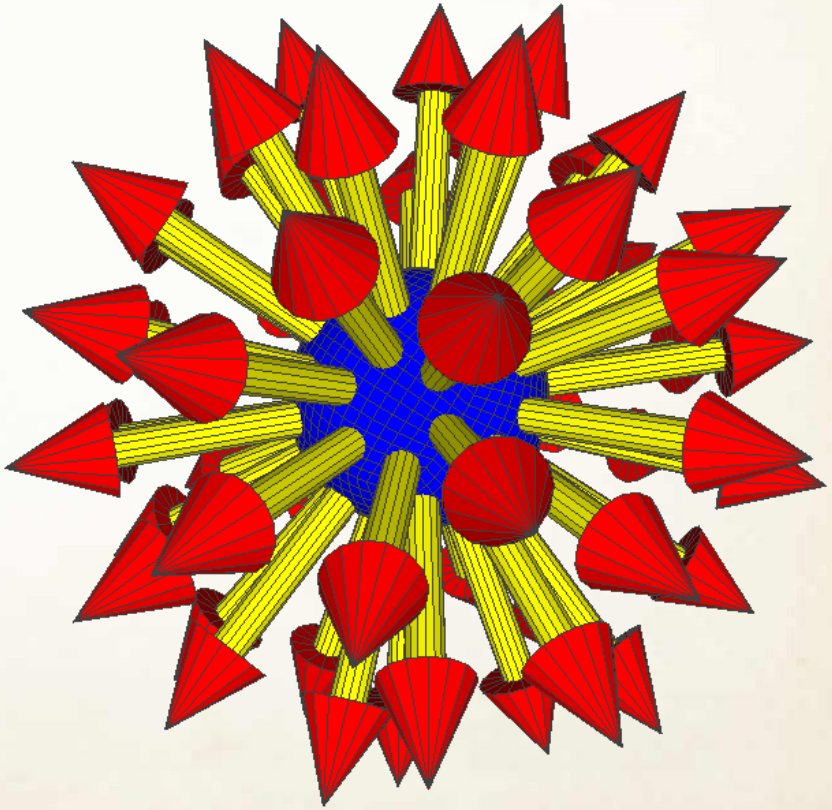
Algoritmai ir duomenų struktūros

12 paskaita



2023-05-03

Svarbi informacija

- Gegužės 3 d. paskaita skirta plačiau susipažinti kompiuteriniu kreivių ir paviršių modeliavimu.
- Gegužės 10 d. paskaita skirta kursui užbaigti.
- Gegužės 17 d. paskaita dedikuota Jūsų pasirinktų temų pristatymams.
- Gegužės 24 d. paskaita skirta sukurtų 3D modelių pristatymams ir gražiausių 3D modelių išrinkimui.
- Gegužės 31 d. paskaita – konsultacija.

- Atsiskaityti praktines užduotis galima iki gegužės galo.
- **Egzaminas birželio 16 d. 12 val. nuotoliniu būdu**
- **Egzamino trukmė – 2 val.**

12 paskaitos tikslas

- Plačiau susipažinti su kūrybine užduotimi ir naudinga medžiaga 3D modelio kūrimui:
  - Parametrinės kreivės:
    - Apskritimas;
    - Elipsė;
    - Bezjė kreivės.
  - Parametriniai paviršiai:
    - Sfera, cilindras, toras, sukinys ir kt.
    - Bezjė paviršiai.
  - Paviršių dalijimo algoritmai:
    - Catmullo ir Clarko paviršių dalijimo algoritmas.
- Apžvelgti modulio add.py modulio funkcijas (1.2b versija)

3D modelio kūrimas

UŽDUOTIS:

- Sukurti 3D modelį naudojant tik pirminį programos tekstą.
- Modelio failo formatas – „OFF“.
- Modelį galima kurti grupėje iki 3 studentų.

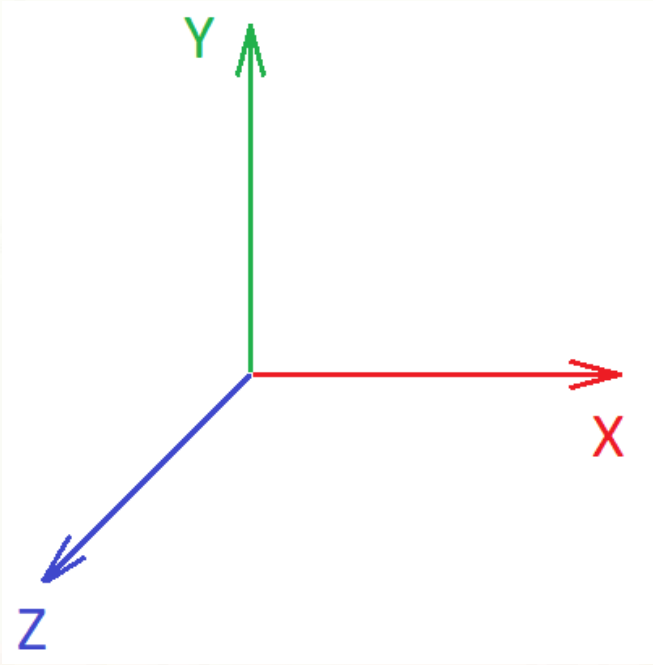
REIKALAVIMAI:

- Gautas 3D modelis turi būti sudarytas bent iš 1000 daugiakampių.
- Sukurtame algoritme privalo būti bent 1 parametras, nuo kurio priklausytų 3D modelio forma.
- Negalima naudoti 3D modeliavimui skirtų programų.
- Keičiant parametrus sukurtas 3D modelis negali sutapti su kursiokų modeliais arba paskaitų metu nagrinėjamais pavyzdžiais.

Vertinimo strategija

- Jei sukurtas 3D modelis tenkins reikalavimus, už jį bus skirta **po 1 balą kiekvienam** šio modelio **autoriui** (atsiskaitymas pratybų metu). Taip pat bus suteikta galimybė dalyvauti gražiausio 3D modelio konkurse.
- Sukurtų 3D modelių pristatymas, gražiausio modelio rinkimas – gegužės 24 d. (per teorijos paskaitą).
- Jūsų pačių išrinkti 5 geriausi modeliai atskirai bus įvertinti **0.25** premija. Jei keli autoriai, premija dalinama lygiomis dalimis.

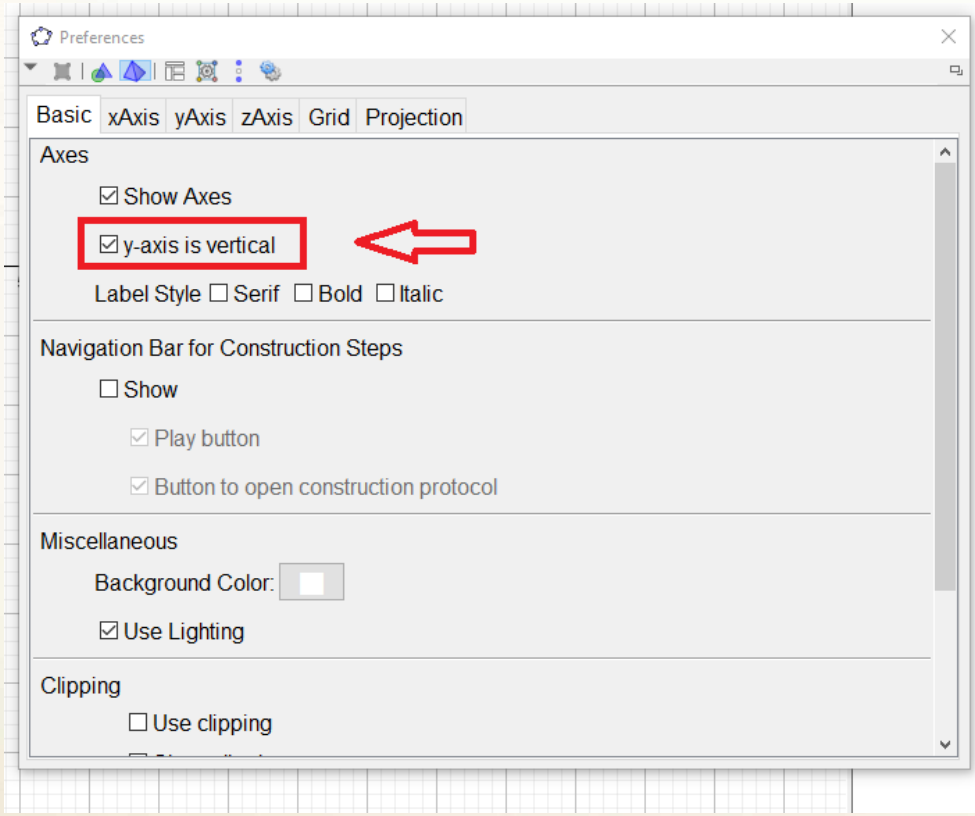
Naudosime šį Dekarto koordinačių ašių išdėstymą

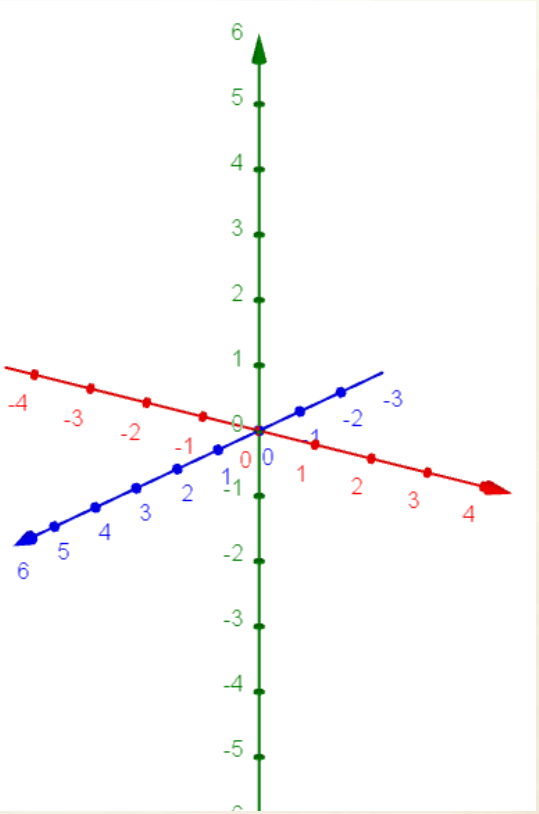


Pagal susitarimą X ašis visada žymima raudona spalva,
Y ašis – žalia ir Z ašis – mėlyna spalva.

Ašių sukeitimas Geogebra programoje

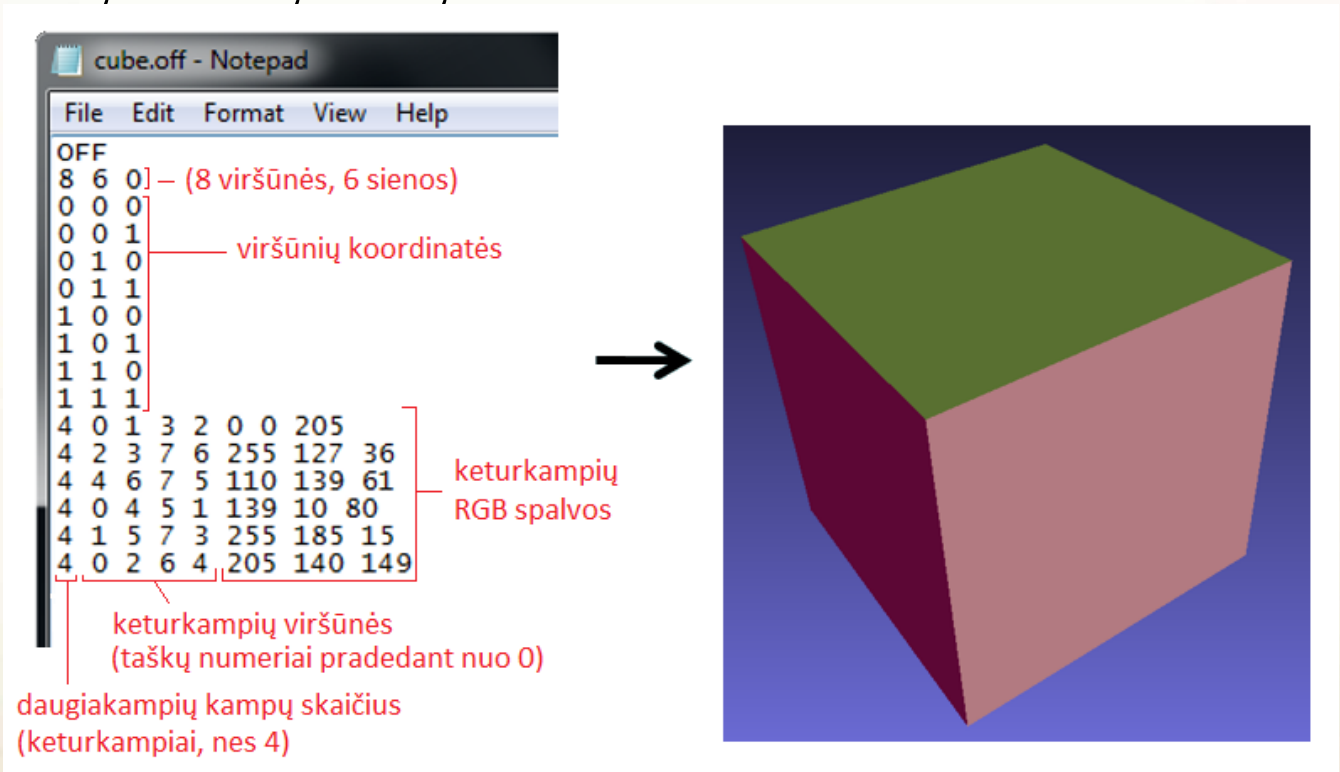
Options → Advanced → Preferences - 3D Graphics → Basic





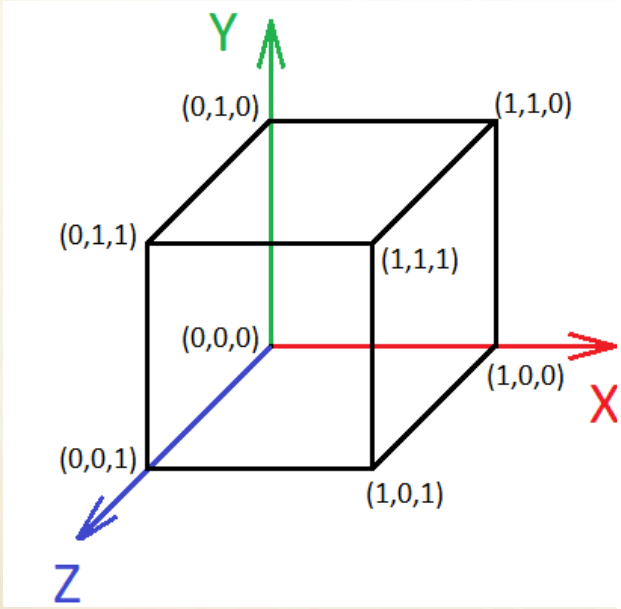
Skaitmeninių modelių formatai

*.ply, *.stl, *.obj, *.qobj, ***.off**, *.ptx, *.vmi, *.bre, *.dae, *.ctm, *.pts, *.apts, ***.xyz**, *.pcl, *.gts, *.pdb, *.tri, *.asc, *.x3d, *.x3dv, *.wrl, *.bw.

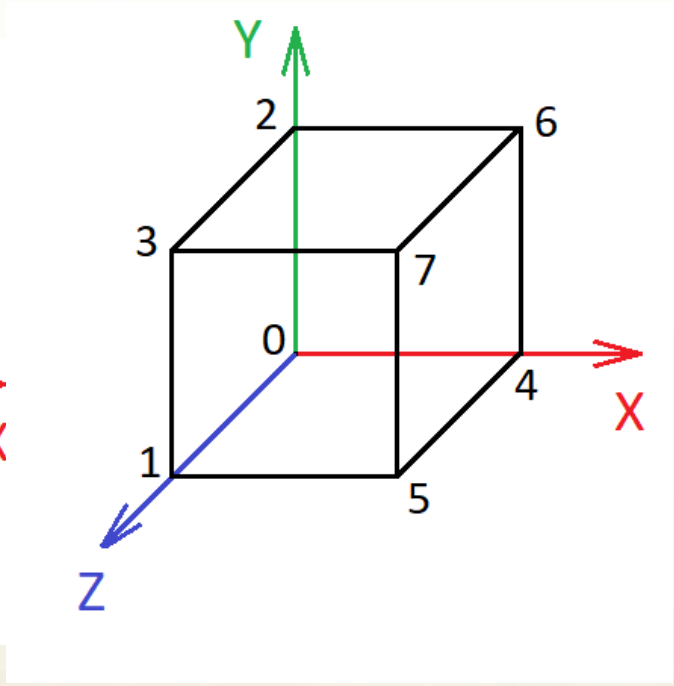


Kubo sudarymas off formatu

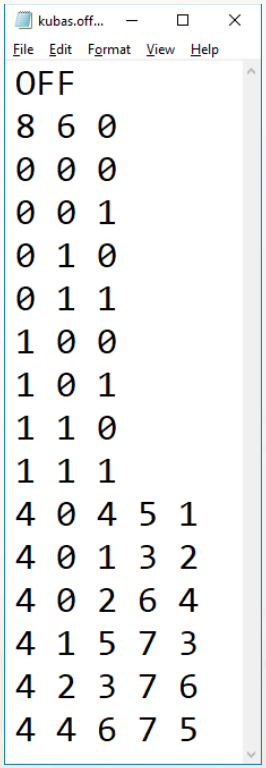
1 etapas: apskaičiuojamos kubo viršūnių koordinatės.



2 etapas: viršūnės indeksuojamos pradedant 0.

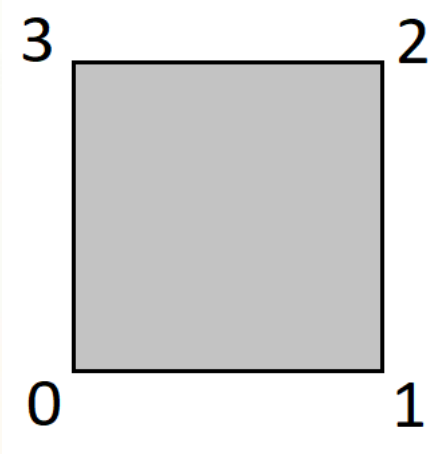


3 etapas: sukuriamas OFF failas, į kurį įrašomos viršūnių koordinatės ir viršūnių indeksų sekos.



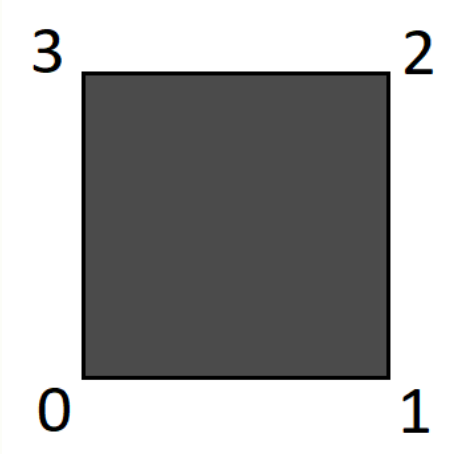
Vidinė ir išorinė siena

Išorinė keturkampio siena



4 0 1 2 3

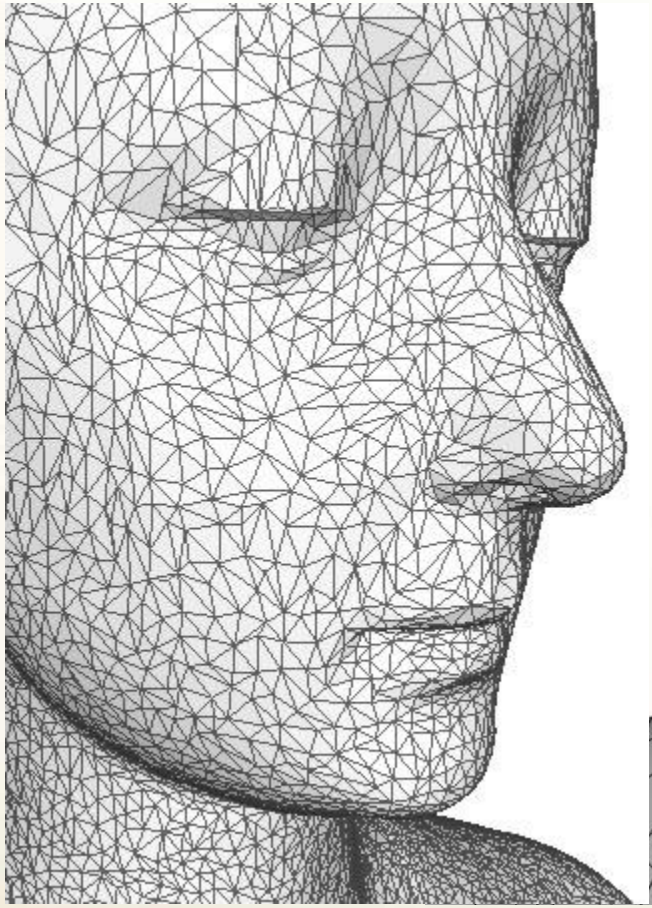
Vidinė keturkampio siena

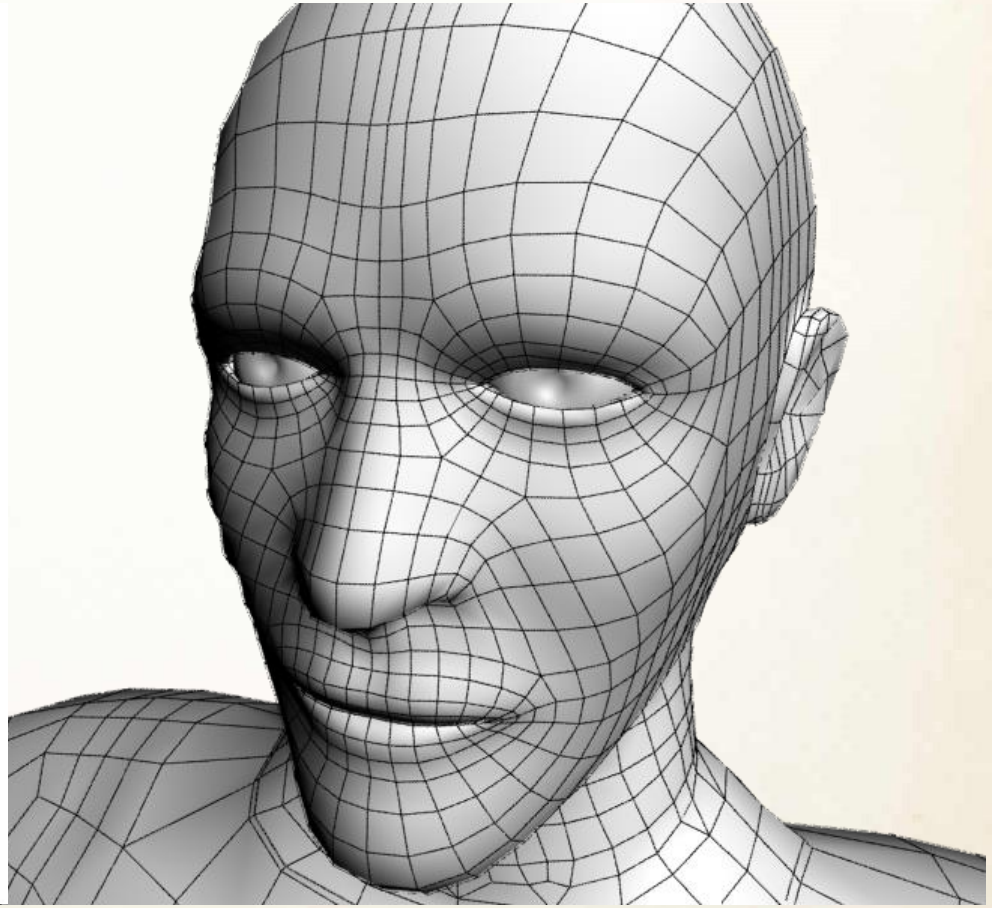


4 3 2 1 0

Žvelgiant į 3D modelį iš išorės, viršūnių indeksus reikia išdėstyti prieš laikrodžio rodyklę.
Žvelgiant į 3D modelį iš vidaus, viršūnių indeksus reikia išdėstyti pagal laikrodžio rodyklę.

