RAČUNALNA GEOMETRIJA I ROBOTSKI VID

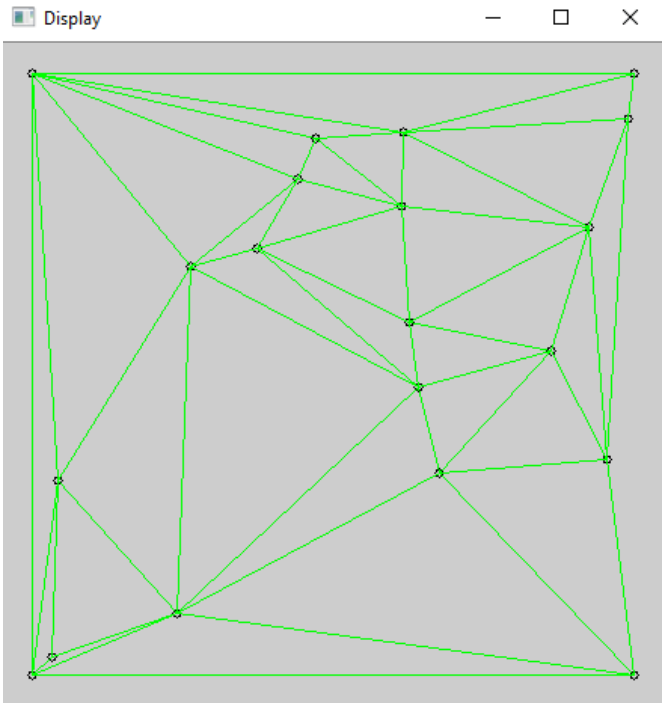
VJEŽBA 1: OPIS POLOŽAJA TIJELA U PROSTORU

Tomislav Rekić, 1.DRB

GitHub: https://github.com/tomislavrekic/lv2-rgrv

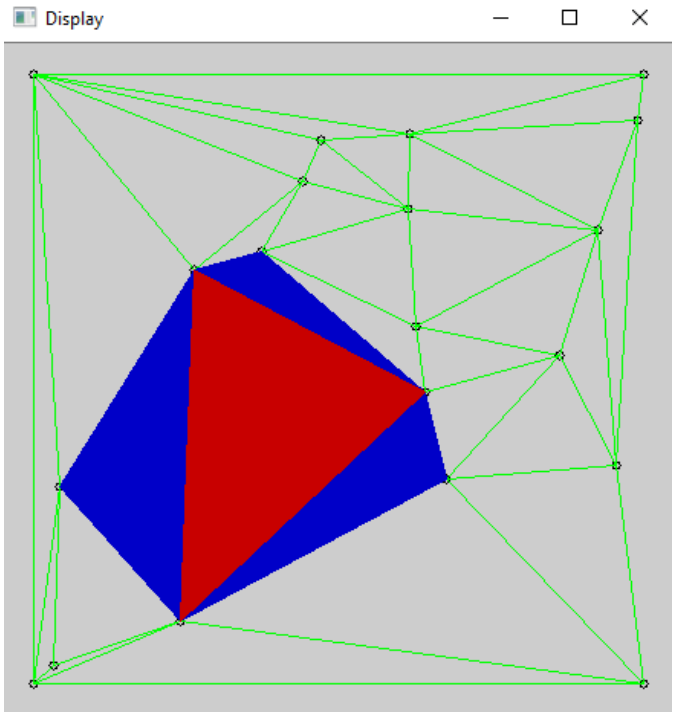
Zadatak:
Izraditi konzolnu aplikaciju koja:
1. generira 20 nasumično izabranih točaka unutar kvadrata dimenzije 400 piksela;
2. generira Delaunay triangulaciju skupa točaka iz koraka 1;
3. prikazuje triangulaciju dobivenu u koraku 2 u prozoru dimenzija 440 x 440;
4. omogućuje izbor nekog od trokuta triangulacije klikom lijeve tipke miša;
5. izabrani trokut prikazuje crvenom bojom, a njemu susjedne trokute (one koji s njim dijele zajedničku stranicu) prikazuje plavom bojom.

Delaunay triangulacija se lagano dobiva pomoću klase **Subdiv2D** koja se nalazi unutar bibiloteke OpenCV. Ona se inicijalizira nad pravokutnikom koji se predaje kao parametar konstruktoru Subdiv2D objekta. Kada je **Subdiv2D** objekt inicijaliziran, pozivom njegove metode **insert()** unosimo nove točke u prostor gdje se Delaunay triangulacija odvija. Unosom novih točaka Delaunay triangulacija se ažurira uzimajući u obzir novu točku. U sklopu ove vježbe morali smo unijeti 4 rubne točke smještene na koordinatama (20, 20), (20, 420), (420, 20), (420, 420) i 16 točaka smještenih nasumično unutar rubnih točaka. Stvorena Delaunay triangulacija se može prikazati koristeći funkciju **imshow()**. Takav prikaz prikazan je na slici 1.



