# Crtanje simplicijalnog kompleksa na ekran

Marko Vojinović

26.04.2022.

## Sadržaj



## Uvod

Kada imamo jedan simplicijalni kompleks dimenzije $D$ instanciran u memoriji, postupak računanja koordinata za crtanje na ekranu vrši se u tri glavna koraka:

1. embedding kompleksa u odgovarajući topološki prostor dimenzije $D$,
2. preslikavanje kompleksa iz topološkog prostora u veliki ambijentalni euklidski prostor dimenzije $D_{\text{amb}}$,
3. projektovanje kompleksa iz ambijentalnog prostora na euklidsku ravan (dimenzije 2) koja predstavlja ravan ekrana.

U nastavku ćemo detaljnije prodiskutovati svaki korak ponaosob, i definisati sve relevantne detalje.

## 1 Embedding kompleksa u topološki prostor

Smisao ovog prvog koraka je da se zadaju inicijalne koordinate za svaki verteks u simpleksu, u skladu sa topologijom kompleksa. Drugim rečima, ako je topologija kompleksa npr. sfera, onda svi verteksi treba da dobiju koordinate koje leže na geometrijskoj sferi, kako bi crtež na kraju ličio na sferu. Ili ako je topologija kompleksa torus, svi verteksi treba da leže na torusu, itd.

Topologija kompleksa se incijalno zadaje seed funkcijom koja generiše kompleks, i pamti se kao string u varijabli `topology` klase `SimpComp`. Verovatno joj nećemo menjati vrednost tokom života datog kompleksa, tj. biće konstantna (osim ako ne odlučimo da uvedemo funkcije koje će da menjaju topologiju, što nije plan za sada).

Za svaki konkretan tip topologije definišemo skup koordinata $q_1, \ldots, q_D$, sa odgovarajućim domenima, koje ćemo da dodelimo svakom verteksu u kompleksu kao jednu boju.

**Zadatak 25.** Definisati klasu

```
TopologicalCoordinatesColor
```

koja će da ima tip

```
TYPE_TOPOLOGICAL_COORDINATES 130
```

kao child-klasu klase `Color`. Ona treba da pamti koordinate $q_1, \ldots, q_D$, najbolje kao vektor tipa `double`, tako da možemo da je konstruišemo za svako zadato $D$. Osim toga, treba da pamti dozvoljen domen svake koordinate, tj. vektor sa vrednostima $q_1^{\min}, \ldots, q_D^{\min}$ i vektor sa vrednostima $q_1^{\max}, \ldots, q_D^{\max}$. Minimalne i maksimalne vrednosti su konstantne i iste za sve vertekse datog kompleksa, dok se same koordinate $q_i$ razlikuju od jednog verteksa do drugog, pri čemu je uvek $q_i^{\min} \leq q_i \leq q_i^{\max}$.

Na primer, recimo da algoritam seed-uje kompleks čija je topologija 2-sfera. Tada će u klasi `SimpComp` da budu setovane vrednosti `topology="sphere"` i `D=2`. Kada to znamo, vertekse tog kompleksa obojimo klasom `TopologicalCoordinatesColor`, i za dati verteks zadamo sledeće koordinate:

$$(q_1, q_2) = (2.48, 4.77)\,, \qquad (q_1^{\min}, q_2^{\min}) = (0.0, 0.0)\,, \qquad (q_1^{\max}, q_2^{\max}) = (3.14, 6.28)\,.$$

Ove koordinate interpretiramo kao koordinate na sferi, geografsku širinu i dužinu, $(\theta, \varphi)$. Domen za $\theta$ je $[0, \pi]$, dok je za $\varphi$ domen $[0, 2\pi)$.

U opštem slučaju, i domen i geometrijski smisao koordinata $q_1, \ldots, q_D$ se zadaju i interpretiraju na osnovu informacije o topologiji i dimenziji datog kompleksa. Za svaku konkretnu topologiju ćemo formulisati funkcije koje će da zadaju vrednosti ovih koordinata u svakoj boji i da manipulišu njima. Ukupan broj različitih topologija neće biti mnogo veliki (imaćemo recimo 5-6 tipova ukupno).

Kada smo obojili sve vertekse gornjom klasom i popunili vrednosti za domene, treba svakom verteksu dodeliti konkretne vrednosti $q_i$. To se radi sledećim algoritmom. Najpre vrednost $q_i$ za svaki verteks izaberemo kao slučajan broj (random generatorom na njenom domenu). Pošto random koordinate neće da izgledaju lepo kad se nacrtaju, ideja je da ih "doteramo" minimizacijom nekog potencijala $V(q_i^{(1)}, \ldots, q_i^{(V)})$, gde je $V$ ukupan broj verteksa u kompleksu. U kontekstu Zadatka 10 sa mailing liste smo diskutovali raznorazne izbore potencijala, i ima smisla da isprobamo više njih (i da ostavimo korisniku da bira koji će da koristi), ali primera radi da citiram recimo Igorov predlog:

$$V(q_i^{(1)}, \ldots, q_i^{(V)}) = \sum_{e=1}^{E} c_1 (L_e(q) - c_2)^2\,, \qquad L_e(q) = \sqrt{\sum_{i=1}^{D} \left( q_i^{(v_1)} - q_i^{(v_2)} \right)^2}\,. \tag{1}$$

Ovaj potencijal ima sledeći smisao — duž svakog edge-a $e$ u kompleksu (kojih ima ukupno $E$) postavimo oprugu dužine $c_2 > 0$ i konstante opruge $c_1 > 0$. Zatim izračunamo dužinu $L_e$ svakog edge-a po gornjoj formuli, u kojoj $v_1$ i $v_2$ označavaju dva verteksa koji su susedi datog edge-a $e$, a $q_i^{(v)}$ su koordinate datog verteksa. Ideja je da te koordinate izaberemo tako da potencijal $V$ bude minimalan, čime ćemo postići da su verteksi pozicionirani tako da svaki edge u kompleksu bude približno iste dužine $c_2$. Naravno, neki će biti duži a neki kraći, ali je ideja da sve opruge budu u međusobnoj "ravnoteži".

Osim što možemo da implementiramo raznorazne potencijale $V(q)$ koji će da nam daju različite konačne "oblike" naseg kompleksa, i formulu za dužinu edge-a $L_e(q)$ treba da implementiramo na nekoliko različitih načina, u zavisnosti od topologije — rastojanje koje računamo Dekartovim koordinatama u ravni ne računa se po istoj formuli kao rastojanje koje računamo sfernim koordinatama tj. uglovima na sferi. Svaka topologija će da ima svoju funkciju $L_e(q)$, a potencijale ćemo da isprobavamo na raznorazne načine.

**Zadatak 26.** Napisati funkcije

```
double evaluate_spring_potential( SimpComp *G )
double evaluate_coordinate_length( KSimplex *edge , SimpComp *G )
```

koje primaju kao input pointer na simplicijalni kompleks i edge za koji računamo dužinu, a kao output daju vrednosti $L_e(q)$ i $V(q)$ prema gornjim formulama. Vrednosti konstanti $c_1$ i $c_2$ ćemo kasnije da štimujemo, za početak stavite npr. vrednosti $c_1 = 2.0$ i $c_2 = 10.0$. Funkciju za $L_e(q)$ ćemo kasnije da dorađujemo tako da pogleda vrednost stringa `topology` u kompleksu, pa na osnovu toga da primeni

odgovarajuću formulu za rastojanje u koordinatama koje odgovaraju toj topologiji. Zasad uradite samo slučaj `topology="linear"` po formuli (1) gore, a formule za ostale topologije ću vam zadavati naknadno.

Minimizacija potencijala se radi tako što računamo gradijent potencijala — izračunamo vrednost potencijala u datim koordinatama $q_i^{(v)}$, a zatim i u "malo pomerenim" koordinatama — svaku koordinatu zaljuljamo npr. za $\pm 0.1$, i izračunamo novu vrednost za $V$. Skupimo sve takve vrednosti, i najmanja među njima pobeđuje — koordinate setujemo na novu vrednost u kojoj je potencijal manji. Postupak ponavljamo dok ne budemo zadovoljni — ili dok ne pronađemo minimum (sve okolne vrednosti $V$ su veće od one u kojoj se nalazimo), ili se prosto zaustavimo nakon 100-200 iteracija.

Pošto proveravanje svih kombinacija svih koordinata pomerenih za $\pm 0.1$ može da bude veoma skup posao ukoliko postoji mnogo verteksa u kompleksu i/ili imaju veliku dimenziju, gradijent može da se traži i Monte-Carlo metodom — slučajnim izborom pomeriti koordinate tačaka za $\pm 0.1$ na neku random stranu, i izračunati potencijal za 10-100-1000 tako slučajno izabranih susednih koordinata, čime se sa određenom verovatnoćom pronađe pravac koji je blizak gradijentu. Broj slučajnih pokušaja može da se oceni proporcionalno sa dimenzijom ukupnog konfiguracionog prostora $V^D$, gde je $V$ broj verteksa u kompleksu a $D$ dimenzija kompleksa.

**Zadatak 27.** Napisati funkciju

```
void evaluate_potential_minimum( SimpComp *G )
```

koja kao input prima pointer na simplicijalni kompleks, a output nema. Pravi rezultat funkcije pamti se u rezultujućim koordinatama $q_i$ u odgovarajućoj boji svakog verteksa. Algoritam treba da pođe od neke početne vrednosti koordinata (koja može da bude zadata random), i da iterativno evaluira potencijal $V(q)$ za koordinate koje su "bliske" početnim koordinatama, izabrane Monte-Carlo metodom. U svakoj iteraciji, ako je najmanja dobijena vrednost potencijala manja od $V(q)$, setovati te koordinate kao nove početne koordinate i nastaviti u sledeću iteraciju. Iteracije se završavaju ukoliko su sve okolne vrednosti potencijala veće od $V(q)$ (lokalni minimum), ili nakon nekog unapred zadatog broja iteracija (npr. 200).