Trianguliarusis ir kvadrianguliarusis paviršius

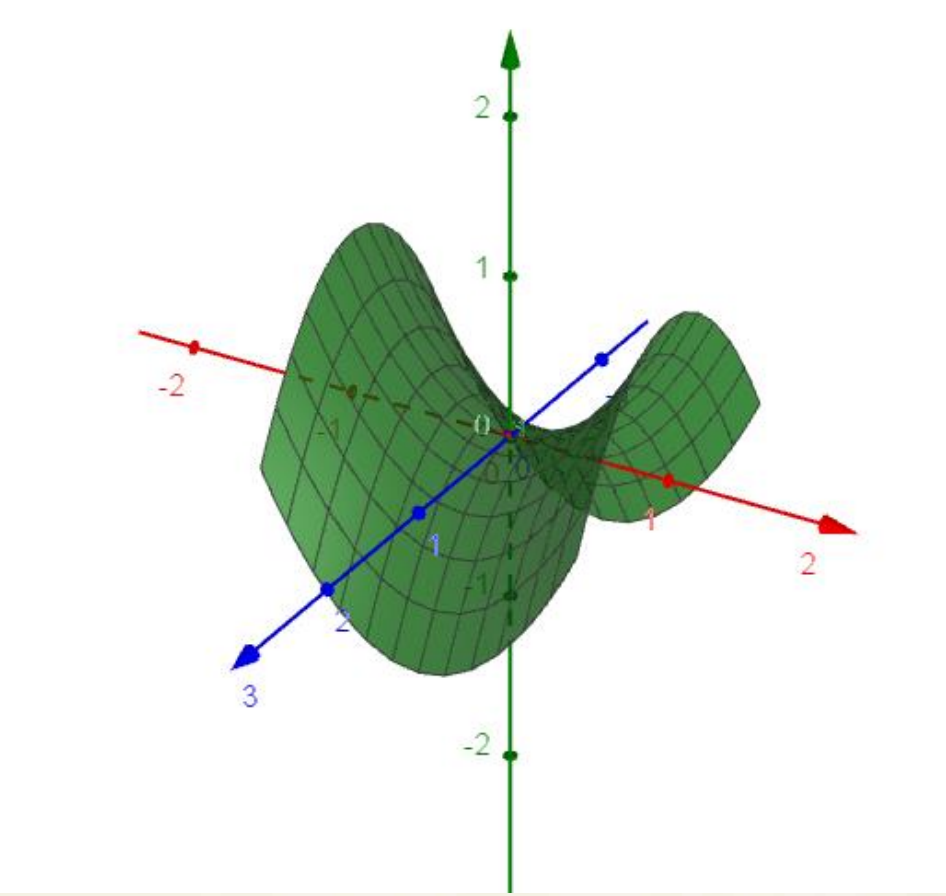




Rekomenduojami 3D modelio kūrimo etapai

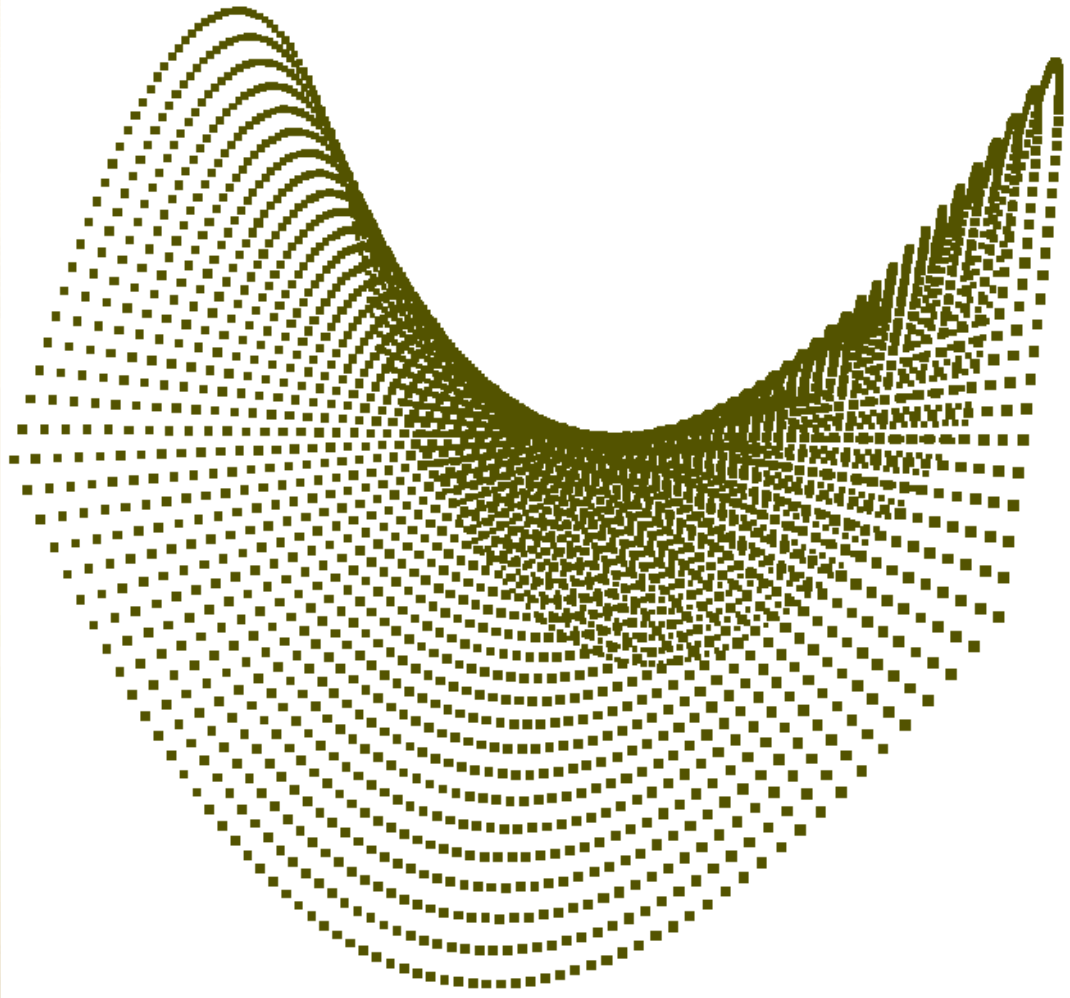
1. 3D eskizo sudarymas;
2. Taškų debesies generavimas (*.xyz formatas);
3. Taškų susiejimas sienomis (*.off formatas);
4. Sienų nuspalvinimas (*.off formatas).

3D eskizo sudarymas (Geogebra)



Surface(u, u² - v², v, u, -1, 1, v, -1, 1)

Taškų debesies generavimas (*.xyz formatas)

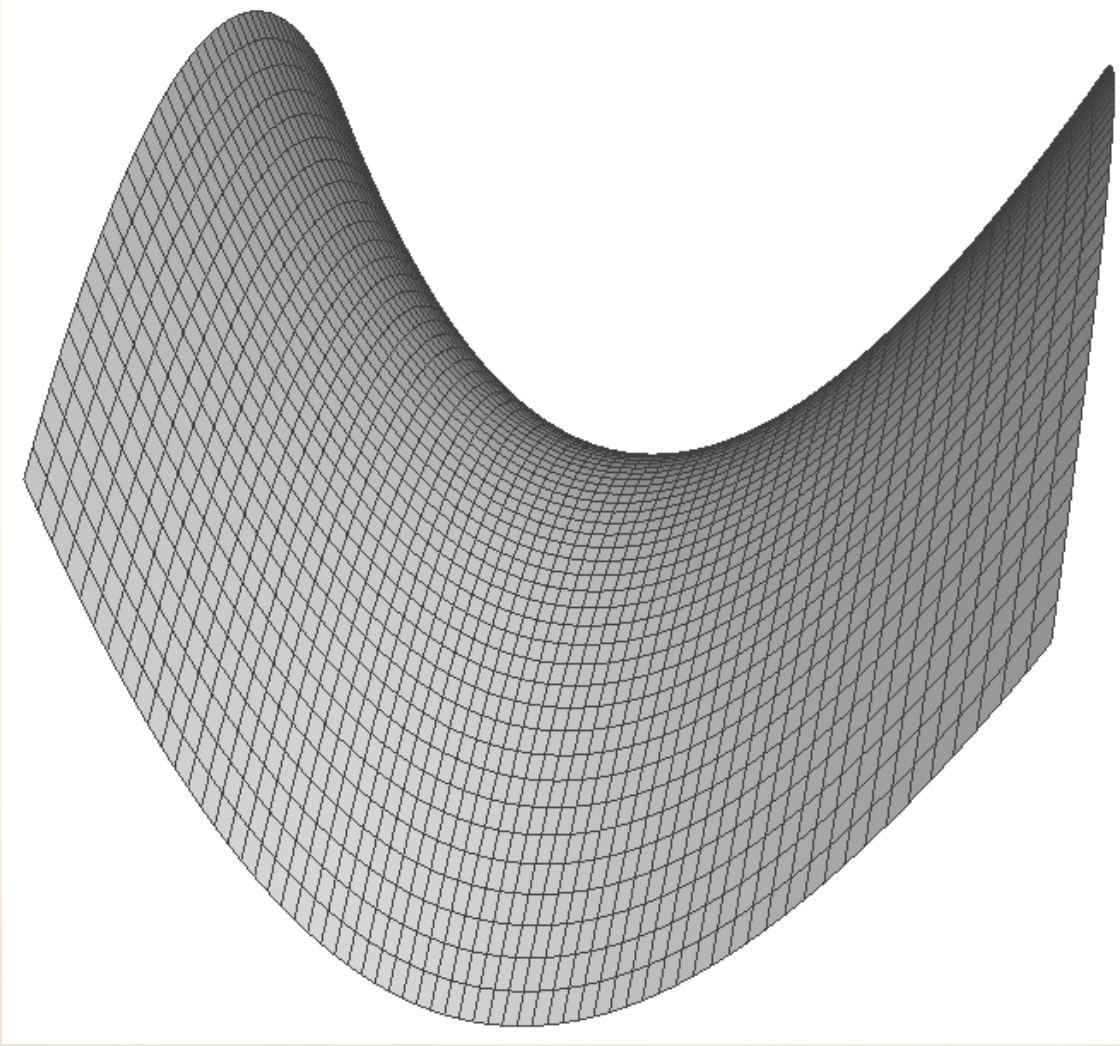


balnas123.xyz - Notepad

File Edit Format View Help

```
-1.000000 0.000000 -1.000000
-1.000000 0.061523 -0.968750
-1.000000 0.121094 -0.937500
-1.000000 0.178711 -0.906250
-1.000000 0.234375 -0.875000
-1.000000 0.288086 -0.843750
-1.000000 0.339844 -0.812500
-1.000000 0.389648 -0.781250
-1.000000 0.437500 -0.750000
-1.000000 0.483398 -0.718750
-1.000000 0.527344 -0.687500
-1.000000 0.569336 -0.656250
-1.000000 0.609375 -0.625000
-1.000000 0.647461 -0.593750
-1.000000 0.683594 -0.562500
-1.000000 0.717773 -0.531250
-1.000000 0.750000 -0.500000
-1.000000 0.780273 -0.468750
-1.000000 0.808594 -0.437500
-1.000000 0.834961 -0.406250
-1.000000 0.859375 -0.375000
-1.000000 0.881836 -0.343750
-1.000000 0.902344 -0.312500
```

Taškų susiejimas sienomis (*.off formatas)

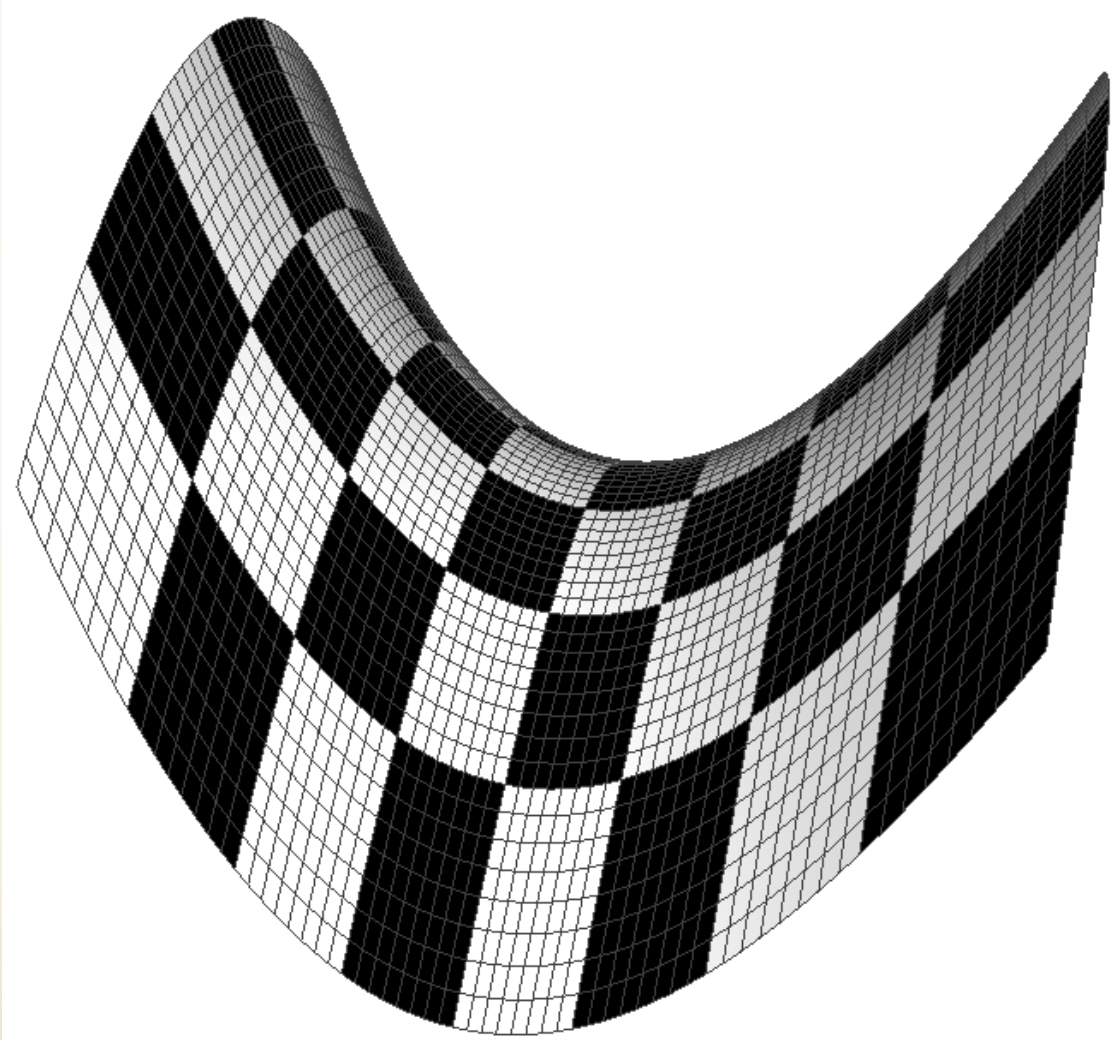


balnas123 - Notepad

File Edit Format View Help

```
OFF
4225 4096 12288
-1.000000 0.000000 -1.000000
-1.000000 0.061523 -0.968750
-1.000000 0.121094 -0.937500
-1.000000 0.178711 -0.906250
-1.000000 0.234375 -0.875000

<...>

1.000000 0.234375 0.875000
1.000000 0.178711 0.906250
1.000000 0.121094 0.937500
1.000000 0.061523 0.968750
1.000000 0.000000 1.000000
4 0 1 66 65
4 1 2 67 66
4 2 3 68 67
4 3 4 69 68
4 4 5 70 69
4 5 6 71 70
4 6 7 72 71
4 7 8 73 72
4 8 9 74 73

<...>
```

Sienų nuspalvinimas (*.off formatas)

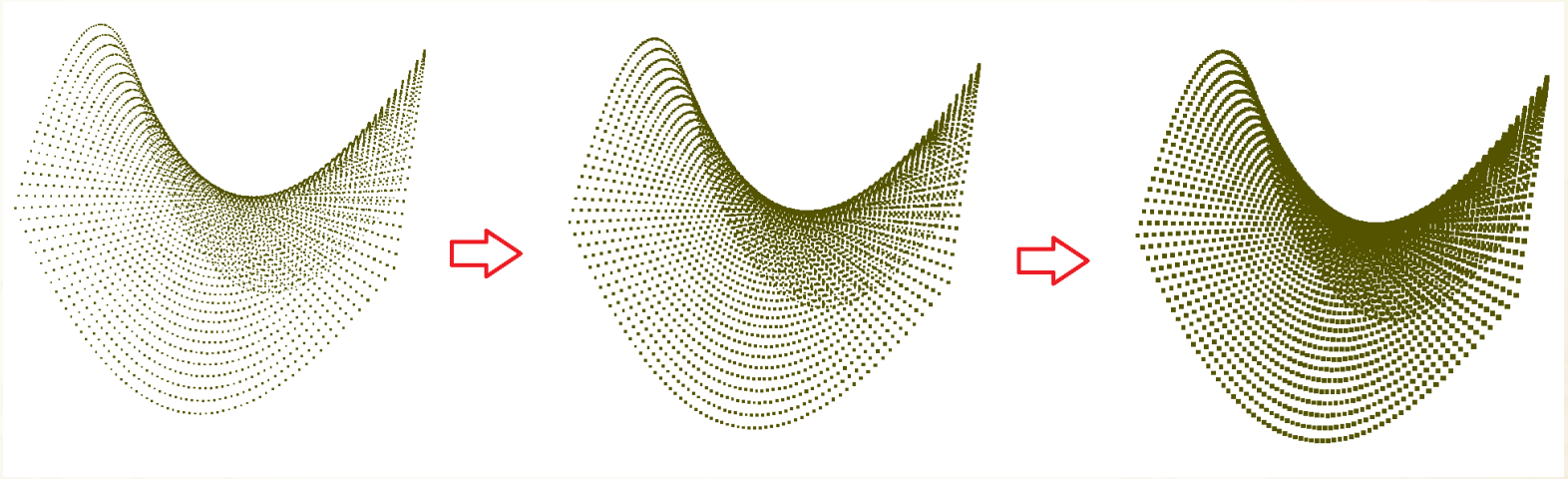


balnas123 - Notepad

File Edit Format View Help

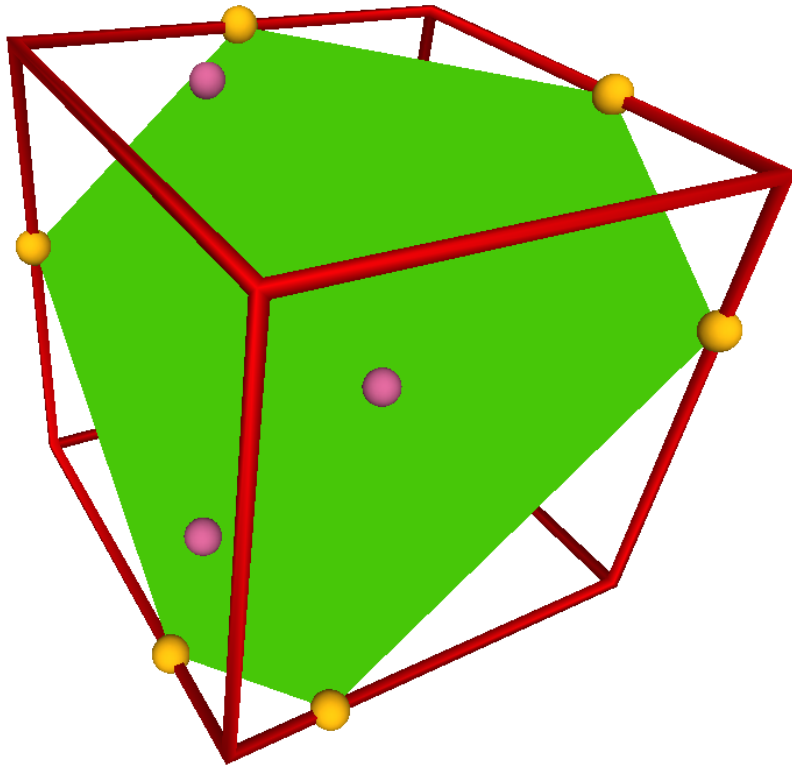
```
OFF
4225 4096 12288
-1.000000 0.000000 -1.000000
-1.000000 0.061523 -0.968750
-1.000000 0.121094 -0.937500
-1.000000 0.178711 -0.906250
-1.000000 0.234375 -0.875000

<...>

1.000000 0.234375 0.875000
1.000000 0.178711 0.906250
1.000000 0.121094 0.937500
1.000000 0.061523 0.968750
1.000000 0.000000 1.000000
4 0 1 66 65 0 0 0
4 1 2 67 66 0 0 0
4 2 3 68 67 0 0 0
4 3 4 69 68 0 0 0
4 4 5 70 69 0 0 0
4 5 6 71 70 0 0 0
4 6 7 72 71 0 0 0
4 7 8 73 72 0 0 0
4 8 9 74 73 255 255 255

<...>
```

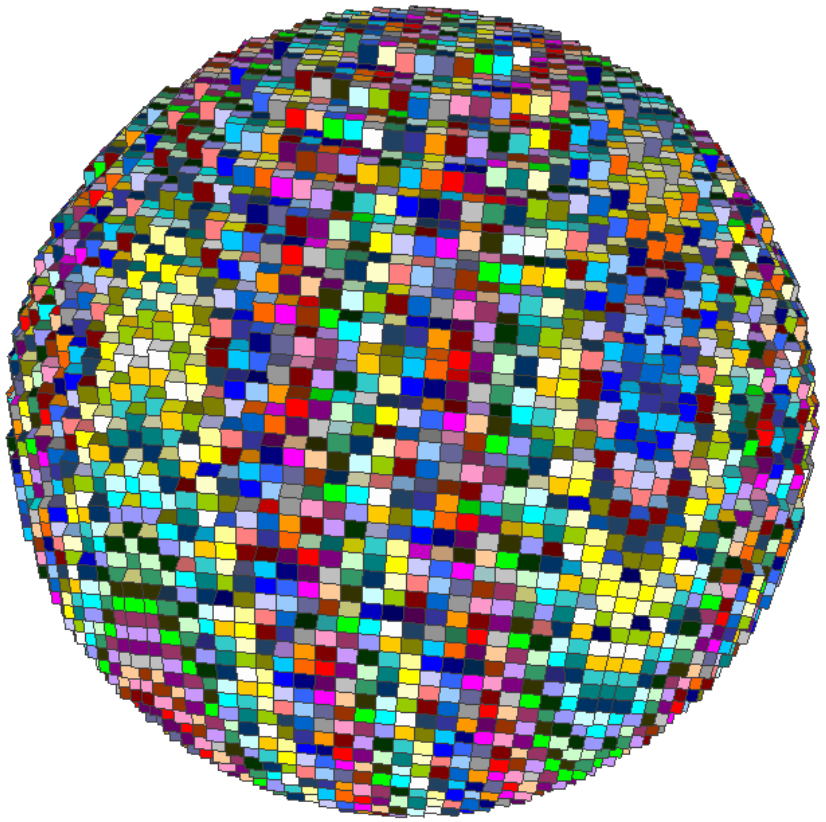
Taškų paryškinimas MeshLab programoje



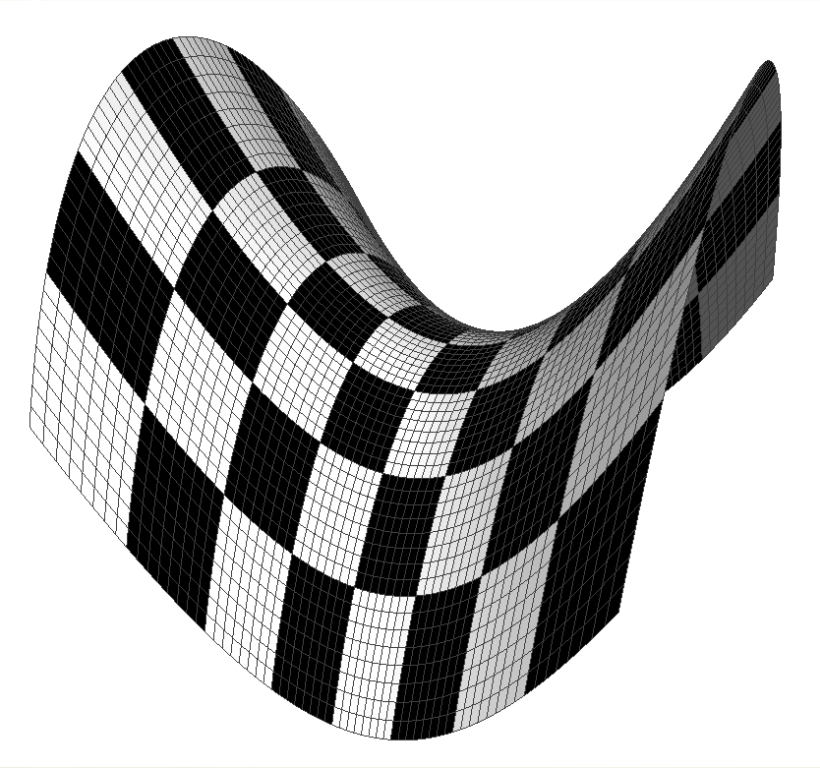
Alt + Mouse Wheel Up,
Alt + Mouse Wheel Down.

