SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 2926

3D interaktivna vizualizacija simulacije vibracija

Nikola Vugdelija

Zagreb, lipanj 2022.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 11. ožujka 2022.

DIPLOMSKI ZADATAK br. 2926

Pristupnik:	**Nikola Vugdelija (0036509409)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	prof. dr. sc. Krešimir Matković

Zadatak: **3D interaktivna vizualizacija simulacije vibracija**

Opis zadatka:

Vibracije mehaničkih dijelova raznih uređaja jedan su od glavnih izvora buke. S porastom svijesti o štetnosti buke i sve strožim normama, analiza vibracija postaje sve važnija. Rezultati simulacije vibracija predstavljaju velike i složene podatke. Interaktivna vizualizacija znatno olakšava njihovu analizu. Osim dijagrama koji prikazuju pregled cijelog spektra u ovisnosti od radne točke i frekvencije, često je potrebno prikazati rezultate i na 3D modelu simuliranog uređaja. Vaša je zadaća proučiti metode interaktivne 3D vizualizacije općenito, osnovne zadatke kod analize vibracija i, na temelju osnovnih zadataka, osmisliti i implementirati metode koje će omogućiti prikaz vibracija za nekoliko frekvencija simultano na 3D modelu. Posebni pažnju treba posvetiti interakciji. Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Rok za predaju rada: 27. lipnja 2022.

Zahvala će biti ovdje kad je formuliram u glavi.

SADRŽAJ

POPIS SLIKA

1. Uvod

Buka i vibriranje motora su jedni od ključnih faktora koji utječu na osjećaj udobnosti putnika u osobnom automobilu. No, iako se može reći kako je osjećaj udobnosti subjektivne prirode, propisi koji kontroliraju dozvoljene razine buke motora vozila pri određenim brzinama su objektivni i egzaktni. Zbog navedenih razloga je inženjerima iznimno važno održati vibriranje motora na što nižoj razini prilikom rada, a najjednostavniji način na koji to mogu provjeriti u fazi projektiranja motora je pomoću NVH (engl. *Noise, Vibration, Harshness*) simulacija. NVH simulacije inženjerima omogućuju precizan uvid u razinu vibriranja pojedinih dijelova motora, a probleme koji mogu uzrokovati vibraciju mogu rješavati prilikom projektiranja.

No, računanje simulacije je samo dio problema. Računske simulacije ove vrste, nerijetko proizvode podatke koje se ne može jednostavno vizualizirati i analizirati [10]. Kako bi stručnjaci mogli navedene podatke koristiti na što efikasniji način, potrebno ih je intuitivno i razumljivo prikazati. Iako su podaci prikazani u obliku tablica i grafova korisni, vizualizacijom istih podataka na 3D modelu motora se postiže jasniji, intuitivniji i potpuniji prikaz rada motora.

Ovaj rad opisuje izradu alata za vizualizaciju podataka dobivenih iz NVH simulacije pomoću 3D modela motora i 2D grafova. Fokus predstavljenog alata je na vizualizaciji vibracija motora prilikom rada na višestrukom broju frekvencija, kako bi se razine vibracije različitih taktova motora jednostavnije uspoređivale.

2. Pregled područja

Iako često zanemarena, vizualizacija podataka je iznimno važno područje analize podataka. Pošto isti skup podataka može prenijeti potpuno drugačije informacije ovisno o načinu prikaza, od velike je važnosti odabrati optimalne vizualizacijske tehnike za konkretni skup podataka. Ali kako je naglašeno u [11], kod vizualizacije ne postoji jedna, "optimalna" grafika koja vrijedi za svaki skup podataka, nego je često potrebno pronaći grupu vizualizacijskih tehnika koje se međusobno upotpunjuju i prikazuju cjelovitu sliku. Stoga je tijekom osmišljavanja alata za vizualizaciju izrazito važno utvrditi za koju svrhu se taj alat koristi, te koje informacije je potrebno njime vizualizirati iz danog skupa podataka. Odgovori na navedena pitanja postaju kompleksniji za odgonetnuti s porastom kompleksnosti skupa podataka koji alat vizualizira. Zbog toga je pomoć stručnjaka u domeni itekako korisna.

Kao što je već spomenuto, glavna svrha alata razvijenog u sklopu ovog rada je vizualizacija NVH podataka na modelu motora. Prilikom izrade ovog rada je korišten isti skup podataka koji je korišten kod [10]. Spomenuti podaci su izračunati NVH simulacijom pomoću alata AVL EXCITE™[1]. Motor je zadan kao skup ćelija, a za svaku ćeliju su poznate koordinate okolnih točaka u 3D prostoru, te kojom snagom ćelija vibrira pri različitim frekvencijama rada.

Kako je spomenuto u [10], stručnjake pri analizi NVH podataka ponajprije zanimaju tri stavke:

- Usporedba podataka na različitim frekvencijama rada
    - Kako se vibracije mijenjaju sa promjenom frekvencije rada motora?
- Lokalizacija značajki
    - Koje ćelije motora pri zadanoj frekvenciji vibriraju iznosom koji pripada zadanom rasponu?
- Lokalna pretraga

- Kojom snagom vibriraju odabrane ćelije?

Alat razvijen u [10] dobro adresira navedene značajke. Koristeći paralelne koordinate, omogućen je detaljan pregled usporedbe podataka na različitim frekvencijama. Adresiranje druge i treće značajke je olakšano omogućavanjem korištenja višestrukih, međusobno povezanih prozora sa 3D prikazom motora. Lokalna pretraga je ostvarena uz pomoć "kista", kojeg korisnik može koristiti za biranje raspona iznosa vibracija za određene frekvencije, a svi prozori automatski ažuriraju prikaz modela motora označujući ćelije koje pripadaju zadanim rasponima. Konačno, u [10] je lokalna pretraga omogućena bojanjem modela motora bazirano na vrijednostima vibracija pojedinačnih ćelija pri radu na zadanoj frekvenciji. Detaljniji uvid u snagu vibriranja pojedine ćelije modela je omogućen odabirom interesne ćelije, nakon čega se istakne 2D graf vibracija zadane ćelije.

No, iako je referirani rad cjelokupno odlično pokrio sve tri značajke, u području usporedbe podataka na različitim frekvencijama rada još ima prostora za napredak. Naime, trenutna usporedba podataka na različitim frekvencijama je moguća samo pomoću 2D grafova, koji iako korisni, imaju problema sa čitljivošću i jasnoćom. Alat koji je razvijen u sklopu ovog rada implementira niz vizualizacijskih tehnika koje potencijalno povećavaju intuitivnost usporedbe vibracija motora na različitim frekvencijama u odnosu na izvedbu kod [10].

3. Analiza zadatka i zahtjeva

Pošto dizajniranje alata za vizualizaciju nije jednostavan zadatak, potrebno je poduzeti određene predprodukcijske korake kako bi se osigurala zadovoljavajuća implementacija glavnog zadatka. Navedeni je zadatak potrebno dodatno analizirati, čime bi se uočili zahtjevi potrebni za njegovo uspješno odrađivanje. Identificirane zahtjeve je također potrebno analizirati, kako bi se utvrdila njihova povezanost sa glavnim zadatkom, te osiguralo njihovo uspješno i cjelovito zadovoljavanje.

3.1. Identificirani zadaci

Ključni zadatak alata razvijenog u sklopu ovog rada je prikazivanje razlika i sličnosti između različith frekvencija rada ćelija motora na intuitivan i razumljiv način. Kao što je već navedeno u prethodnom poglavlju, alat iz [10] ima problema sa čitljivošću i jasnoćom usporedbe utjecaja više frekvencija na vibraciju motora. Spomenuta usporedba je jedino moguća korištenjem grafova kod kojih svaka linija predstavlja funkciju vibracije jedne ćelije (x-os predstavlja sve frekvencije za koje postoje podaci, a y-os predstavlja snagu vibracije). Takav način rada stvara nezanemarive poteškoće sa korištenjem alata: teže je usporediti frekvencije koje nisu susjedne na grafu, čitljivost grafa je uvelike smanjena kada je odabrano više ćelija i otežano je uočiti gdje se točno na modelu motora nalaze razlike u radu motora.

3.2. Identificirani zahtjevi

Kako bi se usporedbu učinilo intuitivnijom i jasnijom, očiti korak je omogućiti iscrtavanje rezultata usporedbe na 3D modelu motora. Naime, iako se u [10] ćelije bojaju na temelju jačine vibracija pri radu na zadanoj frekvenciji, nije moguće iscrtati utjecaj više frekvencija na modelu motora. Takvim iscrtavanjem bi se uveliko olakšalo uočavanje promjena na dijelovima motora. Naravno, kod više odabranih frekvencija, postavlja se pitanje: kako iz niza podataka o snagama vibracije ćelija pri odabranim

frekvencijama odabrati jedan konačni iznos koji će pomoću zadanog raspona i gradijenta odrediti konačnu boju ćelije kojoj pripada. Alat razvijen u sklopu ovog rada korisniku nudi nekoliko funkcija koje može odabrati, a koje primaju niz iznosa vibracija te vraćaju jednu vrijednost vibracije, koja predstavlja "sažetak" navedenog niza. Za tu svrhu su u sklopu alata implementirane metode:

- MIN - najmanji iznos vibracije
- MAX - najveći iznos
- AVERAGE - prosjek vibracija
- MEDIAN - medijan vibracije
- SPREAD - razlika između najvećeg i najmanjeg iznosa vibracije

Korisnik zatim može odabrati funkciju koja najbolje odgovara njegovim potrebama. Na primjer, ako korisnik treba saznati maksimalno opterećenje za svaku ćeliju pri odabiru različitih frekvencija za to mu može koristiti funkcija MAX, a ako ga zanima kolike će vibriranje ćelija varirati tijekom rada na višestrukim frekvencijama može koristiti funkciju SPREAD itd.

No, kada navedene funkcije iz niza vrijednosti izračunaju konačnu vrijednost, taj iznos treba prebaciti u interval [0, 1] kako bi se omogućilo uzorkovanje gradijenta. Za ostvarivanje prijelaza u taj interval potrebno je znati koji broj označava najmanju vrijednost intervala, odnosno 0, a koji broj označava najveću vrijednost intervala, odnosno 1. U alatu su ponuđena tri načina računanja navedenog raspona:

- GLOBAL - donju i gornju granicu raspona čine najmanji i najveći iznos snage vibracije kojom neka ćelija može vibrirati tijekom rada na bilo kojoj od mogućih frekvencija
- LOCAL - donju i gornju granicu raspona čine najmanji i najveći iznos snage vibracije kojom neka ćelija može vibrirati tijekom rada na bilo kojoj od odabranih frekvencija
- USER DEFINED - korisnik sam definira donju i gornju granicu raspona

Alat korisniku omogućuje definiranje boje gradijenta na temelju kojeg će se modelu odrediti boje ćelija.

U sklopu alata je implementiran još jedan način "bojanja" modela motora koji je baziran na unaprijed određenim ograničenjima vibracija. Navedena ograničenja dijele raspon vibracija na 3 zone: bezopasnu, rizičnu i opasnu. Rasponi su definirani za određene frekvencije (ne nužno za sve) u posebnoj datoteci. Ovaj način bojanja je odvojen od normalnog načina rada, objašnjenog u prijašnjem odlomku. Boje ćelija u načinu rada s ograničenjima se određuju na sljedeći način:

- Zelena - ako se snaga vibracije trenutne ćelije tijekom rada na svim odabranim frekvencijama nalazi u bezopasnoj zoni
- Gradijent žute - ako se snaga vibracije trenutne ćelije tijekom rada na barem jednoj odabranoj frekvenciji nalazi u rizičnoj zoni, a nijedna u opasnoj zoni
- Gradijent crvene - ako se snaga vibracije trenutne ćelije tijekom rada na barem jednoj odabranoj frekvenciji nalazi u opasnoj zoni

Navedeni gradijenti se uzorkuju pomoću omjera broja frekvencija prilikom kojih snaga vibracije ćelije spada u zonu gradijenta i ukupnog broj odabranih frekvencija. Navedena boja i gradijenti su početno postavljeni na spomenute vrijednosti, a korisnik ih može mijenjati kako njemu odgovara.

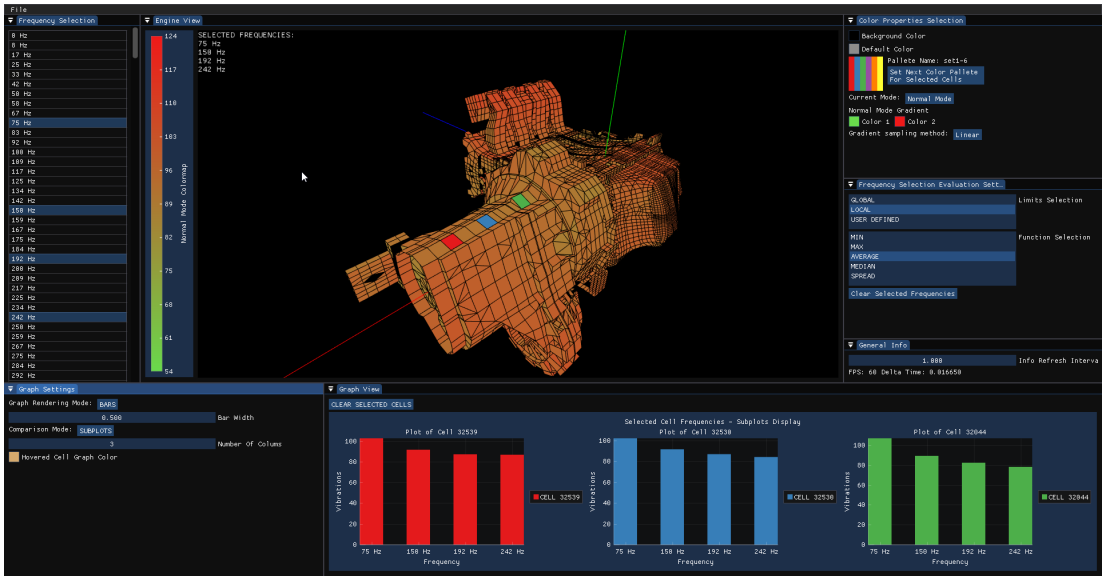
Kako bi se adresirao problem sa uspoređivanjem nesusjednih frekvencija, na grafu se iscrtavaju samo podaci za odabrane frekvencije. Konačno, kako bi se poboljšala čitljivost grafa u sklopu alata su implementirana dva načina iscrtavanja podataka: stupčasti i linijski, te tri načina uspoređivanja:

- DEFAULT - svi grafovi su prikazani unutar jednog prikaza
- SUBPLOTS - svaki graf je prikazan unutar zasebnog prikaza
- RELATIVE - kao DEFAULT, ali je dodatno moguće odabrati jednu ćeliju s obzirom na koju će se prikazati grafovi svih ostalih ćelija

Navedeno rješenje predstavlja poboljšanje na području čitljivosti zbog više razloga. Grafovi svih ćelija više nisu prikazani odjednom, već su samo prikazani grafovi odabranih ćelija. Moguće je prikazati svaki graf na zasebnom prikazu, čime je pojednostavljeno uspoređivanje u slučaju više odabranih ćelija. Sa relativnim uspoređivanjem, korisnik može lakše usporediti odabranu ćeliju sa ostalim ćelijama. Konačno, pošto su omogućena dva načina iscrtavanja podataka, korisnik može odabrati način koji najbolje odgovara njegovim potrebama i preferencijama, te nije prisiljen koristiti linije za iscrtavanje grafova.

4. Dizajn vizualizacije

Zahtjevi navedeni u prijašnjem poglavlju daju naslutiti kako će sučelje alata biti kompleksno, pošto korisniku trebaju biti na raspolaganju brojne opcije. Kako bi se svi navedeni zahtjevi zadovoljili, a upotrebljivost alata se ne bi smanjila, potrebno je alat organizirati u smislene prozore, koje će korisnik moći premiještati, povećavati i smanjivati kako mu najbolje odgovara. Na slici 4.1 se može vidjeti kako je alat zapravo organiziran u sedam prozora. Navedeni prozori su detaljno opisani u nastavku.