Kada se minimizacija potencijala završi, svaki verteks u simplicijalnom kompleksu ima boju koja sadrži pogodno određene koordinate $q_1, \ldots, q_D$. Ove vrednosti ima smisla menjati samo ako se menja struktura kompleksa (Pachner-ovim potezima ili lepljenjem novih simpleksa na granicu kompleksa). Ako se promeni struktura kompleksa, treba ponovo minimizovati potencijal $V(q)$ po gornjem algoritmu, polaženjem od vrednosti koordinata $q_i$ koje već imamo.

U tom smislu, topološke koordinate $q_i$ ćemo računati relativno retko, pa se njihovo računanje može tretirati kao "one time effort", i brzina izračunavanja nije previše kritična.

# 2 Embedding kompleksa u ambijentalni euklidski prostor

Nakon što smo simplicijalni kompleks određene topologije i dimenzije uronili u mnogostrukost iste takve topologije i dimenzije, određivanjem koordinata $q_i$ za svaki verteks u kompleksu, sada u drugom koraku potrebno je celu tu mnogostrukost zajedno sa kompleksom uroniti u ravan euklidski prostor, tzv. ambijentalni prostor, čija je dimenzija $D_{\text{amb}} \geq D$, i u kome se mnogostrukost vidi kao neka iskrivljena hiperpovrš.

Na primer, ako smo seed-ovali kompleks dimenzije $D = 2$ sa topologijom sfere, u prvom koraku smo odredili koordinate svakog verteksa na mnogostrukosti koja predstavlja 2-sferu. Sada u drugom koraku celu tu 2-sferu predstavljamo kao jednu površ dimenzije $D = 2$ u ravnom euklidskom prostoru dimenzije $D_{\text{amb}} = 3$.

U opštem slučaju, svaka mnogostrukost dimenzije $D$ može se uroniti u ambijentalni ravan prostor dimenzije $D_{\text{amb}} \leq 2D$, pri čemu minimalna dimenzija ambijentalnog prostora zavisi od konkretne topologije mnogostrukosti koju uranjamo. Na primer, mnogostrukost dimenzije $D = 2$ napravljena od jednog pojedinačnog trougla ima `topology="linear"`, i može se uroniti u ambijentalni prostor dimenzije $D_{\text{amb}} = 2$, tj. u ravan. Ali zato sfera dimenzije $D = 2$ može da se uroni samo u ambijentalni prostor dimenzije $D_{\text{amb}} = 3$, dok se npr. Klein-ova boca dimenzije $D = 2$ može uroniti samo u ambijentalni prostor dimenzije $D_{\text{amb}} = 4$ (u slučaju vernog uranjanja, bez samopresecanja).

Drugim rečima, dimenzija ambijentalnog prostora $D_{\text{amb}}$ zavisi od dimenzije i topologije simplicijalnog kompleksa, i uranjanje mora da se zada zasebno za svaku konkretnu topologiju. Mi ćemo sada definisati uranjanja za topologije `"linear"` i `"sphere"`, a ostale ćemo razmatrati kasnije.

Mnogostrukost dimenzije $D$ sa linearnom topologijom možemo da uronimo u prostor iste dimenzije, $D_{\text{amb}} = D$. Ako Dekartove koordinate ambijentalnog prostora obeležimo kao:

$$(x_1, x_2, \ldots, x_D)\,, \qquad x_i \in \mathbb{R}\,,$$

onda se uranjanje vrši tako što se one prosto izjednače sa topološkim koordinatama $q_i$ iz prethodnog odeljka, za svaki verteks pojedinačno:

$$x_1 = q_1\,, \qquad x_2 = q_2\,, \qquad \ldots\,, \qquad x_D = q_D\,. \tag{2}$$

**Zadatak 28.** Definisati klasu

```
EmbeddingCoordinatesColor
```

koja će da ima tip

```
TYPE_EMBEDDING_COORDINATES 131
```

kao child-klasu klase `Color`. Ona treba da pamti koordinate $x_1, \ldots, x_{D_{\text{amb}}}$, najbolje kao vektor tipa `double`, tako da možemo da je konstruišemo za svako zadato $D_{\text{amb}}$. Domen svake koordinate je po definiciji ceo skup realnih brojeva, $\mathbb{R}$, i nema potrebe da se pamti. Ovu boju takođe dodeljujemo svakom verteksu datog simplicijalnog kompleksa.

U slučaju kompleksa dimenzije $D$ sa topologijom sfere, prirodne topološke koordinate $q_1, \ldots, q_D$ su uglovi sfernih koordinata u $D_{\text{amb}} = D + 1$ dimenzija, koje standardno obeležavamo kao $\theta_1, \ldots, \theta_{D-1}, \varphi$:

$$(q_1, q_2, \ldots, q_{D-1}, q_D) \equiv (\theta_1, \theta_2, \ldots, \theta_{D-1}, \varphi)\,.$$

Tada je veza između ambijentalnih i topoloških koordinata zadata kao:

$$\begin{aligned}
x_1 &= R\cos\theta_1\,, \\
x_2 &= R\sin\theta_1\cos\theta_2\,, \\
x_3 &= R\sin\theta_1\sin\theta_2\cos\theta_3\,, \\
&\vdots \\
x_D &= R\sin\theta_1\ldots\sin\theta_{D-1}\cos\varphi\,, \\
x_{D+1} &= R\sin\theta_1\ldots\sin\theta_{D-1}\sin\varphi\,,
\end{aligned} \qquad \theta_1, \ldots, \theta_{D-1} \in [0, \pi]\,, \qquad \varphi \in [0, 2\pi)\,. \tag{3}$$

Ovde se $R$ tretira kao konstantan parametar (radijus $D$-sfere uronjene u $(D+1)$-dimenzionalan prostor), koji možemo da setujemo na $R = 1$ ili $R = 10$ ili bilo koju drugu pozitivnu vrednost, u zavisnosti od veličine slike koju hoćemo da nacrtamo.

Druga notacija za sferne koordinate, koja je takođe ponekad zgodna za korišćenje u praksi, definiše čak $D+2$ ugla $\xi_0, \xi_1, \ldots, \xi_{D+1}$, pri čemu su prvi i poslednji ugao fiksirani, dok su ostali jednaki uglovima $\theta_1, \ldots, \theta_{D-1}, \varphi$:

$$\xi_0 = \frac{\pi}{2}\,, \qquad \xi_1 = \theta_1\,, \qquad \ldots\,, \qquad \xi_{D-1} = \theta_{D-1}\,, \qquad \xi_D = \varphi\,, \qquad \xi_{D+1} = 0\,.$$

Tada se veza (3) može zapisati na kompaktan i elegantan način:

$$x_i = R\cos\xi_i \prod_{j=0}^{i-1} \sin\xi_j\,, \qquad i = 1, \ldots, D+1\,. \tag{4}$$

**Zadatak 29.** Napisati funkciju

```
void evaluate_embedding_coordinates( SimpComp *G )
```

kojoj je input pointer na simplicijalni kompleks, a output nema. Rezultat funkcije treba da se smesti u boju `EmbeddingCoordinatesColor` za svaki verteks iz kompleksa. Ukoliko je za dati kompleks varijabla

`topology="linear"`, embedding koordinate $x_i$ svakog verteksa izračunati na osnovu topoloških koordinata $q_i$ prema formuli (2). Ukoliko je za dati kompleks `topology="sphere"`, koordinate $x_i$ izračunati iz koordinata $q_i$ prema formuli (3) ili ekvivalentno (4). Setovati na primer $R = 10$ za početak, pa ćemo videti kasnije da tu vrednost štelujemo ako bude trebalo.

Slično kao i topološke koordinate $q_i$, embedding koordinate $x_i$ se računaju ponovo samo kada se promeni struktura simplicijalnog kompleksa (npr. Pachner-ovim potezom), što se događa relativno retko, a inače su konstantne. U tom smislu, brzina njihovog izračunavanja nije kritična, i gornja funkcija koja ih računa može takođe da se tretira kao "one time effort".

Na ovom mestu je zgodno da zadamo i funkciju $L_e(q)$ kojom se računa rastojanje između dva verteksa duž datog edge-a, koju treba koristiti umesto izraza za $L_e(q)$ iz (1) kada je `topology="sphere"`. Neka dati edge ima dva susedna verteksa (1) i (2), zadate koordinatama $\xi_i^{(1)}$ i $\xi_i^{(2)}$ na simplicijalnom kompleksu dimenzije $D$. Tada je rastojanje $L_e(\xi)$ zadato kao dužina luka po $D$-sferi koji spaja dva verteksa, i računa se kao:

$$L_e(\xi) = R\delta(\xi)\,, \tag{5}$$

gde je $\delta(\xi)$ ugao koji zaklapaju pravci dva verteksa sa "centrom $D$-sfere" (koji nije nijedna tačka na sferi nego tačka u embedding prostoru). Taj ugao se računa po sledećoj formuli:

$$\delta(\xi) = \arccos\left(\sum_{i=1}^{D+1}\cos\xi_i^{(1)}\cos\xi_i^{(2)}\prod_{j=0}^{i-1}\sin\xi_j^{(1)}\sin\xi_j^{(2)}\right)\,. \tag{6}$$

Obratiti pažnju da i za jedan i za drugi verteks važi

$$\xi_0 = \frac{\pi}{2}\,, \qquad \xi_1 = q_1\,, \qquad \ldots\,, \qquad \xi_D = q_D\,, \qquad \xi_{D+1} = 0\,,$$

odnosno da je ugao $\gamma(\xi)$ u potpunosti definisan kao funkcija topoloških koordinata $q_i$ (embedding koordinate $x_i$ ne figurišu nigde u izrazu, izuzev parametra $R$ koji definiše skalu).

