInterface in mu podaj spr. database kot argument
5 zbriši tabeli gostotaPopulacije in gostotaPlemena
6 ustvari tabeli gostotaPopulacije in gostotaPlemena
7 for x in range [-250, 250], x + 10
8         for y in range [-250, 250], y + 10
9                 iz baze beremo podakte o gostoti na območju [x, x + 10][y, y + 10]
10                podatke shrani v spremenljivko val
11                v tabelo gostote zapiši val / 100
```

5. NALOGA (ODBC)

*(**Bonus dodatnih 10% za demonstracijo)** V Pythonu napišite GUI aplikacijo (Qt ali podobno), ki se priključi na podatkovno bazo in v obliki grafov izriše rezultate izračunane gostote poselitev iz četrte naloge. V okviru te naloge lahko realizirate tudi celotno četrto nalogo, brez shranjevanja vmesnih rezultatov.

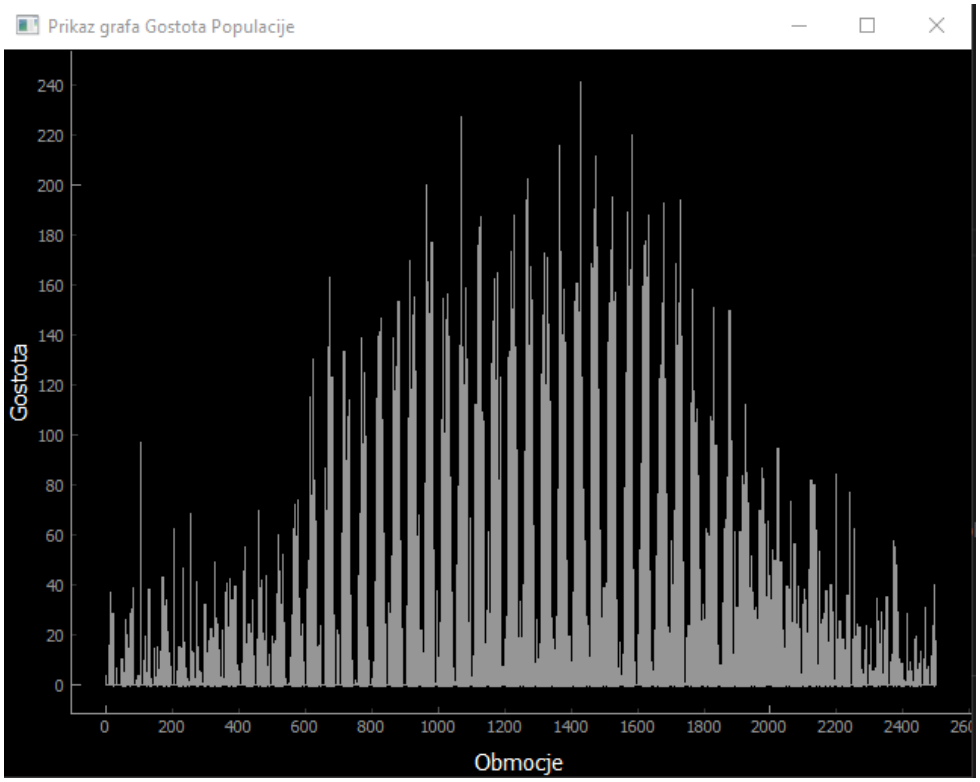
5.1 OPIS REŠITVE

Za reševanje te naloge sem uporabil Python knjižnico pyqtgraph, saj je preprosta za uporabo, zelo učinkovita, privzeto pa tudi omogoča premikanje po grafu in približevanje/oddaljevanje slike, kar nam omogoči makro in mikroskopski pogled na podatke. Knjižnico namestimo z ukazom "*pip install pyqtgraph*".

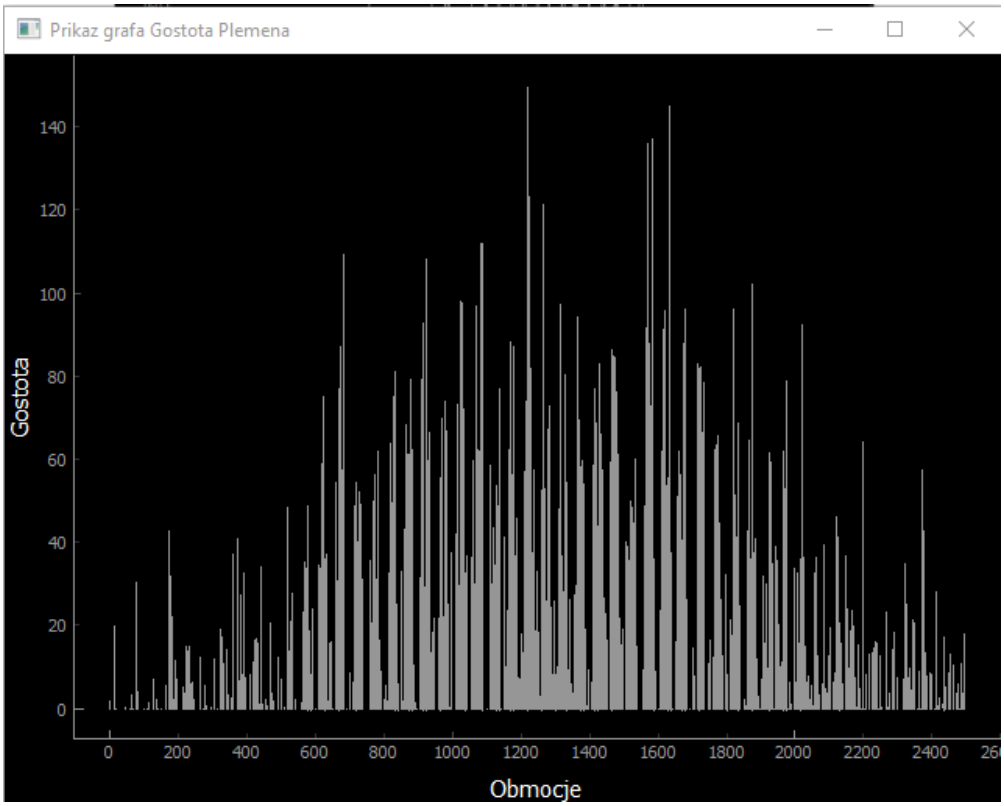
Tudi to nalogo sem reševal z objektno orientiranim pristopom, tako da sem ustvaril razred DataBaseGraphingInterface, ki vsebuje 2 atributa **window_ple** in **window_pop** (tipa PlotWidget, en za vsak graf), 2 atributa **xy_pop** in **xy_ple** (tipa dict, hranita vrednosti x in y za risanje grafov, x – zaporedna št. območja, y – gostota na tem območju), atribut **conn** (tipa pyodbc.Connection, hrani povezavo do podatkovne baze) in **connection_string** (tipa string, vsebuje podatke, potrebne za povezavo). Ta razred vsebuje konstruktor, ki prejme ime podatkovne baze, ki jo želimo uporabiti, "javno" metodo **draw()** (uporablja se za pripravo (branje) podatkov in izris grafov na pravilno okno). Ostale "privatne" metode le bolj podrobno razdelajo samo delovanje in se kličejo znotraj javnih metod avtomatsko.