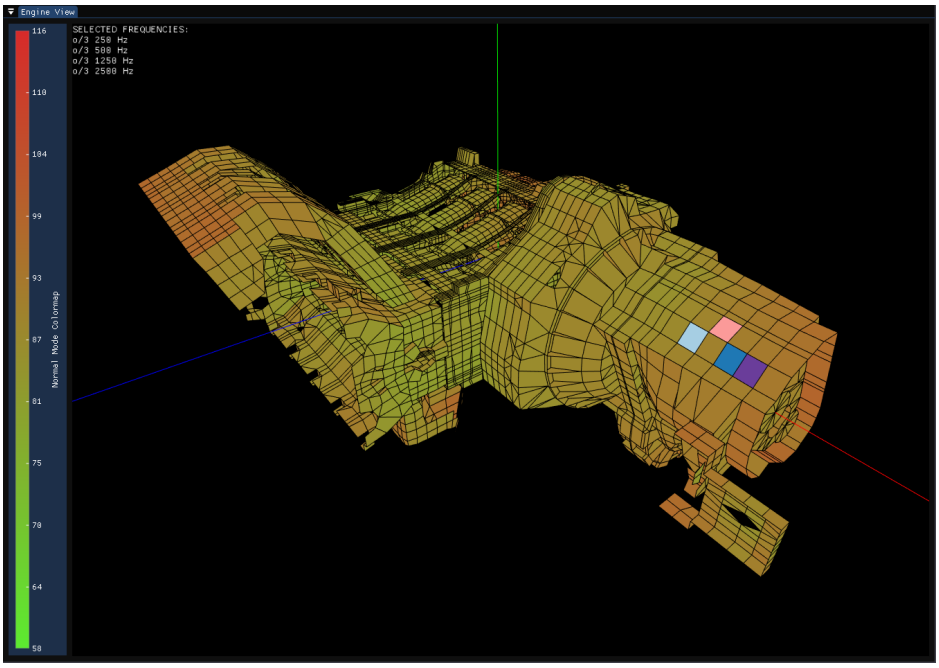


Slika 4.1: Snimka zaslona alata

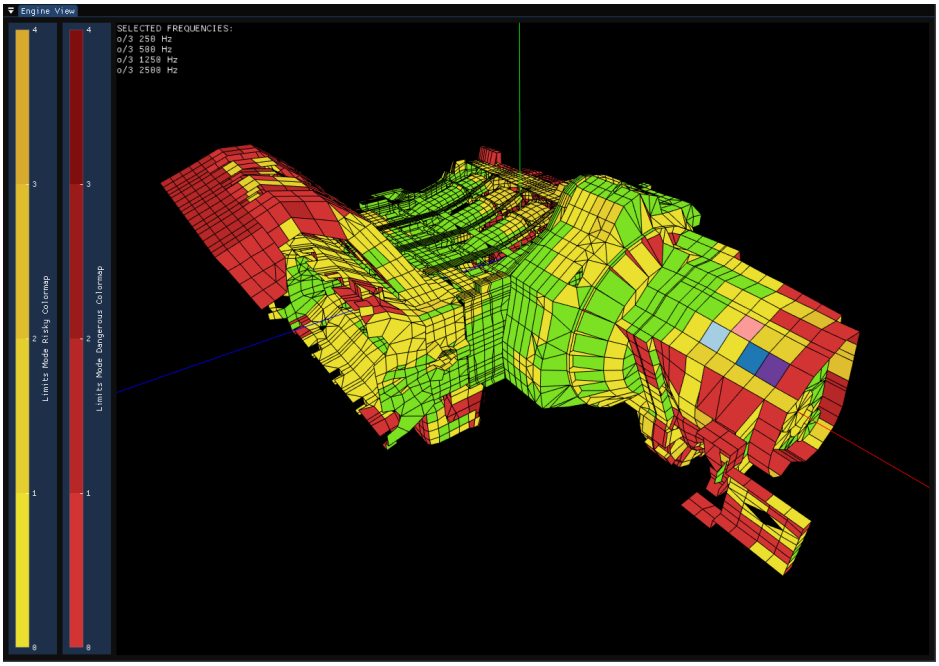
4.1. Prikaz motora

Zadaća prikaza motora je jasno vizualizirati podatke o vibracijama ćelija, pritom uzimajući u obzir brojne postavke vizualizacije poput: načina rada, odabranih frekvencija, načina računanja konačnih podataka ćelije, zadanih raspona podataka, gradijenata koje se koristi za vizualizaciju i načina uzorkovanja tih gradijenata. Korisnik ovu kompo-

nentu također može koristiti i za označavanje interesnih ćelija.



Slika 4.2: Prikaz motora tijekom normalnog načina rada

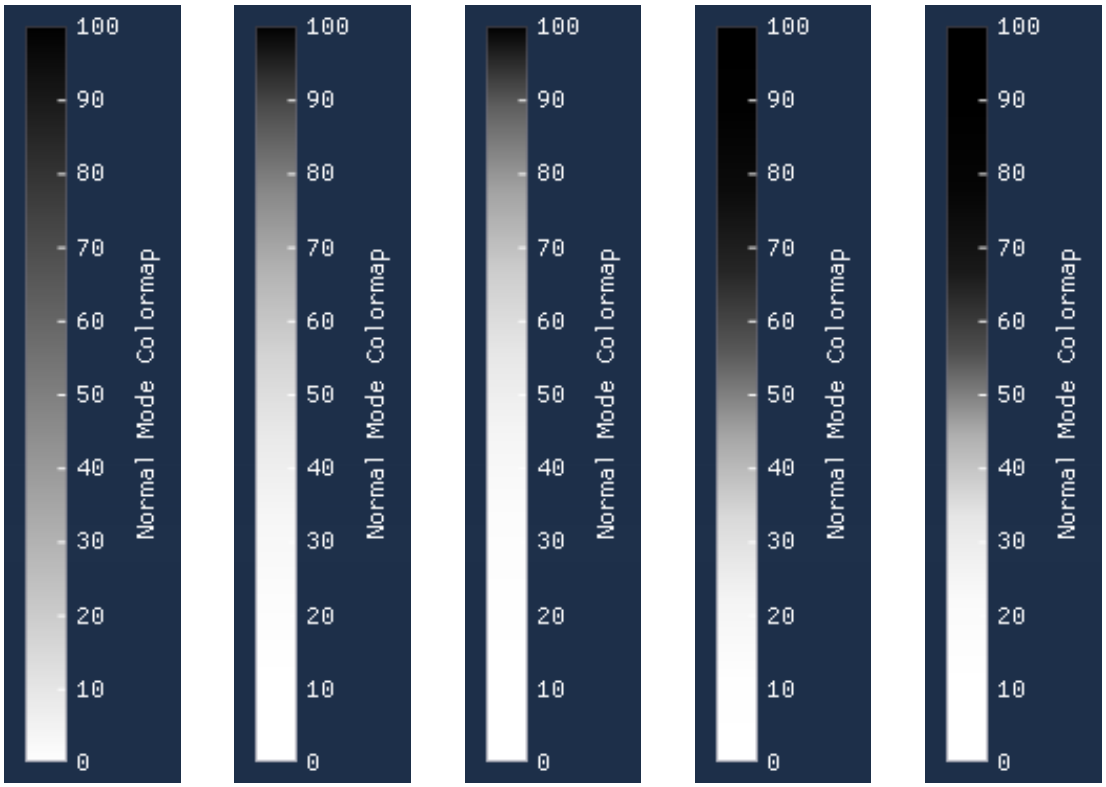


Slika 4.3: Prikaz motora tijekom načina rada sa preddefiniranim ograničenjima

Na slikama 4.2 i 4.3 se može vidjeti primjer ovog prikaza tijekom dva moguća načina rada. Uz prikaz obojanog modela ova komponenta alata je zadužena i za prikaz

legende gradijenta i nabrajanje odabranih frekvencija, kako bi snimak prikaza prenosio sve relevantne informacije potrebne za razumijevanje rezultata vizualizacije. Osi koordinatnog sustava su nacrtane uz model motora kako bi korisnik dobio osjećaj za 3D prostor u kojem se model nalazi. Označavanje ćelija se odrađuje pozicioniranjem pokazivača miša iznad interesne ćelije te klikom na lijevu tipku miša. Kada korisnik pokazivačem miša prijeđe preko ćelije bez klikanja, boja ćelije postaje inverz prijašnje boje, kako bi se korisniku dala povratna informacija o ćeliji koja će biti odabrana odluči li se korisnik kliknuti lijevu tipku miša. Opisana ćelija će se u nastavku rada nazivati "lebdeća" ćelija. Jednom kada korisnik klikne na ćeliju, ćelija se označava jednom od boja iz trenutne palete (detaljno objašnjeno u poglavlju 4.3) te je korisniku prikazan detaljan uvid u podatke označene ćelije putem prikaza grafa (vidi poglavlje 4.2). Ponovnim klikom na već odabranu ćeliju, ćelija se briše iz liste odabranih ćelija.

Kao što je vidljivo iz slika 4.2 i 4.3, razlike u dva načina rada se ne manifestiraju samo u vizualizaciji motora, već i u sučelju ovog prikaza. Naime, kako je objašnjeno u poglavlju 3.2, u normalnom načinu rada alatu je potreban jedan kontinuirani gradijent, a u načinu rada sa preddefiniranim ograničenjima su potrebna dva diskretna gradijenta.



(a) Linearno **(b)** Kubno v1 **(c)** Kvartičko v1 **(d)** Kubno v2 **(e)** Kvartičko v2

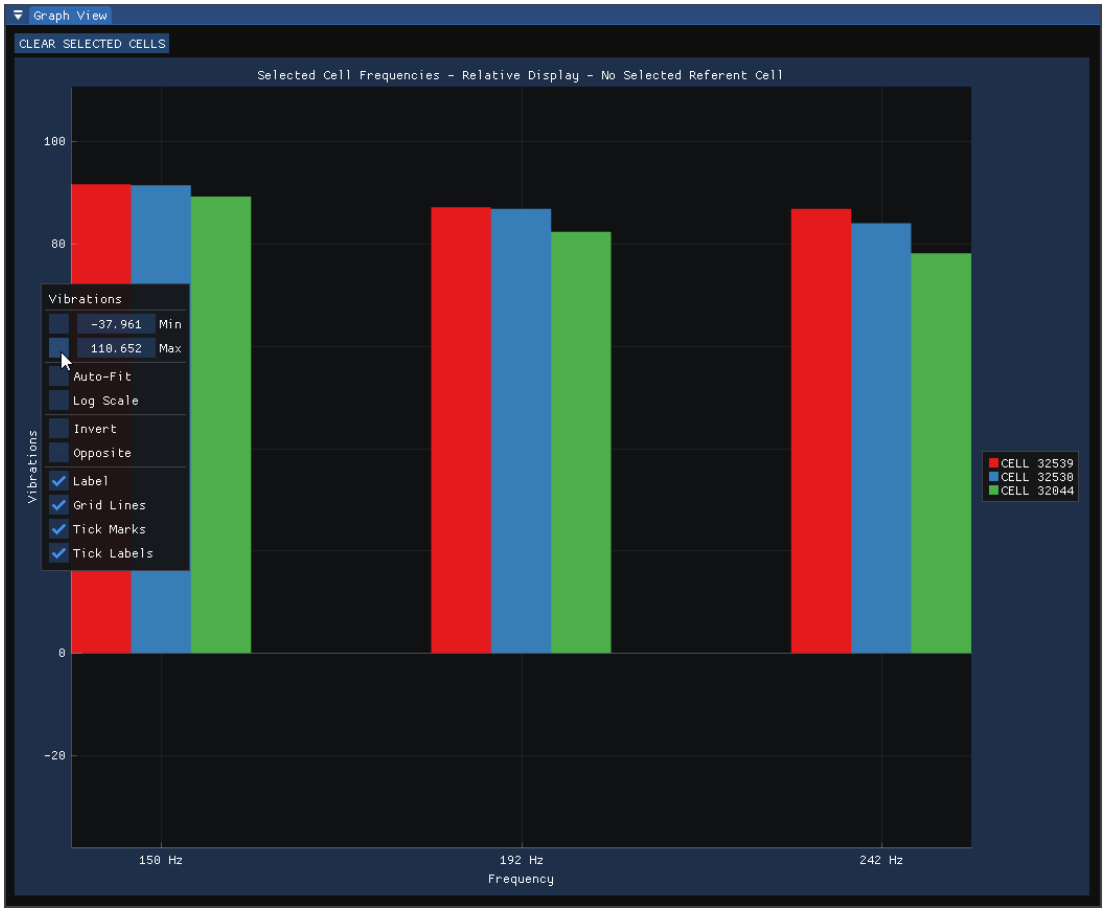
Slika 4.4: Drugačije postavke uzorkovanja gradijenta

Osim što je navedene gradijente moguće mijenjati pomoću dvije kontrolne boje, moguće ih je mijenjati i promjenom funkcije uzorkovanja gradijenta, a utjecaj različitih funkcija uzorkovanja na kontinuirane gradijente se može vidjeti na slici 4.4. Različite funkcije uzorkovanja su direktno preuzete sa [3].

4.2. Prikaz grafa

Komponenta prikaza grafa služi za iscrtavanje jednog ili više prikaza grafova odabranih ćelija, ali i "lebdeće" ćelije. Prilikom iscrtavanja spomenutih grafova uzimaju se u obzir brojne postavke i načini crtanja. Boje grafova odabranih ćelija odgovaraju bojama koje te ćelije imaju unutar prikaza motora (vidi poglavlje 4.1), a boja grafa "lebdeće" ćelije je zadana unutar postavki grafa (vidi poglavlje 4.4). Kao što je navedeno u poglavlju 3.2, alat implementira tri načina usporedbe i dva načina iscrtavanja grafova, a izgled ove komponente uz sve moguće kombinacije postavki grafa su prikazane na slikama u dodatku A. Na navedenim slikama se može vidjeti kako ovu komponentu čini jedan ili više prikaza grafova i gumb za brisanje liste odabranih ćelija.

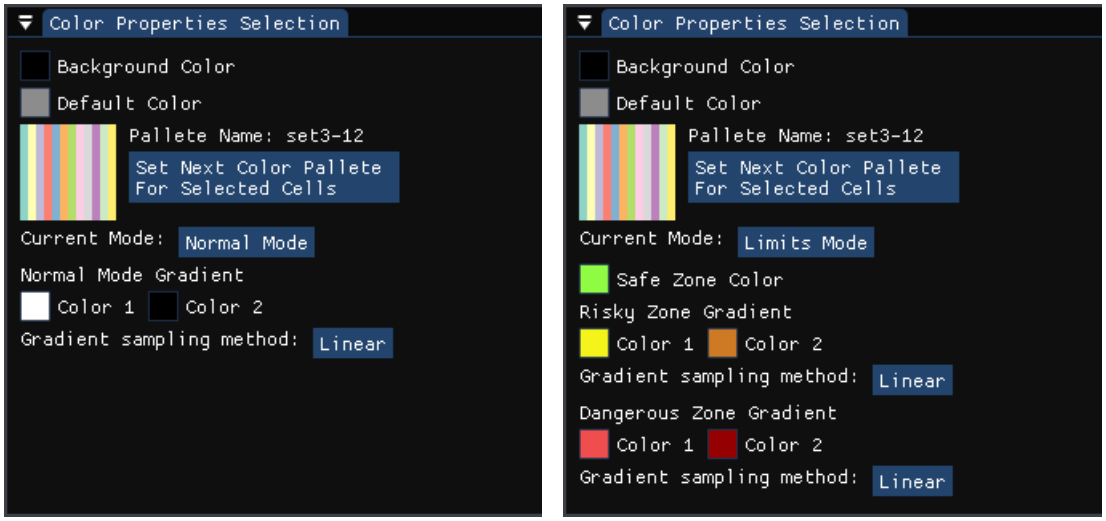
Sa slika u dodatku A je također vidljivo kako su rasponi osi prikaza grafova postavljeni tako da obuhvaćaju podatke koji se u njima nalaze, te nisu fiksno određeni. Implementacija ovakvog načina određivanja raspona je odabrana zbog smanjene preglednosti razlika u "tokovima" grafova kada su postavljeni fiskni rasponi, pošto pri fiksnim rasponima grafovi zauzimaju puno manji dio prikaza, a ostatak prikaza je potrošen na prazan prostor. Spomenuta primjedba je posebno uočljiva na slikama A.3 i A.4 gdje se vidi kako, zbog različitih raspona, grafovi koji su iscrtani pomoću stupaca bolje pokazuju razliku u iznosima podataka, ali grafovi iscrtani pomoću linija bolje pokazuju razlike u "tokovima" grafova. Konkretno, na slici A.3 se na prvi pogled čini kako se zelena ćelija (ćelija 10982) i žuta ćelija (ćelija 11182) ponašaju slično, no uvidom u graf na slici A.4 se otkriva kako to nije istina. Ali ako se korisniku ovakav način određivanja raspona ne sviđa, u alatu je moguće otvoriti postavke bilo koje osi prikaza grafa pozicioniranjem pokazivača miša iznad željene osi te klikom na desnu tipku miša. Kao što se vidi na slici 4.5, unutar spomenutih postavki je moguće uređivati brojna svojstva osi poput raspona, načina interpoliranja između granica raspona, smjera osi i mnoge druge postavke.