**Zadatak 30.** U zadatku iz prethodne sekcije smo napisali funkciju

```
double evaluate_coordinate_length( KSimplex *edge , SimpComp *G )
```

koja računa dužinu datog edge-a $L_e(q)$ u simplicijalnom kompleksu po formuli (1) kada je za dati kompleks `topology="linear"`. Proširiti tu funkciju i na slučaj kada je `topology="sphere"`, tako da dužinu edge-a $L_e(q)$ računa po formulama (5) i (6). Setovati i ovde $R = 10$ isto kao i u prethodnom zadatku.

# 3 Projektovanje kompleksa na ravan ekrana

Kada smo jednom izračunali sve što je trebalo u prethodna dva odeljka, naš simplicijalni kompleks sadrži vertekse koji su obojeni bojom `EmbeddingCoordinatesColor`, i za svaki verteks su određene njegove Dekartove koordinate $x_i$ u $D_{\text{amb}}$-dimenzionalnom euklidskom ambijentalnom prostoru. Sada je naš zadatak da u tom prostoru na određen način zadamo dvodimenzionalnu ravan ekrana, i zatim pravilo kojim ćemo projektovati položaje svakog verteksa na tu ravan. Ta projekcija će biti zadata sa dve koordinate $(X, Y)$ datog verteksa u ravni, i te koordinate koristimo da nacrtamo verteks na ekran.

Kada odredimo sve koordinate svih verteksa u našoj ravni, možemo da ih nacrtamo kao (debele) tačke na ekranu. Zatim prođemo kroz spisak svih edge-eva u simplicijalnom kompleksu, i za svaki edge nacrtamo (tanku) liniju između njegova dva susedna verteksa. Na taj način formiramo tzv. "wireframe" sliku celog simplicijalnog kompleksa na ekranu.

## 3.1 Definisanje ravni ekrana u prostoru velike dimenzije

Da bismo mogli da izračunamo koordinate $(X, Y)$ datog verteksa na ekranu, prvo moramo da preciziramo kako se zadaje ravan ekrana. Neka je koordinatni sistem na ekranu zadat tako da se koordinatni početak nalazi u centru polja za crtanje, tako da leva i desna ivica ekrana budu određene tačkama $(-X_{\max}, Y)$ i $(X_{\max}, Y)$, dok su donja i gornja ivica ekrana određene tačkama $(X, -Y_{\max})$ i $(X, Y_{\max})$,

respektivno. Vrednosti konstanti $X_{\max}$ i $Y_{\max}$ treba da budu određene rezolucijom (brojem piksela) datog polja za crtanje. Ako je polje za crtanje zadato sa rezolucijom $m \times n$ piksela (brojevi $m$ i $n$ su definisani oblikom prozora u kome se crtanje vrši, i menjaju se dinamički resize-ovanjem prozora pomoću miša), onda je

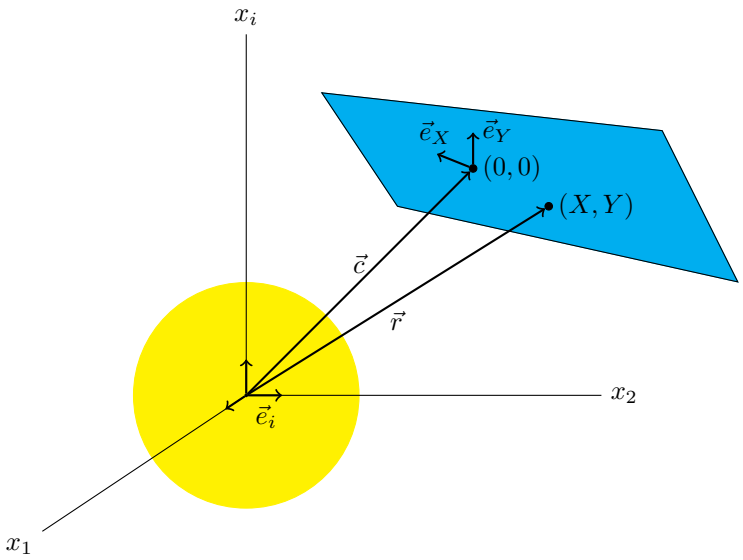
$$X_{\max} = \left\lfloor \frac{m}{2} \right\rfloor , \qquad Y_{\max} = \left\lfloor \frac{n}{2} \right\rfloor .$$

U takvoj geometriji, pretpostavljamo da će wireframe figura našeg simplicijalnog kompleksa da bude okvirno "centrirana" oko koordinatnog početka ambijentalnog prostora, $x_i = 0$. Tada želimo da ravan ekrana bude orijentisana uvek tako da "gleda" ka koordinatnom početku, odnosno da bude ortogonalna na vektor koji spaja koordinatni početak prostora, $x_i = 0$, sa koordinatnim početkom (centrom) ekrana, $X = Y = 0$. Drugim rečima, vektor položaja centra ekrana, $\vec{c}$, predstavlja prvi skup parametara za definiciju ravni ekrana. Kada njega zadamo, treba još da zadamo vektore $\vec{e}_X$ i $\vec{e}_Y$ koji definišu $X$-osu ekrana i $Y$-osu ekrana, tako da ta dva vektora budu ortogonalni na vektor položaja centra, i istovremeno ortogonalni jedan na drugog:

$$\vec{c} \cdot \vec{e}_X = 0 , \qquad \vec{c} \cdot \vec{e}_Y = 0 , \qquad \vec{e}_X \cdot \vec{e}_Y = 0 .$$

Drugim rečima, uređena trojka vektora $(\vec{c}, \vec{e}_X, \vec{e}_Y)$ treba da sačinjava jedan ortogonalni triedar, tako da $\vec{c}$ bude ortogonalan na ravan, dok $\vec{e}_X$ i $\vec{e}_Y$ leže i ravni. Tada proizvoljna tačka na ekranu, sa koordinatama $(X, Y)$ ima u ambijentalnom prostoru vektor položaja

$$\vec{r} = \vec{c} + X\vec{e}_X + Y\vec{e}_Y . \tag{7}$$




Na dijagramu su ilustrovani svi ovi vektori, ravan ekrana (plava površ), i oblast ambijentalnog prostora u kojoj će se nalaziti embeddovan simplicijalni kompleks (žuta sfera).

Kada bi ambijentalni prostor bio trodimenzionalni euklidski prostor, izbor vektora $\vec{c}$ bi jednoznačno fiksirao položaj cele ravni u prostoru, dok bi vektori $\vec{e}_X$ i $\vec{e}_Y$ samo definisali koordinatni sistem u ravni. Međutim, za razliku od trodimenzionalnog slučaja, u $D_{\text{amb}}$-dimenzionalnom prostoru postoji u principu više od jedne ravni, tako da su one sve istovremeno ortogonalne na $\vec{c}$. U tom smislu, zadavanje vektora $\vec{c}$ nije dovoljno da izaberemo ravan ekrana, već moramo da parametrizujemo i izbor vektora $\vec{e}_X$ i $\vec{e}_Y$ kako bismo jednoznačno izabrali ravan u prostoru sa mnogo dimenzija. Ova parametrizacija se najlakše zadaje nizom uglova koji odgovaraju sfernim koordinatama, i definišu orijentaciju i parametre svakog od tri vektora.

Neka je zadat vektor $\vec{c}$ nizom svojih komponenti u Dekartovim koordinatama ambijentalnog prostora,

$$\vec{c} = \sum_{i=1}^{D_{\text{amb}}} c_i \, \vec{e}_i ,$$

gde su $\vec{e}_i$ ortornormirani bazisni vektori Dekartovog koordinatnog sistema. Komponente vektora $\vec{c}$ zadajemo na sledeći način:

$$c_i = d \cos\alpha_i \prod_{j=0}^{i-1} \sin\alpha_j \,, \qquad i = 1, \ldots, D_{\text{amb}} \,. \tag{8}$$

Ovde prepoznajemo sledeći skup parametara kojima zadajemo $\vec{c}$:

- parametar $d \in \mathbb{R}^+$, koji određuje rastojanje centra ekrana od koordinatnog početka ambijentalnog prostora,
- parametre $\alpha_1, \ldots, \alpha_{(D_{\text{amb}}-2)} \in [0, \pi]$ i $\alpha_{(D_{\text{amb}}-1)} \in [0, 2\pi)$, koji određuju orijentaciju vektora $\vec{c}$ u ambijentalnom prostoru,
- konstante $\alpha_0 = \frac{\pi}{2}$ i $\alpha_{D_{\text{amb}}} = 0$.