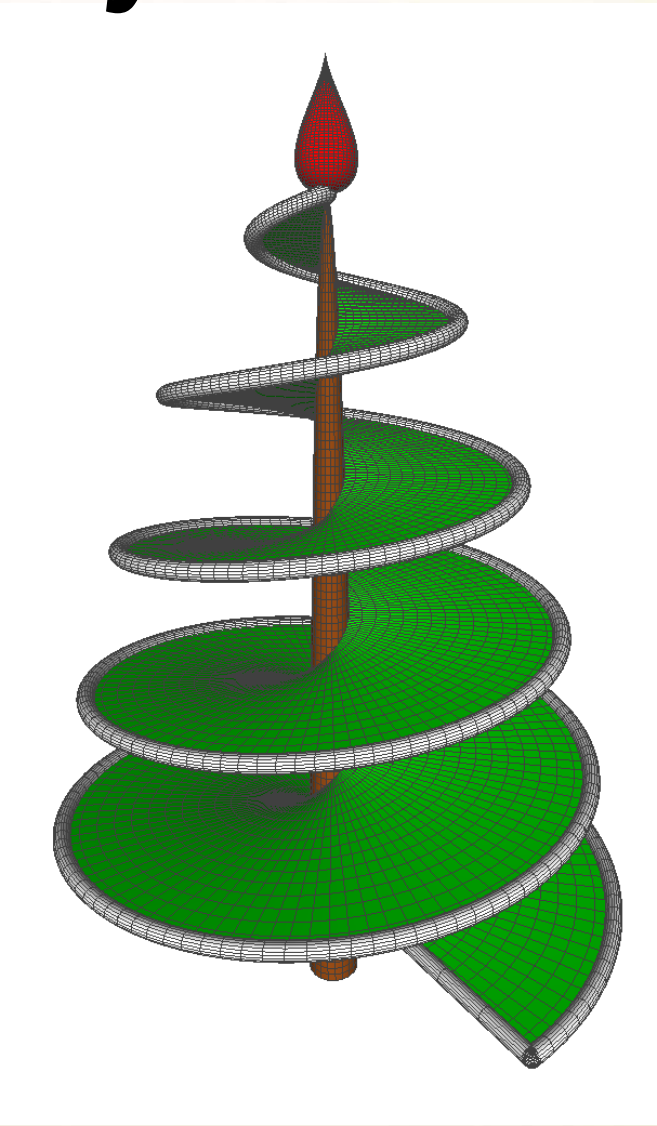
3D modelių pavyzdžiai





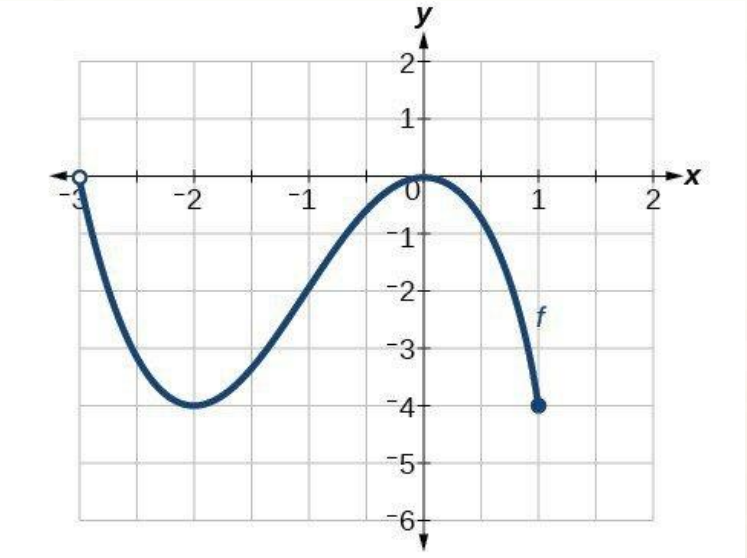
3D modelių pavyzdžiai





Kas yra funkcija?

Funkcija – tai taisyklė, pagal kurią kiekvienai jos apibrėžimo srities reikšmei priskiriama vienintelė reikšmė: f(x) = y.



Funkcija ir jos apibendrinimas

f(x) = y, pažymėkime:

x = t,
y = f(t).

Bendruoju atveju:

x = x(t),
y = y(t),
t ∈ [a, b].

t = time

Parametrinės kreivės sąvoka

- 2D parametrinės kreivės užrašymas:

$$x = x(t),$$
$$y = y(t).$$

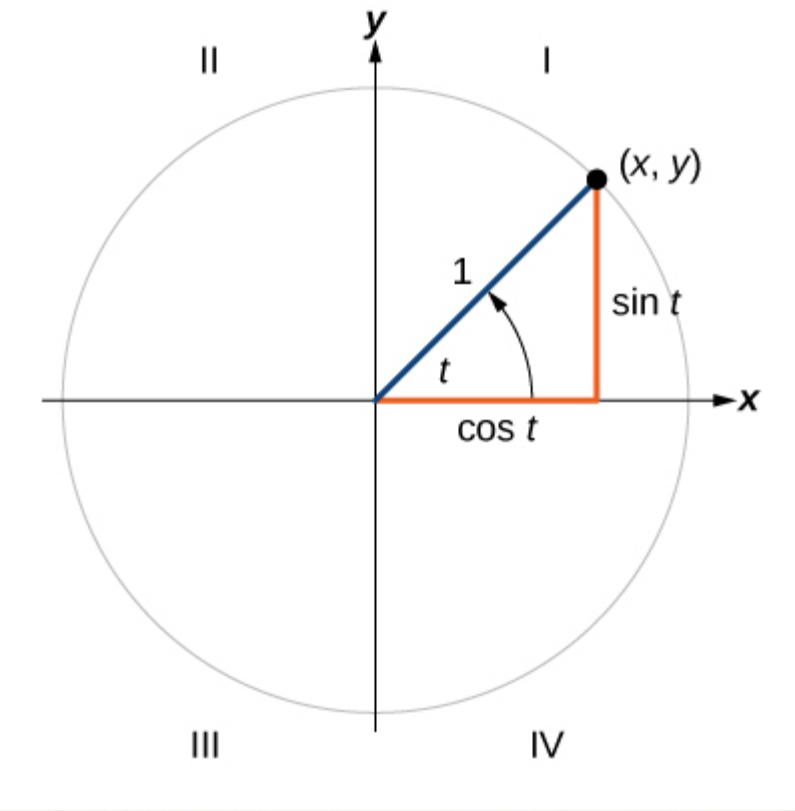
- 3D parametrinės kreivės užrašymas:

$$x = x(t),$$
$$y = y(t),$$
$$z = z(t).$$

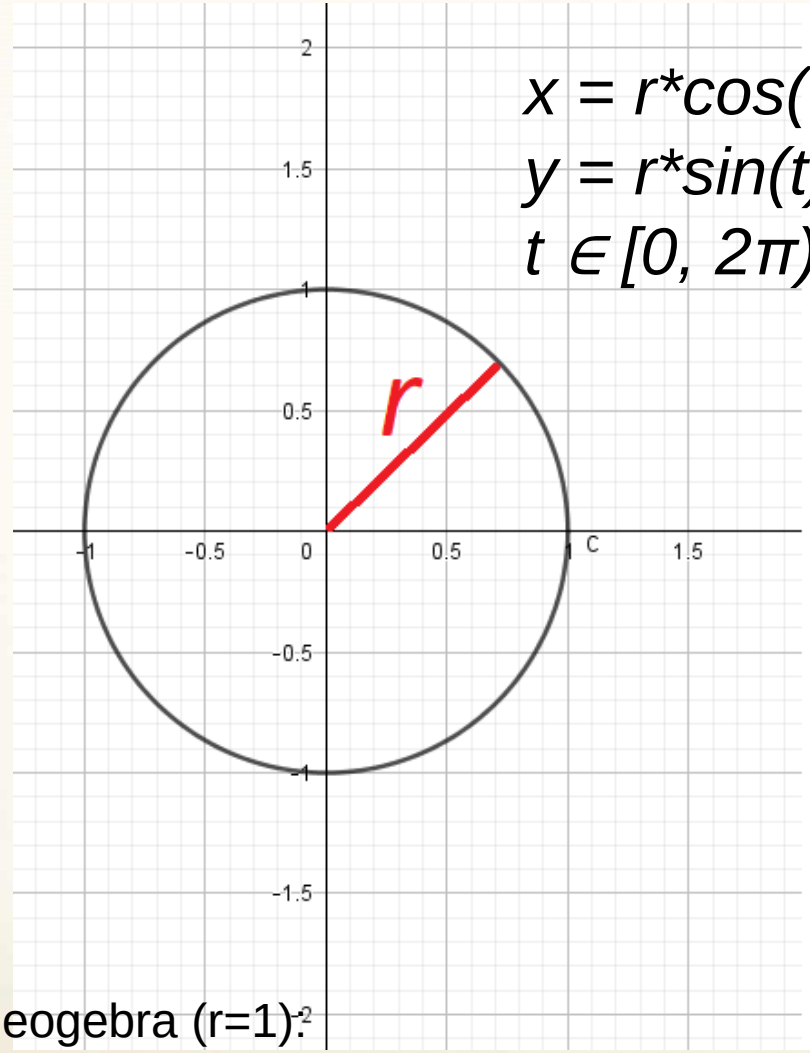
t – parametras, priklausantis intervalui $[t_{min}, t_{max}]$.

Sinuso ir kosinuso apibrėžimai

Parametrinės kreivės užuominos mokykloje

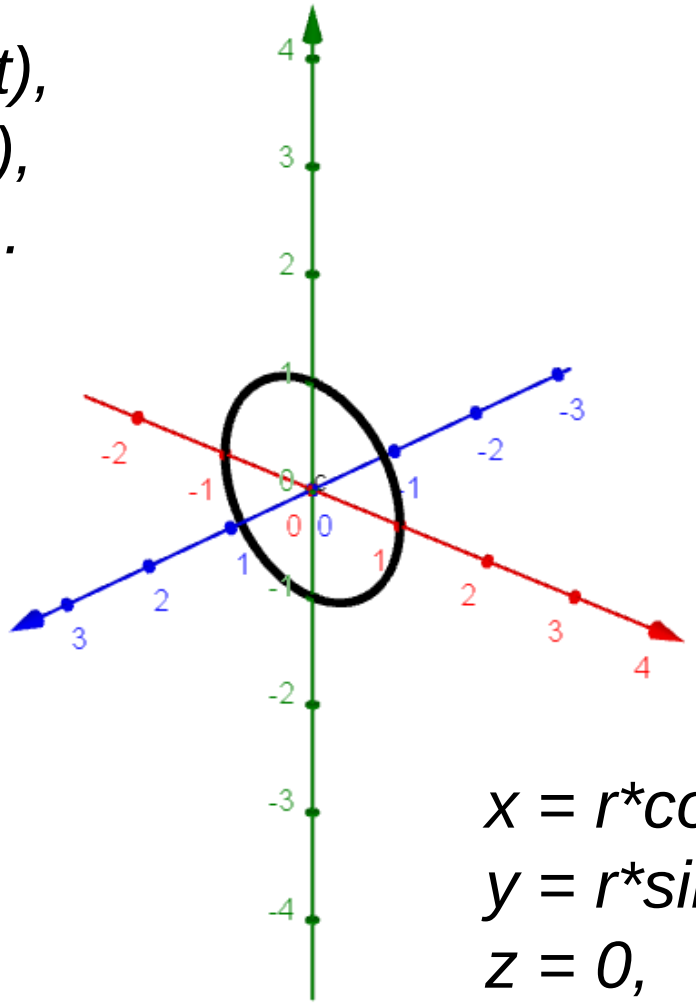


Apskritimo lygtis XY plokštumoje



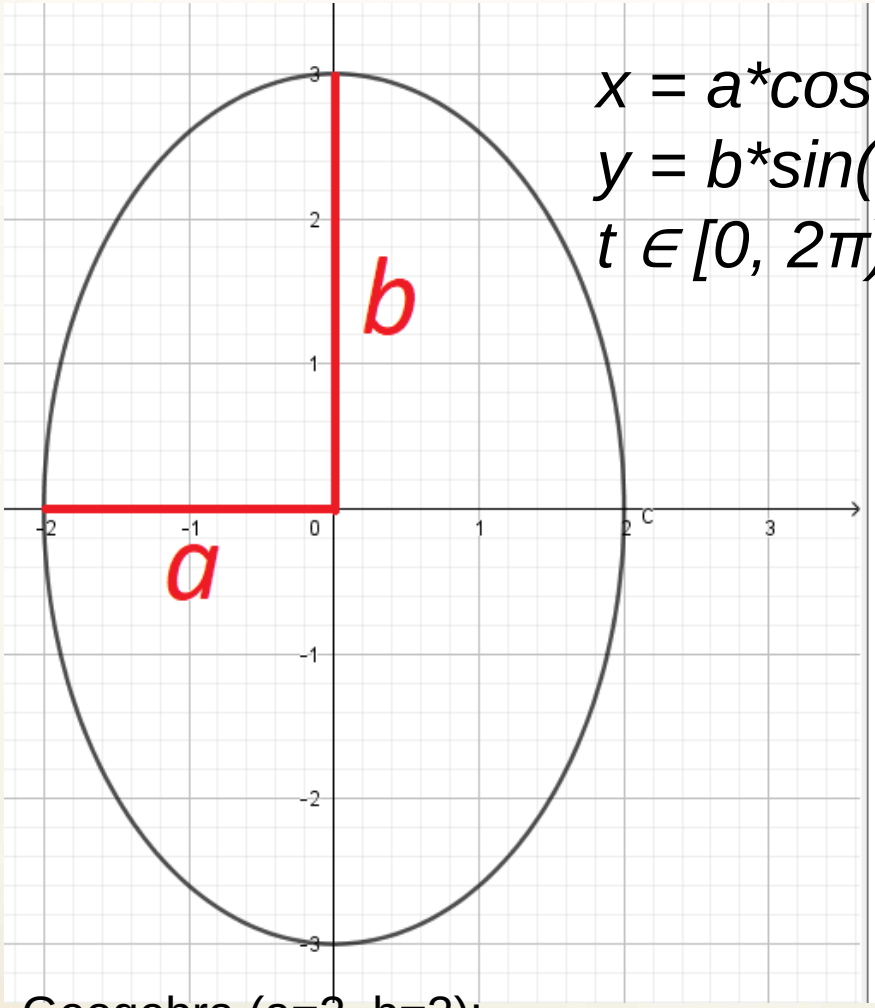
$x = r*\cos(t),$
$y = r*\sin(t),$
$t \in [0, 2\pi).$

Geogebra (r=1):
Curve(cos(t), sin(t), t, 0, 2π)

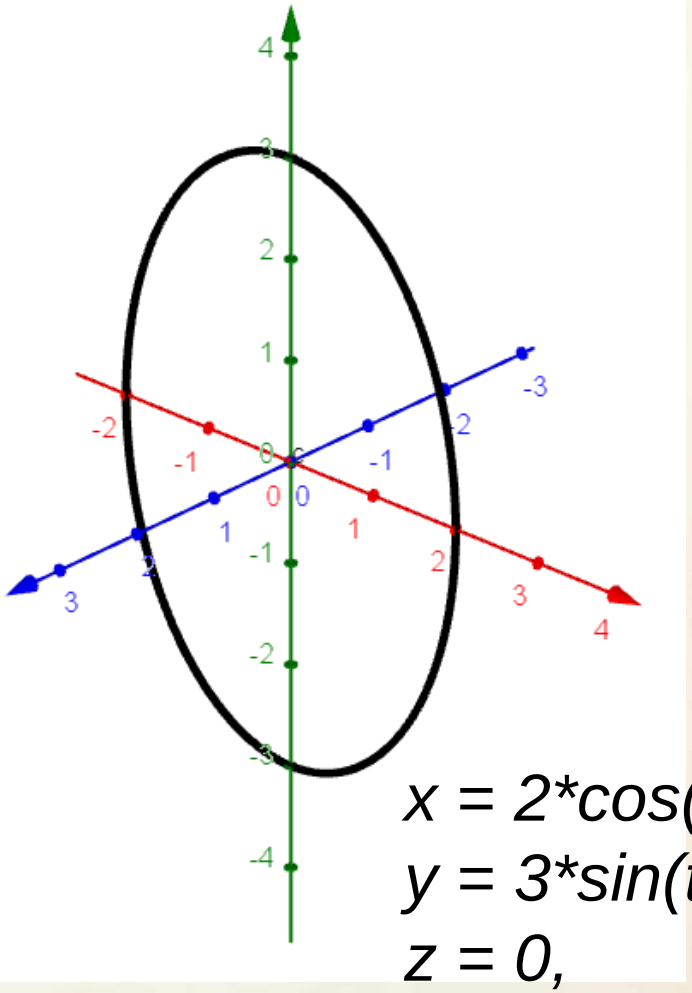


$x = r*\cos(t),$
$y = r*\sin(t),$
$z = 0,$
$t \in [0, 2\pi).$

Elipsės lygtis XY plokštumoje



$x = a*\cos(t),$
$y = b*\sin(t),$
$t \in [0, 2\pi).$

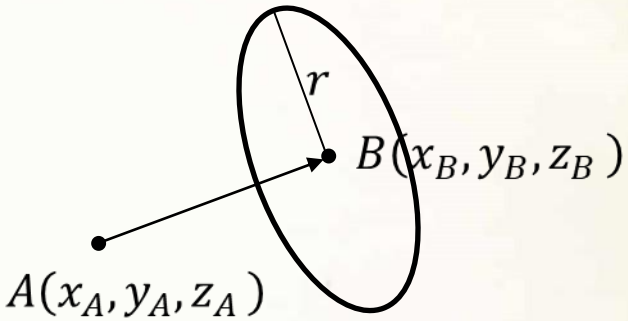


$x = 2*\cos(t),$
$y = 3*\sin(t),$
$z = 0,$
$t \in [0, 2\pi).$

Geogebra (a=2, b=3):
Curve(2 cos(t), 3 sin(t), t, 0, 2π)

Apskritimo lygtis erdvėje

http://demonstrations.wolfram.com/ParametricEquationOfACircleIn3D/



Apskritimo (kurio spindulys r ir centrinis taškas B), ortogonalaus vektoriui AB, parametrinė lygtis:

$$x(t) = x_B - \frac{r\cos(t)(y_B - y_A)\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2} + r\sin(t)(x_B - x_A)(z_B - z_A)}{\sqrt{\left((x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2\right)\left((x_B - x_A)^2 + (y_B - y_A)^2\right)}}$$

$$y(t) = y_B + \frac{r\cos(t)(x_B - x_A)\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2} - r\sin(t)(z_B - z_A)(y_B - y_A)}{\sqrt{\left((x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2\right)\left((x_B - x_A)^2 + (y_B - y_A)^2\right)}}$$

$$z(t) = z_B + \frac{r\sin(t)\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}}{\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2 + (z_B - z_A)^2}}, \qquad t \in [0, 2\pi)$$

$$\operatorname{sgn}(x) = \begin{cases} -1, x < 0 \\ 0, x = 0 \\ 1, x > 1 \end{cases}$$

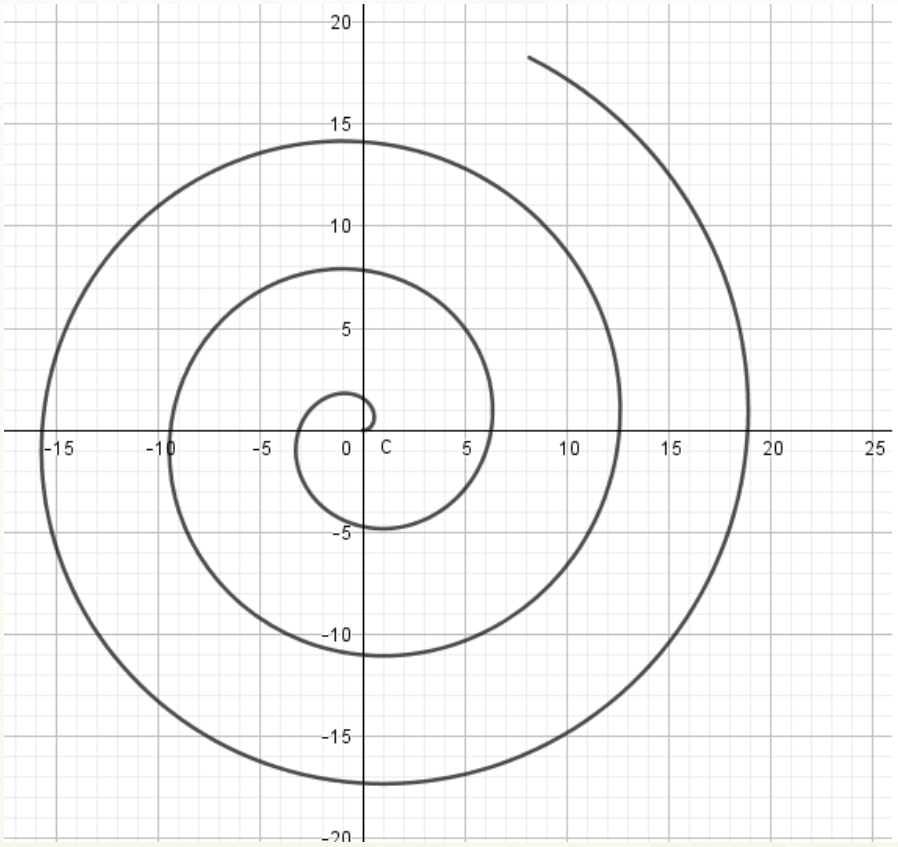
Jei $x_A = x_B$ ir $y_A = y_B$, tai:

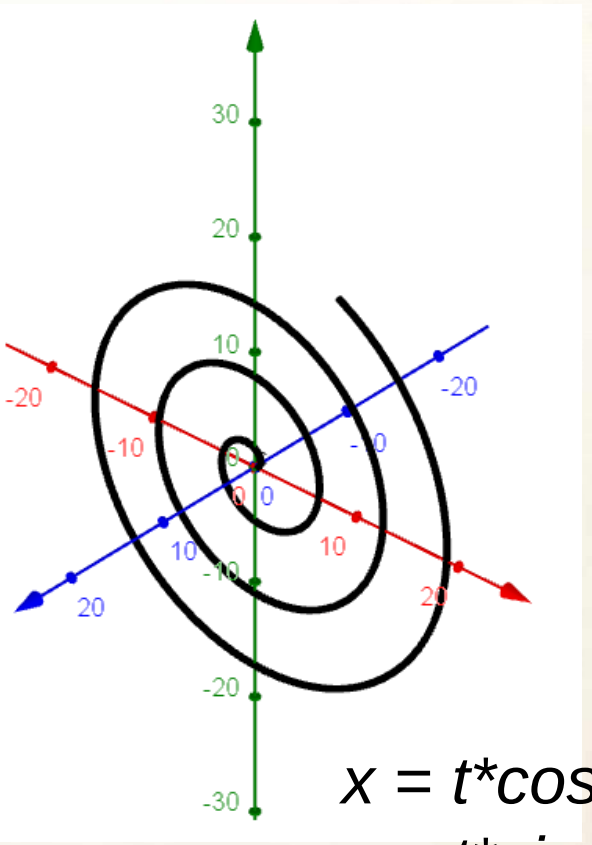
$$x(t) = x_B - \frac{r\sqrt{2}}{2}(\operatorname{sgn}(z_B - z_A)\sin(t) + \cos(t))$$

$$y(t) = y_B - \frac{r\sqrt{2}}{2}(\operatorname{sgn}(z_B - z_A)\sin(t) - \cos(t))$$

$$z(t) = z_B, \qquad t \in [0, 2\pi)$$

Kitų parametrinių kreivių pavyzdžiai



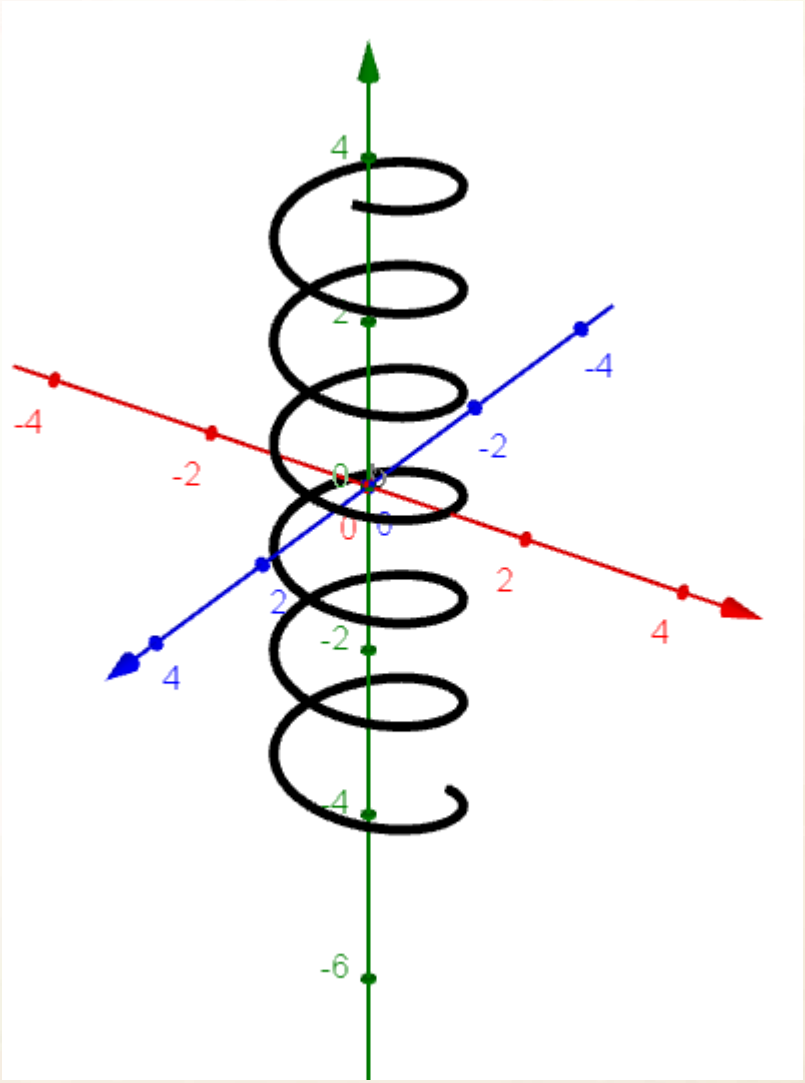


Geogebra:
Curve(t cos(t), t sin(t), t, 0, 20)

$x = t*\cos(t),$
$y = t*\sin(t),$
$z = 0,$
$t \in [0, 20].$

Kitų parametrinių kreivių pavyzdžiai

$x = cos(t)$,
$y = t / 5$,
$z = sin(t)$,
$t \in [-20, 20]$.