Slika 4.5: Postavke osi prikaza grafa

4.3. Postavke boja

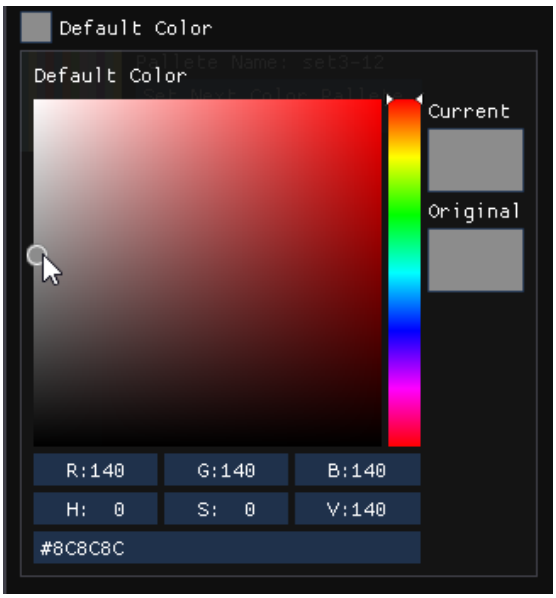
Kao što samo ime ove komponente kaže, glavna svrha joj je čuvanje postavki koje bi mijenjale boje i gradijente unutar prikaza motora. Na slici 4.6 mogu se vidjeti razlike u komponenti između dva moguća načina rada. U oba načina rada moguće je promijeniti pozadinsku boju prikaza motora, početnu boju motora (boju koju motor ima kad nijedna frekvencija nije odabrana), paletu za odabrane ćelije i trenutni način rada. No dva prikaza postavki boja se ipak djelomično razlikuju, pošto je za normalni način rada potrebno definirati samo jedan gradijent, a za način rada sa preddefiniranim ograničenjima je potrebno definirati jednu boju (za sigurnu zonu) i dva gradijenta (za rizičnu i opasnu zonu).



(a) Normalni način rada **(b)** Način rada sa preddefiniranim ograničenjima

Slika 4.6: Postavke boja tijekom dva moguća načina rada

Što se tiče mijenjanja postavki, korisnik može mijenjati boje klikom na obojani kvadrat čime se otvara prozor za mijenjanje boja prikazan na slici 4.7. U otvorenom prozoru boju se može mijenjati pomoću interaktivnog GUI sučelja ili unosom RGB ili HSV vrijednosti boje, a također je moguće i unošenje boja u heksadekadskom formatu.



Slika 4.7: Prozor za odabir boje

Palete boja za odabrane ćelije su učitane izravno iz datoteke *default_pallete.txt* koja se mora nalaziti u istom direktoriju kao i izvršna datoteka alata. Početne palete su preuzete sa [2] i unesene u navedenu datoteku u sljedećem formatu:

```
p ime_palete
rgb_vrijednosti_boje1_u_rasponu_0_255
rgb_vrijednosti_boje2_u_rasponu_0_255
rgb_vrijednosti_boje3_u_rasponu_0_255
rgb_vrijednosti_boje4_u_rasponu_0_255
```

Na slici 4.6 se može vidjeti kako je za svaku paletu u kvadratiću prikazan spektar boja, desno od kojeg je navedeno ime palete, a ispod imena se nalazi gumb za prelazak na sljedeću paletu. Kako palete sadrže ograničen broj boja, u alatu je implementiran način brisanja ćelija kada je broj odabranih ćelija veći od broja boja u paleti. Postoje dva slučaja kod kojih se to dogodi: pri dodavanju nove ćelije kada su već sve boje palete "zauzete" i pri mijenjanju trenutne palete na paletu čiji je broj boja manji od broja trenutno odabranih ćelija. U prvom slučaju alat briše zadnju odabranu ćeliju i dodaje novoodabranu ćeliju, a u drugom slučaju alat briše onoliko zadnjih odabranih ćelija kolika je razlika između broja odabranih ćelija i broja boja nove palete.

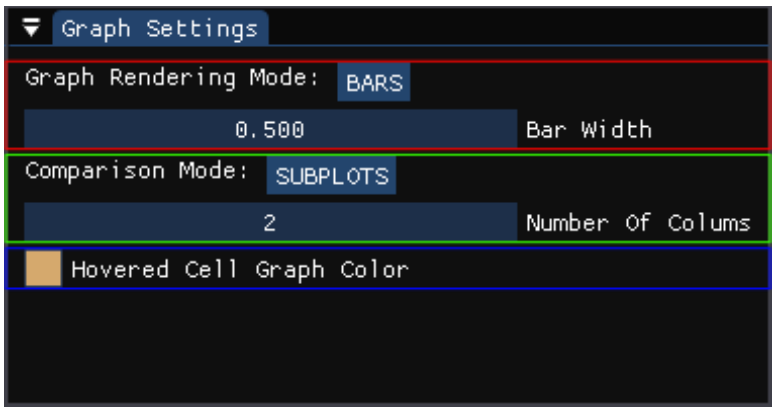
Ispod sučelja za mijenjanje palete se nalazi gumb za mijenjanje načina rada. Na gumbu piše ime trenutnog načina rada, a klikom na gumb alat prelazi u sljedeći način rada. Kako su moguća samo dva načina rada, gumb će alternirati između normalnog načina rada i načina rada sa preddefiniranim ograničenjima.

Konačno, sučelje za odabir gradijenta se sastoji od dva standardna kvadrata za mijenjanje boja pomoću kojih se mijenja početna i krajnja kontrolna boja gradijenta. Ispod navedenih kvadrata se nalazi gumb za mijenjanje metode uzorkovanja gradijenta. Implementirane metode uzorkovanja i njihov utjecaj na gradijent su objašnjene i prikazane u poglavlju 4.2.

4.4. Postavke grafa

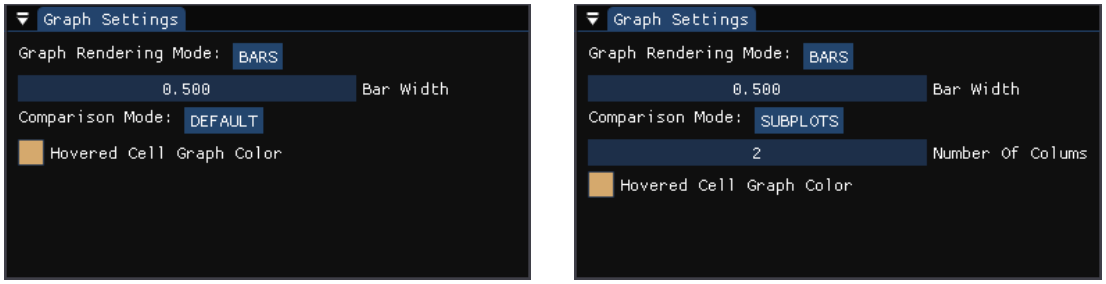
Postavke grafa se dijele na tri dijela koji su jasno naznačeni na slici 4.8. Dio u kojem se mijenja boja grafa lebdeće ćelije se ne mijenja s obzirom na različite načine crtanja ili uspoređivanja jer se taj graf uvijek iscrtava. Kao što je već napomenuto u poglavlju 3.2, alat implementira dva načina iscrtavanja grafa: linijski i stupčasti, a razlika u postavkama između ta dva načina je što stupčastom grafu korisnik može odrediti ši-

rinu stupca, kao što se može vidjeti na slici 4.8, dok na linijskom grafu korisnik ne može mijenjati ništa. Način iscrtavanja se mijenja klikom na gumb na kojem piše ime trenutno aktivnog načina iscrtavanja.

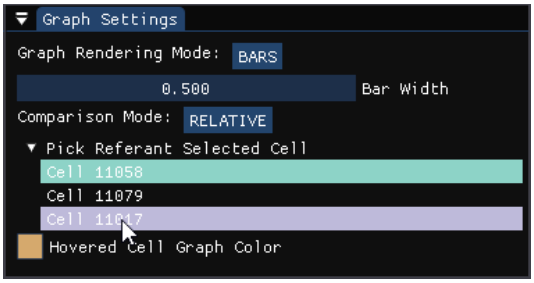


Slika 4.8: Postavke grafa rastavljene na dijelove: postavke načina iscrtavanja (crveno), postavke načina usporedbe (zeleno) i postavke boje grafa "lebdeće ćelije" (plavo)

Ali za razliku od postavki načina iscrtavanja, razlike između postavki načina uspoređivanja su puno uočljivije. Tijekom normalnog načina usporedbe, korisnik nema nikakvih dodatnih postavki osim gumba za mijenjanje načina usporedbe koji je prisutan tijekom svih načina uspoređivanja i funkcionira na identičan način kao i gumb za mijenjanje načina iscrtavanja. Tijekom načina usporedbe sa višestrukim prikazima, korisnik može odabrati u koliko će se stupaca prikazi organizirati. Konačno, kada je aktivan relativni način usporedbe, korisnik može odabrati jednu od već odabranih ćelija s obzirom na koju će se prikazati grafovi ostalih ćelija. Gumb ćelije koju korisnik odabere, ali i gumb ćelije iznad koje korisnik postavi pokazivač miša poprima boju odgovarajuće ćelije iz prikaza motora. Navedeni dodatak je implementiran kako bi korisnik imao bolju ideju o tome koja ćelija je trenutno odabrana i/ili koja će ćelija biti odabrana za relativno iscrtavanje.



(a) Normalni način usporedbe **(b)** Način usporedbe sa višestrukim prikazima

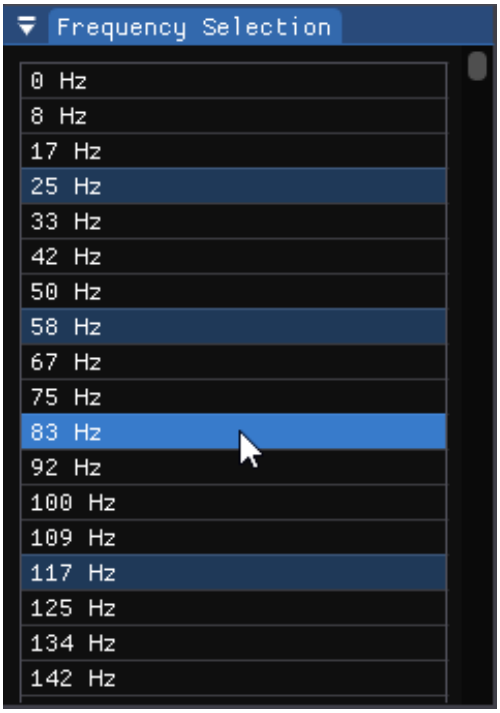


(c) Relativni način usporedbe

Slika 4.9: Postavke grafa tijekom tri različita načina usporedbe

4.5. Odabir frekvencija

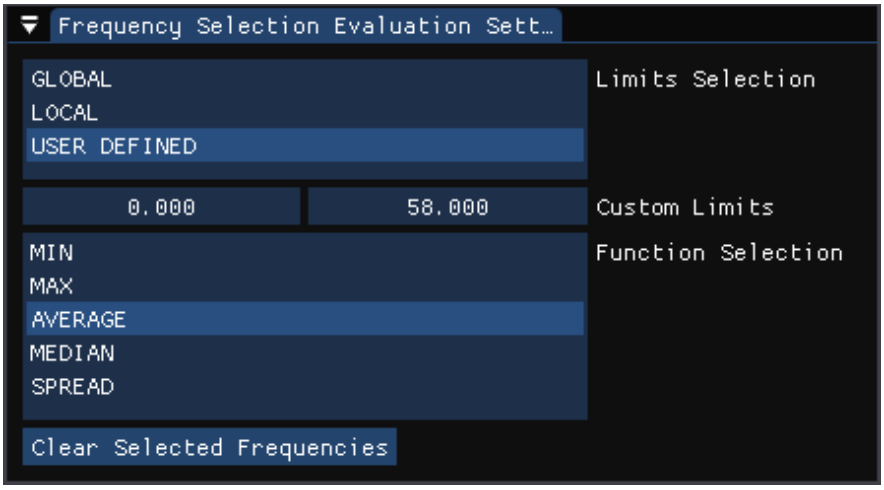
Ova komponenta služi isključivo za odabir frekvencija (kao što i samo ime govori), a postavljena je u odvojenu komponentu od postavki evaluacije odabranih frekvencija (opisanih u poglavlju 4.6) radi poboljšanja lakoće korištenja alata. Izgled komponente se može vidjeti na slici 4.10. U normalnom načinu rada komponenta prikazuje sve frekvencije za koje su učitani podaci o vibriranju ćelija, dok su u načinu rada sa predde-finiranim ograničenjima prikazane samo frekvencije za koje je definirano ograničenje u datoteci sa ograničenjima.



Slika 4.10: Prozor za odabir frekvencija

4.6. Postavke evaluacije odabranih frekvencija

Kao što je već opisano u poglavlju 3.2, korisnik u normalnom načinu rada može dodatno prilagoditi prikaz utjecaja odabranih frekvencija na vibraciju motora odabirom različitih metoda pomoću kojih se iz niza podataka o vibraciji ćelije dobije jedan konkretan podatak, ali i odabirom raspona pomoću kojeg se izračunati podatak pretvori u podatak iz raspona [0, 1] kako bi se olakšalo uzorkovanje gradijenta. Navedene prilagodbe nisu moguće prilikom načina rada sa preddefiniranim ograničenjima, pošto je algoritam već "strogo" zadan i nema dijelova algoritma koje korisnik može mijenjati, ali spomenute parametre normalnog načina rada, korisnik može mijenjati unutar komponente postavki odabranih frekvencija koja je prikazana na slici 4.11.



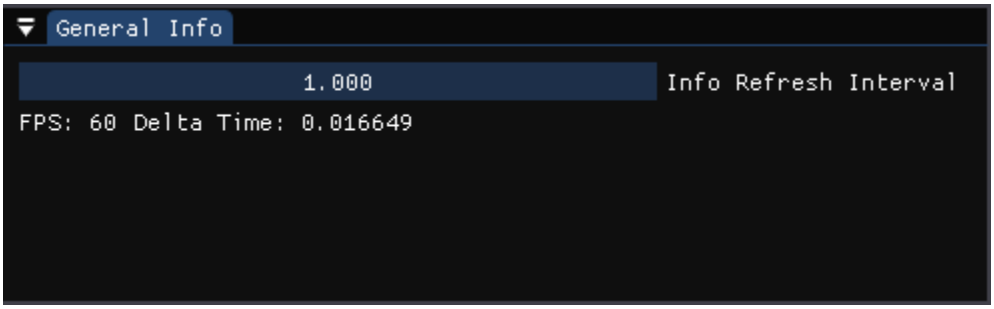
Slika 4.11: Prozor postavki evaluacije odabranih frekvencija sa svim mogućim parametrima

Prozor postavki evaluacije odabranih frekvencija nije vidljiv sve dok korisnik ne odabere jednu od ponuđenih frekvencija unutar komponente iz poglavlja 4.5. Kada se odabere barem jedna frekvencija, prozor postavki se prikazuje sa parametrima koji su vidljivi ovisno o broju odabranih frekvencija i načinu rada u kojem korisnik koristi alat.

Oba načina rada u postavkama imaju gumb za brisanje liste trenutno odabranih frekvencija. Način rada sa preddefiniranim ograničenjima osim tog gumba u postavkama neće imati niti jedan drugi element, a normalni način rada će moći mijenjati već spomenute metode računanja konačnih podataka i raspone izračunatih podataka. Lista za odabir raspona je vidljiva kada korisnik odabere barem jednu frekvenciju, a mijenjanje raspona definiranog od korisnika je omogućeno kada je navedeni raspon odabran u spomenutoj listi. Lista za odabir metoda računanja konkretnih podataka je vidljiva kada korisnik odabere dvije ili više frekvencije, pošto se pri jednoj odabranoj frekvenciji sve funkcije ponašaju identično.

4.7. Općenite informacije

Na slici 4.12 se može vidjeti komponenta koja služi za prikaz općenitih informacija vezanih za performanse alata poput broja sličica po sekundi i vremena proteklog između iscrtavanja dviju sličica. Korisnik dodatno može promijeniti interval nakon kojeg se navedene informacije ažuriraju.