Kada ambijentalni prostor ima $D_{\text{amb}}$ dimenzija, postoji ukupno $D_{\text{amb}} - 1$ vektora koji su ortogonalni na $\vec{c}$, i obeležavaćemo ih sa $\vec{u}_a$, gde indeks $a$ prebrojava te vektore i uzima vrednosti $1, \ldots, D_{\text{amb}} - 1$. Svaki od tih vektora se može prikazati u Dekartovom bazisu $\vec{e}_i$ kao suma po komponentama,

$$\vec{u}_a = \sum_{i=1}^{D_{\text{amb}}} u_{ai}\, \vec{e}_i \,,$$

i njihove komponente glase ovako (neću da objašnjavam ovde zašto izgledaju baš tako, samo vam dajem gotove formule):

$$u_{ai} = \begin{cases} 0 & \text{kada je} \quad i < a \,, \\ -\dfrac{1}{N_a} \displaystyle\prod_{b=1}^{a} \sin\alpha_b & \text{kada je} \quad i = a \,, \\ \dfrac{1}{N_a} \cos\alpha_a \cos\alpha_i \displaystyle\prod_{\substack{b=0 \\ b\neq a}}^{i-1} \sin\alpha_b & \text{kada je} \quad i > a \,. \end{cases} \tag{9}$$

U ovim izrazima $N_a$ je norma vektora $\vec{u}_a$, i data je ovako:

$$N_a \equiv \sqrt{\vec{u}_a \cdot \vec{u}_a} = \sqrt{\prod_{c=1}^{a} \sin^2\alpha_c + \cos^2\alpha_a \sum_{i=a+1}^{D_{\text{amb}}} \cos^2\alpha_i \prod_{\substack{c=0 \\ c\neq a}}^{i-1} \sin^2\alpha_c} \,.$$

Kada su vektori $\vec{u}_a$ ovako zadati, ukupan skup $(\vec{u}_1, \ldots, \vec{u}_{(D_{\text{amb}}-1)})$ čini jedan ortonormirani skup vektora ortogonalnih na $\vec{c}$,

$$\vec{c} \cdot \vec{u}_a = 0 \,, \qquad \vec{u}_a \cdot \vec{u}_a = 1 \,, \qquad \vec{u}_a \cdot \vec{u}_b = 0 \qquad (b \neq a) \,,$$

i zajedno sa njim čine bazis u ambijentalnom prostoru. U celoj konstrukciji je značajno to što su svi ovi vektori u potpunosti definisani parametrima $d$ i $\alpha_1, \ldots, \alpha_{(D_{\text{amb}}-1)}$ kojima smo definisali vektor $\vec{c}$.

Razlog za uvođenje ortornormiranih vektora $\vec{u}_a$ sastoji se u tome što sada možemo da ih iskoristimo za parametrizaciju vektora $\vec{e}_X$, a zatim i $\vec{e}_Y$ koji definišu našu ravan. Vektor $\vec{e}_X$ mora da bude ortogonalan na $\vec{c}$, pa se samim tim može zapisati kao linearna kombinacija vektora $\vec{u}_a$:

$$\vec{e}_X = \sum_{a=1}^{D_{\text{amb}}-1} e_{Xa} \vec{u}_a \,.$$

Pošto vektori $\vec{u}_a$ čine bazis za potprostor dimenzije $D_{\text{amb}} - 1$ velikog ambijentalnog prostora, vektor $\vec{e}_X$ možemo izraziti u odgovarajućim sfernim koordinatama u tom potprostoru, na sledeći način:

$$e_{Xa} = s_X \cos\beta_a \prod_{b=0}^{a-1} \sin\beta_b \,, \qquad a = 1, \ldots, D_{\text{amb}} - 1 \,. \tag{10}$$

Ovde prepoznajemo sledeći skup parametara kojima zadajemo $\vec{e}_X$:

- parametar $s_X \in \mathbb{R}^+$, koji određuje "skalu" duž $X$-ose na ekranu, tj. intenzitet vektora $\vec{e}_X$,
- parametre $\beta_1, \ldots, \beta_{(D_{\text{amb}}-3)} \in [0, \pi]$ i $\beta_{(D_{\text{amb}}-2)} \in [0, 2\pi)$, koji određuju orijentaciju vektora $\vec{e}_X$ u ambijentalnom prostoru,
- konstante $\beta_0 = \frac{\pi}{2}$ i $\beta_{D_{\text{amb}}-1} = 0$.

Primetite da uglova $\beta_1, \ldots, \beta_{(D_{\text{amb}}-2)}$ ima tačno jedan manje nego uglova $\alpha$ koji definišu vektor $\vec{c}$. Konstrukcija je napravljena tako da izborom $\beta$-uglova možemo da izaberemo bilo koju orijentaciju vektora $\vec{e}_X$ u ambijentalnom prostoru, ali tako da mu ne menjamo intenzitet $s_X$ i da uvek ostane ortogonalan na vektor $\vec{c}$, ma kako ih orijentisali.

Konačno, kada smo fiksirali $\vec{c}$ i $\vec{e}_X$ parametrima $d, s_X, \alpha_i, \beta_a$, sada treba celu konstrukciju ponoviti još jednom u preostalom potprostoru dimenzije $D_{\text{amb}} - 2$, čime ćemo parametrizovati vektor $\vec{e}_Y$. U tom prostoru sada uvedemo bazisne vektore $\vec{v}_r$, gde $r = 1, \ldots, D_{\text{amb}} - 2$, kao linearne kombinacije vektora $\vec{u}_a$, po analogiji sa vektorima $\vec{u}_a$ koje smo uveli gore. Svaki od tih vektora se može prikazati u bazisu $\vec{u}_a$ kao suma po komponentama,

$$\vec{v}_r = \sum_{a=1}^{D_{\text{amb}}-1} v_{ra}\, \vec{u}_a \,,$$

i njihove komponente glase ovako (po analogiji sa formulama za kompoenente $u_{ai}$, samo sve za jednu dimenziju manje):

$$v_{ra} = \begin{cases} 0 & \text{kada je} \quad a < r \,, \\ -\dfrac{1}{M_r} \displaystyle\prod_{t=1}^{r} \sin\beta_t & \text{kada je} \quad a = r \,, \\ \dfrac{1}{M_r} \cos\beta_r \cos\beta_a \displaystyle\prod_{\substack{t=0 \\ t\neq r}}^{a-1} \sin\beta_t & \text{kada je} \quad a > r \,. \end{cases} \tag{11}$$

U ovim izrazima $M_r$ je norma vektora $\vec{v}_r$, i data je ovako:

$$M_r \equiv \sqrt{\vec{v}_r \cdot \vec{v}_r} = \sqrt{\prod_{t=1}^{r} \sin^2\beta_t + \cos^2\beta_r \sum_{a=r+1}^{D_{\text{amb}}-1} \cos^2\beta_a \prod_{\substack{t=0 \\ t\neq r}}^{a-1} \sin^2\beta_t} \,.$$

Kada su vektori $\vec{v}_r$ ovako zadati, ukupan skup $(\vec{v}_1, \ldots, \vec{v}_{(D_{\text{amb}}-2)})$ čini jedan ortonormirani skup vektora, po konstrukciji ortogonalnih na $\vec{c}$ i na $\vec{e}_X$,

$$\vec{c}\cdot\vec{v}_r = 0 \,, \qquad \vec{e}_X \cdot \vec{v}_r = 0 \,, \qquad \vec{v}_r \cdot \vec{v}_r = 1 \,, \qquad \vec{v}_r \cdot \vec{v}_t = 0 \qquad (t \neq r) \,,$$

i zajedno sa vektorima $\vec{c}$ i $\vec{e}_X$ čine bazis u ambijentalnom prostoru. U celoj konstrukciji je značajno to što su i ovi vektori u potpunosti definisani parametrima $s_X$ i $\beta_1, \ldots, \beta_{(D_{\text{amb}}-2)}$ kojima smo definisali vektor $\vec{e}_X$.

Sada vektore $\vec{v}_r$ možemo konačno da iskoristimo za parametrizaciju vektora $\vec{e}_Y$, kojim kompletiramo zadavanje ravni. Vektor $\vec{e}_Y$ mora da bude ortogonalan na $\vec{c}$ i $\vec{e}_X$, pa se samim tim može zapisati kao linearna kombinacija vektora $\vec{v}_r$:

$$\vec{e}_Y = \sum_{r=1}^{D_{\text{amb}}-2} e_{Yr} \vec{v}_r \,.$$

Pošto vektori $\vec{v}_r$ čine bazis za potprostor dimenzije $D_{\text{amb}} - 2$ velikog ambijentalnog prostora, vektor $\vec{e}_Y$ možemo izraziti u odgovarajućim sfernim koordinatama u tom potprostoru, na sledeći način:

$$e_{Yr} = s_Y \cos\gamma_r \prod_{t=0}^{r-1} \sin\gamma_t \,, \qquad r = 1, \ldots, D_{\text{amb}} - 2 \,. \tag{12}$$

Ovde prepoznajemo sledeći skup parametara kojima zadajemo $\vec{e}_Y$:

- parametar $s_Y \in \mathbb{R}^+$, koji određuje "skalu" duž $Y$-ose na ekranu, tj. intenzitet vektora $\vec{e}_Y$,
- parametre $\gamma_1, \ldots, \gamma_{(D_{\text{amb}}-4)} \in [0, \pi]$ i $\gamma_{(D_{\text{amb}}-3)} \in [0, 2\pi)$, koji određuju orijentaciju vektora $\vec{e}_Y$ u ambijentalnom prostoru,
- konstante $\gamma_0 = \frac{\pi}{2}$ i $\gamma_{D_{\text{amb}}-2} = 0$.

Primetite da uglova $\gamma_1, \ldots, \gamma_{(D_{\text{amb}}-3)}$ ima tačno jedan manje nego uglova $\beta$ koji definišu vektor $\vec{e}_X$. Konstrukcija je napravljena tako da izborom $\gamma$-uglova možemo da izaberemo bilo koju orijentaciju vektora $\vec{e}_Y$ u ambijentalnom prostoru, ali tako da mu ne menjamo intenzitet $s_Y$ i da uvek ostane ortogonalan na vektore $\vec{c}$ i $\vec{e}_X$, ma kako ih orijentisali.