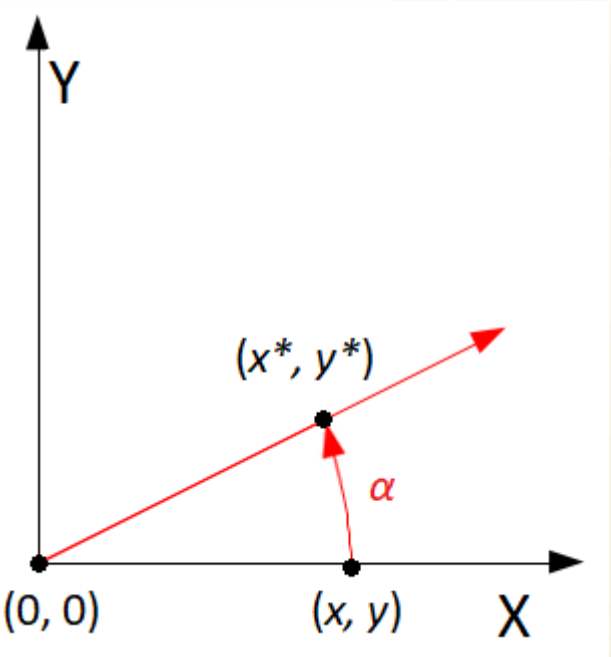


Geogebra:
Curve(cos(t), t / 5, sin(t), t, -20, 20)

Koordinačių sistemos pasukimas

Taško (x, y) naujos koordinatės (x^*, y^*) po posūkio kampu α:

$$x^* = x\cos(\alpha) - y\sin(\alpha)$$
$$y^* = x\sin(\alpha) + y\cos(\alpha)$$



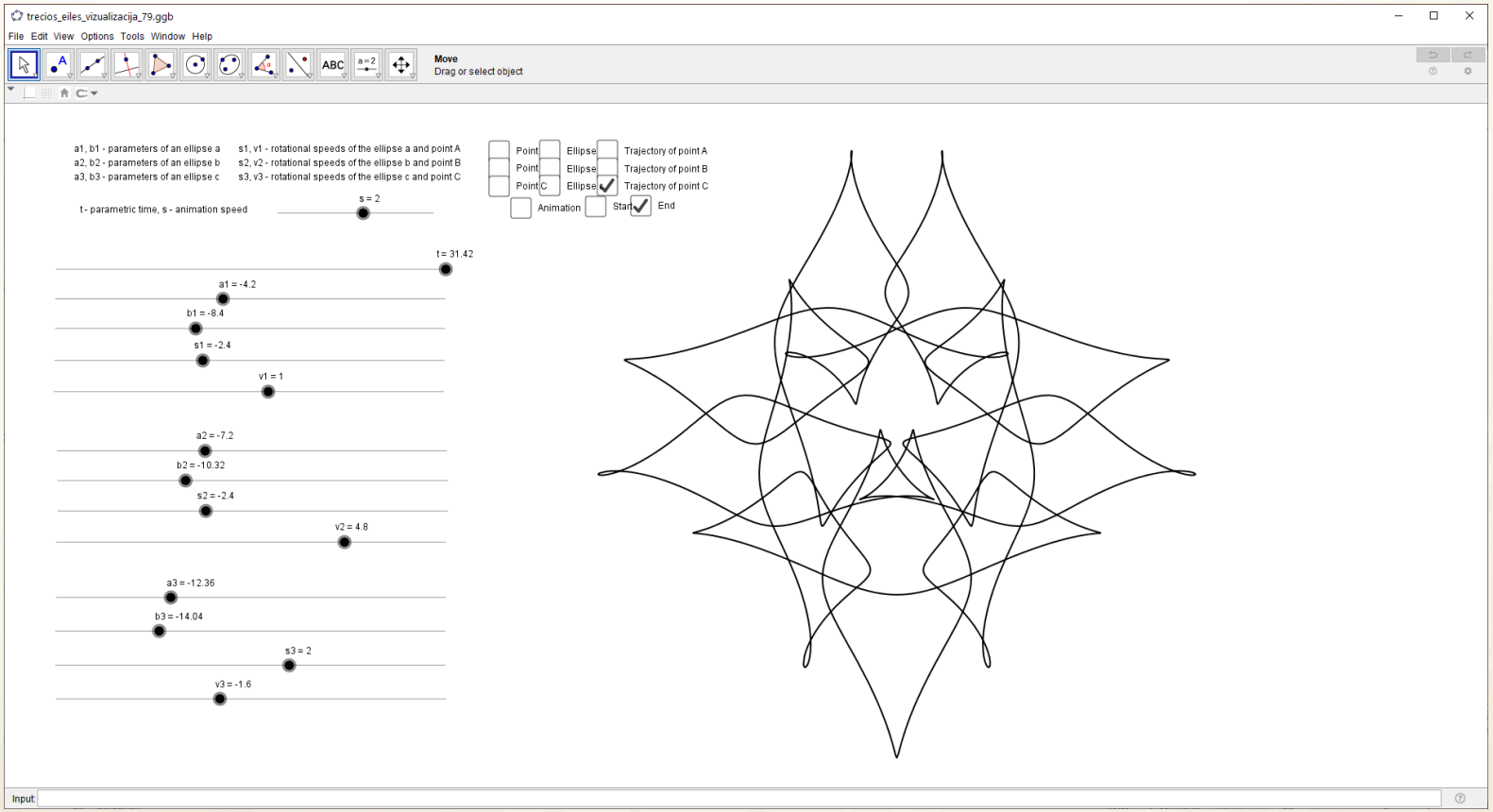
Užduotis: išveskime funkcijos f(x) = x^2 parametrinę kreivę ir pasukime ją kampu α. Pasitikrinkime gautą rezultatą su GeoGebra programa.

Elipsinės parametrinės 2D kreivės

$$x(t) = \sin(s_1\,t)\,b_1\cos(v_1\,t) + \cos(s_1\,t)\,a_1\sin(v_1\,t) + \sin(s_2\,t)\,b_2\cos(v_2\,t) + \cos(s_2\,t)\,a_2\sin(v_2\,t) + \sin(s_3\,t)\,b_3\cos(v_3\,t) + \cos(s_3\,t)\,a_3\sin(v_3\,t)$$

$$y(t) = -a_1\sin(v_1\,t)\sin(s_1\,t) + b_1\cos(v_1\,t)\cos(s_1\,t) - a_2\sin(v_2\,t)\sin(s_2\,t) + b_2\cos(v_2\,t)\cos(s_2\,t) - a_3\sin(v_3\,t)\sin(s_3\,t) + b_3\cos(v_3\,t)\cos(s_3\,t)$$

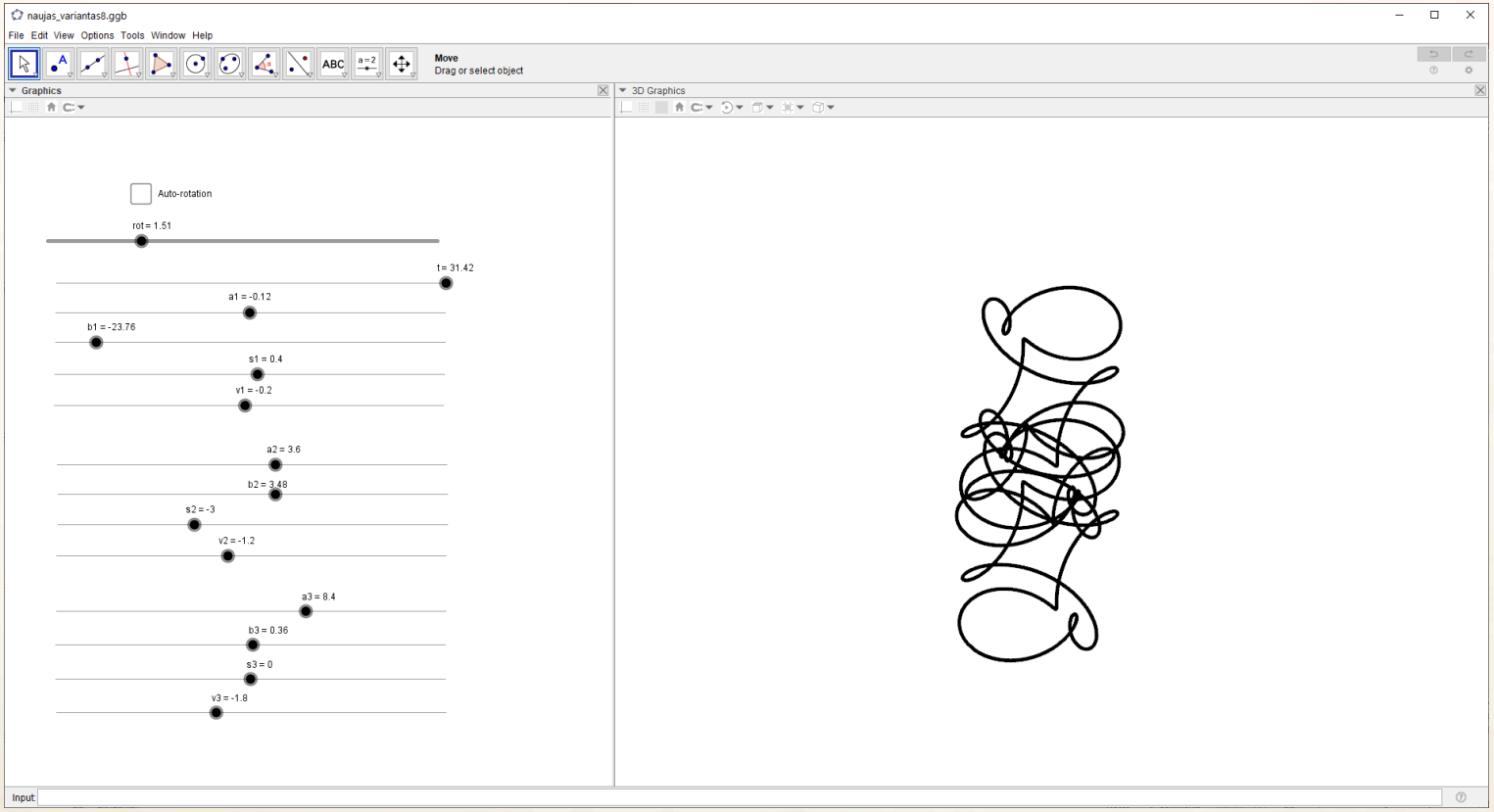
$$t \in [0,\ 10\,\pi]$$



https://www.geogebra.org/m/j6d6ws6e

Elipsinės parametrinės 3D kreivės

$$x(t) = \sin(s_3\,t)\,b_3\cos(v_3\,t) + \cos(s_3\,t)\,a_3\sin(v_3\,t) - a_2\sin(v_2\,t)\sin(s_2\,t) + b_2\cos(v_2\,t)\cos(s_2\,t)$$
$$y(t) = \sin(s_1\,t)\,b_1\cos(v_1\,t) + \cos(s_1\,t)\,a_1\sin(v_1\,t) - a_3\sin(v_3\,t)\sin(s_3\,t) + b_3\cos(v_3\,t)\cos(s_3\,t)$$
$$z(t) = \sin(s_2\,t)\,b_2\cos(v_2\,t) + \cos(s_2\,t)\,a_2\sin(v_2\,t) - a_1\sin(v_1\,t)\sin(s_1\,t) + b_1\cos(v_1\,t)\cos(s_1\,t)$$
$$t \in [0,\ 10\,\pi]$$



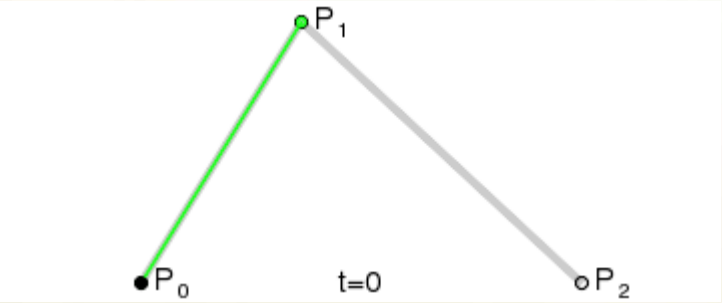
https://www.geogebra.org/m/bemyhdzw

Bezjė kreivės

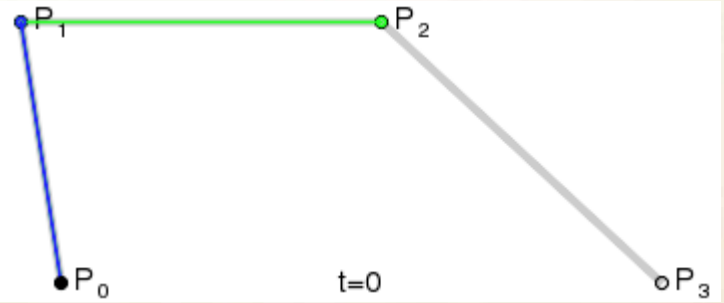
$$\mathbf{B}(t) = \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i$$

P_i – kontroliniai taškai, n – kreivės eilė.

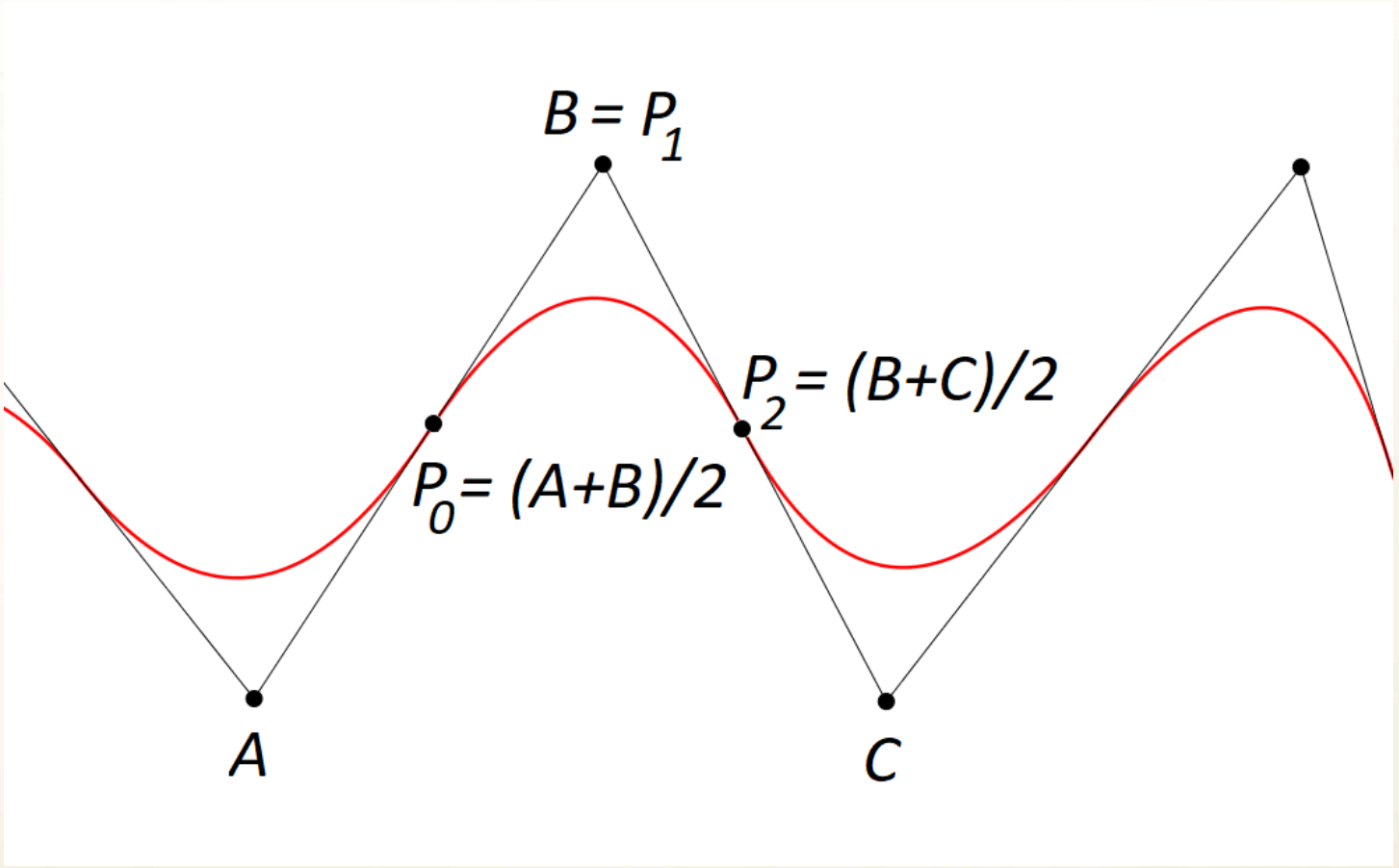
n = 2:



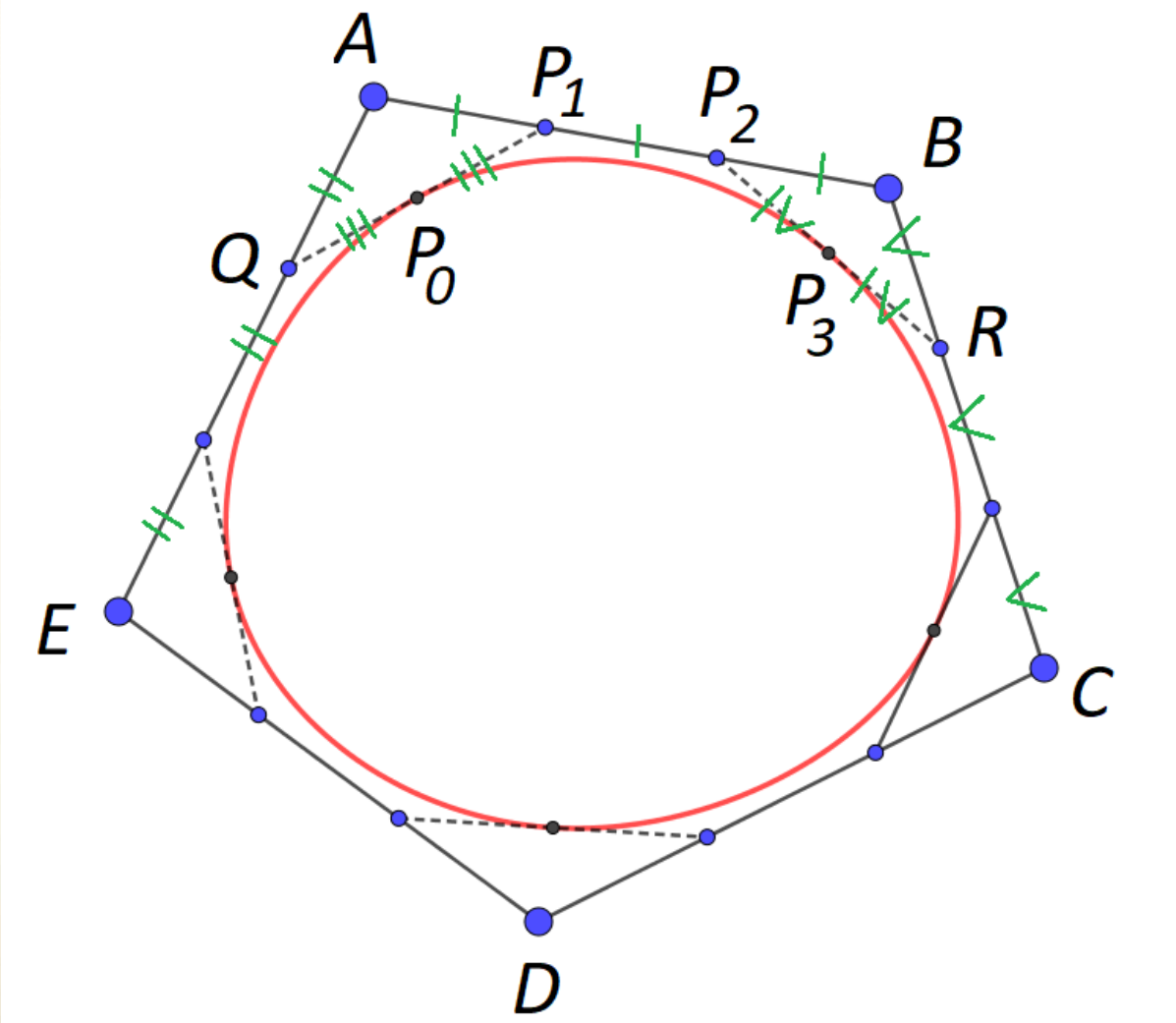
n = 3:



2 eilės Bezjė kreivių kontūras

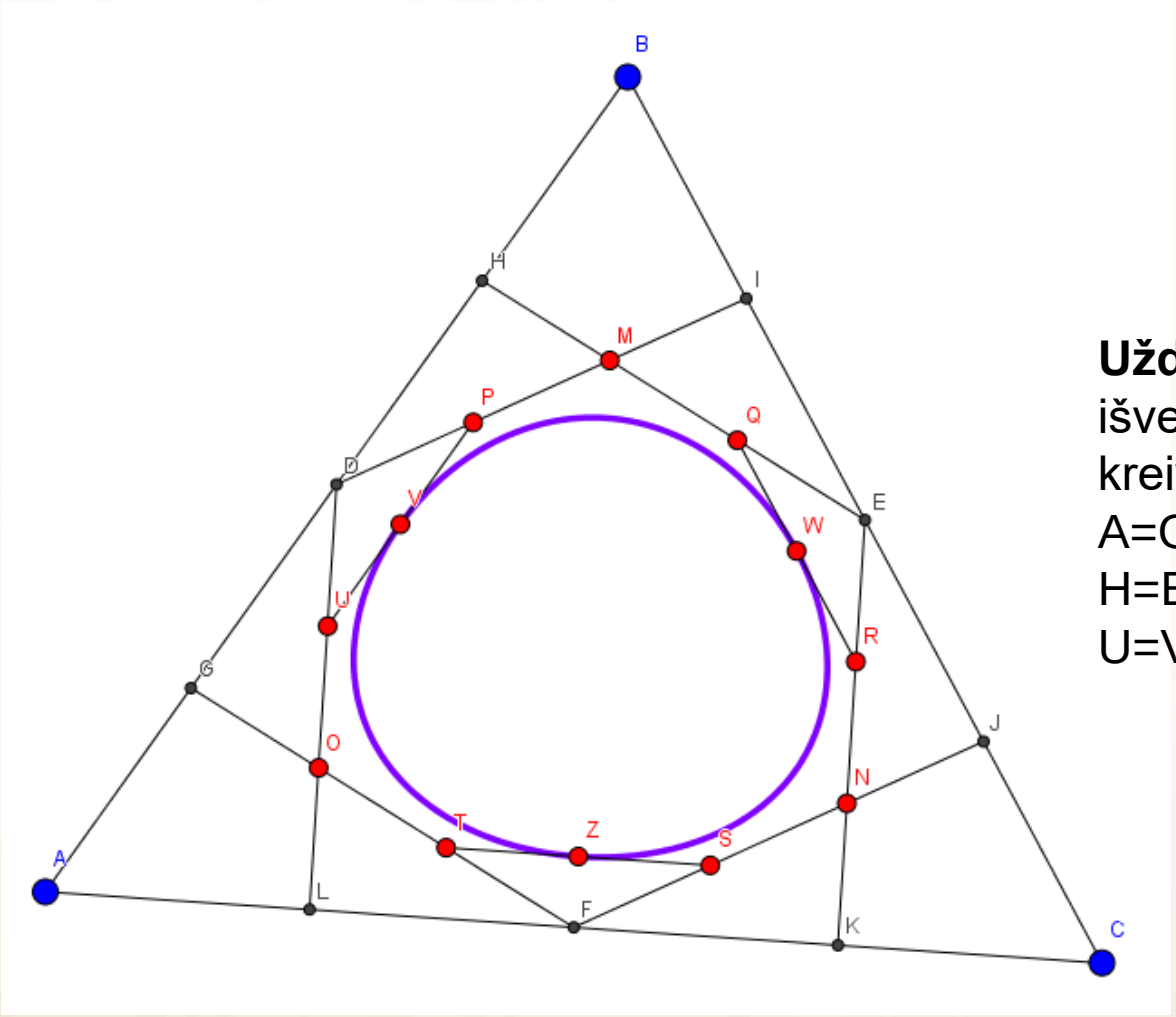


3 eilės Bezjė kreivių kontūras



$P_1=2/3*A+1/3*B,$
$P_2=1/3*A+2/3*B,$
$P_0=(Q+P_1)/2,$
$P_3=(P_2+R)/2.$

4 eilės Bezjė kreivių kontūras



Užduotis: pagal brėžinį išvesti 4 eilės Bezjė kreivių lygtis, kai A=G, G=D, D=H, H=B, D=P, P=M, U=V, V=P ir t. t.