Slika 4.12: Prozor za općenite informacije

5. Implementacija

Alat je razvijen u programskom jeziku C++ uz korištenje biblioteke OpenGL za komuniciranje sa grafičkom karticom. No, osim OpenGL-a za razvoj alata je korišten i niz drugih biblioteka, čiji su detaljniji opisi i razlozi za korištenje navedeni u poglavlju 5.1. Pošto su se tijekom razvoja alata konstantno pojavljivali novi zahtjevi i funkcionalnosti koje se treba implementirati, potrebno je bilo uspostaviti arhitekturu rješenja koje će se rješenje pridržavati kako ne bi postalo kruto i krhko, a ta arhitektura je opisana u poglavlju 5.2. Konačno, kako je način slanja podataka sa procesora na grafičku karticu iznimno važna komponenta izvedbe svake grafičke aplikacije, poglavlje 5.3 opisuje kako je organiziran tok podataka sa procesora na grafičku karticu.

5.1. Korištene biblioteke

5.1.1. OpenGL

OpenGL[9] je višeplatformska i višejezična biblioteka koja služi za prikazivanje 2D i 3D vektorske grafike. Biblioteka podržava dva distinktna načina rada: *compatibility* i *core*. U *compatibility* načinu rada klijent definira poziciju i boju točaka primitiva koje želi iscrtati pozivajući funkcije API-a, nakon čega se navedene točke propuštaju kroz fiksni grafički protočni sustav na kraju kojega se iscrtava slika. *Core* način rada, često zvani moderni OpenGL, dopušta klijentu mijenjanje određenih dijelova grafičkog protočnog sustava pomoću sjenčara (engl. *shader*). Sjenčari su programi napisani u GLSL-u (engl. *OpenGL Shading Language*) koji se izvode na grafičkoj kartici, te manipuliraju podacima poslanim od strane klijenta preko OpenGL API-a u obliku tzv. *buffer object*-a kako bi se postigli različiti vizualni efekti.

Za razvoj alata opisanog u ovom radu je korišten OpenGL 3.3 u *core* načinu rada. Iako je u sklopu implementacije alata ova biblioteka prvenstveno korištena za iscrtavanje 3D modela motora, implementacije biblioteka Dear ImGui (poglavlje 5.1.2) i

ImPlot (poglavlje 5.1.3) korištene u ovom radu također koriste OpenGL za iscrtavanje elemenata korisničkog sučelja i 2D grafova.

5.1.2. Dear ImGui

Dear ImGui[6] je C++ biblioteka koja služi za iscrtavanje grafičkog korisničkog sučelja ili GUI-a (engl. *Graphical User Interface*).

GUI biblioteke se obično mogu razvrstati na zadržane (engl. *retained*) i izravne (engl. *immediate*) s obzirom na korištene obrasce ažuriranja prikaza. Zadržane biblioteke interno čuvaju podatke o primitivima korisničkog sučelja koje treba iscrtati, a klijent pozivima metoda biblioteke ne može izravno utjecati na iscrtavanje tih primitiva, već jedino može mijenjati apstraktni model korisničkog sučelja. Na ovakav način zadržane biblioteke mogu dodatno optimizirati kako i kada će se izvršiti iscrtavanje elemenata. S druge strane, izravne GUI biblioteke omogućuju klijentu izravno pozivanje metoda za iscrtavanje elemenata korisničkog sučelja. Zbog toga klijent treba pozivati funkcije za iscrtavanje primitiva u svakoj iteraciji petlje ažuriranja aplikacije.

Dear ImGui spada u kategoriju izravnih GUI biblioteka. Ova biblioteka omogućuje jednostavno i brzo iteriranje GUI aplikacija, te unatoč tome što joj nedostaju određene funkcionalnosti koje se mogu naći u drugim GUI bibliotekama, i dalje je iznimno popularna u području razvoja GUI aplikacija zbog jednostavnosti korištenja i nadogradnje.

5.1.3. ImPlot

Iako Dear ImGui službeno podržava crtanje grafova, funkcionalnosti grafova implementiranih u sklopu ove biblioteke su iznimno ograničene. Iz tog razloga, za implementaciju grafova u ovom radu je korištena biblioteka ImPlot[7]. Ova biblioteka je proširenje Dear ImGui-a koje omogućava prikaz raznovrsnih interaktivnih grafova jednostavnim izravnim pozivima metoda za iscrtavanje.

5.1.4. Ostale biblioteke

Osim već spomenutih biblioteka, za razvoj ovog alata je korištena još nekolicina drugih biblioteka koje su navedene u ovom poglavlju.

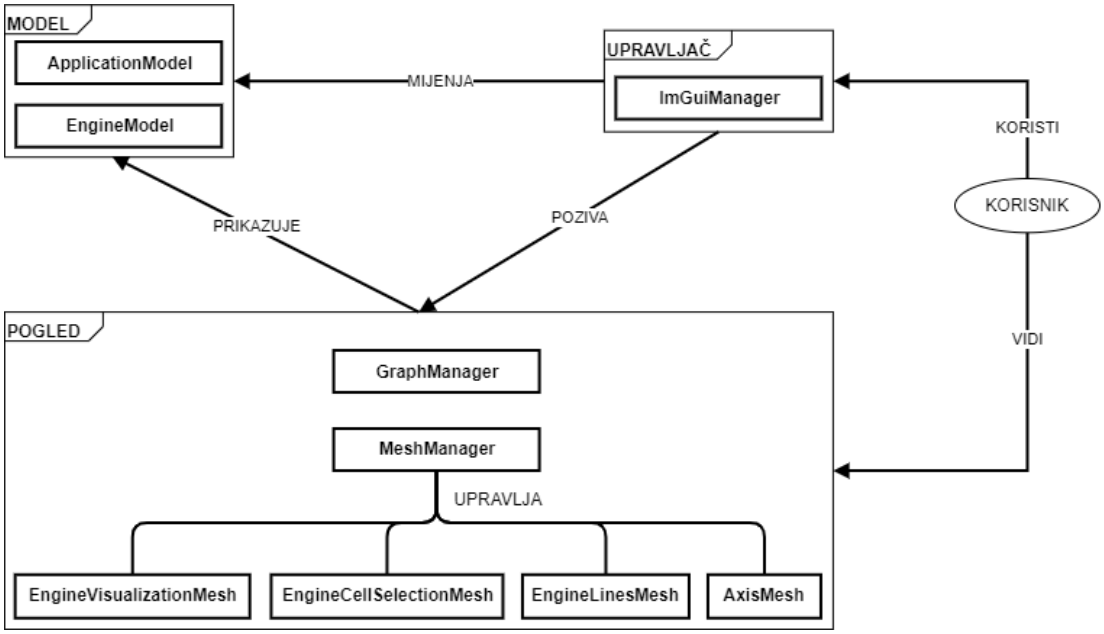
GLFW[4] je višeplatformska biblioteka za razvoj OpenGL aplikacija, a u sklopu ove aplikacije biblioteka je korištena za stvaranje prozora, konteksta, javljanje događaja i primanje ulaza od korisnika.

GLM[5] je matematička biblioteka za grafičke aplikacije pisane u OpenGL-u. GLM nudi brojne klase i funkcije koje su dizajnirane i implementirane sa istim konvencijama imenovanja i funkcionalnostima kao odgovarajući tipovi podataka i funkcije u GLSL-u. GLM je u ovom alatu korišten za operacije nad vektorima, te računanje matrica transformacije i projekcije.

NFD[8] (engl. *Native File Dialog*) je višeplatformska C biblioteka koja pruža mogućnost otvaranja izvornog dijaloga za datoteke, a u sklopu alata je korištena prilikom učitavanja različitih datoteka sa podacima o motoru.

5.2. Arhitektura rješenja

Organizacija koda alata je generalno inspirirana obrascem Model-Pogled-Upravljač ili MVC (engl. *Model-View-Controller*) kako bi se omogućilo lakše održavanje programa. Na slici 6.18 se može vidjeti pojednostavljeni prikaz MVC organizacije alata sa većinom ključnih klasa alata.



Slika 5.1: Pregled MVC organizacije alata

Važno je spomenuti kako na slici 6.18 nedostaje glavna klasa alata, a to je klasa *App*, pošto ne sudjeluje direktno u MVC arhitekturi. Glavna odgovornost ove klase je povezivanje prikazanih komponenti, obrađivanje GLFW događaja i pozivanje metoda za ažuriranje spomenutih komponenti tijekom iteracija petlje ažuriranja. Spomenuta klasa stvara instance komponenti MVC arhitekture, te ih međusobno povezuje slanjem preko argumenata konstruktora ili povezivanjem preko sustava događaja i signala (detaljno opisanih u dodatku B). U poglavljima koje slijede su opisane uloge, odgovornosti i dijelovi svake komponente MVC organizacije ovog alata.

5.2.1. Model

Osnovna odgovornost svakog Modela u sklopu MVC arhitekture je čuvanja podataka aplikacije i pravila za njihovu manipulaciju. Radi lakšeg razumijevanja koda i raspodjele odgovornosti, Model ovog alata je podijeljen u dvije klase: *EngineModel* i *ApplicationModel*. Objekt klase *EngineModel* čuva postavke i podatake vezane za motor i prikaz motora, poput odabranih frekvencija, odabranih ćelija, "trenutne" lebdeće ćelije, pozicija točaka motora, ćelija motora itd., te definira metode za mijenjanje navedenih podataka. S druge strane, odgovornost klase *ApplicationModel* je čuvanje postavki aplikacije koje nisu izravno vezane za podatke o motoru ili prikaz tih podataka, te definiranje metoda za vanjsko mijenjanje tih postavki. Spomenute postavke i podaci uključuju razne postavke grafova, postavke bojanja motora, pozadinsku boju alata te poziciju i rotaciju kamere.

Navedene klase nemaju izravne reference na nijedan drugi dio MVC arhitekture, stoga svaki put kada se dogodi promjena u Modelu, oni je javljaju "pretplatnicima" na tu promjenu pomoću događaja i/ili signala, čime se smanjuje krutost programskog rješenja. Što se mijenjanja postavki tiče, objekti spomenutih klasa interno sadrže objekte klase *VariableMap* (detaljno opisana u dodatku C) čija je odgovornost čuvanje varijabli postavki, te pozivanje zadane metode kada se dogodi promjena u nekoj varijabli. Na taj način su omogućeni povratni pozivi, koje Dear ImGui za većinu elemenata korisničkog sučelja zapravo ne podržava. Kako bi se promjene u Modelu ispitale prilikom svake iteracije petlje ažuriranja alata, već spomenuta klasa *App* poziva odgovarajuće metode objekata klasa *EngineModel* i *ApplicationModel* koje interno zovu metode vlastitih *VariableMap* objekata.

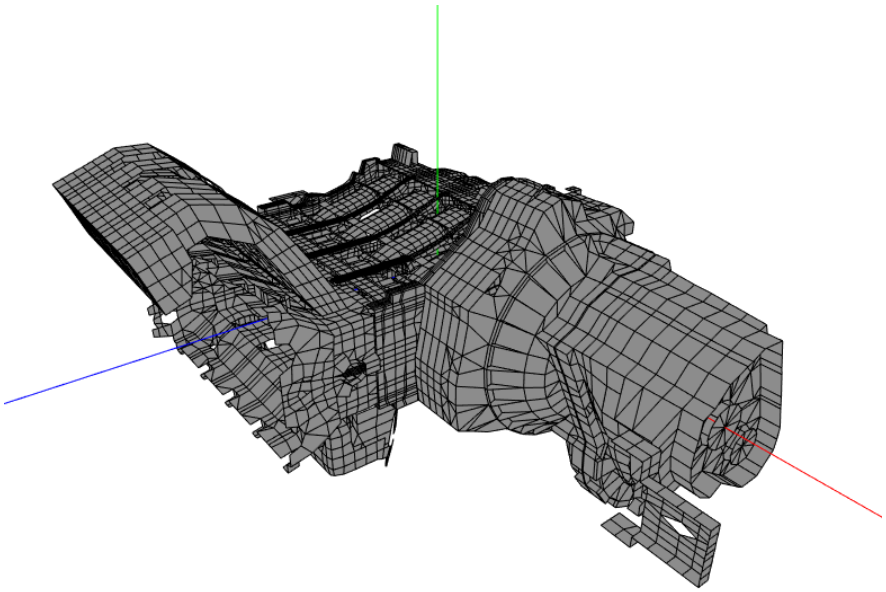
5.2.2. Pogled

Pogled je, kao komponenta MVC-a, uglavnom odgovoran za prikaz podataka komponente Model na koristan i praktičan način. Na slici 6.18 se može vidjeti kako je komponenta Pogled, baš kao i Model, implementirana pomoću dvije klase: *MeshManager* i *GraphManager*.

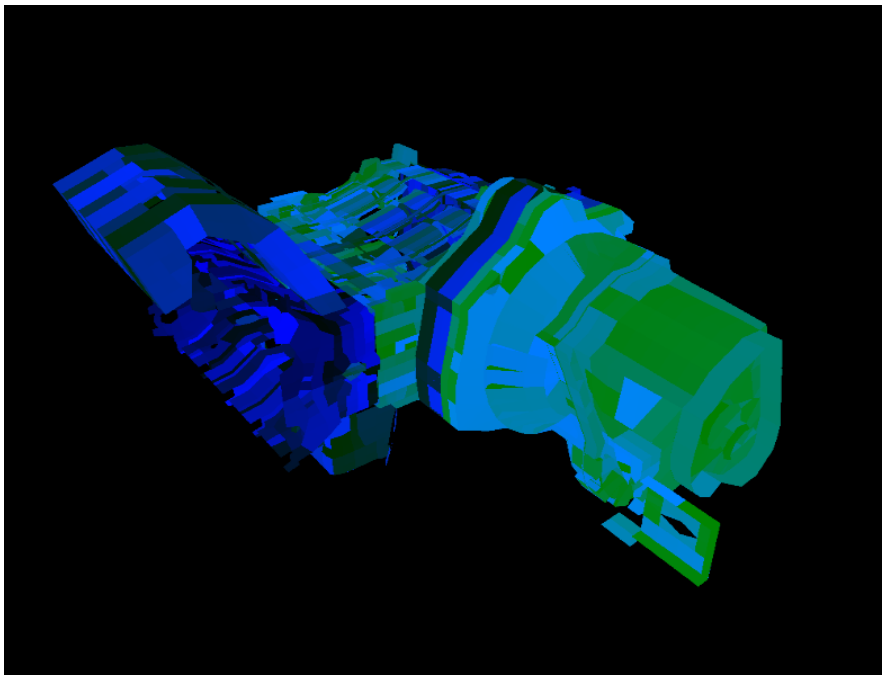
Ključna odgovornost klase *MeshManager* je iscrtavanje poligona i linija uz pomoć OpenGL-a na temelju podataka iz komponente Model. Kako se može vidjeti na skici organizacije arhitekture koda, klasa *MeshManager* ovu odgovornost razlaže na četiri dodatne klase: *EngineVisualizationMesh*, *EngineCellSelectionMesh*, *EngineLinesMesh* i *AxisMesh*. Navedene klase nasljeđuju apstraktnu klasu *AbstractMesh*, s time da sve klase osim klase *AxisMesh* klasu ne nasljeđuju izravno, već preko apstraktne klase *AbstractEngineMesh*. Implementirana je i klasa *Shader* kako bi se apstrahirale operacije vezane za sjenčare, pa svaka od spomenutih komponenata *MeshManager*-a sadrži vlastitu instancu klase *Shader* i definira vlastite *buffer object*-e koje šalje na grafičku karticu (više u poglavlju 5.3). Klasa *App* u svakoj iteraciji petlje ažuriranja pozove metodu *render* klase *MeshManager* čija je odgovornost ažuriranje prikaza modela motora u stvarnom vremenu.