**Rezime.** Sada sve ove rezultate možemo da pokupimo i sklopimo u jednu celinu. Budući da su vektori $\vec{v}_r$ izraženi preko $\vec{u}_a$, koji su opet izraženi preko originalnog Dekartovog bazisa $\vec{e}_i$ ambijentalnog prostora, možemo sva tri vektora koji definišu ravan, $\vec{c}$, $\vec{e}_X$ i $\vec{e}_Y$ da napišemo preko komponenti u velikom bazisu $\vec{e}_i$, ovako:

$$\vec{c} = \sum_{i=1}^{D_{\text{amb}}} c_i \, \vec{e}_i \,,$$

$$\vec{e}_X = \sum_{i=1}^{D_{\text{amb}}} e_{Xi} \, \vec{e}_i \,, \qquad \text{gde je} \qquad e_{Xi} = \sum_{a=1}^{D_{\text{amb}}-1} e_{Xa} u_{ai} \,,$$

$$\vec{e}_Y = \sum_{i=1}^{D_{\text{amb}}} e_{Yi} \, \vec{e}_i \,, \qquad \text{gde je} \qquad e_{Yi} = \sum_{a=1}^{D_{\text{amb}}-1} \sum_{r=1}^{D_{\text{amb}}-2} e_{Yr} v_{ra} u_{ai} \,.$$

Svi koeficijenti koji ovde figurišu, $c_i$, $e_{Xa}$, $e_{Yr}$, $u_{ai}$, $v_{ra}$, zadati su preko tri realna broja $d$, $s_X$, $s_Y$ koji određuju intenzitete vektora $\vec{c}$, $\vec{e}_X$, $\vec{e}_Y$, i preko tri skupa uglova, $\alpha_i$, $\beta_a$, $\gamma_r$, koji određuju njihove orijentacije. Cela konstrukcija je urađena tako da su vektori $\vec{c}$, $\vec{e}_X$, $\vec{e}_Y$ međusobno uvek ortogonalni, bez obzira na to kako izaberemo sve ove parametre.

Takođe skrećem pažnju da koeficijenti $c_i$, $e_{Xa}$, $e_{Yr}$, $u_{ai}$, $v_{ra}$ mogu da se shvate kao (ne-kvadratne) matrice, i da se gornje formule prepišu koristeći matrično množenje tih matrica. To ćemo iskoristiti niže kad budemo zadali konačne jednačine za projektovanje neke proizvoljne zadate tačke prostora (verteksa) na našu ravan.

Ovime smo kompletirali postupak zadavanja dvodimenzionalne ravni u $D_{\text{amb}}$-dimenzionalnom ambijentalnom prostoru. Sve je zadato intenzitetima $d$, $s_X$, $s_Y$, uglovima $\alpha_i$, $\beta_a$, $\gamma_r$, i gomilom komplikovane trigonometrije, tako da imamo jasnu geometrijsku interpretaciju svakog od tih parametara, i tako da ravan možemo da smestimo bilo gde u prostoru i orijentišemo je bilo kako, tako da ona uvek "gleda" u koordinatni početak prostora, i da imamo dobro definisane ose $X$ i $Y$ za crtanje ravni na ekran.

Sada prelazimo na poslednji korak u celoj priči — formule za projekciju tačke prostora na ravan, tj. funkciju koja na odgovarajući način preslikava skup koordinata $(x_1, \ldots, x_{D_{\text{amb}}})$ date tačke u koordinate $(X, Y)$ u našoj ravni.

## 3.2 Projektovanje datog verteksa na ravan ekrana

Neka je zadat položaj nekog verteksa našeg simplicijalnog kompleksa njegovim vektorom položaja u ambijentalnom prostoru,

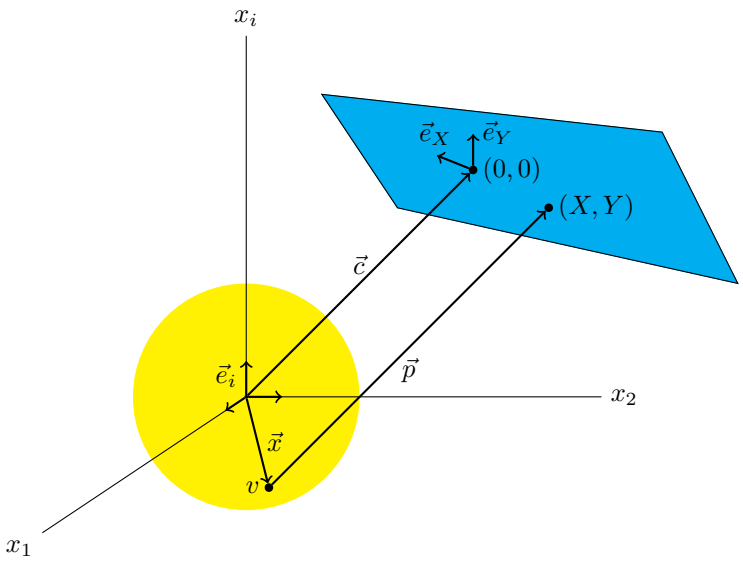
$$\vec{x} = \sum_{i=1}^{D_{\text{amb}}} x_i \, \vec{e}_i \,,$$

gde su $x_i$ koordinate koje smo dobili embedding-om iz topoloških koordinata $q_i$, koje smo odredili minimizovanjem nekog izraza za potencijal. Sada je naš zadatak da definišemo funkciju koja će koordinate $x_i$ da preslika u koordinate ekrana $(X, Y)$, projekcijom na ravan definisanu vektorima $\vec{c}$, $\vec{e}_X$, $\vec{e}_Y$ iz prethodnog odeljka.

Postoje *dve* glavne funkcije koje obavljaju taj posao — tzv. *ortogonalna projekcija*, i tzv. *projekcija u perspektivi*. Intuitivno, prva predstavlja "tehnički crtež" date figure, tj. projekciju koja čuva paralelnost i uglove horizontalnih i vertikalnih linija figure (u odnosu na ravan projekcije). Druga predstavlja "crtež iz mog ugla" date figure, tj. projekciju koja realistično prikazuje šta vidi ljudsko oko kada sa nekog rastojanja pogleda figuru, i ne čuva ništa od paralelnosti, rastojanja i uglova.

### 3.2.1 Ortogonalna projekcija

Na dijagramu je ilustrovan metod ortogonalne projekcije tačke na ravan.




Zadata je ravan ekrana vektorima $\vec{c}$, $\vec{e}_X$ i $\vec{e}_Y$, i zadat je verteks $v$ svojim vektorom položaja $\vec{x}$. Po definiciji, ortogonalna projekcija konstruiše vektor projekcije $\vec{p}$ tako da bude paralelan sa $\vec{c}$, ali da polazi iz tačke $v$ umesto iz koordinatnog početka, i da dostiže tačku $(X, Y)$ u ravni. Tu tačku onda zovemo ortogonalna projekcija verteksa $v$ na datu ravan. Sa slike se vidi da vektor položaja verktesa $\vec{x}$ možemo paralelno da prenesemo iz koordinatnog početka ambijentalnog prostora duž vektora $\vec{c}$ u koordinatni početak ravni $(0, 0)$, i tada koordinate $(X, Y)$ projekcije verteksa određujemo projektovanjem (tj. računanjem skalarnog proizvoda) vektora $\vec{x}$ na bazisne vektore $\vec{e}_X$ i $\vec{e}_Y$:

$$X = \left\lfloor \frac{\vec{x}\cdot\vec{e}_X}{s_X^2} \right\rfloor = \left\lfloor \frac{1}{s_X^2} \sum_{i=1}^{D_{\text{amb}}} x_i\, e_{Xi} \right\rfloor , \qquad Y = \left\lfloor \frac{\vec{x}\cdot\vec{e}_Y}{s_Y^2} \right\rfloor = \left\lfloor \frac{1}{s_Y^2} \sum_{i=1}^{D_{\text{amb}}} x_i\, e_{Yi} \right\rfloor . \tag{13}$$

Ovde računamo ceo deo datih izraza jer su $X$ i $Y$ koordinate tipa `int`, budući da predstavljaju koordinate piksela na ekranu. Projekcije moraju da se podele faktorima $s_X^2$ i $s_Y^2$ jer su vektori $\vec{e}_X$ i $\vec{e}_Y$ nisu normirani, nego imaju intenzitete $s_X$ odnosno $s_Y$, respektivno.

Uzimajući u obzir kako se određuju komponente bazisnih vektora $\vec{e}_X$ i $\vec{e}_Y$ iz prethodnog odeljka, ovi rezultati se mogu izraziti množenjem odgovarajućih matrica, na sledeći način:

$$\begin{aligned} X &= \frac{1}{s_X^2}\big[\quad x_i \quad\big]_{1\times D_{\text{amb}}} \Big[\quad u_{ia} \quad\Big]_{D_{\text{amb}}\times(D_{\text{amb}}-1)} \Big[e_{Xa}\Big]_{(D_{\text{amb}}-1)\times 1} , \\ Y &= \frac{1}{s_Y^2}\big[\quad x_i \quad\big]_{1\times D_{\text{amb}}} \Big[\quad u_{ia} \quad\Big]_{D_{\text{amb}}\times(D_{\text{amb}}-1)} \Big[\quad v_{ar} \quad\Big]_{(D_{\text{amb}}-1)\times(D_{\text{amb}}-2)} \Big[e_{Ya}\Big]_{(D_{\text{amb}}-2)\times 1} , \end{aligned} \tag{14}$$

(Ovde nisam napisao oznake za ceo deo da ne komplikujem izraz, podrazumevaju se.) Matrica $[u_{ia}]$ je transponovana matrica matrice $[u_{ai}]$ čije su komponente zadate kao funkcije parametara $\alpha$ u jednačini (9). Matrica $[v_{ar}]$ je transponovana matrica matrice $[v_{ra}]$ čije su komponente zadate kao funkcije parametara $\beta$ u jednačini (11). Komponente matrice-kolone $[e_{Xa}]$ su zadate kao funkcije parametara $s_X, \beta$ u jednačini (10), dok su komponente matrice-kolone $[e_{Yr}]$ zadate kao funkcije parametara $s_Y, \gamma$ u jednačini (12). Kada su zadati svi ovi podaci, množenjem gornjih matrica dobijamo kompletno preslikavanje koordinata date tačke $(x_1, \ldots, x_{D_{\text{amb}}})$ u tačku $(X, Y)$ na ravan definisanu parametrima $s_X, s_Y, \alpha, \beta, \gamma$. Time je definisana ortogonalna projekcija.