Parametriniai paviršiai

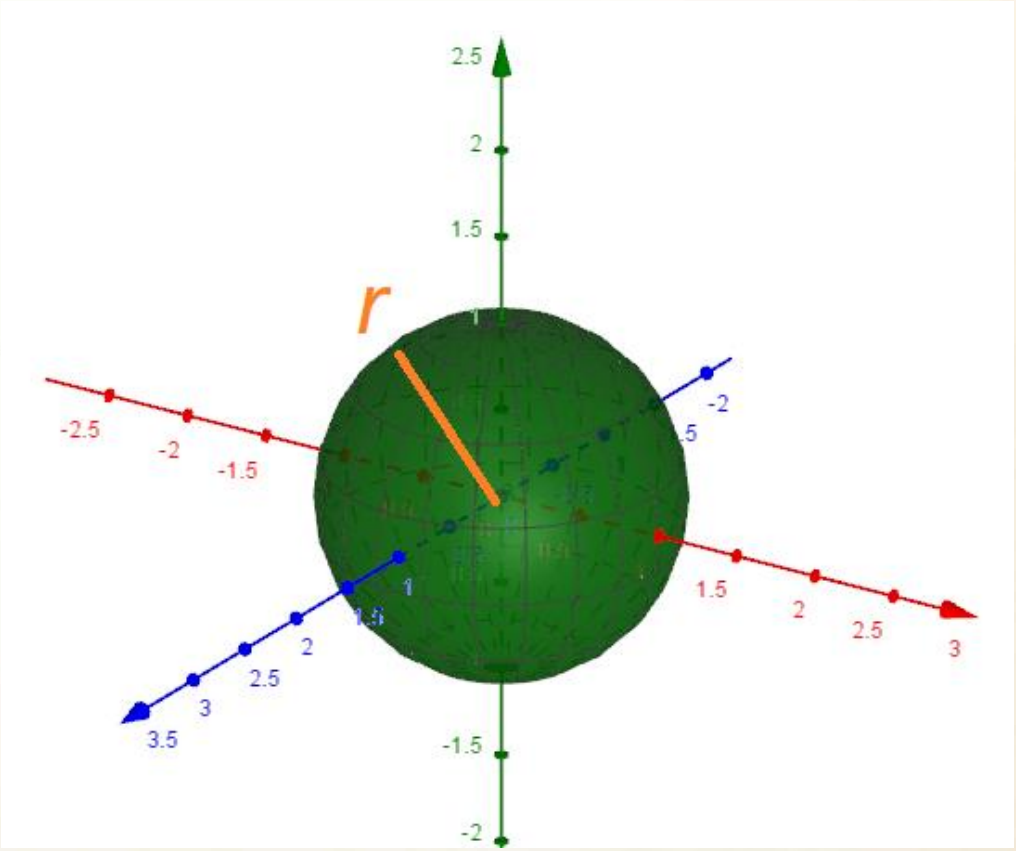
- Parametrinio paviršiaus užrašymas:

$$x = x(u,v),$$
$$y = y(u,v),$$
$$z = z(u,v).$$

u – parametras, priklausantis intervalui $[u_{min}, u_{max}]$,
v – parametras, priklausantis intervalui $[v_{min}, v_{max}]$.

Sfera

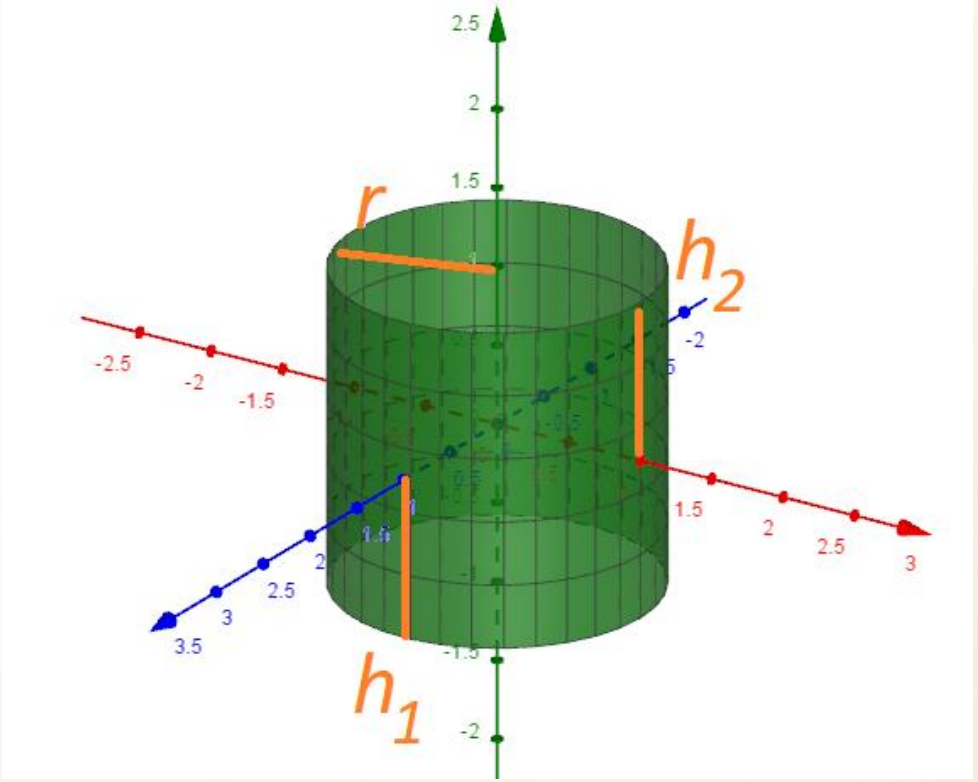
Parametrinis sferos paviršius:
*x=r*cos(u)*sin(v),*
*y=r*cos(v),*
*z=r*sin(u)*sin(v),*
u∈[0, 2π), v∈[0, π).



Geogebra (r=1):
Surface(cos(u) sin(v), cos(v), sin(u) sin(v), u, 0, 2π, v, 0, π)

Cilindras

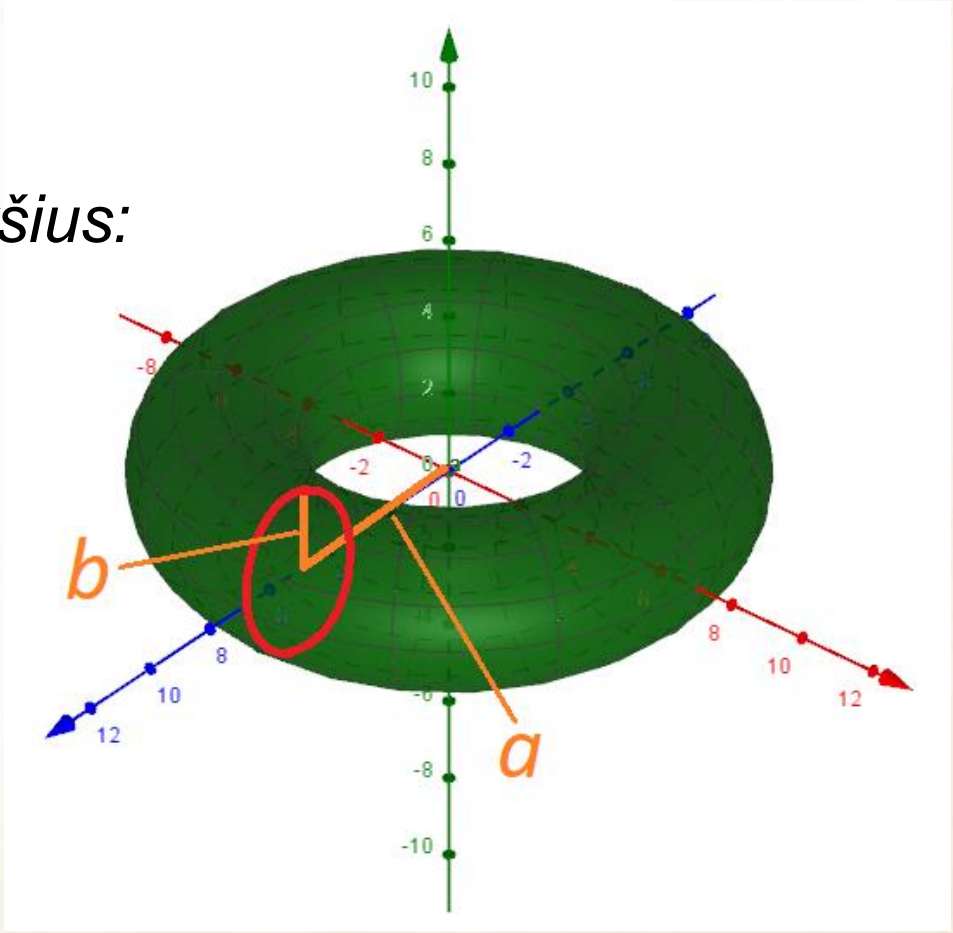
Parametrinis cilindro paviršius:
$x=r*\cos(u)$,
$y=v$,
$z=r*\sin(u)$,
$u \in [0,2\pi), v \in [h_1,h_2]$.



Geogebra ($r=1$, $h_1=-1$, $h_2=1$):
Surface(cos(u), v, sin(u), u, 0, 2π, v, -1, 1)

Toras

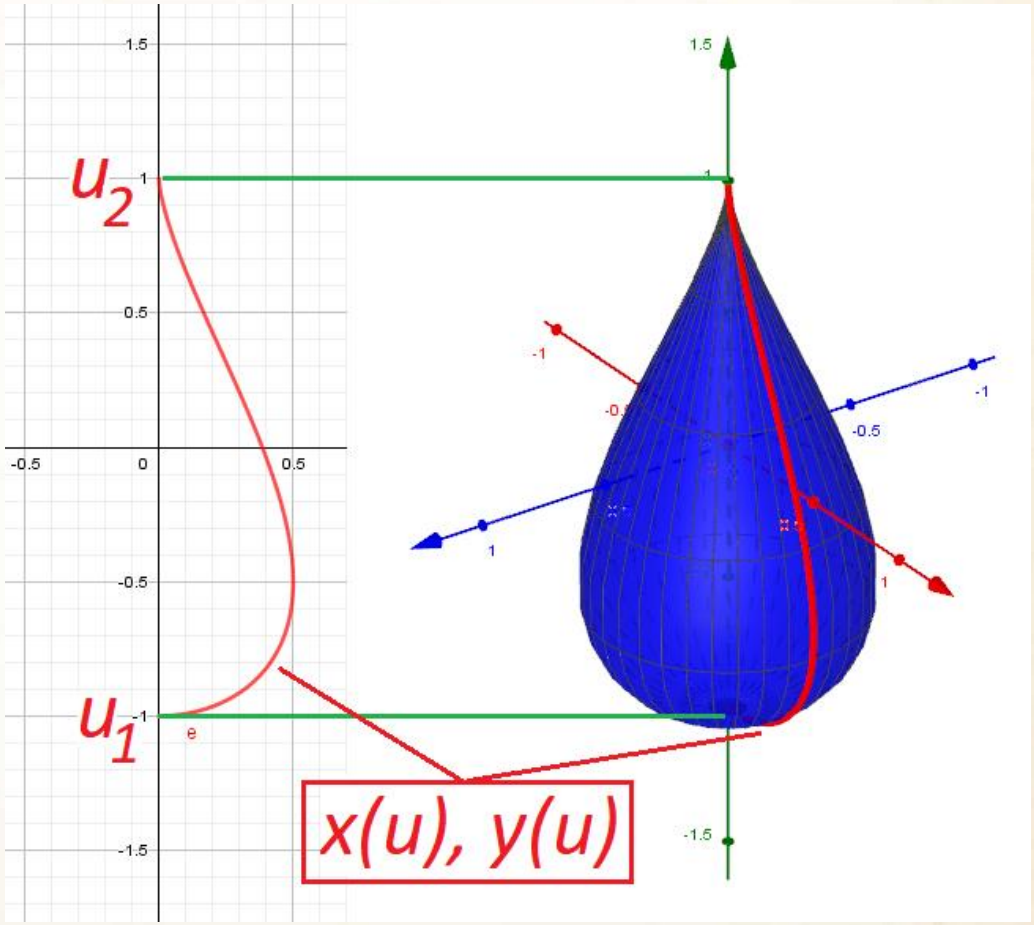
Parametrinis toro paviršius:
*x=(a+b*cos(u))*cos(v),*
*y=b*sin(u),*
*z=(a+b*cos(u))*sin(v),*
u∈[0, 2π), v∈[0, 2π).



Geogebra (a=5, b=2):
Surface((5 + 2cos(u)) cos(v),
2sin(u), (5 + 2cos(u)) sin(v),
u, 0, 2π, v, 0, 2π)

Sukinys

Parametrinis sukinio paviršius:
$x=x(u)*\cos(v)$,
$y=y(u)$,
$z=x(u)*\sin(v)$,
$u \in [u_1, u_2)$, $v \in [0, 2\pi)$.



Geogebra (x(u)=sqrt(1 - u²) (1 - u) / (3 / 2 sqrt(3)), y(u)=u, u_1=-1, u_2=1):
Surface(sqrt(1 - u²) (1 - u) / (3 / 2 sqrt(3)) cos(v), u, sqrt(1 - u²) (1 - u) / (3 / 2 sqrt(3)) sin(v), u, -1, 1, v, 0, 2π), Curve(sqrt(1 - u²) (1 - u) / (3 / 2 sqrt(3)), u, u, -1, 1)

Bezjė paviršiai

$$p(u, v) = \sum_{i=0}^{n}\sum_{j=0}^{m}\binom{n}{i}u^{i}(1-u)^{n-i}\binom{m}{j}v^{j}(1-v)^{m-j}K_{i,j}$$

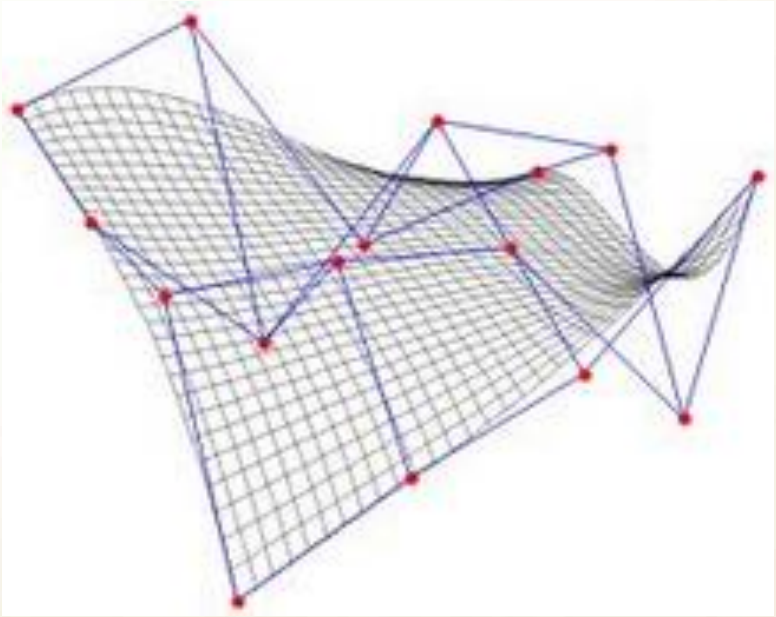
$$u, v \in [0, 1]$$

Pavyzdys, kai *n=m=3.*

Raudona spalva pažymėti kontroliniai taškai.

Mėlyna spalva pažymėtas kontrolinis tinklas.

Juoda spalva pažymėta Bezjė paviršiaus aproksimacija.



Trečios eilės Bezjė paviršius

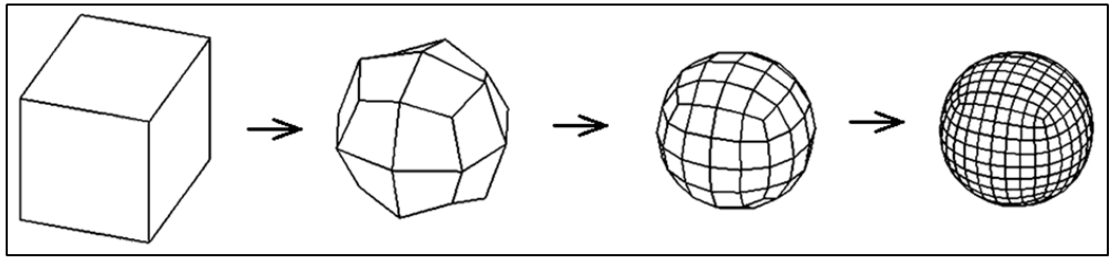
Tegu $B(u,v)$ – trečios eilės Bezjė paviršius, kurio parametrinė forma

$$B(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} \binom{3}{i} u^i (1-u)^{n-i} \binom{3}{j} v^j (1-v)^{m-j} K_{i,j}.$$

Apskaičiavus binominius koeficientus, paviršiaus parametrinę lygtį galima suvesti į matricinę forma:

$$B(u,v) = (1 \;\; u \;\; u^2 \;\; u^3) \begin{pmatrix} 1 & 0 & 0 & 0 \\ -3 & 3 & 0 & 0 \\ 3 & -6 & 3 & 0 \\ -1 & 3 & -3 & 1 \end{pmatrix} \begin{pmatrix} K_{0,0} & K_{0,1} & K_{0,2} & K_{0,3} \\ K_{1,0} & K_{1,1} & K_{1,2} & K_{1,3} \\ K_{2,0} & K_{2,1} & K_{2,2} & K_{2,3} \\ K_{3,0} & K_{3,1} & K_{3,2} & K_{3,3} \end{pmatrix} \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ v \\ v^2 \\ v^3 \end{pmatrix}.$$

$K_{i,j}$ – kontroliniai taškai, pagal kurių išsidėstymą suformuojamas Bezjė paviršius (arba C2 paviršiaus fragmentas).



Catmull-Clark algoritmo 3 iteracijų vizualizacija.

Catmullo ir Clarko algoritmas (1978 m.)

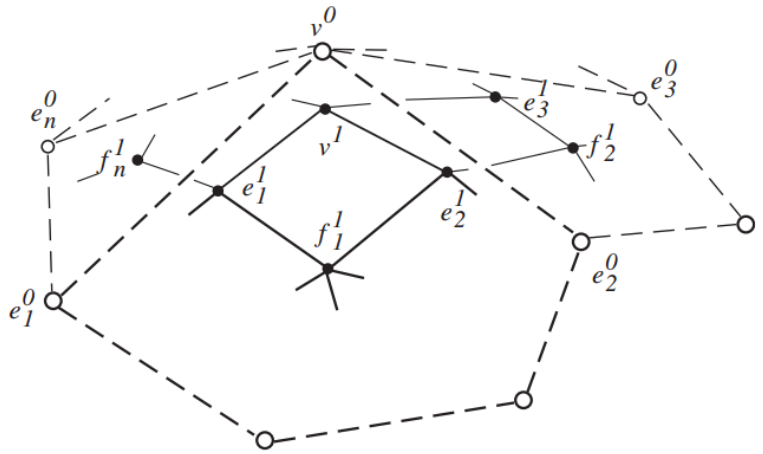
Subdivision Rules

The Catmull-Clark smooth subdivision rules for face, edge, and vertex points, are defined as:

—Faces rule: f^{i+1} is the centroid of the vertices surrounding the face.

—Edge rule: $e_j^{i+1} = \frac{1}{4}(v^i + e_j^i + f_{j-1}^{i+1} + f_j^{i+1})$,

—Vertex rule: $v^{i+1} = \frac{n-2}{n} v^i + \frac{1}{n^2} \sum_j e_j^i + \frac{1}{n^2} \sum_j f_j^{i+1}$.

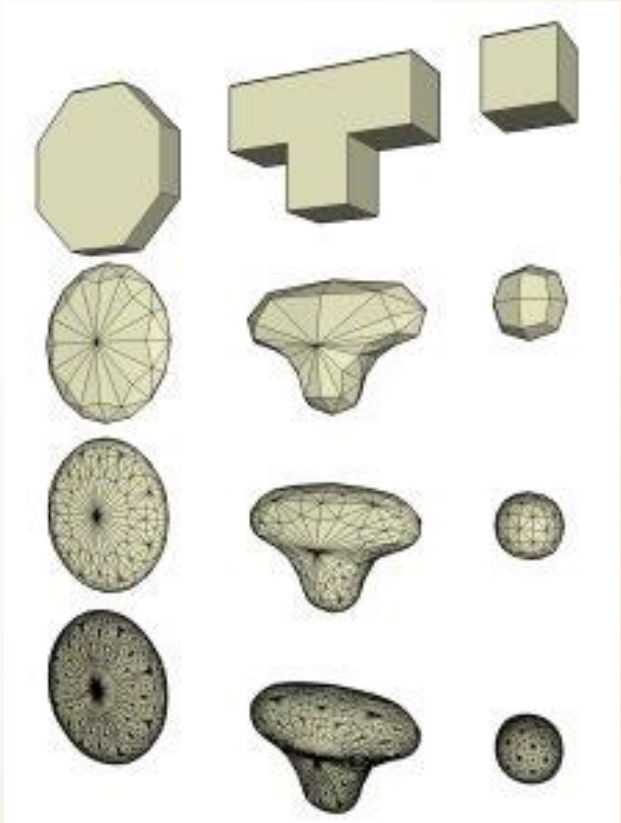


The following *sharp rules* are used for both boundaries and sharp edges (crease rules):

—$e_j^{i+1} = \frac{1}{2}(v^i + e_j^i)$

—$v_j^{i+1} = \frac{1}{8}(e_j^i + 6v^i + e_k^i)$

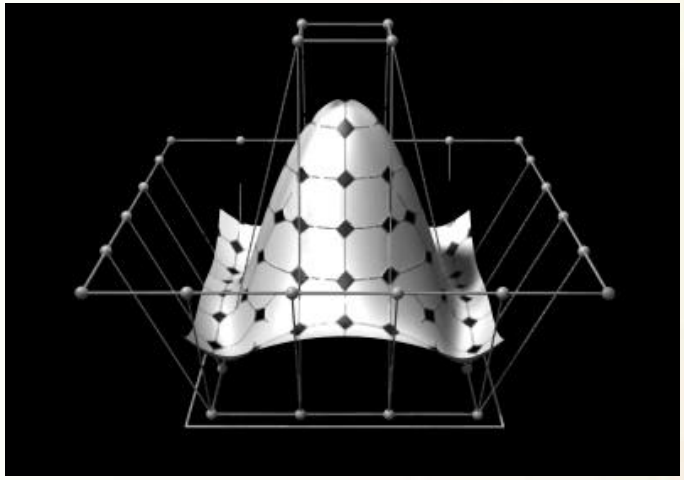
Pavyzdžiai:



C2 paviršius

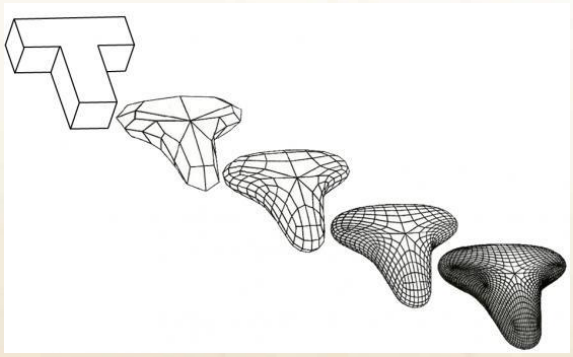
Glodus paviršius, sudarytas iš atskirų mažesnių paviršių, kurių sandūroje sutampa antros eilės išvestinės (kreivumas κ) kiekvieno tokio paviršiaus atžvilgiu:

$$\kappa = \frac{|x'y'' - y'x''|}{(x'^2 + y'^2)^{\frac{3}{2}}}.$$

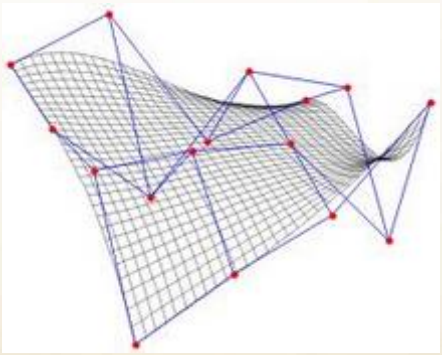


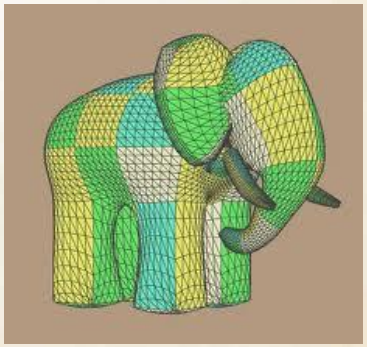
C2 paviršiaus konstravimo metodai:

Catmullo ir Clarko algoritmas



Aproksimavimo Bezjė paviršiais metodas





C2 paviršiaus konstravimas iš Bezjė paviršių (1)

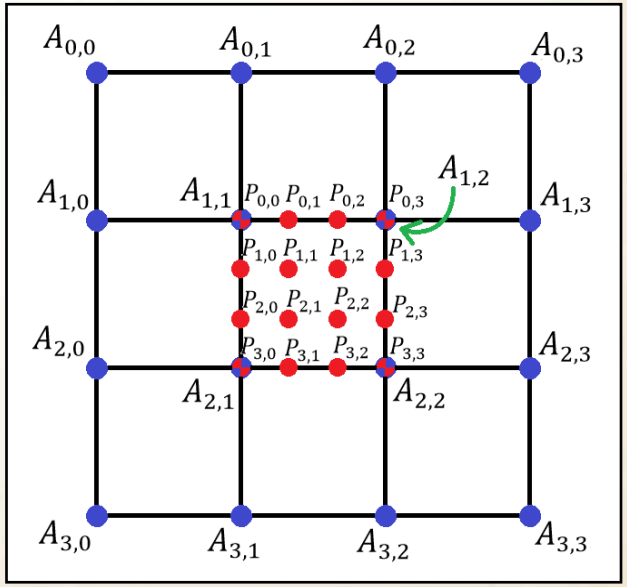
Trečios eilės Bezjė kreivių tenzorinės sandaugos koeficientai (kaukės):

$$\left[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right] \times \left[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right] = \begin{bmatrix} \frac{1}{36} & \frac{1}{9} & \frac{1}{36} \\ \frac{1}{9} & \frac{4}{9} & \frac{1}{9} \\ \frac{1}{36} & \frac{1}{9} & \frac{1}{36} \end{bmatrix}$$

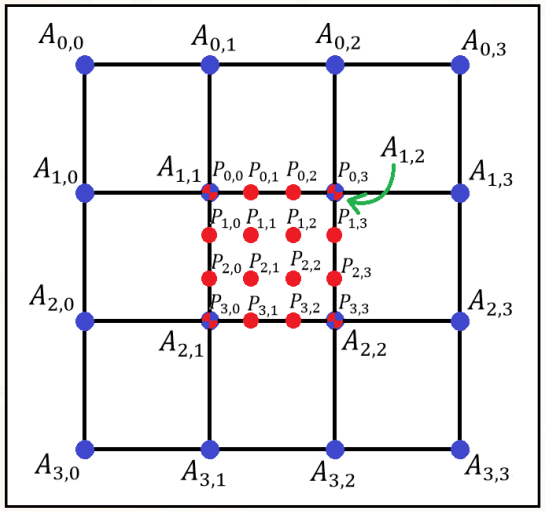
$$\left[\frac{1}{3}, \frac{2}{3}\right] \times \left[\frac{1}{3}, \frac{2}{3}\right] = \begin{bmatrix} \frac{1}{18} & \frac{2}{9} & \frac{1}{18} \\ \frac{1}{9} & \frac{4}{9} & \frac{1}{9} \end{bmatrix}$$

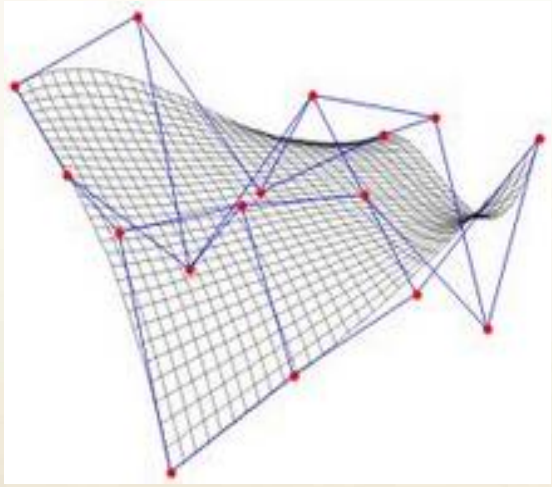
$$\begin{bmatrix} \frac{1}{9} & \frac{2}{9} \\ \frac{2}{9} & \frac{4}{9} \end{bmatrix} = \left[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right] \times \left[\frac{1}{3}, \frac{2}{3}\right]$$

Pagal šias kaukes apskaičiuojami Bezjė paviršiaus kontroliniai taškai.