Rezultati iscrtavanja se ne spremaju izravno u glavni grafički međuspremnik (engl. *framebuffer*), već u dva sporedna međuspremnika. U prvi međuspremnik rezultate iscrtavanja upisuju objekti klasa *EngineVisualizationMesh*, *EngineLinesMesh* i *AxisMesh*. Klasa *EngineVisualizationMesh* računa boje ćelija pomoću podataka iz Modela te iscrtava "obojani" motor. Klasa *EngineLinesMesh* iscrtava linije koje obrubljuju ćelije kako bi se lakše prepoznale njihove granice, a klasa *AxisMesh* služi za iscrtavanje koordinatnih osi kako bi korisnik dobio osjećaj za 3D prostor u kojem se motor nalazi. Opisani grafički međuspremnik se prikazuje korisniku tako što se u prikazu motora zapravo crta tekstura u koju se spremaju RGB vrijednosti međuspremnika. Drugi međuspremnik se koristi isključivo za prepoznavanje trenutne "lebdeće" ćelije, pa je "skriven" od korisnika. Razlika između ovog međuspremnika i međuspremnika vidljivog korisniku je što su ćelije motora u ovom međuspremniku konstantno obojane istim bojama koje ovise isključivo o indeksu ćelije. Stoga se u svakom trenutku ovaj grafički međuspremnik može uzorkovati na pikselu na kojem se nalazi pokazivač miša kako bi se iz dobivene RGB vrijednosti piksela saznao indeks ćelije na kojoj je pokazivač miša pozicioniran. Iscrtavanje ovih podataka u "skriveni" međuspremnik, te uzorko-

vanje istog je zadatak klase *EngineCellSelectionMesh*. Razlike između prikaza ovih međuspremnika su vidljive na slici 5.2.



(a) Grafički međuspremnik koji je vidljiv korisniku



(b) Grafički međuspremnik za detekciju "lebdeće" ćelije

Slika 5.2: Razlike između prikaza grafičkih međuspremnika

Druga klasa koja čini komponentu Pogleda je *GraphManager*, a njena odgovornost je prikaz 2D grafova i legende gradijenta s obzirom na podatke koji su dostupni u

Modelu poput: odabranih postavki, frekvencija i ćelija. Instanca ove klase izravno poziva metode biblioteke ImPlot koje su dodatno izmijenjene kako bi se omogućilo eksplicitno definiranje boja koje se koriste za crtanje grafova i legende gradijenta preko argumenata metoda. Klasa se pretplaćuje na događaje i signale Modela, tj. instanci sastavnih klasa Modela, koji su relevantni za ažuriranje instance pomoćne klase *GraphData* unutar koje se čuvaju podaci važni za iscrtavanje grafova. Na ovaj način se objekt klase *GraphData* ne mora ponovno "puniti" podacima u svakoj iteraciji petlje ažuriranja. No, *GraphManager* izravno dohvaća podatke Modela, tj. ne pretplaćuje se na promjene, koji mijenjaju izgled grafa, ali ne zahtjevaju ponovno postavljanje objekta klase *GraphData*. Konkretne metode iscrtavanja i uspoređivanja grafova, te uzorkovanja legende gradijenta su implementirane pomoću obrasca Strategija, čime se olakšava potencijalno dodavanje novih metoda iste namjene. Za razliku od do sada objašnjenih klasa, klasa *App* ne poziva direktno metodu za iscrtavanje grafova i legendi klase *GraphManager*, već to radi klasa *ImGuiManager* kada definira prikaze korisničkog sučelja alata unutar kojih se te komponente trebaju nalaziti.

5.2.3. Upravljač

Komponenta Upravljač je u arhitekturi MVC uvijek odgovorna za komunikaciju s korisnikom, a kako je većina spomenute komunikacije u ovom alatu omogućena pomoću korisničkog sučelja koje pruža biblioteka Dear ImGui, glavna komponenta Upravljača je klasa *ImGuiManager*. Navedena klasa je odgovorna za sve pozive metoda koje su dio biblioteke Dear ImGui, stoga ova klasa definira prozore korisničkog sučelja i njihove sadržaje, gumbe i funkcije koje se pozivaju na aktivaciju gumba, sučelja za mijenjanje boja i brojeva, izbornike i dijaloge za učitavanje datoteka itd. Osim navedenog, dužnost ove klase je i obavještavanje ostalih komponentama o događajima koji su vezani za korisničko sučelje poput promjene veličine prikaza motora ili definiranje direktorija datoteke za učitavanje podataka o motoru. Instanca ove klase čuva pokazivač na objekt klase *MeshManager* kako bi pristupila teksturi u koju je spremljen grafički međuspremnik koji se prikazuje korisniku (opisan u poglavlju 5.2.2). Spomenutu teksturu zatim može prikazati u prozoru zaduženom za prikaz motora. Uz to instanca klase *ImGuiManager* čuva i pokazivač na objekt klase *GraphManager* čije metode koristi za iscrtavanje grafova i legende gradijenta.

Važno je naglasiti kako, unatoč tome što je veliki dio interakcije s korisnikom u alatu implementiran pomoću biblioteke Dear ImGui, veliki dio stvarne interakcije koju korisnik obavlja sa alatom nije vezan za dijelove korisničko sučelja već za događaje poput klika i kretanja miša, te pomicanja kotačića miša. Ti su događaji važni kako bi se korisniku omogućilo označavanje "lebdeće" ćelije, odabiranje "lebdeće" ćelije i kontrola kamere. Kako je već spomenuto na početku poglavlja 5.2, klasa *App* zapravo obrađuje GLFW događaje koji javljaju ovakav tip korisničkog unosa. Kako su takvi događaji važni za obraditi *ImGuiManager*-u, objekt klase *App* prvo pozove metodu objekta klase *ImGuiManager* koja je zadužena za obrađivanje specifičnog događaja. U spomenutoj metodi se obrađivanje događaja delegira biblioteci Dear ImGui, a zatim se ispita je li pokazivač miša iznad prikaza modela motora, pošto su sve navedene interakcije vezane za prikaz modela motora. *ImGuiManager* vraća odgovor na taj upit preko povratne vrijednosti pozvane funkcije. Ako je pokazivač miša iznad prikaza motora u trenutku pojave događaja, objekt klase *App* će događaj dodatno obraditi pomoću ostalih komponenti MVC arhitekture, tako da je jednim dijelom i klasa *App* dio komponente Upravljač.

5.3. Organizacija podataka na grafičkoj kartici

U poglavlju 5.2.2 je objašnjeno kako četiri klase šalju vlastitim sjenčarima različite podatke na grafičku karticu pošto imaju različite uloge. Cilj ovog poglavlja je detaljnije objašnjavanje toka podataka između procesora i grafičke kartice za svaku od navedenih klasa, a kako bi se omogućilo što razumljivije objašnjavanje tog toka, važno je objasniti nekoliko generalnih razlika između načina slanja i iscrtavanja podataka korištenih u ovom alatu.

U alatu razvijenom u sklopu ovog rada se podaci na grafičku karticu šalju na dva načina: pomoću *vertex buffer object*-a i pomoću uniformnih varijabli. *Vertex buffer object*-i, skraćeno VBO, sadrže polje podataka koji opisuju svaki vrh modela na način koji je relavantan sjenčaru vrhova. U grafičkom protočnom sustavu sjenčar vrhova procesira elemente VBO-a, a rezultate prosljeđuje u daljnji dio protočnog sustava. Za jedan sjenčar se može definirati proizvoljan broj VBO-a samo je važno uvijek povezati strukturu elementa koji je sačuvan u VBO-u sa atributima sjenčara vrhova pomoću funkcija OpenGL-a poput *glVertexAttribPointer*. Uniformne varijable su iste za svaki vrh, te su zajedničke sjenčaru vrhova i fragmenata. Njih klijent ne šalje na grafičku karticu učitavanjem u VBO-e, već izravnim definiranjem lokacije i "pakirane "vrijednosti

uniformne varijable. Klijent saznaje lokaciju uniformne varijable u sjenčaru pozivanjem funkcije *glGetUniformLocation* kojoj kao argumente predaje indeks sjenčara i ime uniformne varijable.

Također, u alatu se koriste i dvije različite metode za iscrtavanje podataka učitanih u VBO: *glDrawArrays* i *glDrawElements*. Metoda *glDrawArrays* svaki element VBO-a šalje jednom u sjenčar vrhova, te pretpostavlja da su vrhovi unutar VBO-a poredani slijedno i prate strukturu koju predstavljaju, npr. tri vrha iz istog trokuta su unutar VBO-a definirani jedan za drugim, te zadnji trokut neće imati broj vrhova koji nije djeljiv sa tri. S druge strane metoda *glDrawElements* zahtjeva dodatno definirani *buffer object* zvani *element buffer object*, skraćeno EBO, unutar kojeg su spremljeni indeksi podataka iz VBO-a koji definiraju poredak iscrtavanja učitanih podataka. Korištenjem ove metode u slučaju kada lica modela imaju velik broj dijeljenih vrhova, značajno se smanjuje broj podataka koji se šalje preko VBO-a, pošto se podaci ponavljajućih vrhova mogu "reciklirati" višestrukim navođenjem indeksa tog vrha unutar EBO-a.

5.3.1. Podaci za bojanje ćelija

Podaci koji su relevantni za bojanje ćelija su organizirani u dva VBO-a i 4 uniformne varijable. Preko VBO-a, tj. podataka koji opisuju pojedini vrh, se šalju pozicije vrhova, indeks ćelije kojoj pripadaju i boja te ćelije. Ti podaci su rastavljeni u dva VBO-a na način da se pozicije i indeks ćelije vrha šalju u sklopu jednog VBO-a, a boja ćelije pomoću drugog. Na taj način se podaci koji su konstantni (pozicija i indeks ćelije) ne moraju slati svaki put kada se mijenjaju boje modela motora, već se šalju samo kad se učitaju iz datoteka.

Uniformne varijable koje se šalju na karticu za potrebe bojanja ćelija su matrice modela, pogleda i projekcije, te indeks trenutne "lebdeće" ćelije. Matrica modela je konstantno postavljena na matricu identiteta, a matrice pogleda i projekcije su tu kako bi se model pravilno iscrtavao i nakon promjene rotacije i pozicije kamere, te promjene dimenzija prozora prikaza motora. Indeks trenutne lebdeće ćelije je poslan kako bi sjenčar znao kojoj ćeliji treba postaviti boju na inverz trenutne boje. Na taj način nije potrebno ažurirati VBO sa bojama ćelija svaki put kada pokazivač miša bude postavljen na novu ćeliju.

Za iscrtavanje ovih podataka se koristi metoda *glDrawElements*, tako da je uz sla-

nje podataka, potrebno poslati i indekse koji definiraju po kojem poretku ti podaci dolaze. Iako se na prvi pogled može primjetiti kako ćelije ne dijele vrhove sa istim indeksom ćelija, važno je napomenuti kako OpenGL u *core* načinu rada podržava samo iscrtavanje trokuta. Stoga ćelije koje su četverokuti treba podijeliti na trokute, a takve ćelije korištenjem metode *glDrawElements* ne definiraju unutar VBO-a više šest točaka već četiri. Kako takve ćelije čine ukupno 90.191% svih ćelija modela motora koji se koristi u ovom radu (ostatak ćelija su trokuti), primjenom ove optimizacije se na grafičku karticu šalje 31.614% manje podataka o vrhovima.

5.3.2. Podaci za detekciju lebdeće ćelije

Podaci koji se koriste za iscrtavanje motora sa ćelijama jedinstvenih boja (opisano u poglavlju 5.2.2) se na grafičku karticu šalju pomoću jednog VBO-a koji za svaki vrh definira njegovu poziciju i indeks ćelije kojoj pripada, te pomoću tri uniformne varijable: matrica modela, pogleda i projekcije. Sjenčar iz indeksa ćelije izračuna boju vrha te ćelije, a klasa *EngineCellSelectionMesh* obavlja inverznu operaciju nakon što uzorkuje boju na pikselu iznad kojeg je postavljen pokazivač miša. Za ovo iscrtavanje se također koristi metoda *glDrawElements*, te je smanjenje broja podataka jednako kao i kod podataka za bojanje ćelija.

5.3.3. Podaci za linije ćelija

Podaci koji se koriste za crtanje linija koje razgraničavaju ćelije se na grafičku karticu šalju pomoću jednog VBO-a koji prenosi samo pozicije vrhova linija i tri uniformne varijable koje su matrice modela, pogleda i projekcije. Ovi podaci se također iscrtavaju pomoću metode *glDrawElements* od koje dobivaju veću uštedu u odnosu na korištenje metode *glDrawArrays* jer se pozicije vrhova mogu dijeliti između ćelija pošto više nisu vezane indeksima i/ili bojama ćelija.

5.3.4. Podaci za koordinatne osi

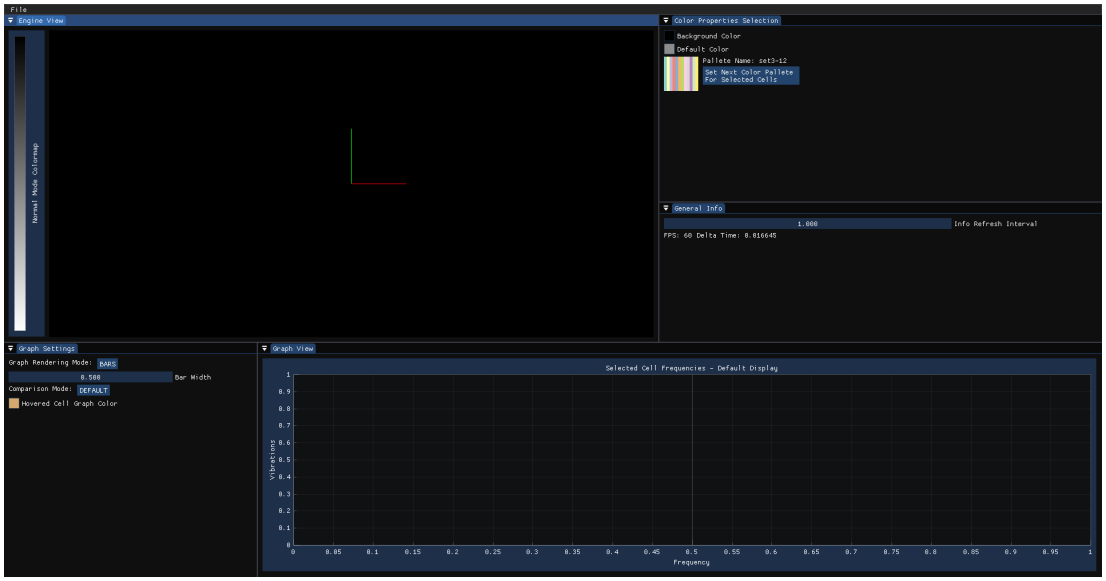
Za crtanje linija koordinatnih osi se koristi jedan VBO unutar kojeg se prenosi polje podataka o vrhovima koji su opisani u dva bajta: prvi bajt je indeks u rasponu od 0 do 3, te služi za identificiranje pozicije vrha koordinatnih osi (točka ishodišta ili vrh x/y/z osi), a drugi bajt je indeks u rasponu od 0 do 2 i definira kojoj osi taj vrh pripada. Sjenčar interno definira konstantna polja koja indeksira pomoću indeksa iz VBO-a. Osim VBO-a sjenčar prima i dvije uniformne varijable: matrice pogleda i projekcije.

Za razliku od dosadašnjih podataka, ovi podaci se crtaju metodom *glDrawArrays*. Ova metoda se koristi zbog jednostavnosti i manje količine potrebnih podataka u odnosu na metodu *glDrawElements*, pošto bi ta metoda zahtjevala dodatno definiranje EBO-a, što za ovako mali broj vrhova nema smisla.