Obratite pažnju na to da parametar $d$ koji definiše rastojanje ravni od koordinatnog početka ambijentalnog prostora ne figuriše nigde u ovim jednačinama. Ovo je intuitivno jasno, jer zbog paralelne prirode vektora $\vec{p}$ i $\vec{c}$ položaj projekcije $(X, Y)$ uopšte ne zavisi od toga koliko je ravan udaljena od koordinatnog početka. Uveličavanje i smanjivanje slike na ekranu (zoom-in/out efekat) kontrolišemo promenom

parametara skale $s_X$ i $s_Y$, menjajući ih proporcionalno. Uglovi $\alpha, \beta, \gamma$ definišu ukupnu orijentaciju ravni u ambijentalnom prostoru.

**Zadatak 31.** Implementirati funkciju

```
vector<vector<int>> evaluate_orthogonal_projection( vector<vector<double>> x,
                                                    double sx,
                                                    double sy,
                                                    vector<double> alpha,
                                                    vector<double> beta,
                                                    vector<double> gamma )
```

koja kao input prima pointer na matricu sa koordinatama $x_i$ svih verteksa koje projektujemo, zatim vrednosti $s_X, s_Y$ i pointere na vektore sa vrednostima uglova $\alpha, \beta, \gamma$, i kao output daje matricu sa koordinatama $(X, Y)$ ortogonalne projekcije svih verteksa. Koordinate se računaju korišćenjem jednačina (14), (9), (11), (10), (12). Matrica `x` je tipa $D_{\text{amb}} \times V$ gde je $V$ broj verteksa u kompleksu, i u svakoj vrsti matrice se nalaze koordinate jednog verteksa $(x_1, \ldots, x_{D_{\text{amb}}})$ koje treba projektovati. Matrica output-a je dimenzije $2 \times V$, gde se u svakoj vrsti nalaze koordinate $(X, Y)$ projekcije, po jedan verteks u vrsti.

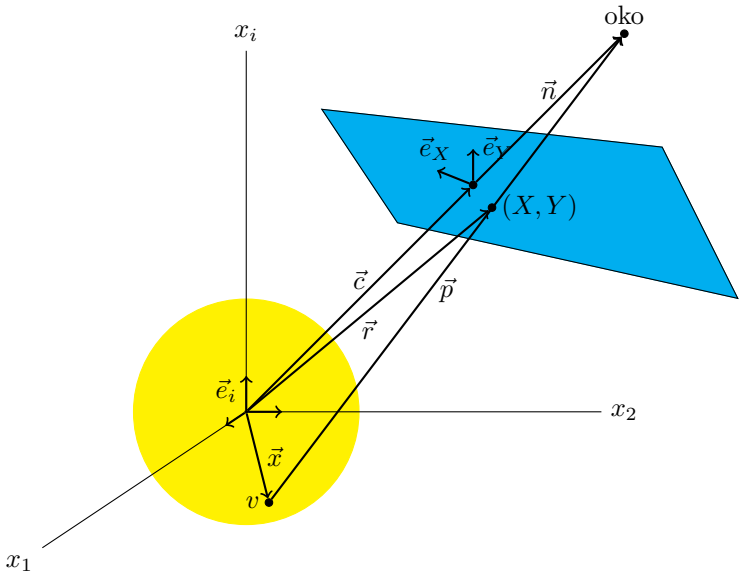
// Skrećem pažnju da nisam siguran da sam dobro napisao sintaksu za deklarisanje pointera na input i output vektore u imenu f-je, popravite to ako treba. //

Ova funkcija *nije* "one-time effort", nego treba da se računa u realnom vremenu, kontinualno dok korisnik pomera slajdere na ekranu kojima zadaje uglove $\alpha, \beta, \gamma$ i $s_X, s_Y$. Ovu funkciju dakle treba implementirati tako da radi u multi-core i multi-thread režimu, tj. da koristi sve procesore u datoj mašini. Takođe, pošto je crtanje kompleksa "uživo" operacija koja se vrši u GUI-ju, a GUI se po pravilu izvršava na *jednoj* mašini, treba koristiti procesorske resurse te mašine, dok nam eventualne druge mašine (klaster, MPI, itd.) ovde nisu relevantne.

U algoritmu najpre treba izračunati "teške" matrice $[u_{ia}]$, $[v_{ar}]$, $[e_{Xa}]$, $[e_{Yr}]$, na osnovu varijabli $s_X, s_Y, \alpha, \beta, \gamma$ (teške su zbog trigonometrije koju treba uraditi), pa zatim primeniti te matrice redom na svaku vrstu matrice sa koordinatama verteksa (što je relativno lako jer je u pitanju samo matrično množenje), i time generisati matricu sa koordinatama projekcija svih verteksa. One se dalje prosleđuju funkciji koja će ih crtati na ekran. Pošto ovu celu funkciju treba izvršiti svaki put kada korisnik mrdne neki slajder i promeni vrednost nekog od ulaznih parametara, važno je da se ova funkcija izvršava *što je moguće brže*. Slika na tipičnom VGA ekranu se osvežava sa frekvencom od 60 Hz, što znači da bi cela ova funkcija (zajedno sa funkcijom koja zapravo iscrtava piksele na ekran) trebalo da se završi pre toga, da i računanje i crtanje budu gotovi između svaka dva video frejma.

### 3.2.2 Projekcija u perspektivi

Na dijagramu je ilustrovan metod projekcije u perspektivi tačke na ravan.



Zadata je ravan ekrana vektorima $\vec{c}$, $\vec{e}_X$ i $\vec{e}_Y$, zadat je verteks $v$ svojim vektorom položaja $\vec{x}$, i zadat je položaj "oka" vektorom $\vec{n}$ definisanim kao

$$\vec{n} = \frac{s_Z}{d}\ \vec{c},$$

gde je $s_Z \in \mathbb{R}^+$ nov parametar koji zadaje udaljenost oka od ravni ekrana, a $d$ je poznati parametar koji zadaje dužinu vektora $\vec{c}$. Po definiciji, projekcija u perspektivi konstruiše vektor projekcije $\vec{p}$ tako da polazi od verteksa $v$ i završava u oku, usput prodirući ravan ekrana u tački $(X, Y)$ u ravni. Tu tačku onda zovemo projekcija verteksa $v$ na datu ravan iz perspektive datog oka, i njen vektor položaja označavamo sa $\vec{r}$.

Sa slike se vidi da je vektor $\vec{p}$ određen jednačinom

$$\vec{p} = \vec{c} + \vec{n} - \vec{x}\,, \tag{15}$$

čime je izražen kao funkcija poznatih veličina. Sa druge strane, sa slike se vidi da vektor položaja projekcije $\vec{r}$ zadovoljava dve jednačine:

$$\vec{r} = \vec{c} + X\,\vec{e}_X + Y\,\vec{e}_Y\,, \qquad \vec{r} = \vec{x} + t\,\vec{p}.$$

Prva jednačina kaže da vrh vektora $\vec{r}$ leži u ravni ekrana — jednačina (7). Druga jednačina kaže da se vrh vektora $\vec{r}$ nalazi na pravoj liniji koja prolazi kroz verteks $v$ sa pravcem duž vektora $\vec{p}$ i parametrom $t \in \mathbb{R}$ koji prebrojava tačke na pravoj. Izjednačavanjem ova dva izraza dobijamo sistem jednačina

$$\vec{c} + X\,\vec{e}_X + Y\,\vec{e}_Y = \vec{x} + t\,\vec{p}, \tag{16}$$

čijim rešavanjem dobijamo parametar $t$, kao i tražene koordinate $X$ i $Y$. Prvo određujemo parametar $t$ tako što celu jednačinu skalarno pomnožimo sa $\vec{p}$ i rešimo po $t$:

$$t = \frac{1}{p^2}\left(\vec{c}\cdot\vec{p} + X\,\vec{e}_X\cdot\vec{p} + Y\,\vec{e}_Y\cdot\vec{p} - \vec{x}\cdot\vec{p}\right),$$

gde smo uveli skraćenu oznaku $p^2 \equiv \vec{p}\cdot\vec{p}$. Ako ovde još zamenimo (15) i malo pospremimo (uzimajući u obzir da su $\vec{c}$ i $\vec{n}$ ortogonalni na $\vec{e}_X$ i $\vec{e}_Y$), dobijamo:

$$t = \frac{1}{p^2}\left(d(d+s_Z) - \frac{2d+s_z}{d}\vec{x}\cdot\vec{c} + \vec{x}\cdot\vec{x} - X\,\vec{x}\cdot\vec{e}_X - Y\,\vec{x}\cdot\vec{e}_Y\right).$$