Kontrolinių Bezjė paviršiaus taškų apskaičiavimas





$$P_{0,0} = \frac{1}{36}A_{0,0} + \frac{1}{36}A_{0,2} + \frac{1}{36}A_{2,0} + \frac{1}{36}A_{2,2} + \frac{1}{9}A_{0,1} + \frac{1}{9}A_{1,0} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{0,1} = \frac{1}{18}A_{0,2} + \frac{1}{18}A_{2,2} + \frac{1}{9}A_{0,1} + \frac{1}{9}A_{2,1} + \frac{2}{9}A_{1,2} + \frac{4}{9}A_{1,1},$$

$$P_{0,2} = \frac{1}{18}A_{0,1} + \frac{1}{18}A_{2,1} + \frac{1}{9}A_{0,2} + \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,1} + \frac{4}{9}A_{1,2},$$

$$P_{0,3} = \frac{1}{36}A_{0,1} + \frac{1}{36}A_{0,3} + \frac{1}{36}A_{2,1} + \frac{1}{36}A_{2,3} + \frac{1}{9}A_{0,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{1,3} + \frac{1}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{1,0} = \frac{1}{18}A_{2,0} + \frac{1}{18}A_{2,2} + \frac{1}{9}A_{1,0} + \frac{1}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{1,1} = \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{1,1},$$

$$P_{1,2} = \frac{1}{9}A_{2,1} + \frac{2}{9}A_{1,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{1,3} = \frac{1}{18}A_{2,1} + \frac{1}{18}A_{2,3} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{1,3} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{1,2},$$

$$P_{2,0} = \frac{1}{18}A_{1,0} + \frac{1}{18}A_{1,2} + \frac{1}{9}A_{2,0} + \frac{1}{9}A_{2,2} + \frac{2}{9}A_{1,1} + \frac{4}{9}A_{2,1},$$

$$P_{2,1} = \frac{1}{9}A_{1,2} + \frac{2}{9}A_{1,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{2,1},$$

$$P_{2,2} = \frac{1}{9}A_{1,1} + \frac{2}{9}A_{1,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{2,2},$$

$$P_{2,3} = \frac{1}{18}A_{1,1} + \frac{1}{18}A_{1,3} + \frac{1}{9}A_{2,1} + \frac{1}{9}A_{2,3} + \frac{2}{9}A_{1,2} + \frac{4}{9}A_{2,2},$$

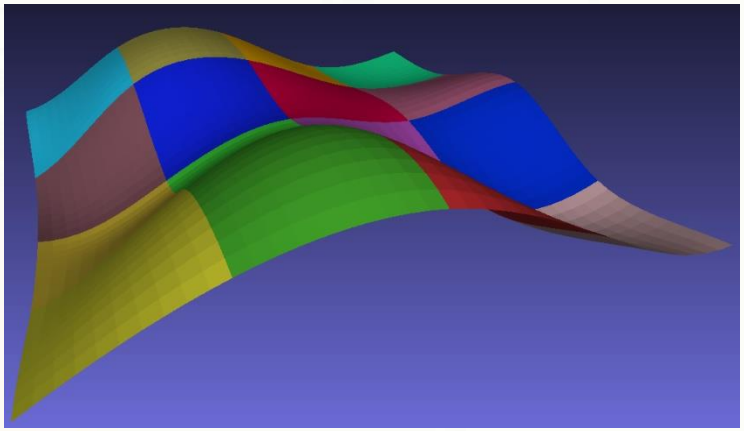
$$P_{3,0} = \frac{1}{36}A_{1,0} + \frac{1}{36}A_{1,2} + \frac{1}{36}A_{3,0} + \frac{1}{36}A_{3,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{2,0} + \frac{1}{9}A_{2,2} + \frac{1}{9}A_{3,1} + \frac{4}{9}A_{2,1},$$

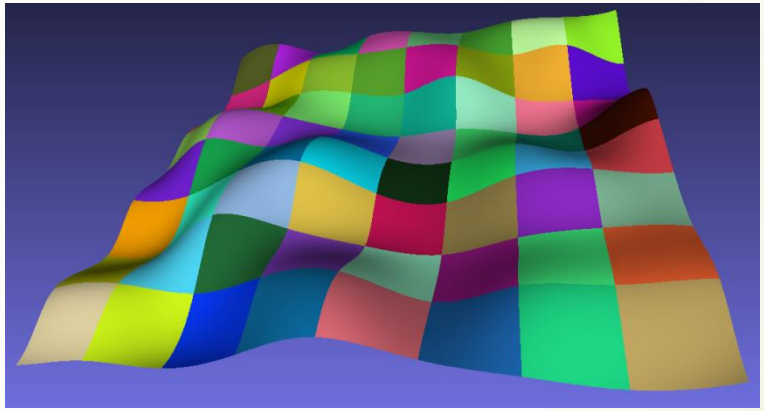
$$P_{3,1} = \frac{1}{18}A_{1,2} + \frac{1}{18}A_{3,2} + \frac{1}{9}A_{1,1} + \frac{1}{9}A_{3,1} + \frac{2}{9}A_{2,2} + \frac{4}{9}A_{2,1},$$

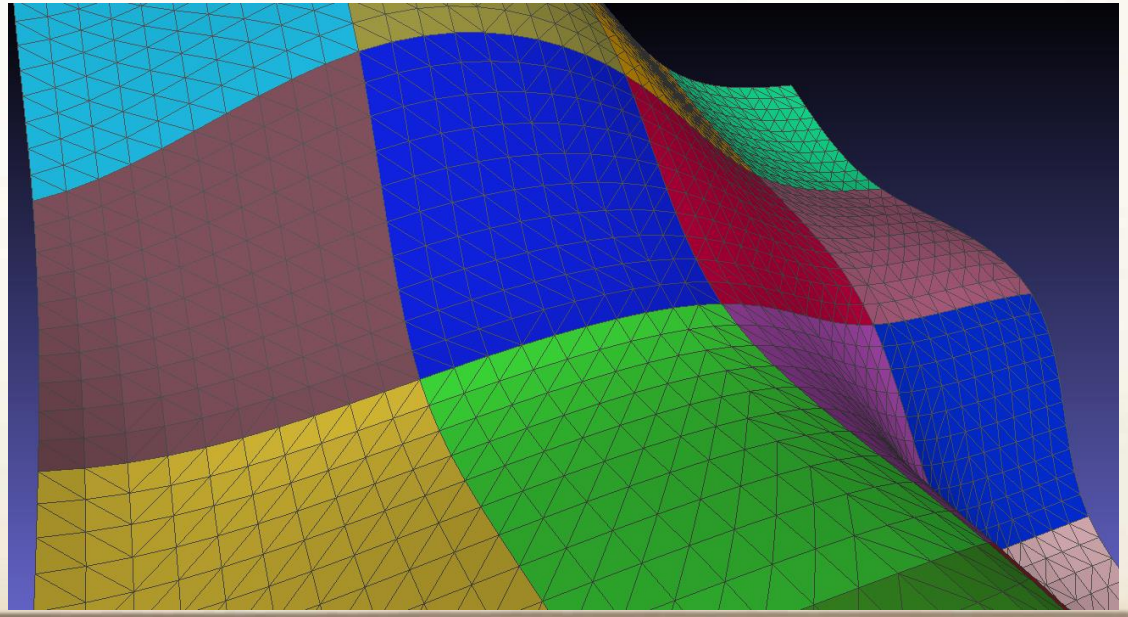
$$P_{3,2} = \frac{1}{18}A_{1,1} + \frac{1}{18}A_{3,1} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{3,2} + \frac{2}{9}A_{2,1} + \frac{4}{9}A_{2,2},$$

$$P_{3,3} = \frac{1}{36}A_{1,1} + \frac{1}{36}A_{1,3} + \frac{1}{36}A_{3,1} + \frac{1}{36}A_{3,3} + \frac{1}{9}A_{1,2} + \frac{1}{9}A_{2,1} + \frac{1}{9}A_{2,3} + \frac{1}{9}A_{3,2} + \frac{4}{9}A_{2,2}.$$

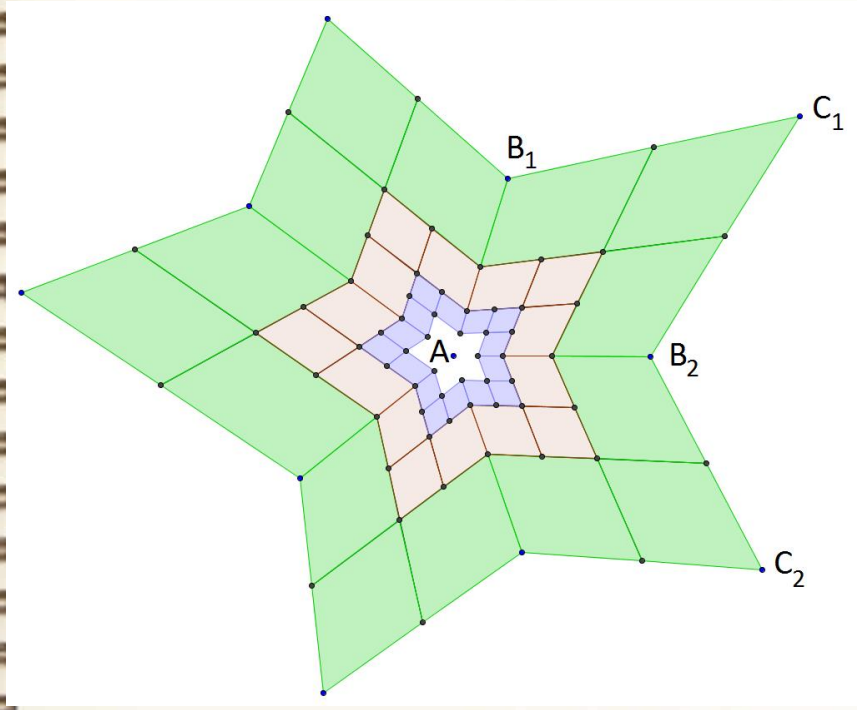
C2 paviršiaus konstravimas iš Bezjė paviršių. Pavyzdžiai







Bezjė paviršių konstravimas ypatingųjų taškų aplinkoje



Naudojamos šios kaukės paviršių dalinimui.

Naujo taško koordinatės viršūnės vietoje:

$$\begin{bmatrix} \frac{1}{4} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{4} \end{bmatrix}$$

Naujo taško koordinatės briaunos vietoje:

$$\begin{bmatrix} \frac{1}{16} & \frac{3}{8} & \frac{1}{16} \\ \frac{1}{16} & \frac{3}{8} & \frac{1}{16} \end{bmatrix}$$

Naujo taško koordinatės sienos vietoje:

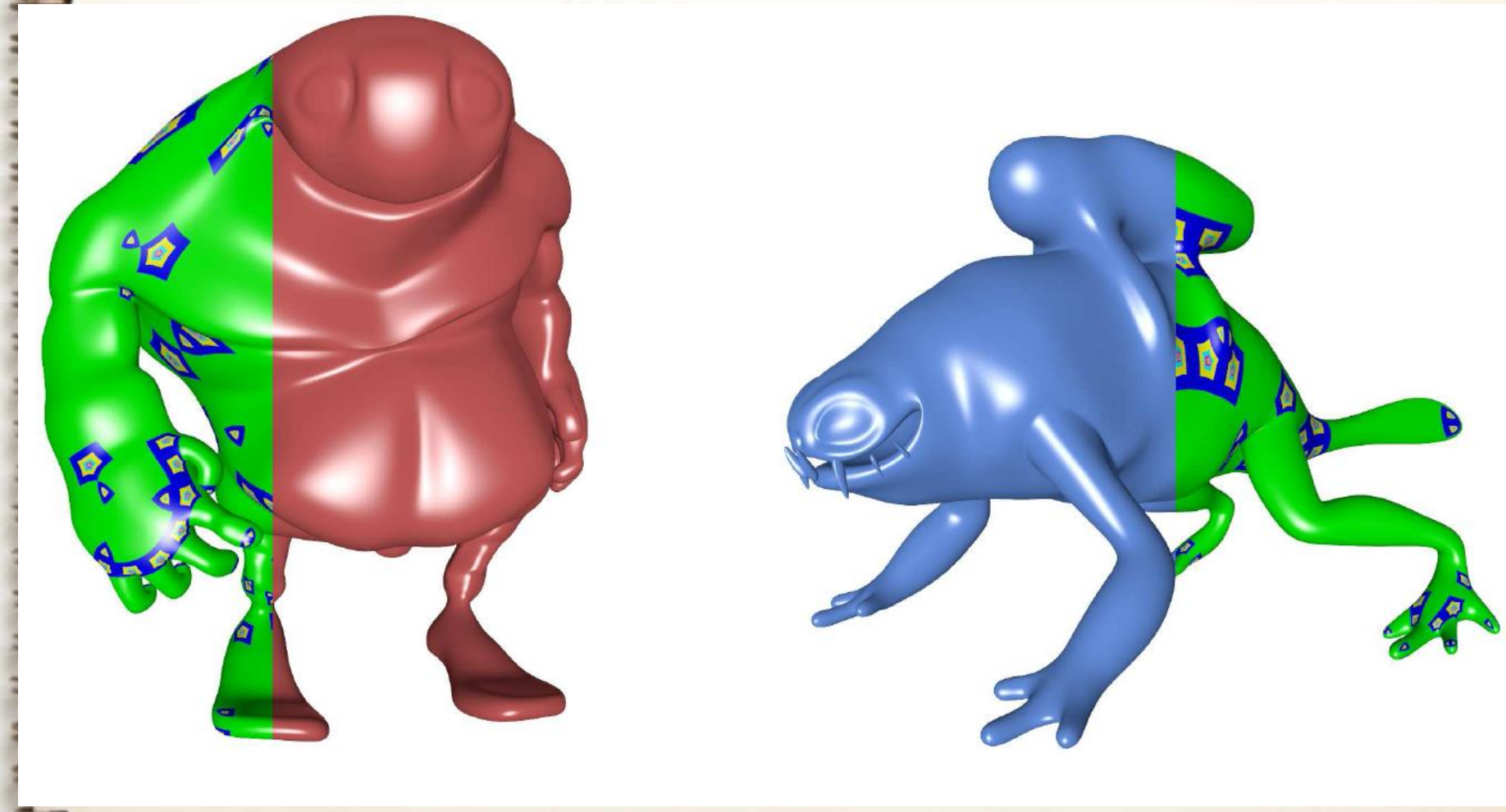
$$\begin{bmatrix} \frac{1}{64} & \frac{3}{32} & \frac{1}{64} \\ \frac{3}{32} & \frac{9}{16} & \frac{3}{32} \\ \frac{1}{64} & \frac{3}{32} & \frac{1}{64} \end{bmatrix}$$

$$centras = \frac{n}{n+2}A + \frac{4n}{n(n+5)}\sum_{i=1}^{n} B_i + \frac{1}{n(n+5)}\sum_{i=1}^{n} C_i$$

K. Karčiauskas, J. Peters (2007)

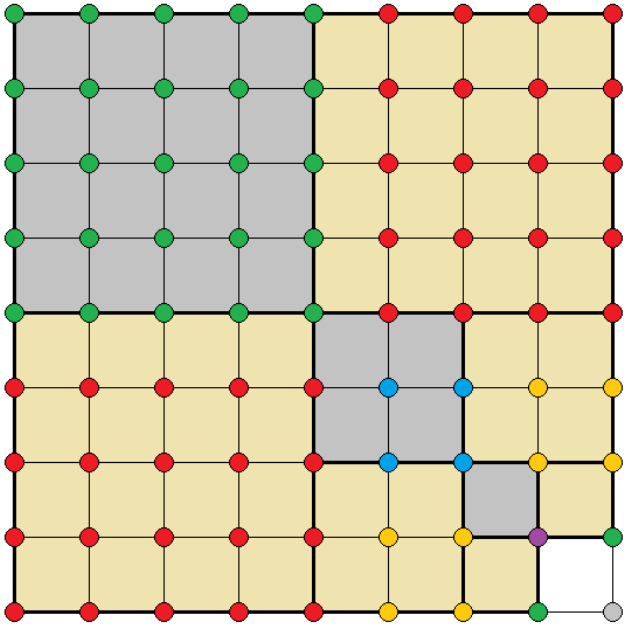
Programa vizualizavimui: **BezierView**,
failo formatas: **.bw**.

Matthias Nießner, Charles Loop, Microsoft Research, Mark Meyer and Tony DeRose (2012)



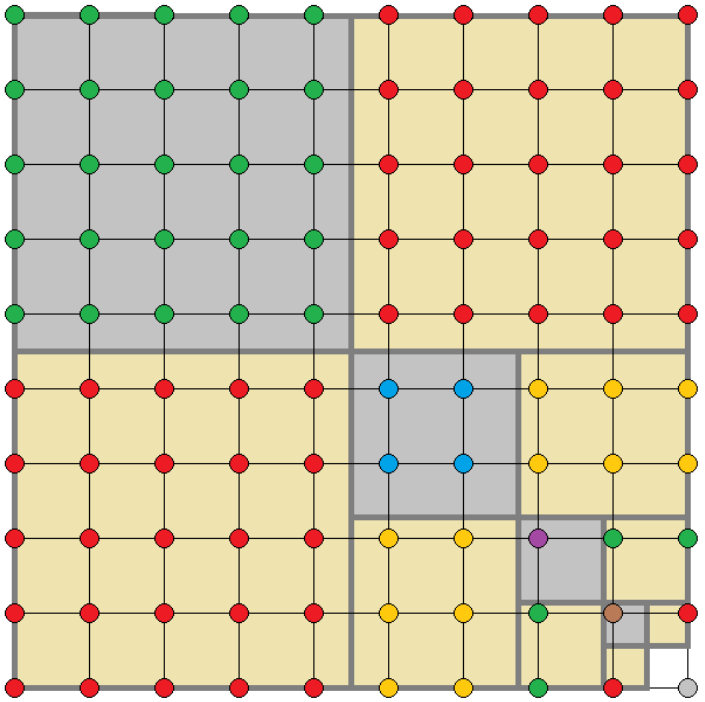
Begalinio proceso keitimas baigtiniu (1)

n = 8



```
0.000000, 0.250000, 0.500000, 0.750000, 1.000000,
0.500000, 1.000000,
1.000000,

0.250000, 0.500000, 0.750000, 1.000000,
0.500000, 1.000000,
1.000000,
```

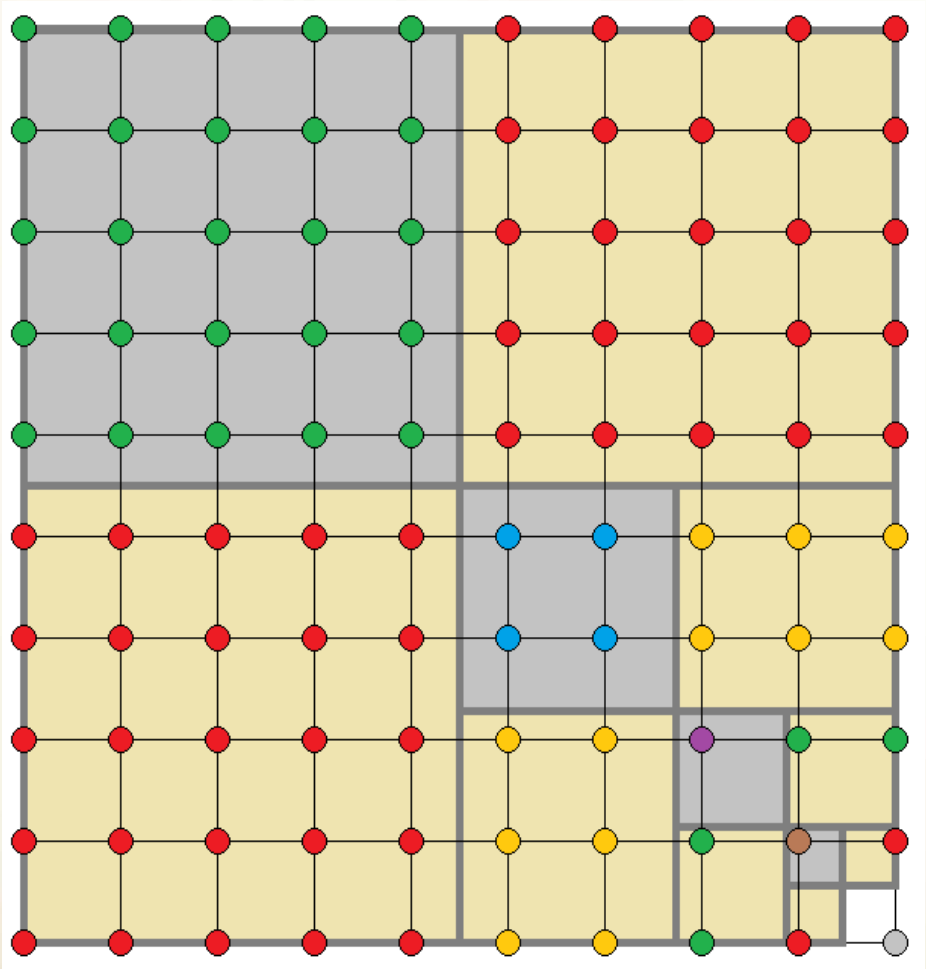
n = 9



```
0.000000, 0.222222, 0.444444, 0.666667, 0.888889,
0.222222, 0.666667,
0.222222,
0.222222,

0.111111, 0.333333, 0.555556, 0.777778, 1.000000,
0.111111, 0.555556, 1.000000,
0.111111, 1.000000,
1.000000,
```