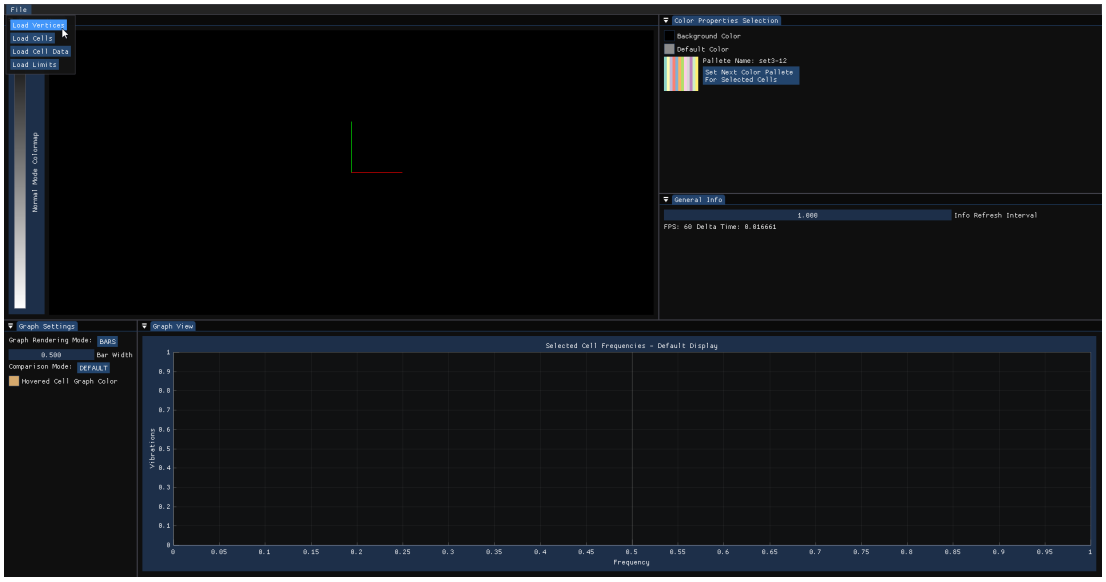
6. Demonstracija rada alata

6.1. Učitavanje datoteka

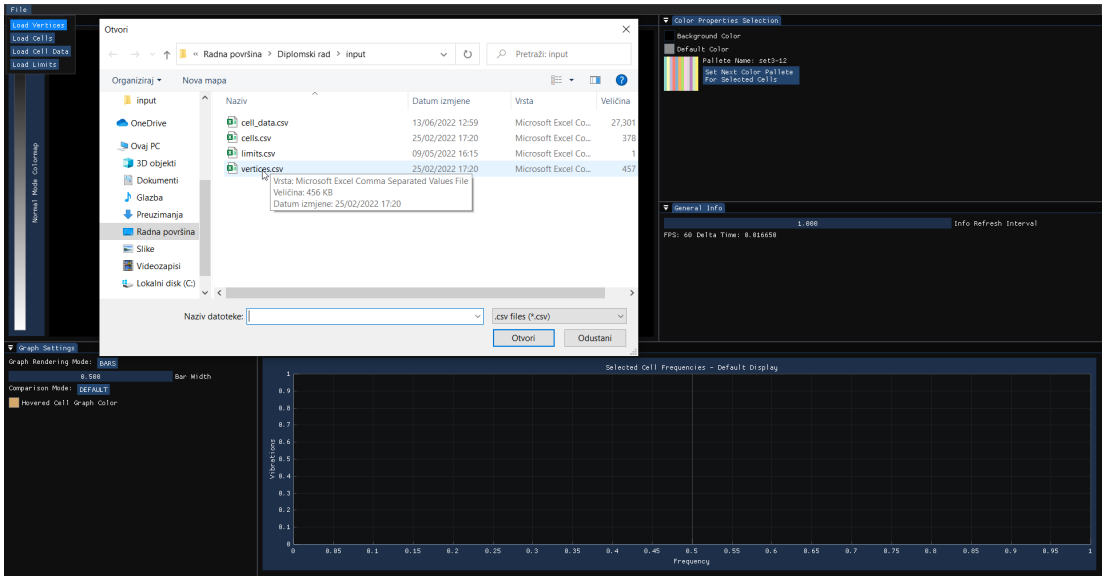
prvo poglavlje prvo inicijalni screen, pa meni za odabrat podatke, pa dijalog za odabrat datoteke, pa kako izgleda kad je sve učitano



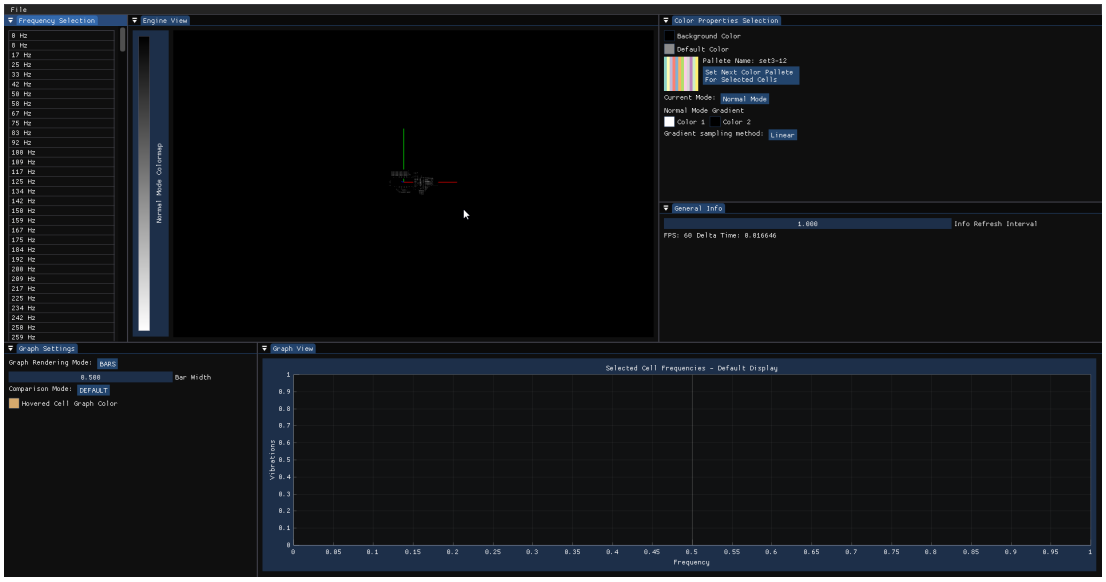
Slika 6.1: Pregled MVC organizacije alata



Slika 6.2: Pregled MVC organizacije alata



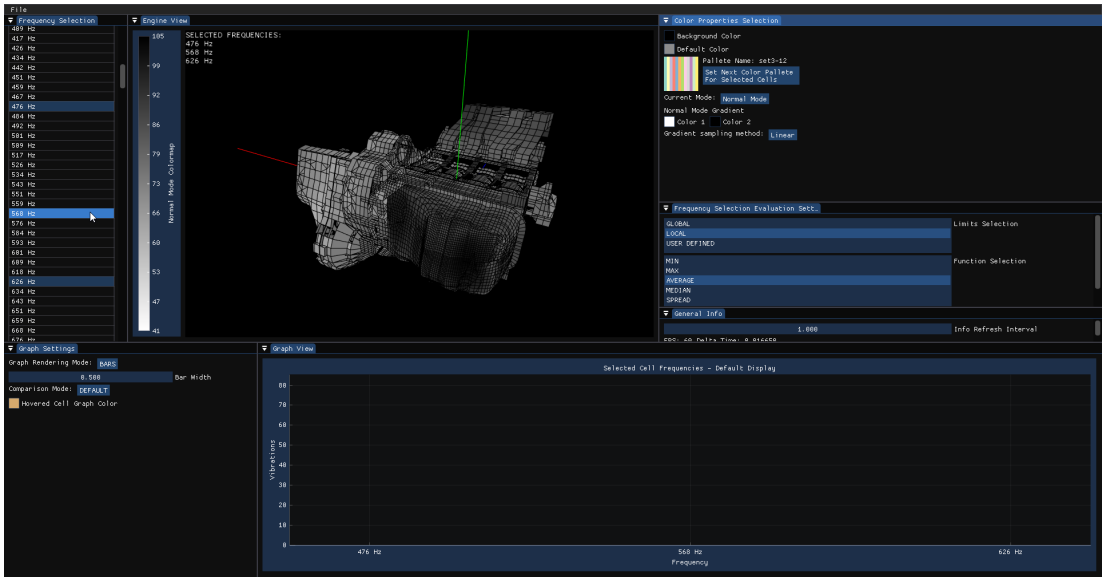
Slika 6.3: Pregled MVC organizacije alata



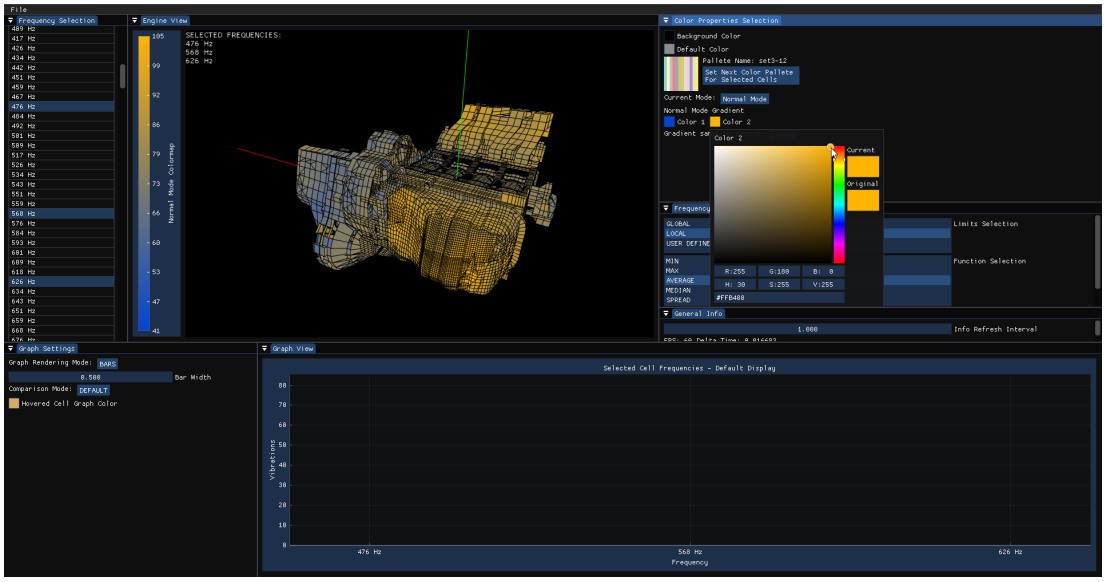
Slika 6.4: Pregled MVC organizacije alata

6.2. Normalni način rada

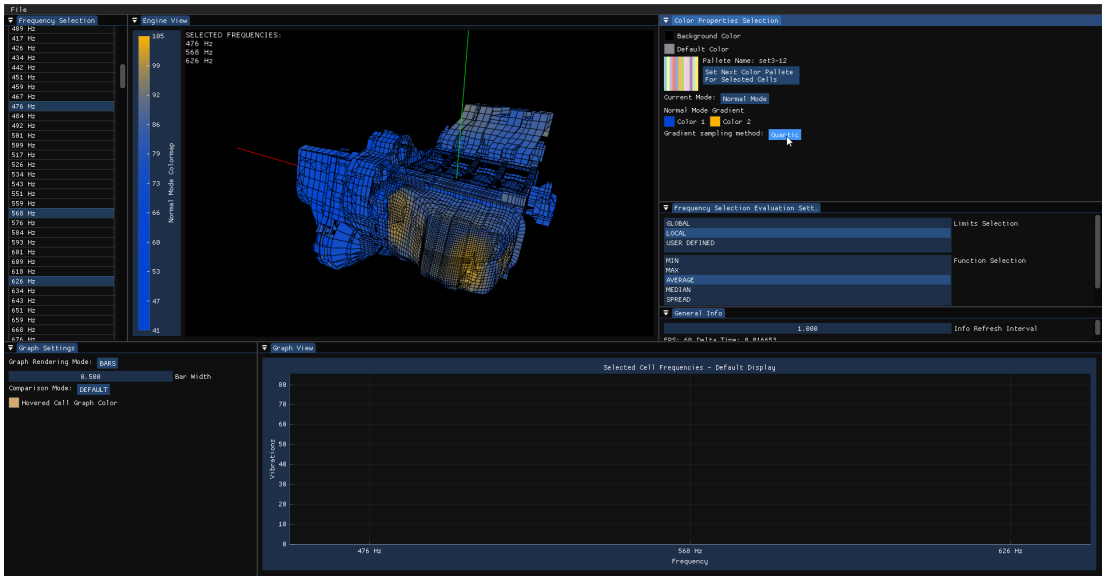
drugo poglavlje kako izgleda kada se odaberu frekvencije, pa promijeni par boja radi vidljivosti, pa promijeni uzorkovanje gradijenta radi vidljivosti (vidimo da je dosta gradijenata opasno otraga ali promjenom uzorkovanja možemo izolirati sami vrh), spomeni da je ovo posebno korisno za max pri lokalno selektiranim rasponima uzet quartic pa se vidi di je max najgori, pa promijeni funkcije i raspone (sličan efekt možemo postići ručnim definiranjem raspona i linearom), pa komentiraj korištenje spread-a



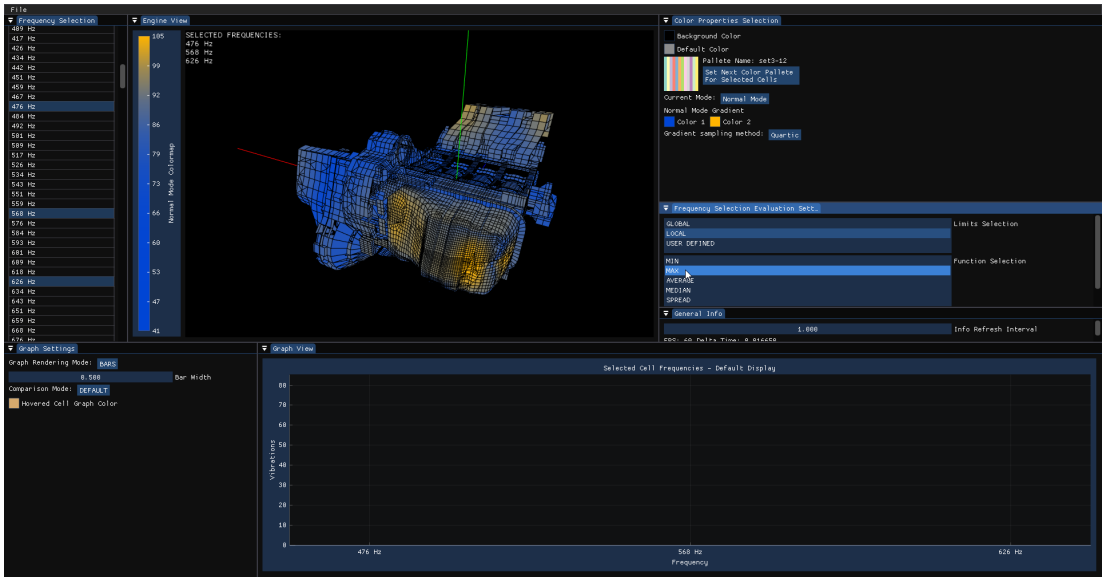
Slika 6.5: Pregled MVC organizacije alata



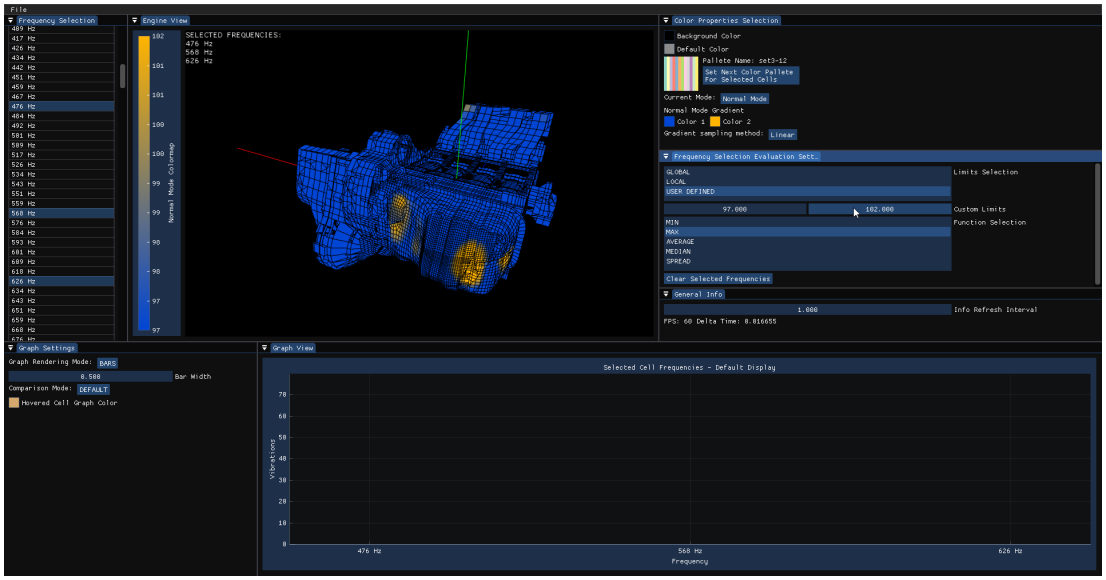
Slika 6.6: Pregled MVC organizacije alata