Zamenom ovog izraza u (16), skalarnim množenjem sa $\vec{e}_X$ i $\vec{e}_Y$ i sređivanjem dobijamo sledeći sistem jednačina za $X$ i $Y$:

$$\begin{aligned}
\left[(\vec{x}\cdot\vec{e}_X)(\vec{x}\cdot\vec{e}_X) - s_X^2 p^2\right] X \quad &+ \quad (\vec{x}\cdot\vec{e}_X)(\vec{x}\cdot\vec{e}_Y)\,Y \quad = \quad s_Z\left(\frac{\vec{x}\cdot\vec{c}}{d} - d - s_Z\right)(\vec{x}\cdot\vec{e}_X)\,,\\
(\vec{x}\cdot\vec{e}_X)(\vec{x}\cdot\vec{e}_Y)\,X \quad &+ \quad \left[(\vec{x}\cdot\vec{e}_Y)(\vec{x}\cdot\vec{e}_Y) - s_Y^2 p^2\right] Y \quad = \quad s_Z\left(\frac{\vec{x}\cdot\vec{c}}{d} - d - s_Z\right)(\vec{x}\cdot\vec{e}_Y)\,,
\end{aligned}$$

Konačno, uvođenjem zgodnije notacije

$$v_X = \vec{x}\cdot\vec{e}_X\,, \qquad v_Y = \vec{x}\cdot\vec{e}_Y\,, \qquad v_c = \vec{x}\cdot\vec{c}, \qquad v^2 = \vec{x}\cdot\vec{x}\,, \tag{17}$$

i rešavanjem sistema dobijamo tražene koordinate $(X, Y)$:

$$X = \left\lfloor F\,s_Y^2\,v_X \right\rfloor, \qquad Y = \left\lfloor F\,s_X^2\,v_Y \right\rfloor, \tag{18}$$

gde je

$$F \equiv \frac{s_Z\left(s_Z + d - \dfrac{v_c}{d}\right)}{s_X^2 s_Y^2 (s_Z+d)\left[s_Z + d - \dfrac{2v_c}{d}\right] + s_X^2 s_Y^2 v^2 - s_Y^2 v_X^2 - s_X^2 v_Y^2}\,.$$

Jednačine (18) zajedno sa (17) predstavljaju konačan rezultat kojim se realizuje projekcija u perspektivi, tj. preslikavanje koordinata $(x_1, \ldots, x_{D_{\text{amb}}})$ datog verteksa u koordinate ekrana $(X, Y)$. Kao i prošli put,

$X$ i $Y$ se računaju kao ceo deo, jer su tipa `int` i prebrojavaju piksele na ekranu. Jednačine (17) se mogu napisati u matričnom obliku, slično kao u prethodnom odeljku,

$$\begin{aligned} v_X &= \left[\ x_i\ \right]_{1\times D_{\text{amb}}} \left[\ u_{ia}\ \right]_{D_{\text{amb}}\times(D_{\text{amb}}-1)} \left[e_{Xa}\right]_{(D_{\text{amb}}-1)\times 1}, \\ v_Y &= \left[\ x_i\ \right]_{1\times D_{\text{amb}}} \left[\ u_{ia}\ \right]_{D_{\text{amb}}\times(D_{\text{amb}}-1)} \left[\ v_{ar}\ \right]_{(D_{\text{amb}}-1)\times(D_{\text{amb}}-2)} \left[e_{Ya}\right]_{(D_{\text{amb}}-2)\times 1}, \\ v_c &= \left[\ x_i\ \right]_{1\times D_{\text{amb}}} \left[c_i\right]_{D_{\text{amb}}\times 1}, \\ v^2 &= \left[\ x_i\ \right]_{1\times D_{\text{amb}}} \left[x_i\right]_{D_{\text{amb}}\times 1}, \end{aligned} \tag{19}$$

pri čemu su komponente matrice-kolone $[c_i]$ date jednačinom (8).

Obratite pažnju na to da parametar $d$ koji definiše rastojanje ravni od koordinatnog početka ambijentalnog prostora sada figuriše u jednačinama, za razliku od slučaja ortogonalne projekcije. Osim njega, figuriše i novi parametar $s_Z$ koji predstavlja rastojanje oka od ravni ekrana. Takođe imamo i stare parametre — parametre skale $s_X$ i $s_Y$ (za zoom-in/out), i skupove uglova $\alpha, \beta, \gamma$ za orijentaciju ravni u ambijentalnom prostoru.

Takođe obratite pažnju da se ortogonalna projekcija može razumeti kao specijalan slučaj projekcije u perspektivi, kada je oko posmatrača udaljeno beskonačno daleko od ravni ekrana. Tada vektor $\vec{p}$ postaje paralelan sa vektorom $\vec{c}$ (vidi sliku), i perspektiva se svodi na ortogonalnu projekciju. Ovo može da se vidi i iz jednačina (18) — u limesu $s_Z \to \infty$ imamo da $F \to 1/s_x^2 s_y^2$, pa se izrazi za $X$ i $Y$ svode na izraze (13) za ortogonalnu projekciju.

**Zadatak 32.** Implementirati funkciju

```
vector<vector<int>> evaluate_perspective_projection( vector<vector<double>> x,
                                                     double d,
                                                     double sx,
                                                     double sy,
                                                     double sz,
                                                     vector<double> alpha,
                                                     vector<double> beta,
                                                     vector<double> gamma )
```

koja kao input prima pointer na matricu sa koordinatama $x_i$ svih verteksa koje projektujemo, zatim vrednosti $d, s_X, s_Y, s_Z$ i pointere na vektore sa vrednostima uglova $\alpha, \beta, \gamma$, i kao output daje matricu sa koordinatama $(X, Y)$ projekcije u perspektivi svih verteksa. Koordinate se računaju korišćenjem jednačina (18), (19), (8), (9), (11), (10), (12). Matrica `x` je tipa $D_{\text{amb}} \times V$ gde je $V$ broj verteksa u kompleksu, i u svakoj vrsti matrice se nalaze koordinate jednog verteksa $(x_1, \ldots, x_{D_{\text{amb}}})$ koje treba projektovati. Matrica output-a je dimenzije $2 \times V$, gde se u svakoj vrsti nalaze koordinate $(X, Y)$ projekcije, po jedan verteks u vrsti.

// I ovde skrećem pažnju da nisam siguran da sam dobro napisao sintaksu za deklarisanje pointera na input i output vektore u imenu f-je, popravite to ako treba. //

Slično kao i u zadatku za ortogonalnu projekciju, ova funkcija *nije* "one-time effort", nego treba da se računa u realnom vremenu, kontinualno dok korisnik pomera slajdere na ekranu kojima zadaje uglove $\alpha, \beta, \gamma$ i parametre $d, s_X, s_Y, s_Z$. Ovu funkciju dakle treba isto implementirati tako da radi u multi-core i multi-thread režimu, tj. da koristi sve procesore u datoj mašini, i da radi što brže, tako da i računanje svih koordinata i crtanje na ekran može da se završi između dva video-frejma na tipičnom VGA monitoru sa vertikalnim osvežavanjem od 60 Hz.

## 3.3 Specijalni slučajevi

Razmotrimo na kraju kakva je struktura input parametara $d, s_X, s_Y, s_Z$ i uglova $\alpha, \beta, \gamma$ za slučaj malih dimenzija ambijentalnog prostora, konkretno $D_{\text{amb}} = 1, 2, 3, 4$. Slučajevi $D_{\text{amb}} \geq 4$ su već generički, jer

se svi uglovi pojavljuju. Sa druge strane, prve dve dimenzije su "degenerisane" u smislu da nisu sve jednačine dobro definisane. Prvi regularan slučaj gde je sve definisano je $D_{\text{amb}} = 3$, pri čemu i u tom slučaju imamo specifično mali broj uglova koji figurišu kao parametri.

### 3.3.1 Slučaj $D_{\text{amb}} = 1$

Jednodimenzionalni slučaj je skroz degenerisan u smislu da ništa od gornjih jednačina za određivanje projekcije $(X, Y)$ nije primenljivo. U ovom slučaju postoji samo jedna koordinata ambijentalnog prostora, $x_1$, i ne postoji druga dimenzija, kao ni razlika između ortogonalne projekcije i projekcije u perspektivi. U oba slučaja projekciju definišemo kao

$$X = \frac{x_1}{s_X}\,, \qquad Y = 0\,. \tag{20}$$

Jedini parametar koji ovde figuriše je $s_X$, i on određuje skalu rastojanja duž $X$-ose. Simpleksi (duži na liniji) su u ovom slučaju poređani na horizontalnoj pravoj liniji koja se prostire s leva na desno po ekranu, dok je vertikalno centrirana po sredini ekrana. Jednodimenzionalne komplekse (oblika lanca) ima smisla samo ovako crtati. Osim lanca, jednodimenzionalni kompleksi mogu da se zatvaraju u krug, ali to već zahteva embedding u ambijentalnom prostoru sa $D_{\text{amb}} = 2$, što je naredni slučaj.