Iteracijų skaičius: $\left[\log_2(n-1)\right]+1$

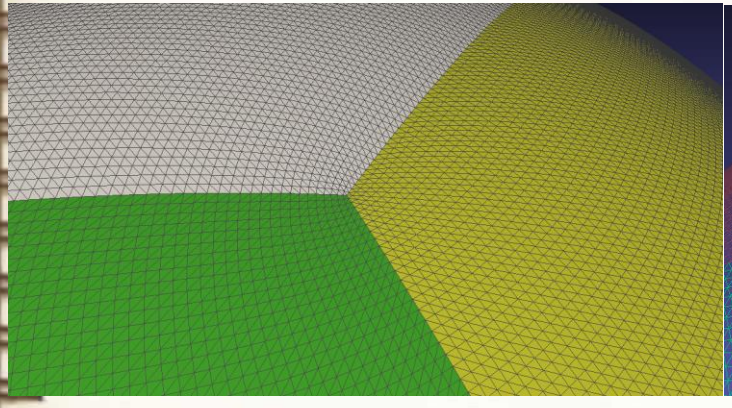
Begalinio proceso keitimas baigtiniu (2)

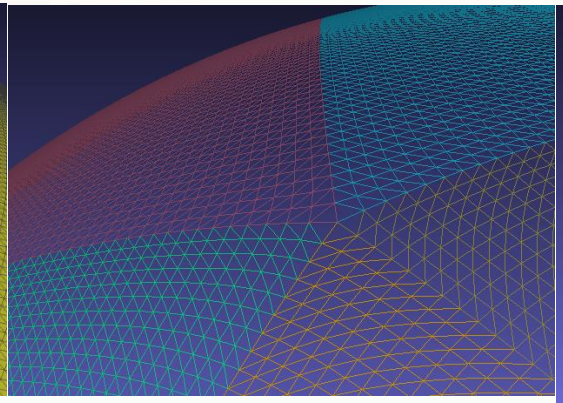


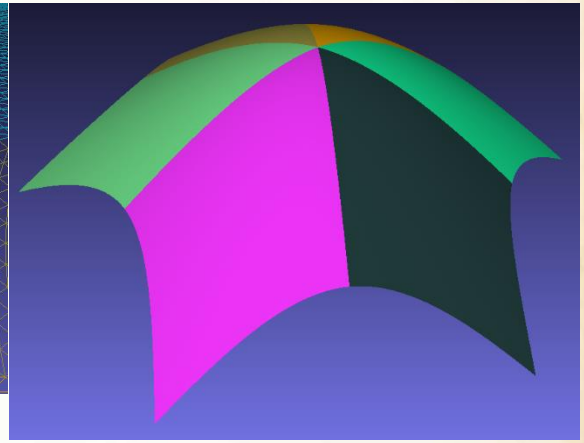
Tinklas(n)

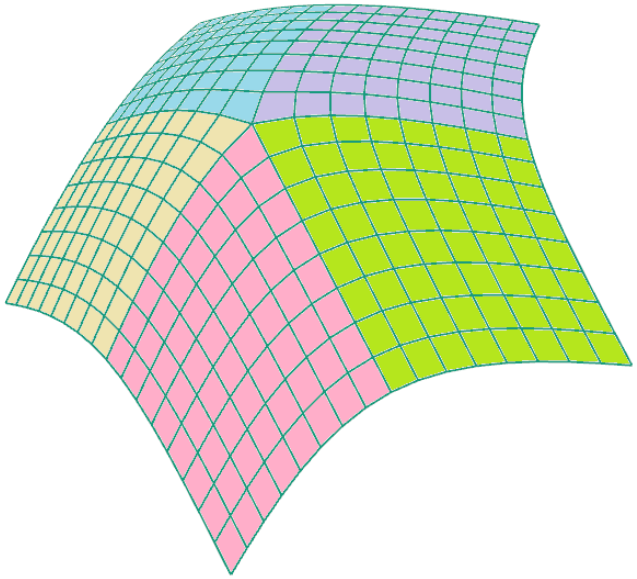
1. $d \to [\log_2(n-1)] + 1$
2. $m_1 \to \left[\frac{n+2}{2}\right]$
3. $m_2 \to n + 1 - m_1$
4. **for** $i \to 1$ **to** d **do**
5. $\quad p \to \frac{2^i m_2}{n} - 1$
6. $\quad$ **for** $j \to 1$ **to** m_1 **do**
7. $\quad\quad T_1[i][m_1 - j + 1] \to 1 - p - \frac{(j-1)2^i}{n}$
8. $\quad$ **for** $k \to 1$ **to** m_2 **do**
9. $\quad\quad T_2[i][m_2 - k + 1] \to 1 - \frac{(k-1)2^i}{n}$
10. $\quad m_1 \to \left[\frac{m_2}{2}\right]$
11. $\quad m_2 \to m_2 - m_1$
12. **return** T_1, T_2

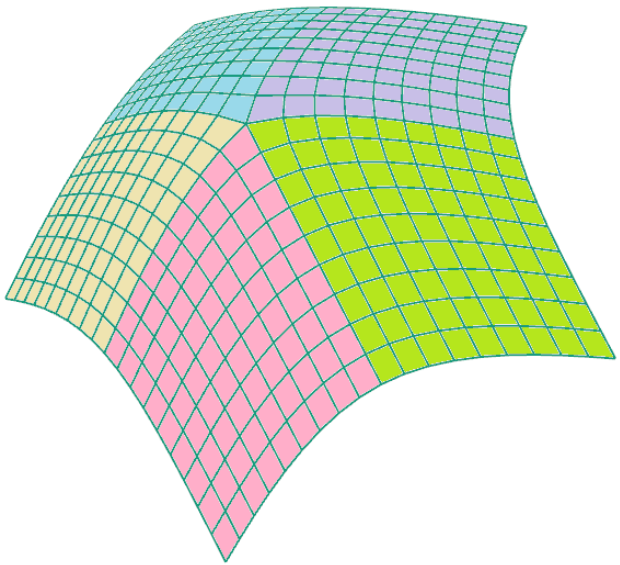
C2 paviršiaus formavimas ypatingųjų taškų srityse







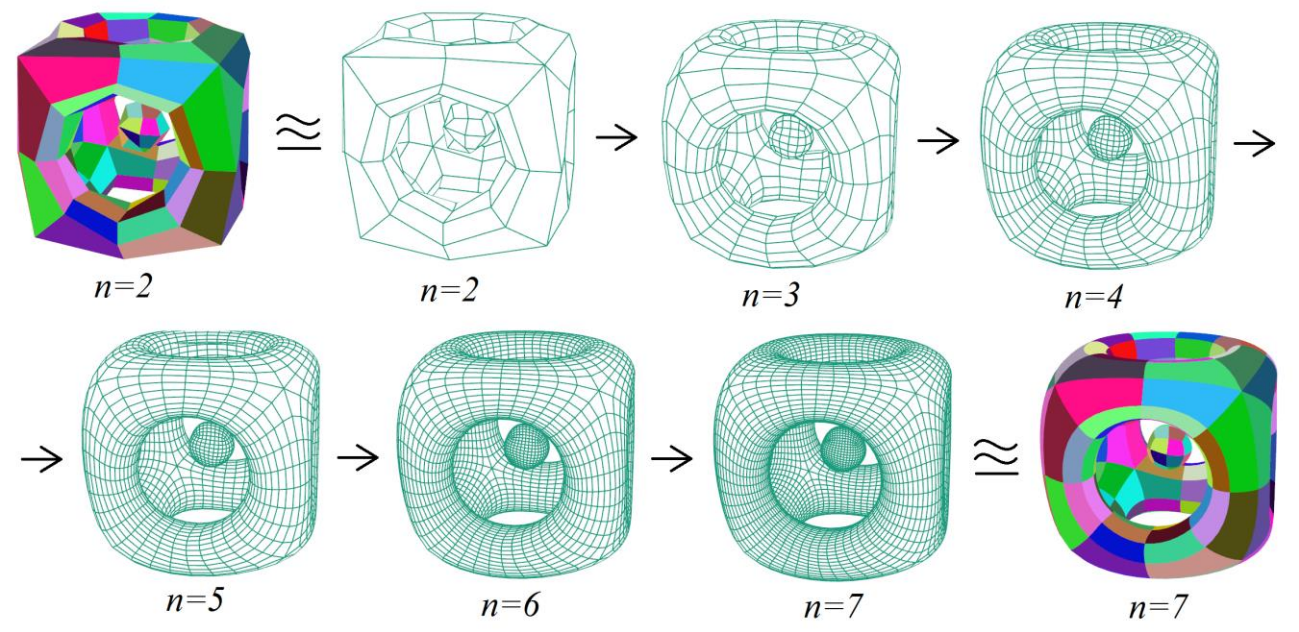




Bendroji algoritmo idėja

C2_paviršius(n)

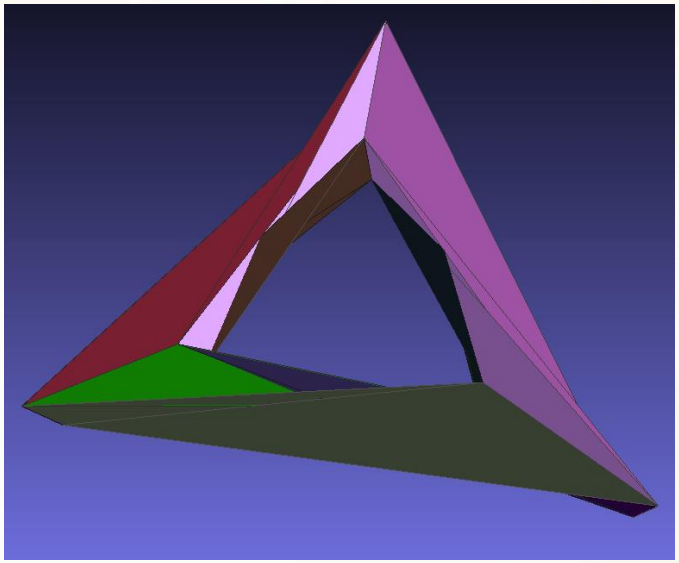
1. Inicializacija,
2. Tinklas(n),
3. Realizuok $[\log_2(n-1)]+1$ kartų Catmull-Clark algoritmą keturkampiams, kurių viršūnės – ypatingieji taškai,
4. Apskaičiuok visų Bezjė paviršių lygtis,
5. Į šias lygtis įsistatyk atitinkamas $u, v \in [0,1]$ parametrų reikšmes ir apskaičiuok C2 paviršiaus taškus,
6. Susiedamas šiuos taškus po 4 išsaugok rezultatą, kuris atitinka C2 paviršiaus aproksimaciją.

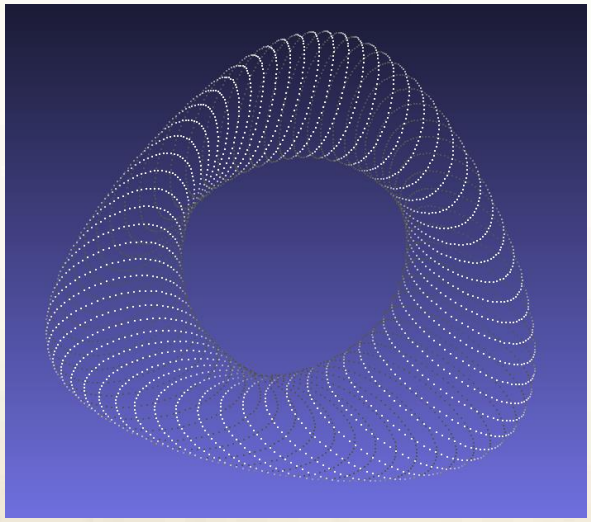


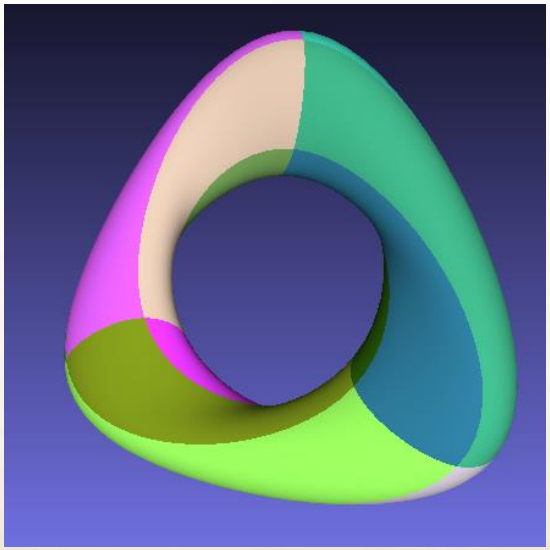
Algoritmo taikymas

- Ši Bezjė paviršių aproksimavimo metodika gali būti taikoma optimizavimo uždaviniuose, kuriuose pagrindinis tikslas maksimizuoti C2 paviršiaus detalumą, kai:
  - kompiuterio pastovioji atmintis, skirta *C2_paviršius(n)* algoritmo rezultato išsaugojimui, turi tam tikrus apribojimus;
  - yra žinoma perteklinė informacijos riba, pavyzdžiui, naudojant 3D spausdintuvą trimačiams objektams suformuoti, papildomas C2 paviršiaus taškų skaičius neturi įtakos rezultato kokybei.

„Neįmanomas“ trikampis







Loop-Schaefer algoritmas (2008)

US 20080043023A1

(19) **United States**

(12) **Patent Application Publication**

Loop et al.

(10) **Pub. No.: US 2008/0043023 A1**

(43) **Pub. Date: Feb. 21, 2008**

(54) **APPROXIMATING SUBDIVISION SURFACES WITH BEZIER PATCHES**

(75) Inventors: **Charles T. Loop**, Bellevue, WA (US); **Scott David Schaefer**, Bryan, TX (US)

Correspondence Address:
KLARQUIST SPARKMAN LLP
121 S.W. SALMON STREET, SUITE 1600
PORTLAND, OR 97204

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(21) Appl. No.: **11/464,800**

(22) Filed: **Aug. 15, 2006**

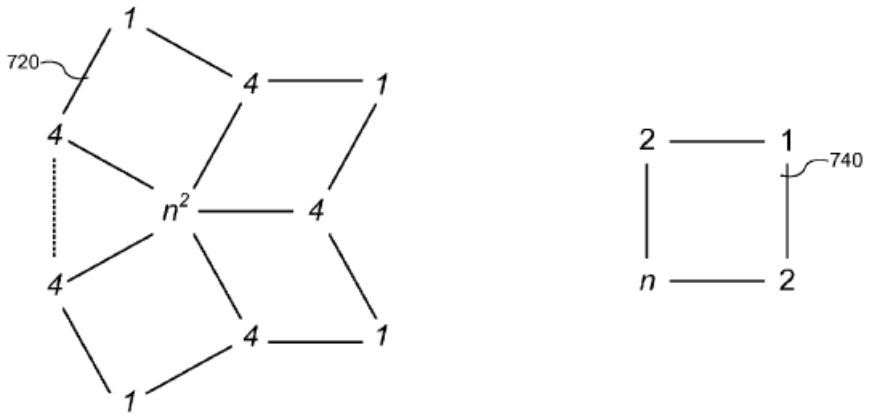
Publication Classification

(51) **Int. Cl.**
G06T 11/20 (2006.01)

(52) **U.S. Cl.** **345/441**

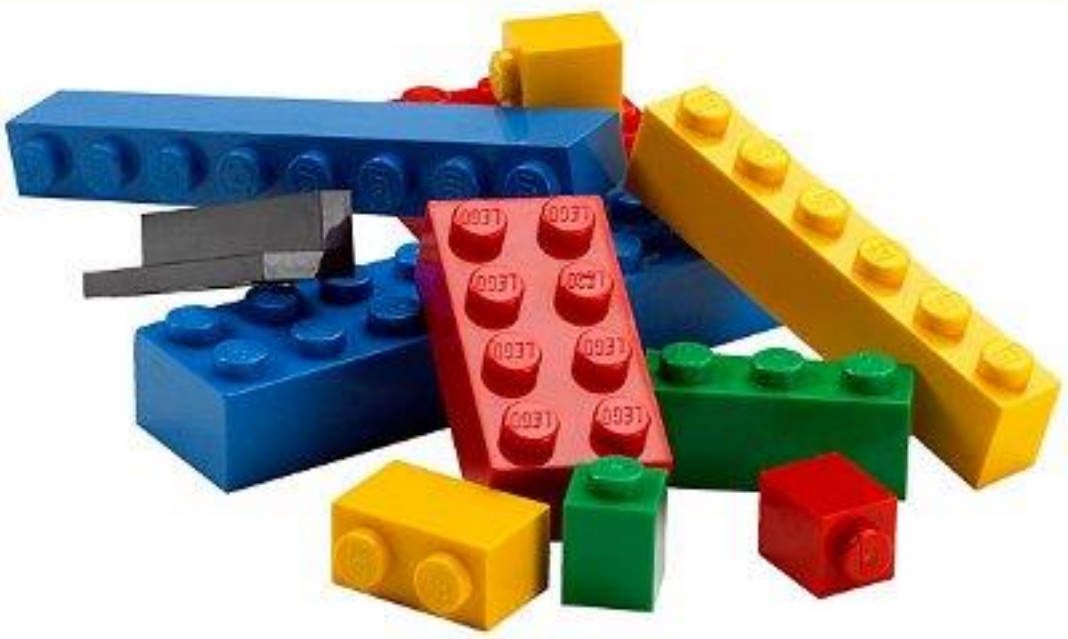
(57) **ABSTRACT**

Surface modeling systems and techniques are described which approximate Catmull-Clark subdivision surfaces. A quadrilateral mesh is analyzed by applying shape and tangent masks to faces on the quad-mesh. Through application of the shape masks, a shape patch is created which approximates the subdivision limit surface. This shape patch can be used for rendering surface shape. Through application of tangent masks, tangent patches are created which comprise tangent vectors which give rise to continuous normal vector fields, which can be used for shading of the surface.



Masks for approximating shape

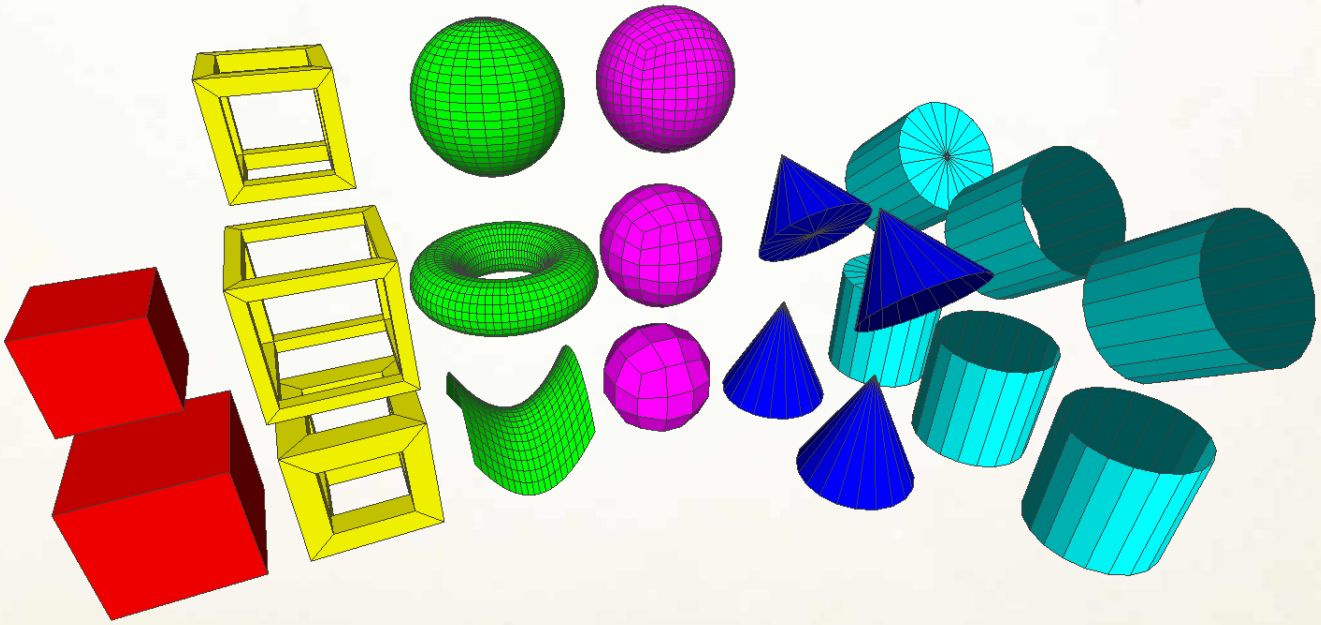
Norint kažką sukonstruoti, reikia...



turėti detalių.

12 paskaitos tikslas

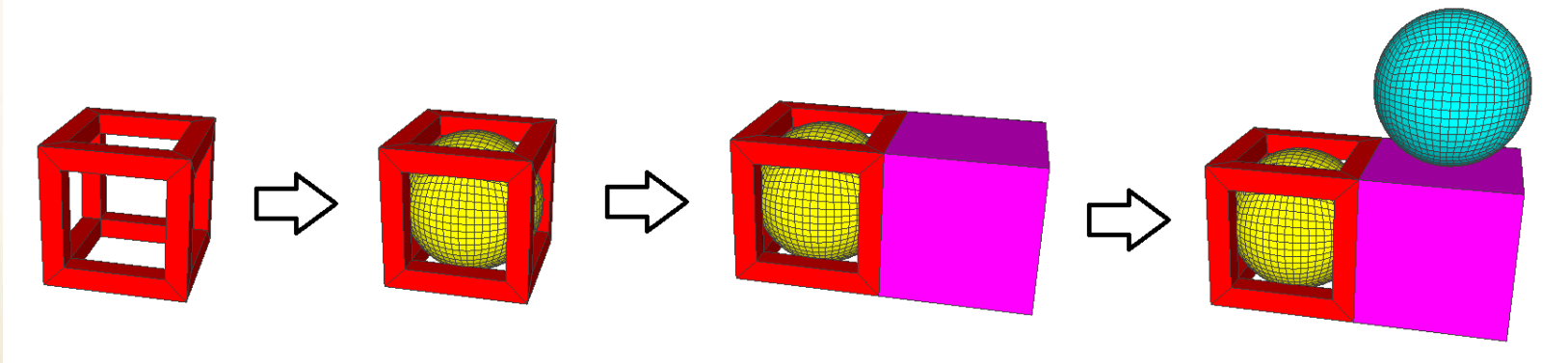
- Susipažinti su *python* modulio ***add.py*** *1.2 versija.*
- Sukurti skaitmeninį modelį naudojantis šiuo moduliu:



- Įkelti sukurtą 3D modelį į https://sketchfab.com svetainę.

python modulis ***add.py***

- Modulis kiekvienais metais atnaujinamas!
- Modulis pritaikytas kurti 3D skaitmeninius modelius **OFF** formatu.
- Būtina Python 3 versija!
- Modelio kūrimas vyksta konstravimo principu:



Pagrindinė idėja

Į sąrašą *vertices* įrašomos viršūnių koordinatės *string* pavidalu, į sąrašą *faces* įrašoma informacija apie kiekvieną 3D modelio sieną irgi *string* pavidalu. Šios informacijos užtenka norint sugeneruoti 3D modelį OFF formatu.

```
vertices = []
faces = []

<...>

def off(mesh):
  global vertices, faces
  file = open(mesh, 'w')
  file.write('%s\n%d %d %d\n' % ('OFF',len(vertices),len(faces),0))
  for i in range (len(vertices)):
    file.write('%s\n' % vertices[i])
  for j in range (len(faces)):
    file.write('%s\n' % faces[j])
  file.close()
```

*Modulio **add.py** funkcijos*

- **def cube(c,e,RGB):** # c = center, e = edge width
- **def cube2(c,e,b,RGB):** # c = center, b = border width, e = edge width
- **def parametric(S,min_u,max_u,grid_u,min_v,max_v,grid_v,RGB):** # S - parametric uv surface, grid - detail, RGB - color
- **def sphere(c,r,k,RGB):** # c - center, r - radius, k - detail, RGB - color
- **def cylinder(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color
- **def cylinder2(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color
- **def cylinder3(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color
- **def cone(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color
- **def cone2(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color
- **def off(mesh):** # mesh – off file

*Modulio **add.py** 1.1 versija*

- **def newface(A,RGB):** # A = set of 3D points
- **def pyramid(c,e,h,RGB):** # c - center, e - edge width, h - high
- **def rectangle3D(c,e,RGB):** # c - center, e - width of edges
- **def circle(A,B,r,k,RGB):** # A - start point, B - end point, r - radius, k - detail
- **def spin3D(A,B,S,min_t,max_t,grid_t,k,RGB):** # A - start point, B - end point, r - radius, k - detail, RGB - color, S – parametric function

*Modulio **add.py** 1.2 versija*

- **def axes(C):** # adds 3D axes to default layer, C - center
- **def layer():** # returns 3D mesh from default layer and deletes default layer
- **def mesh(M):** # adds 3D mesh to default layer, M – selected layer
- **def center(M):** # returns 3D center from selected layer M
- **def rotateX(M,angle,P):** # rotates selected layer M around X-axis,
  angle – rotation angle, P – point of rotation center
- **def rotateY(M,angle,P):** # rotates selected layer M around Y-axis
- **def rotateZ(M,angle,P):** # rotates selected layer M around Z-axis
- **def move(M,V):** # moves selected layer M according to vector V
- **def zoom(M,s):** # zooms selected layer M by scale s
- **def merge(M**): **#** merges selected layers M=[M[0],M[1], …]
- **def load(mesh):** # loads mesh from off file
- **def color(M,RGB):** # sets RGB color to layer M