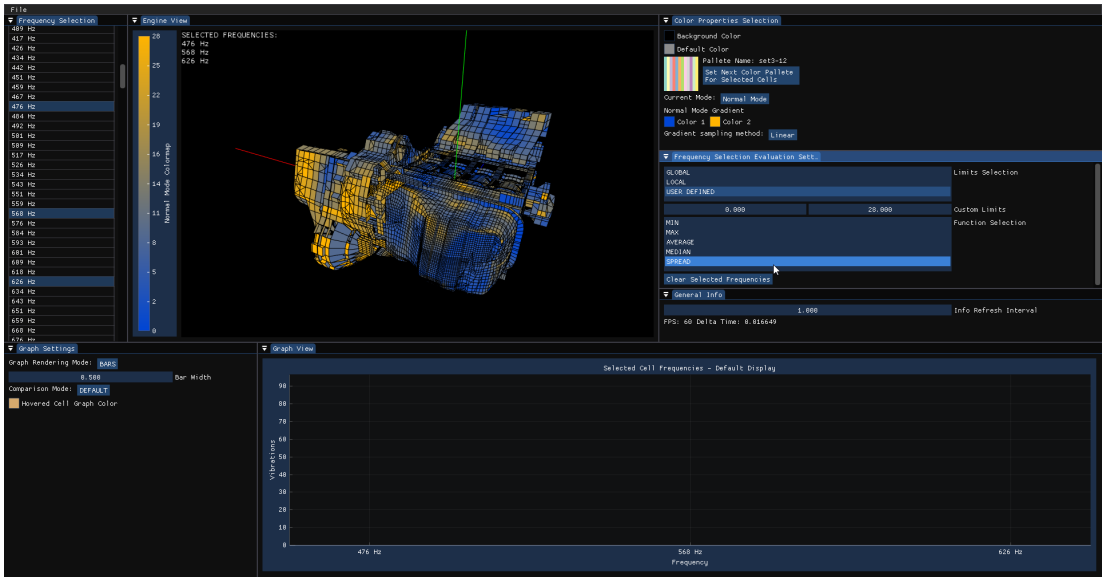
Slika 6.7: Pregled MVC organizacije alata



Slika 6.8: Pregled MVC organizacije alata



Slika 6.9: Pregled MVC organizacije alata

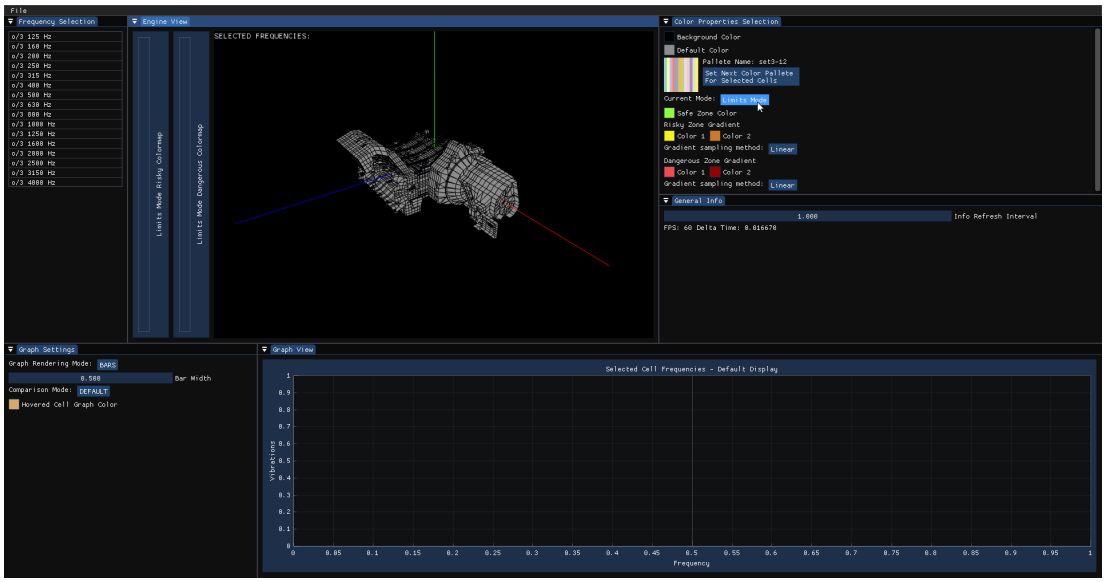


Slika 6.10: Pregled MVC organizacije alata

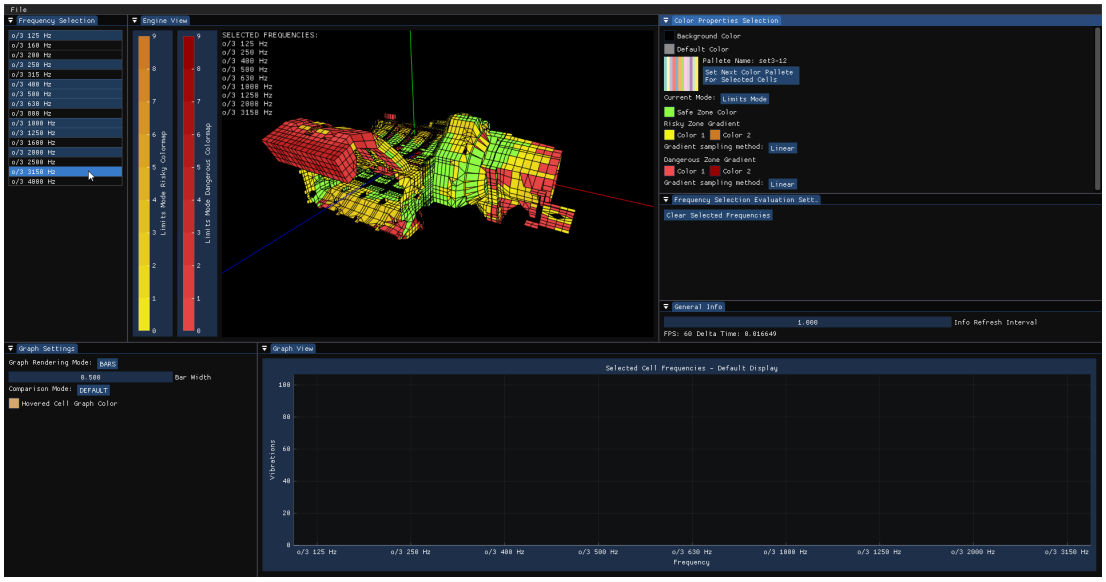
6.3. Način rada sa preddefiniranim ograničenjima

treće poglavlje

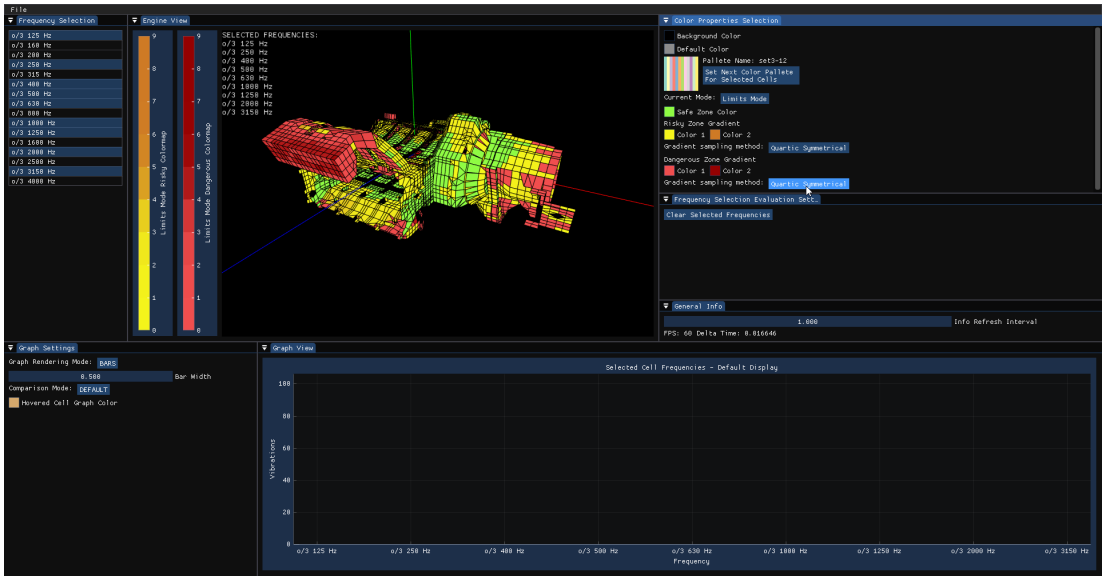
upali limits mode i slikaj kako izgleda na pocetku komentirajuci da su prikazane samo ćelije koje su dokumentirane u ograničenjima, odaberi nekoliko frekvencija, izmijeni uzorkovanje gradijenata da pokazes kako pomazu sa razlikovanjem



Slika 6.11: Pregled MVC organizacije alata



Slika 6.12: Pregled MVC organizacije alata

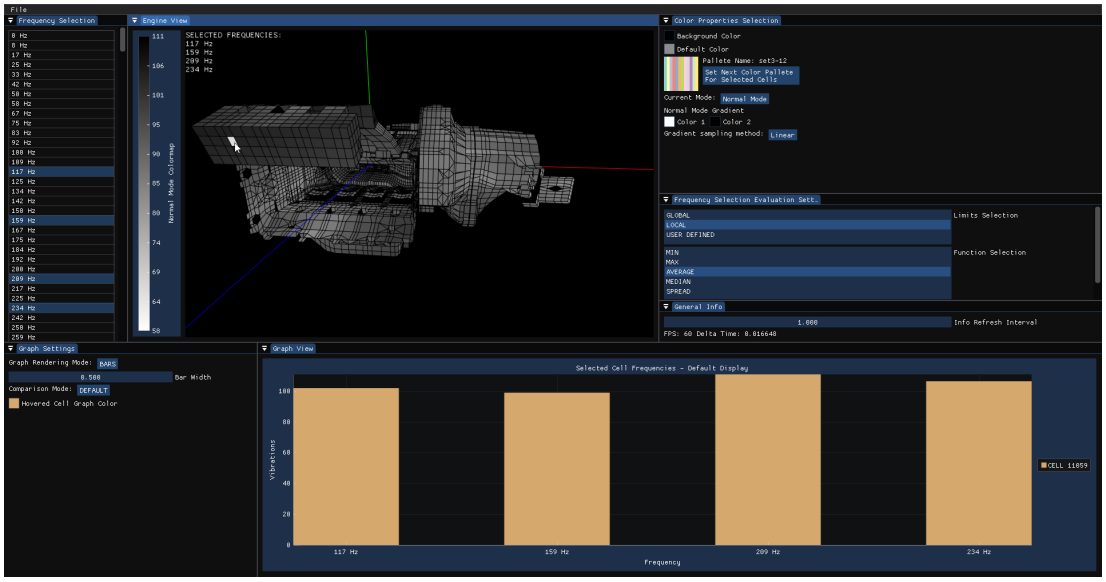


Slika 6.13: Pregled MVC organizacije alata

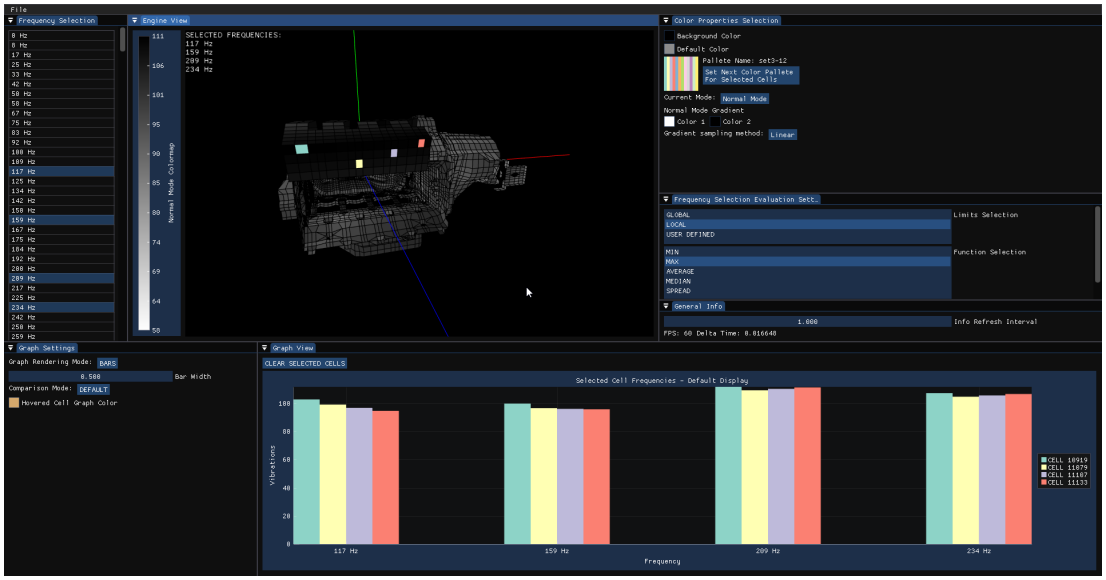
6.4. Uspoređivanje ćelija pomoću grafova

četvrto poglavlje

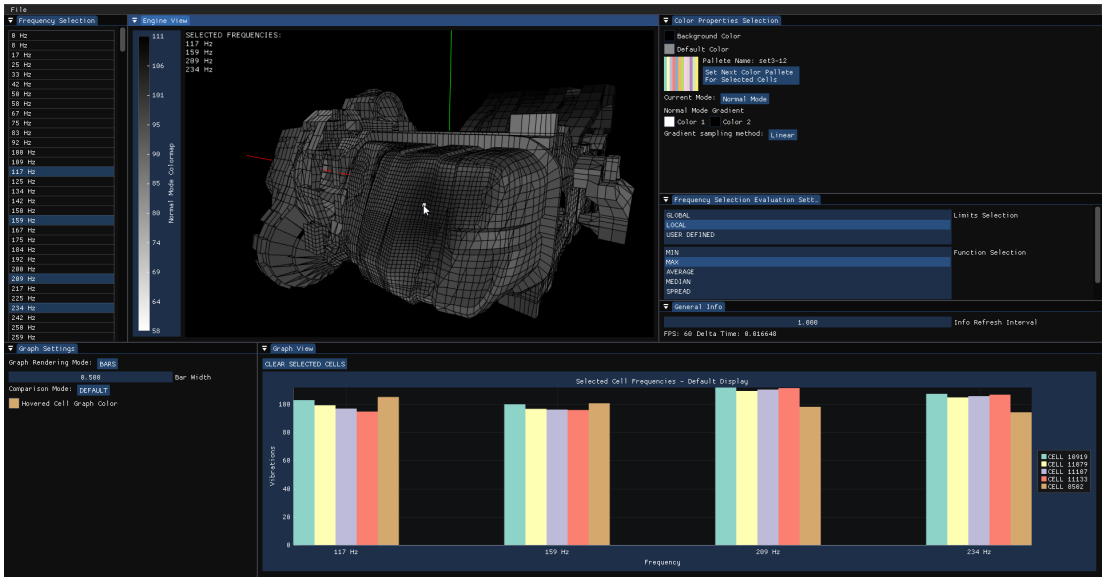
hoveraj ćeliju, odaberi nekoliko ćelija radi detaljnijeg pregleda, uzmi max i reci kako selektirane ćelije imaju sličnu boju ko lebdeća ali se lebdeća drugačije ponaša -> važnost prikaza grafa



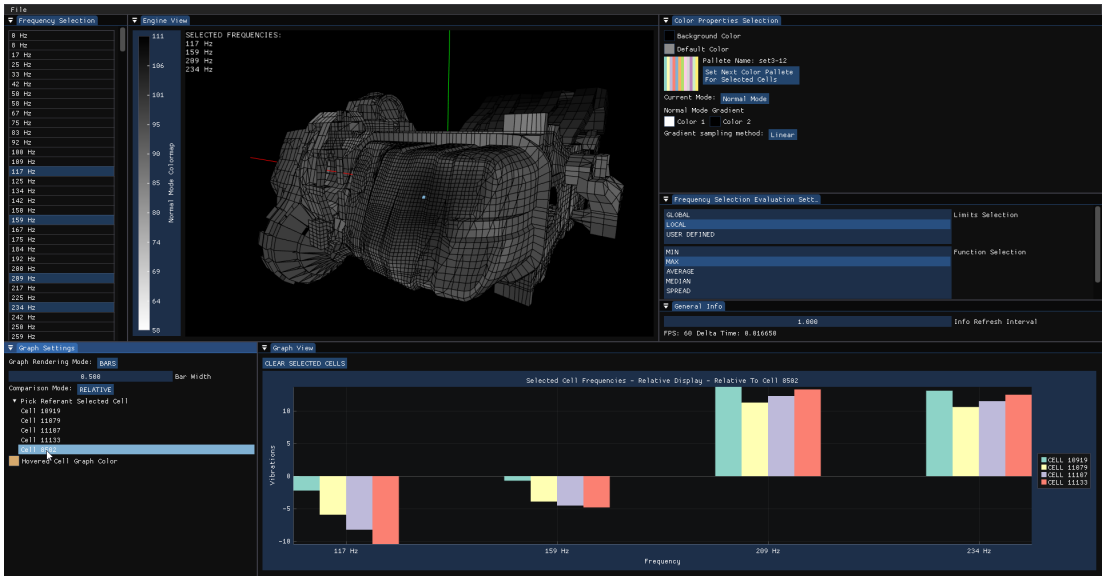
Slika 6.14: Pregled MVC organizacije alata