### 3.3.2 Slučaj $D_{\text{amb}} = 2$

U ovom slučaju vektor $\vec{c} = 0$, jer se ambijentalni prostor u suštini poklapa sa ravni ekrana. Iz jednačine (8) vidimo da je $d = 0$, i da su uglovi $\alpha$ zadati kao

$$\alpha_0 = \frac{\pi}{2}\,, \qquad \alpha_1 \in [0, 2\pi)\,, \qquad \alpha_2 = 0\,,$$

odnosno postoji jedan slobodan parametar $\alpha_1$. Iz jednačine (9) može da se izračuna matrica $[u_{ia}]$:

$$[u_{ia}] = \begin{bmatrix} -\sin\alpha_1 \\ \cos\alpha_1 \end{bmatrix}_{2\times 1}\,.$$

Uglovi $\beta$ sadrže samo konstantne delove,

$$\beta_0 = \frac{\pi}{2}\,, \qquad \beta_1 = 0\,,$$

dok iz jednačine (10) možemo da izračunamo jednu jedinu komponentu matrice-kolone $e_{Xa}$,

$$[e_{Xa}] = [s_X]_{1\times 1}\,.$$

Međutim, što se tiče uglova $\gamma$, matrice $[v_{ar}]$ i matrice-kolone $[e_{Yr}]$, dvodimenzionalni slučaj je degenerisan, i jednačine (11) i (12) nisu dobro definisane, a uglovi $\gamma$ čine prazan skup.

Da bismo prevazišli ove probleme, definisaćemo vektore $\vec{e}_X$ i $\vec{e}_Y$ direktno u ambijentalnom prostoru. Vektor $\vec{e}_X$ čak može da se izračuna proizvodom gornjih matrica $[u_{ia}][e_{Xa}]$, čime dobijamo

$$[e_{Xi}] = \begin{bmatrix} -s_X \sin\alpha_1 \\ s_X \cos\alpha_1 \end{bmatrix}_{2\times 1}\,,$$

ali odgovarajuću matricu za vektor $\vec{e}_Y$ moramo da definišemo "ručno":

$$[e_{Yi}] = \begin{bmatrix} s_Y \cos\alpha_1 \\ s_Y \sin\alpha_1 \end{bmatrix}_{2\times 1}\,.$$

Lako je proveriti da su dva vektora međusobno ortogonalna. Za zadati verteks sa koordinatama $(x_1, x_2)$, koordinate $(X, Y)$ za ortogonalnu projekciju određujemo kao:

$$X = \left\lfloor \frac{\vec{x}\cdot\vec{e}_X}{s_X^2} \right\rfloor = \left\lfloor \frac{1}{s_X^2} [x_1 \;\; x_2] \begin{bmatrix} -s_X \sin\alpha_1 \\ s_X \cos\alpha_1 \end{bmatrix} \right\rfloor = \left\lfloor -\frac{x_1}{s_X}\sin\alpha_1 + \frac{x_2}{s_X}\cos\alpha_1 \right\rfloor, \tag{21}$$

odnosno

$$Y = \left\lfloor \frac{\vec{x}\cdot\vec{e}_Y}{s_Y^2} \right\rfloor = \left\lfloor \frac{1}{s_Y^2} [x_1 \;\; x_2] \begin{bmatrix} s_Y \cos\alpha_1 \\ s_Y \sin\alpha_1 \end{bmatrix} \right\rfloor = \left\lfloor \frac{x_1}{s_Y}\cos\alpha_1 + \frac{x_2}{s_Y}\sin\alpha_1 \right\rfloor. \tag{22}$$

Vidimo da koordinate $(X, Y)$ zavise od parametara $s_X, s_Y$ koji određuju skale duž dve ose, i ugla $\alpha_1$ koji rotira ravan ekrana oko ose ortogonalne na ekran.

Što se tiče projekcije u perspektivi, budući da je $d = 0$, može se izračunati da je u tom slučaju $F = 1/s_X^2 s_Y^2$, odnosno projekcija u perspektivi se ne razlikuje od ortogonalne projekcije, i ne zavisi od parametra $s_Z$. Drugim rečima, za projekciju u perspektivi treba iskoristiti iste jednačine (21) i (22) za računanje projekcije, kao i za ortogonalnu projekciju.

### 3.3.3 Slučaj $D_{\mathbf{amb}} = 3$

Trodimenzionalni slučaj je prvi slučaj koji se regularno ponaša. Vektor $\vec{c}$ sada postoji, i iz jednačine (8) imamo

$$[c_i] = \begin{bmatrix} d\cos\alpha_1 \\ d\cos\alpha_2\sin\alpha_1 \\ d\sin\alpha_2\sin\alpha_1 \end{bmatrix}_{3\times 1},$$

pri čemu je skup uglova $\alpha$ zadat kao

$$\alpha_0 = \frac{\pi}{2}, \qquad \alpha_1 \in [0,\pi], \qquad \alpha_2 \in [0,2\pi), \qquad \alpha_2 = 0,$$

odnosno postoje dva slobodna parametra $\alpha_1, \alpha_2$. Jednačina (9) daje matricu $[u_{ia}]$:

$$[u_{ia}] = \begin{bmatrix} -\sin\alpha_1 & 0 \\ \cos\alpha_1\cos\alpha_2 & -\sin\alpha_2 \\ \cos\alpha_1\sin\alpha_2 & \cos\alpha_2 \end{bmatrix}_{3\times 2}.$$

Uglovi $\beta$ sadrže jedan slobodan parametar,

$$\beta_0 = \frac{\pi}{2}, \qquad \beta_1 \in [0,2\pi), \qquad \beta_2 = 0,$$

dok iz jednačine (10) možemo da izračunamo matricu-kolonu $e_{Xa}$,

$$[e_{Xa}] = \begin{bmatrix} s_X\cos\beta_1 \\ s_X\sin\beta_1 \end{bmatrix}_{2\times 1}.$$

Uglovi $\gamma$ sada jesu definisani, i uzimaju samo konstantne vrednosti,

$$\gamma_0 = \frac{\pi}{2}, \qquad \gamma_1 = 0,$$

dok matrica $[v_{ar}]$ i matrica-kolona $[e_{Yr}]$ mogu da se odrede iz jednačina (11) i (12), redom:

$$[v_{ar}] = \begin{bmatrix} -\sin\beta_1 \\ \cos\beta_1 \end{bmatrix}_{2\times 1}, \qquad [e_{Yr}] = [s_Y]_{1\times 1}.$$

Sve ove matrice sada mogu da se stave u jednačine za ortogonalnu projekciju i projekciju u perspektivi na normalan način.

Interpretacija uglova je sledeća — $\alpha_1$ i $\alpha_2$ su redom geografska širina i geografska dužina koordinatnog početka ekrana u odnosu na koordinatni početak ambijentalnog prostora, odnosno standardni uglovi $\theta, \varphi$ iz sfernog koordinatnog sistema. Ugao $\beta_1$ definiše orijentaciju koordinatnih osa $(X, Y)$ na ekranu.

### 3.3.4 Slučaj $D_{\mathbf{amb}} = 4$ i više

Demonstrirajmo još i četvorodimenzionalni slučaj kao "generički" slučaj, u kome postoje i netrivijalni uglovi $\gamma$. Naime, skupovi uglova $\alpha, \beta, \gamma$ sada izgledaju ovako:

$$\alpha_0 = \frac{\pi}{2}, \qquad \alpha_1 \in [0,\pi], \qquad \alpha_2 \in [0,\pi], \qquad \alpha_3 \in [0,2\pi), \qquad \alpha_4 = 0,$$

$$\beta_0 = \frac{\pi}{2}, \qquad \beta_1 \in [0,\pi], \qquad \beta_2 \in [0,2\pi), \qquad \beta_3 = 0,$$

$$\gamma_0 = \frac{\pi}{2}, \qquad \gamma_1 \in [0,2\pi), \qquad \gamma_2 = 0,$$

odakle vidimo da imamo ukupno šest slobodnih parametara $\alpha_1, \alpha_2, \alpha_3, \beta_1, \beta_2, \gamma_1$, čija geometrijska interpretacija postoji, ali samo u četiri dimenzije, pa je nećemo objašnjavati.

Jednačine (8), (9), (11), (10), (12) mogu da se iskoriste za izračunavanje matrica $[c_i]$, $[u_{ia}]$, $[v_{ar}]$, $[e_{Xa}]$ i $[e_{Yr}]$, redom, koje se zatim mogu zameniti u izraze za ortogonalnu projekciju i projekciju u perspektivi, kako je navedeno u glavnom tekstu. Rezultujuće matrice glase:

$$[c_i] = \begin{bmatrix} d\cos\alpha_1 \\ d\cos\alpha_2\sin\alpha_1 \\ d\cos\alpha_3\sin\alpha_2\sin\alpha_1 \\ d\sin\alpha_3\sin\alpha_2\sin\alpha_1 \end{bmatrix}_{4\times 1}, \qquad [e_{Xa}] = \begin{bmatrix} s_X\cos\beta_1 \\ s_X\cos\beta_2\sin\beta_1 \\ s_X\sin\beta_2\sin\beta_1 \end{bmatrix}_{3\times 1}, \qquad [e_{Yr}] = \begin{bmatrix} s_Y\cos\gamma_1 \\ s_Y\sin\gamma_1 \end{bmatrix}_{2\times 1},$$

$$[u_{ia}] = \begin{bmatrix} -\sin\alpha_1 & 0 & 0 \\ \cos\alpha_1\cos\alpha_2 & -\sin\alpha_2 & 0 \\ \cos\alpha_1\cos\alpha_3\sin\alpha_2 & \cos\alpha_2\cos\alpha_3 & -\sin\alpha_3 \\ \cos\alpha_1\sin\alpha_2\sin\alpha_3 & \cos\alpha_2\sin\alpha_3 & \cos\alpha_3 \end{bmatrix}_{4\times 3}, \qquad [v_{ar}] = \begin{bmatrix} -\sin\beta_1 & 0 \\ \cos\beta_1\cos\beta_2 & -\sin\beta_2 \\ \cos\beta_1\sin\beta_2 & \cos\beta_2 \end{bmatrix}_{3\times 2}.$$