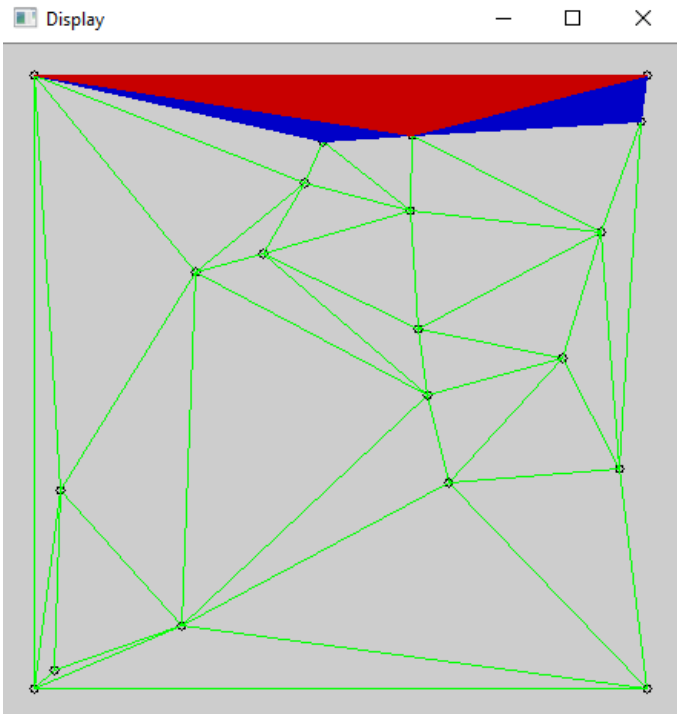
Slika 1 - Primjer Delaunay triangulacije

Klikom miša unutar jednog od trokuta Delaunay triangulacije pronalazi se trokut u kojem se pokazivač miša nalazi, taj trokut se oboja u crveno te se susjedni trokuti oboje u plavo (slika 2).



Slika 2 - Pronalazak odgovarajućih trokuta

Ako se odabere rubni trokut, rub Delaunay triangulacije ne smije se obojati u plavo (slika 3).



Slika 3 - Odabir rubnog trokuta

Aktivacija klika miša vrši se pozivom funkcije **setMouseCallback** prikazane na programskom kodu 1. Toj funkciji se predaje korisnički definirana funkcija **onMouse** te također korisnički definirana instanca strukture **userData** prikazane na programskom kodu 2.

```
cv::setMouseCallback("Display", (MouseCallback)onMouse, userData);
```

Programski kod 1 - funkcija setMouseCallback

```
struct UserData {
	int pointNum;
	Subdiv2D subdiv;
	Mat image;
};
```

Programski kod 2 - Struktura UserData

Funkcija **onMouse** prolazi kroz sve trokute Delaunay triangulacije, računa njihove normale i središta stranica (programski kod 3), te pronalazi onaj koji zadovoljava uvjete opisane na programskom kodu 4.

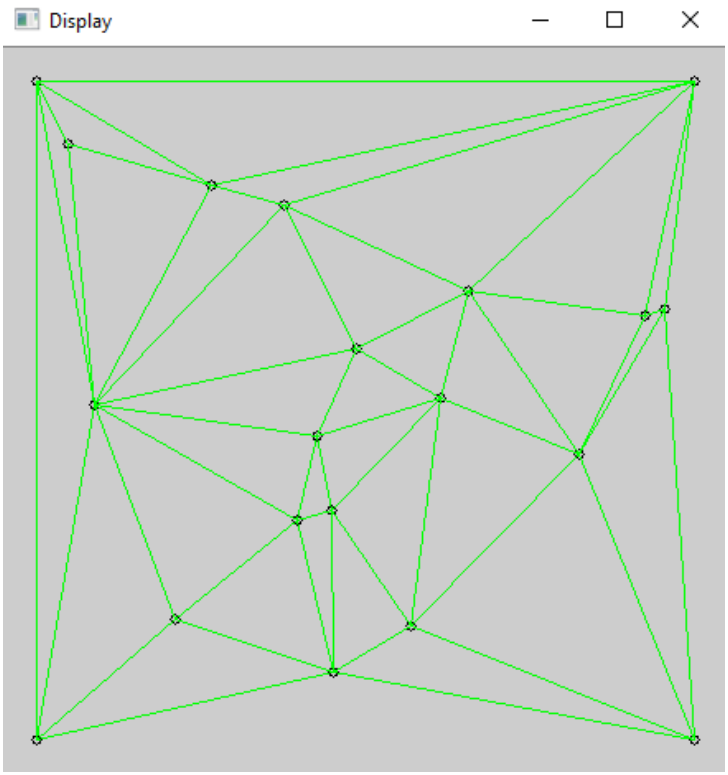
```
int k = 0;
for (int j = 0; j < 3; j++) {
	k = j + 1;
	if (k == 3) k = 0;
	edgeVectors[j] = Point2f(Point2f(verts[k]->x - verts[j]->x, verts[k]->y -
	verts[j]->y));

	float edgeModulus = sqrt((edgeVectors[j].x * edgeVectors[j].x) +
	(edgeVectors[j].y * edgeVectors[j].y));
	edgeNormal[j] = Point2f((edgeVectors[j].y / edgeModulus), (-edgeVectors[j].x /
	edgeModulus));

	edgeMidpoint[j] = Point2f((verts[j]->x + verts[k]->x) / 2, (verts[j]->y +
	verts[k]->y) / 2);
}
```

Programski kod 3 - Pronalazak normala i središta stranica

```
int counter = 0;
for (int j = 0; j < 3; j++) {
	if (((edgeNormal[j].x * (x - edgeMidpoint[j].x)) + (edgeNormal[j].y * (y -
	edgeMidpoint[j].y))) <= 0) {
	counter++;
	}
}
```

Programski kod 4 - Pronalazak odgovarajućeg trokuta

Glavna zadaća je pronalazak odgovarajućeg trokuta, nakon čega slijedi kod koji redom prolazi kroz točke trokuta i pronalazi točke susjednih trokuta. Iz praktičnih razloga se cijeli kod neće objasniti unutar dokumenta, ali se nalazi na GitHub stranici čiji link se nalazi na prvoj strani. Dodatan primjer Delaunay triangulacije je prikazan na slici 4.



Slika 4 - Delaunay triangulacija