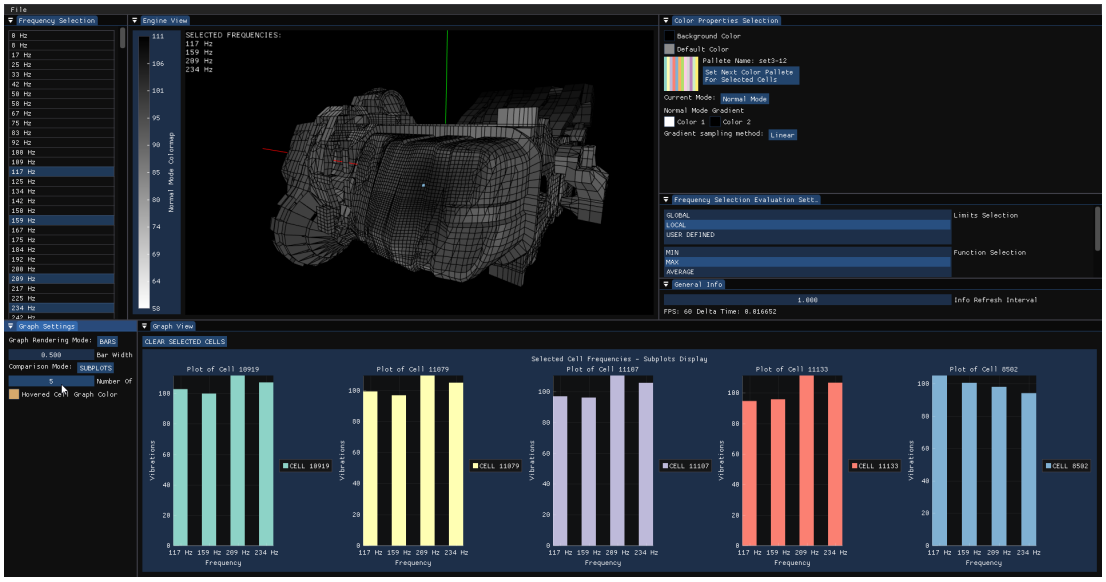
Slika 6.15: Pregled MVC organizacije alata



Slika 6.16: Pregled MVC organizacije alata



Slika 6.17: Pregled MVC organizacije alata



Slika 6.18: Pregled MVC organizacije alata

7. Zaključak

1 stranica

LITERATURA

[1] AVL EXCITE™. https://www.avl.com/excite. [Pristupljeno 3. lipnja 2022].

[2] ColorBrewer: Color Advice for Maps. https://colorbrewer2.org. [Pristupljeno 10. lipnja 2022].

[3] Easing Functions Cheat Sheet. https://easings.net/#. [Pristupljeno 9. lipnja 2022].

[4] Glfw: Main page. https://www.glfw.org/docs/3.3/index.html. [Pristupljeno 15. lipnja 2022].

[5] OpenGL Mathematics (GLM). https://github.com/g-truc/glm. [Pristupljeno 15. lipnja 2022].

[6] Dear ImGui: Bloat-free Graphical User interface for C++ with minimal dependencies. https://github.com/ocornut/imgui. [Pristupljeno 15. lipnja 2022].

[7] implot: Immediate Mode Plotting. https://github.com/epezent/implot. [Pristupljeno 15. lipnja 2022].

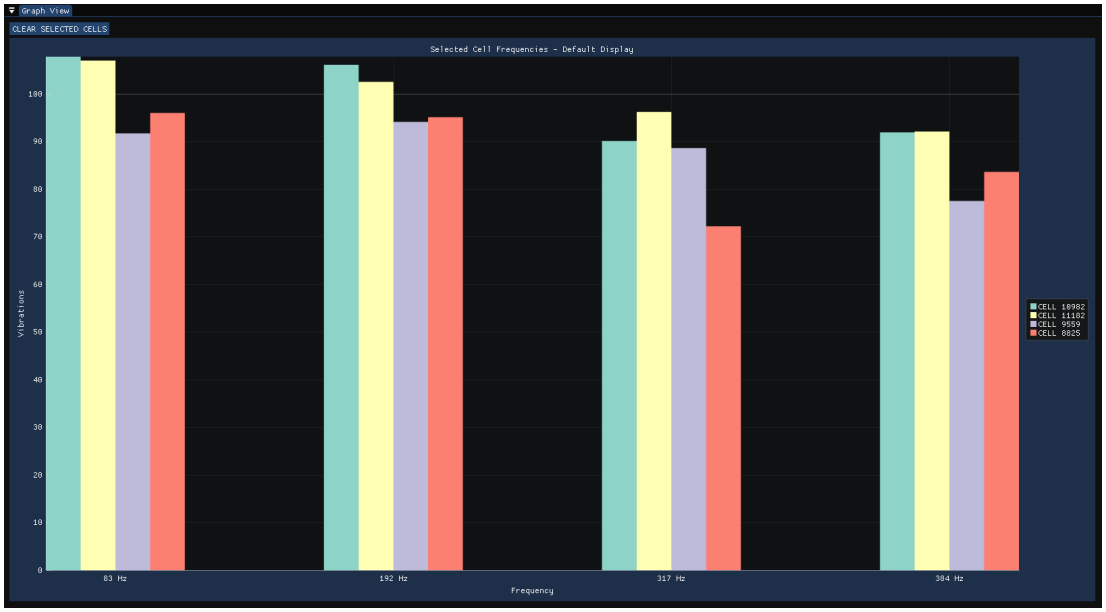
[8] nativefiledialog: A tiny, neat C library that portably invokes native file open and save dialogs. https://github.com/mlabbe/nativefiledialog. [Pristupljeno 15. lipnja 2022].

[9] Opengl - the industry standard for high performance graphics. https://www.opengl.org/. [Pristupljeno 15. lipnja 2022].

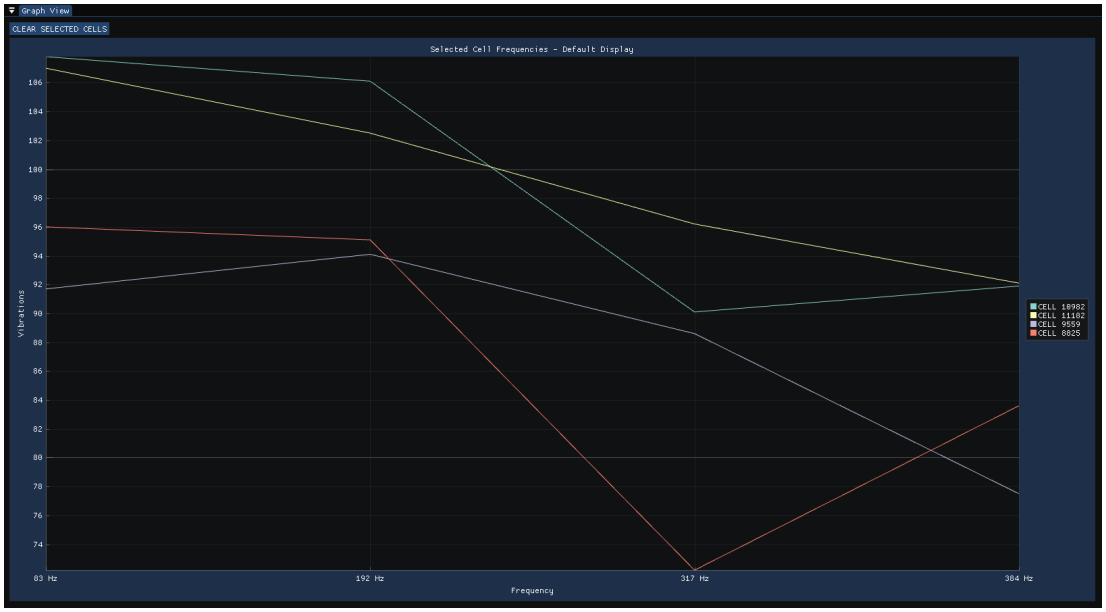
[10] Krešimir Matković, Rainer Splechtna, Denis Gračanin, Goran Todorović, Stanislav Goja, Boris Bedić, i Helwig Hauser. Getting insight into noise, vibration, and harshness simulation data. U *2021 Winter Simulation Conference (WSC)*, stranice 1–12. IEEE, 2021.

[11] Antony Unwin. Why is Data Visualization Important? What is Important in Data Visualization? *Harvard Data Science Review*, 2(1), 31. siječnja 2020. https://hdsr.mitpress.mit.edu/pub/zok97i7p.

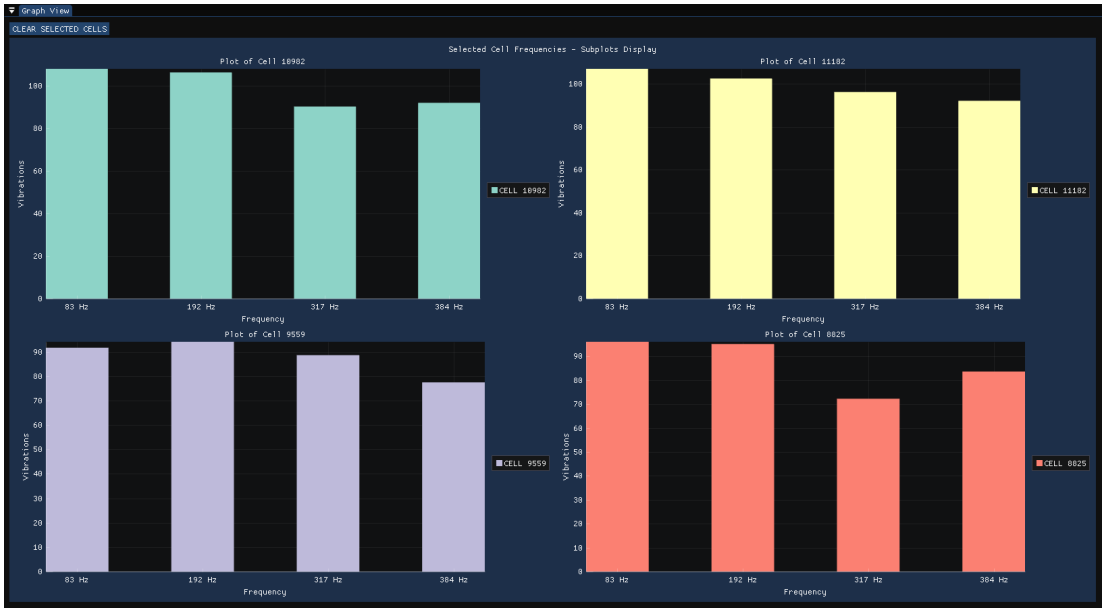
Dodatak A
Različiti načini crtanja grafova



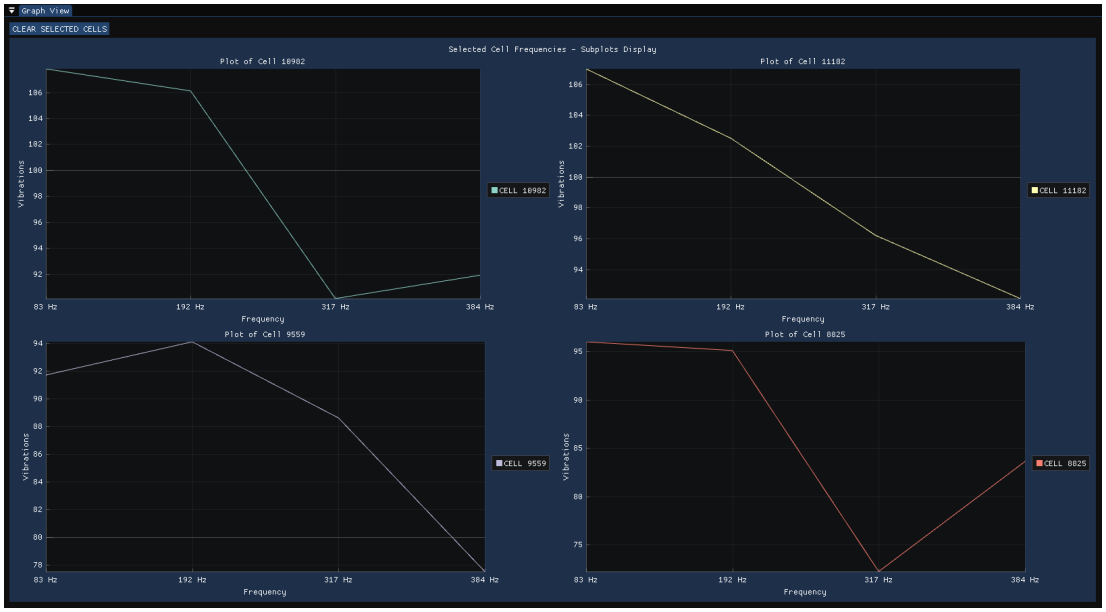
Slika A.1: Normalni način usporedbe grafova u obliku stupaca



Slika A.2: Normalni način usporedbe grafova u obliku linija

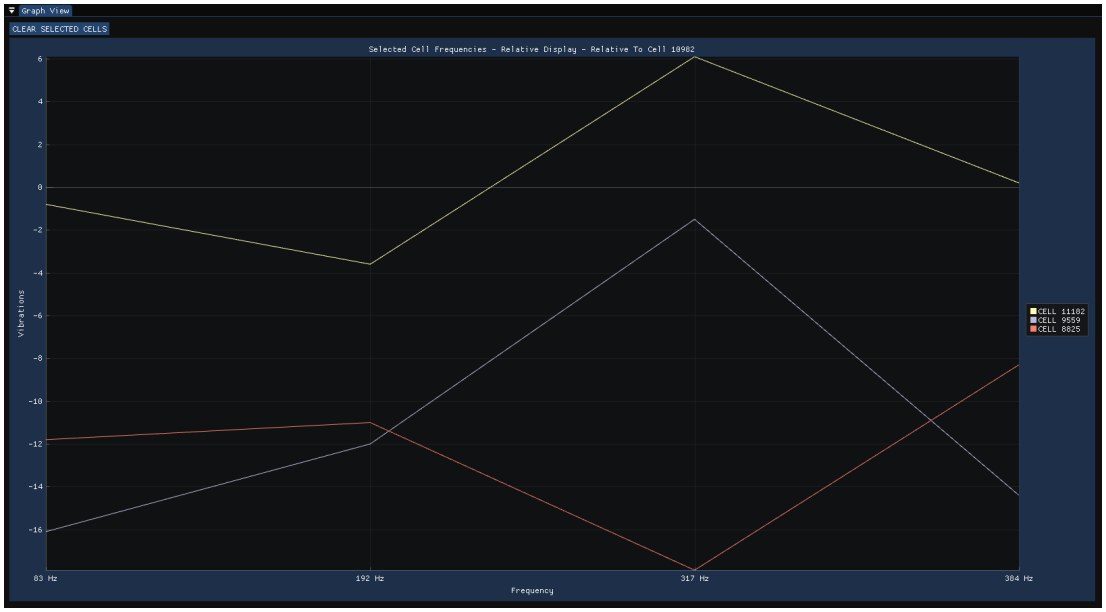


Slika A.3: Način usporedbe grafova sa višestrukim prikazima u obliku stupaca



Slika A.4: Način usporedbe grafova sa višestrukim prikazima u obliku linija

Slika A.5: Relativni način usporedbe grafova u obliku stupaca u kojem su podaci svih ćelija prikazani u odnosu na ćeliju 10982



Slika A.6: Relativni način usporedbe grafova u obliku linija u kojem su podaci svih ćelija prikazani u odnosu na ćeliju 10982

Dodatak B
Sustav događaja i signala

Dodatak C
Klasa *VariableMap*

3D interaktivna vizualizacija simulacije vibracija

Sažetak

Sažetak na hrvatskom jeziku.

Ključne riječi: Ključne riječi, odvojene zarezima.

3D Interactive Visualization of Vibration Simulation

Abstract

Abstract.

Keywords: Keywords.