Cel razred je napisan v skripti **db_graph_interf.py**, prav tako kot skripta pri nalogi 4 pa se poganja preko skripte **run_db_interf.py**. To skripto sem modificiral tako, da na začetku uporabnika vpraša, katero nalogo želi pognati. Zaradi same strukture programa moramo pred zagonom naloge 5 **vsaj enkrat uspešno pognati nalogo 4**, da se kreirajo in napolnijo potrebne tabele.

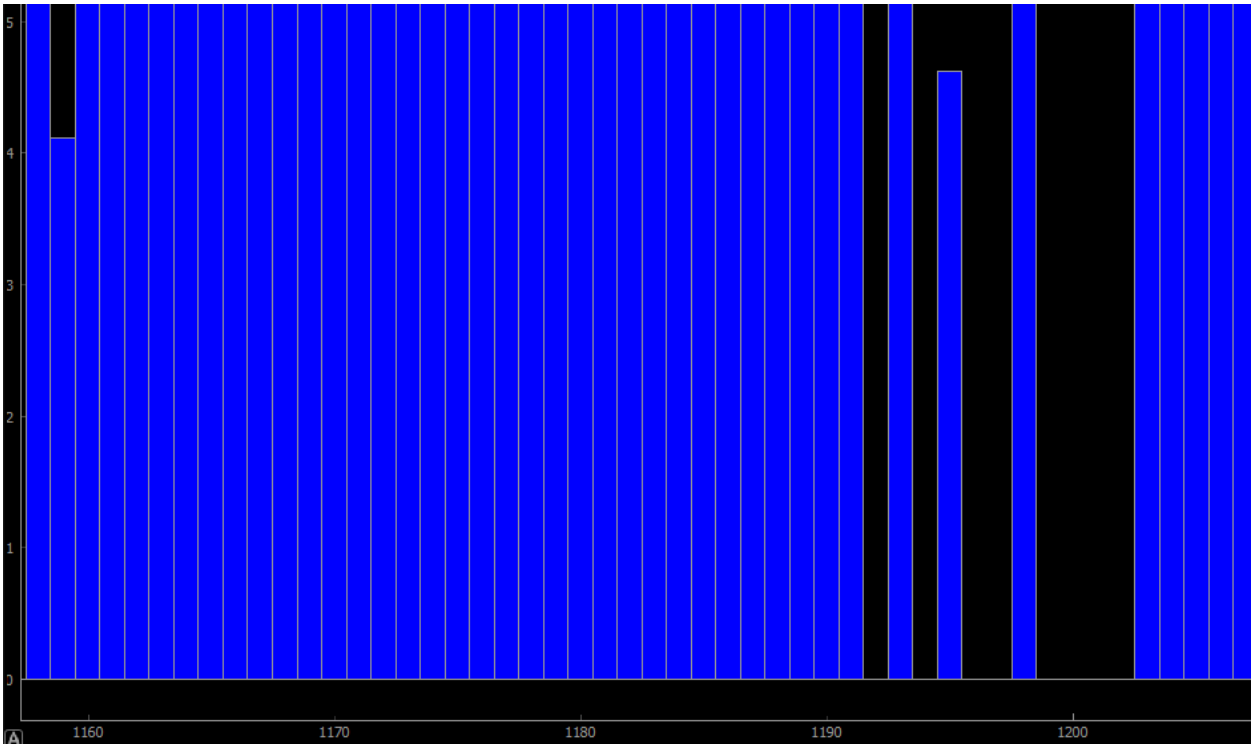
Kot lahko vidimo na naslednjih slikah, so posamezni stolpci dokaj tanki, saj jih je zelo veliko (2500). To niti ne predstavlja velikega problema, saj nam knjižnica pyqtgraph omogoča sprehanje po grafih z uporabo miške (koleščk, drag z levim gumbom, itd.), kar nam pomaga boljši pregled prikazanih podatkov.



Graf gostote populacije



Graf gostote plemena



Primer približanja