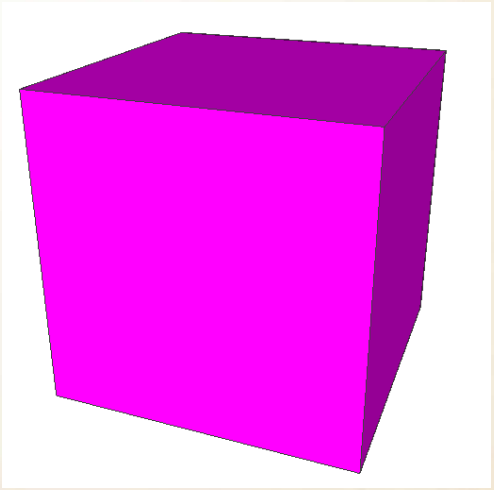
def cube(c, e, RGB)

```
def cube(c,e,RGB): # c - center, e - edge width, RGB - color
  global vertices, faces
  F = [[0,4,5,1],[0,1,3,2],[0,2,6,4],[1,5,7,3],[2,3,7,6],[4,6,7,5]]
  V = [[0,0,0],[0,0,e],[0,e,0],[0,e,e],[e,0,0],[e,0,e],[e,e,0],[e,e,e]]
  for i in range (0,6):
    faces += ['4 '+str(F[i][0]+len(vertices))+' '+str(F[i][1]+len(vertices))+
    ' '+str(F[i][2]+len(vertices))+' '+str(F[i][3]+len(vertices))+
    ' '+str(RGB[0])+' '+str(RGB[1])+' '+str(RGB[2])]
  for j in range (0,8):
    vertices += [str(c[0]+V[j][0]-e/2)+' '+str(c[1]+V[j][1]-e/2)+
    ' '+str(c[2]+V[j][2]-e/2)]
```

c – centro 3D koordinatės, e – briaunos ilgis,
RGB – kubo spalva.

cube.py generuoja cube.off failą:

```
1  import add
2  add.cube([1,0,0],1,[255,0,255])
3  add.off('cube.off')
```



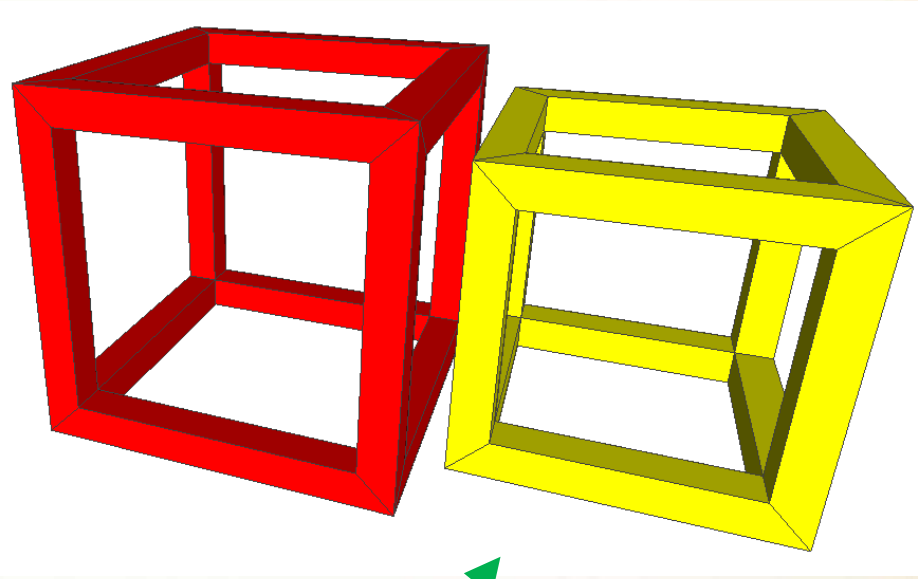
def cube2(c, e, b, RGB)

c – centro 3D koordinatės,

e – briaunos ilgis,

b – briaunos storis,

RGB – kubo spalva.

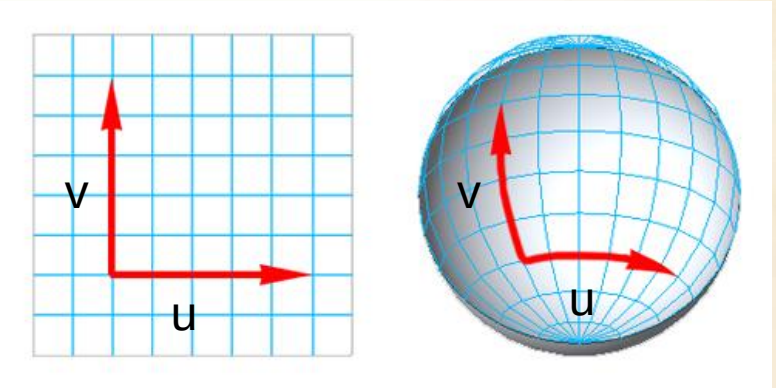


cube2.py generuoja cube2.off failą:

```
1  import add
2  add.cube2([1,0,0],1,0.1,[255,0,0])
3  add.cube2([2,0,0],0.8,0.1,[255,255,0])
4  add.off('cube2.off')
```

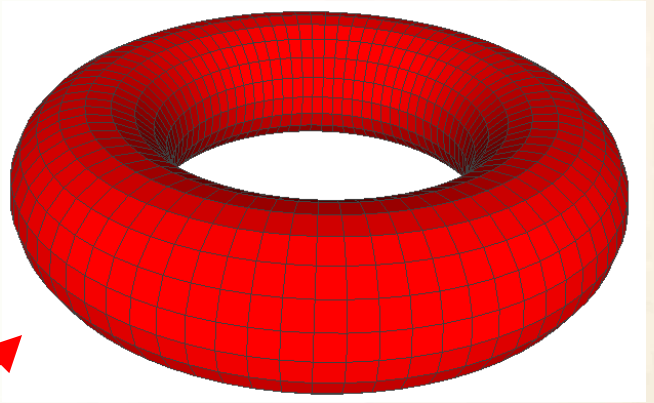
def parametric(S, min_u, max_u, grid_u, min_v, max_v, grid_v, RGB)

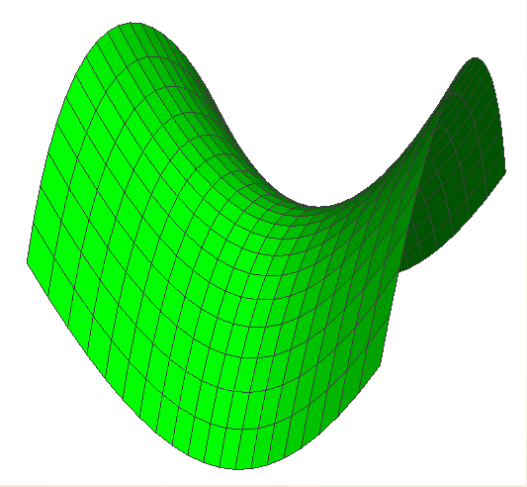
S – parametrinio uv paviršiaus f-ja,
min_u – parametro u mažiausia reikšmė,
max_u – parametro u didžiausia reikšmė,
grid_u – paviršiaus detalumas u atžvilgiu,
min_v – parametro v mažiausia reikšmė,
max_v – parametro v didžiausia reikšmė,
grid_v – paviršiaus detalumas v atžvilgiu,
RGB – paviršiaus spalva.



Parametrinių paviršių pavyzdžiai

```
1  import add
2  def saddle(u,v):
3    x = u
4    y = v*v - u*u
5    z = -v
6    return ([x, y, z])
7  add.parametric(saddle,-1,1,20,-1,1,20,[0,255,0])
8  add.off('saddle.off')
```





```
1  import add
2  import math
3  a = 3
4  b = 1
5  def torus(u,v):
6    x = (a+b*math.cos(u))*math.cos(v)
7    y = b*math.sin(u)
8    z = (a+b*math.cos(u))*math.sin(v)
9    return ([x, y, z])
10 add.parametric(torus,0,2*math.pi,20,0,2*math.pi,80,[255,0,0])
11 add.off('torus.off')
```

def sphere(c, r, k, RGB)

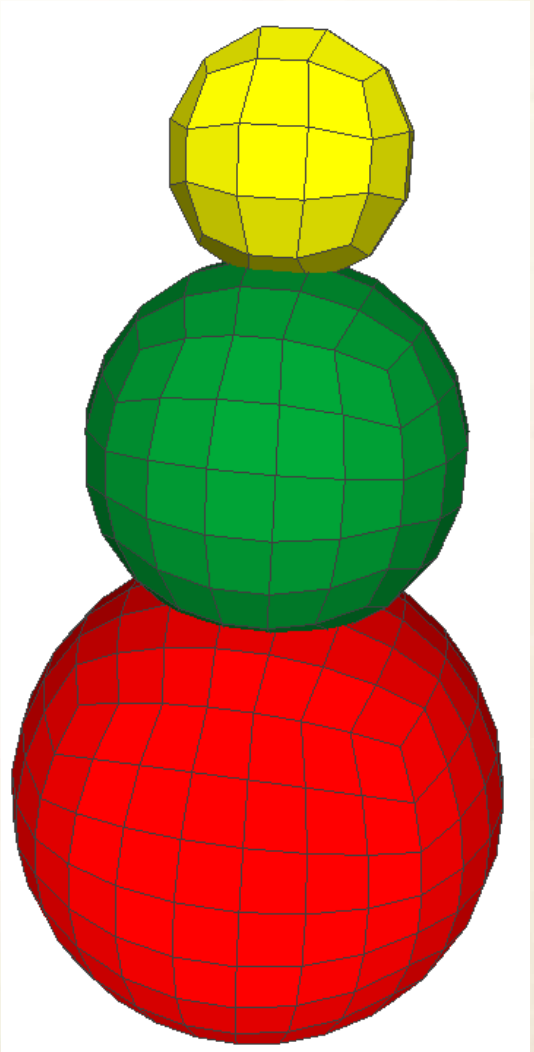
c – centro 3D koordinatės,

r – spindulio ilgis,

k – sferos detalumas,

RGB – sferos spalva.

```
1  import add
2  add.sphere([0,0.72,0],0.3,3,[255,255,0])
3  add.sphere([0,0,0],0.5,5,[0,153,51])
4  add.sphere([0,-1,0],0.7,7,[255,0,0])
5  add.off('LTsenis.off')
```



def cylinder(A, B, r, k, RGB) – uždaras cilindras
def cylinder2(A, B, r, k, RGB) – atviras cilindras
def cylinder3(A, B, r, k, RGB) – pusiau atviras cilindras

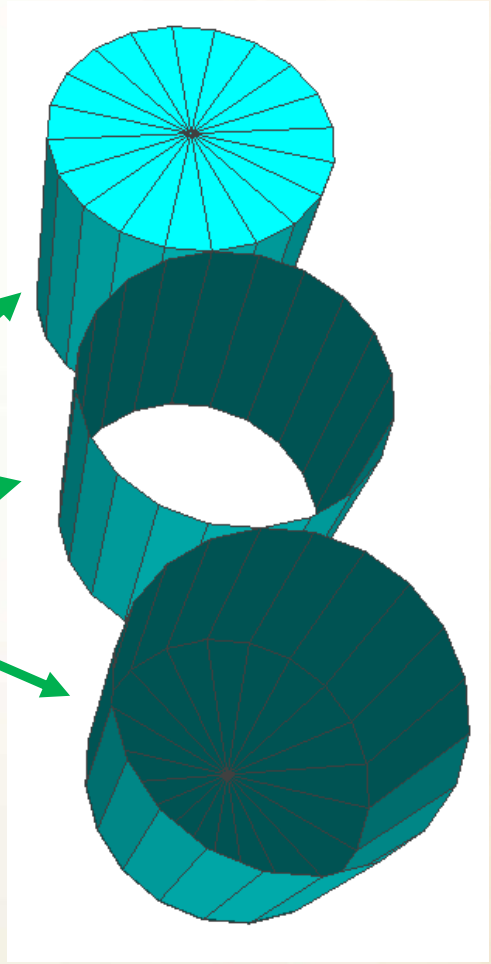
A – cilindro centro pradžios taškas,

B – cilindro centro pabaigos taškas,

r – cilindro spindulio ilgis,

k – cilindro detalumas,

RGB – cilindro spalva.




```
1  import add
2  add.cylinder([0,-0.5,0],[0,0.5,0],0.5,20,[0,255,255])
3  add.cylinder2([0,-0.5,1],[0,0.5,1],0.5,20,[0,255,255])
4  add.cylinder3([0,-0.5,2],[0,0.5,2],0.5,20,[0,255,255])
5  add.off('cylinders.off')
```

def cone(A, B, r, k, RGB) – uždaras kūgis
def cone2(A, B, r, k, RGB) – kūgio šoninis paviršius

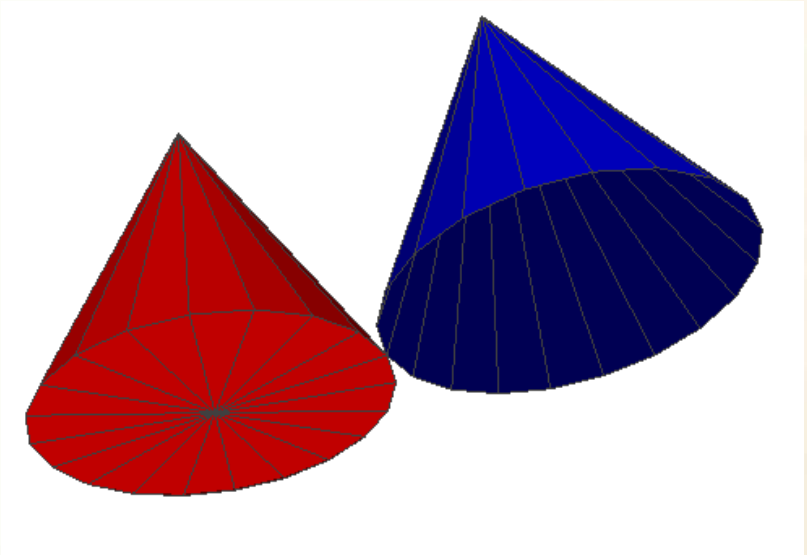
A – kūgio pagrindo centras,

B – kūgio viršūnė,

r – kūgio pagrindo spindulio ilgis,

k – kūgio detalumas,

RGB – kūgio spalva.

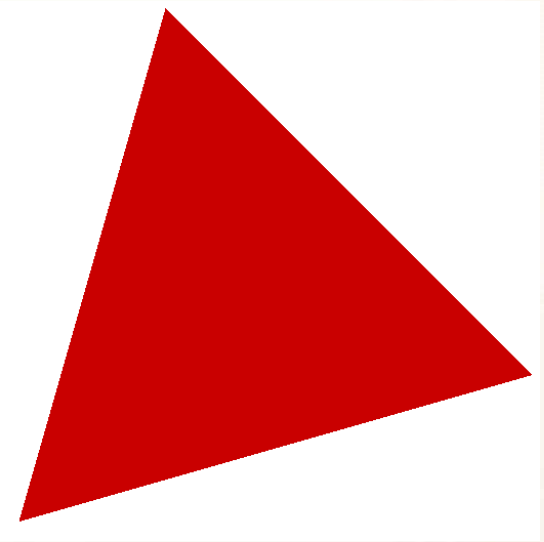


```
1  import add
2  add.cone([0,-0.5,0],[0,0.5,0],0.5,20,[255,0,0])
3  add.cone2([0,-0.5,1],[0,0.5,1],0.5,20,[0,0,255])
4  add.off('cones.off')
```

def newface(A,RGB)

A – 3D taškų seka,

RGB – sienos spalva.



```
1  import add
2  add.newface([[1,0,0],[0,1,0],[0,0,1]],[255,0,0])
3  add.off('trikampis.off')
```

Pastaba: gali būti nebūtinai trikampis.

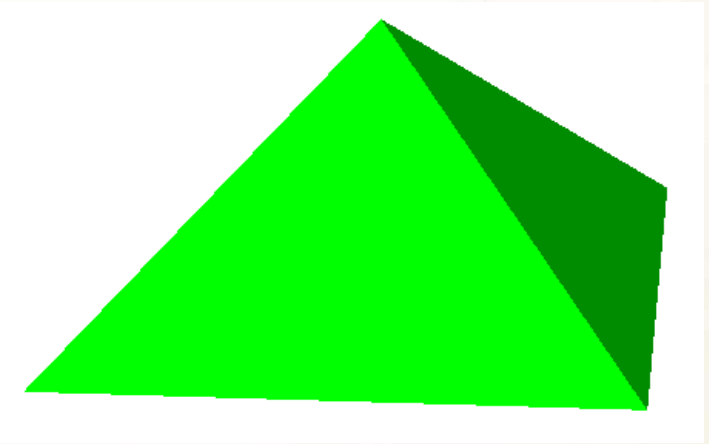
def pyramid(c,e,h,RGB)

c – kvadrato centro 3D koordinatės,

e – pagrindo (kvadrato) briaunos ilgis,

h – piramidės aukštis,

RGB – piramidės spalva.



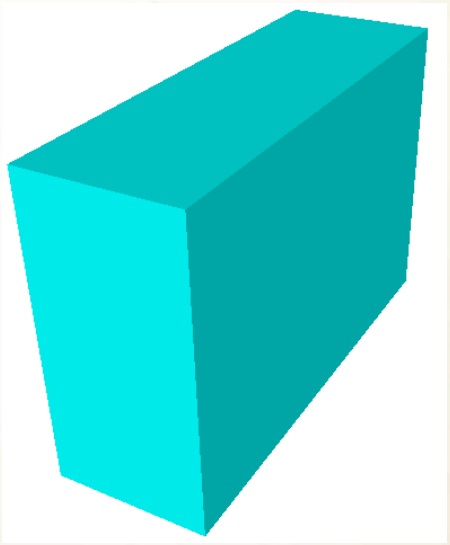
```
1 import add
2 add.pyramid([0,0,0],1,0.5,[0,255,0])
3 add.off('pyramid.off')
```

def rectangle3D(c,e,RGB)

c – centro 3D koordinatės,

e – stačiakampio gretasienio briaunų ilgių seka (atitinkamai X, Y ir Z ašių atžvilgiu),

RGB – stačiakampio gretasienio spalva.



```
1  import add
2  add.rectangle3D([0,0,0],[1,2,3],[0,255,255])
3  add.off('plyta.off')
```

def circle(A,B,r,k,RGB)

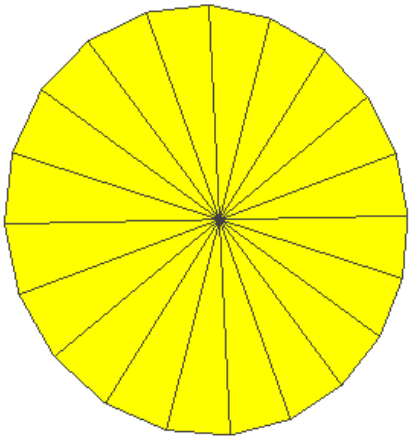
A – vektoriaus AB pradžios taškas,

B – vektoriaus AB pabaigos taškas,

r – apskritimo spindulys,

k – detalumo parametras,

RGB – apskritimo spalva.

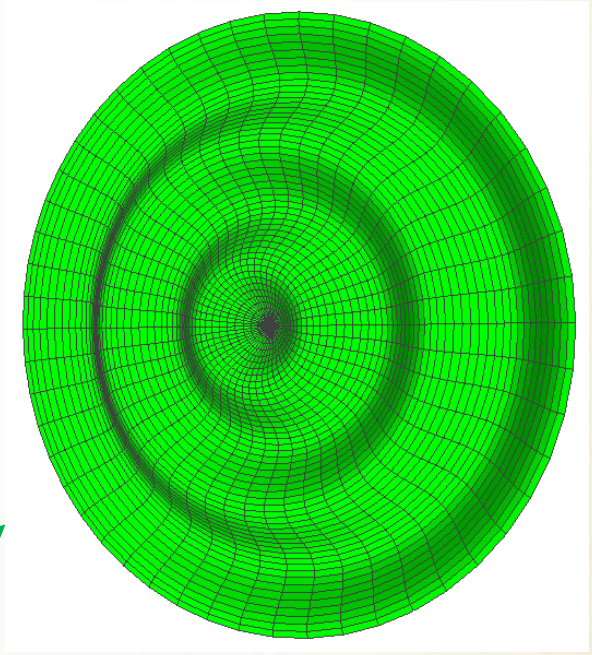


```
1  import add
2  add.circle([0,0,0],[1,1,1],1,20,[255,255,0])
3  add.off('apskritimas.off')
```

Pastaba: vektorius AB statmenas apskritimui, kur A – apskritimo centras.

def spin3D(A,B,S,min_t,max_t,grid_t,k,RGB)

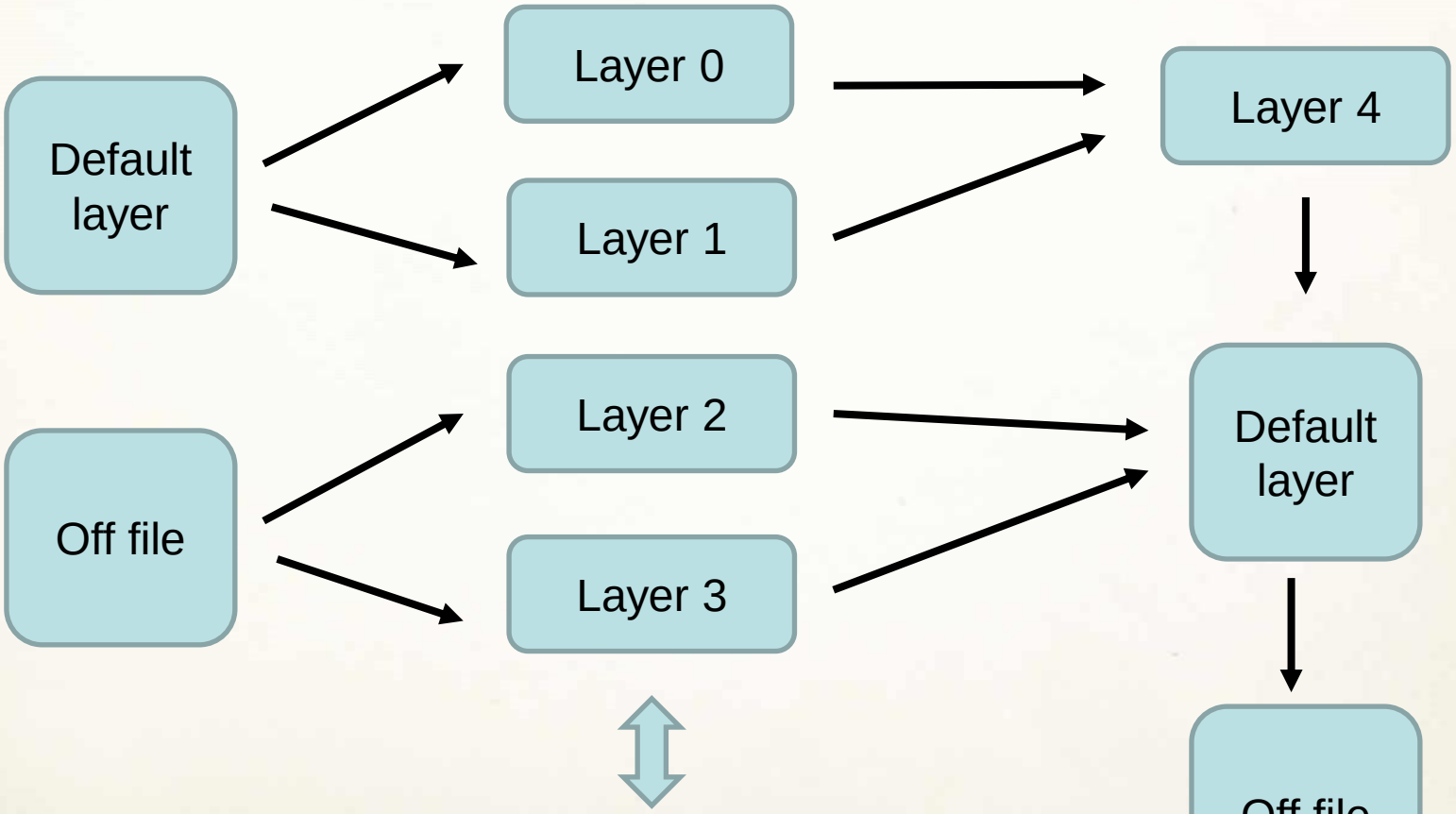
A – vektoriaus AB pradžios taškas,
B – vektoriaus AB pabaigos taškas,
S – parametrinė kreivė,
min_t, max_t – parametrinės kreivės t parametro intervalas,
grid_t – t parametro detalumas,
k – sukinio detalumas,
RGB – sukinio spalva.



```
1 import add
2 import math
3 def curve(t):
4   x = t
5   y = math.cos(t)
6   return ([x, y])
7 add.spin3D([0,0,0],[-1,1,3],curve,0,10*math.pi/2,50,50,[0,255,0])
8 add.off('sukinys.off')
```

Pastaba: kreivė sukama apie vektorių AB, kur A – naujos koordinačių pradžios taškas.

add.py 1.2 versija

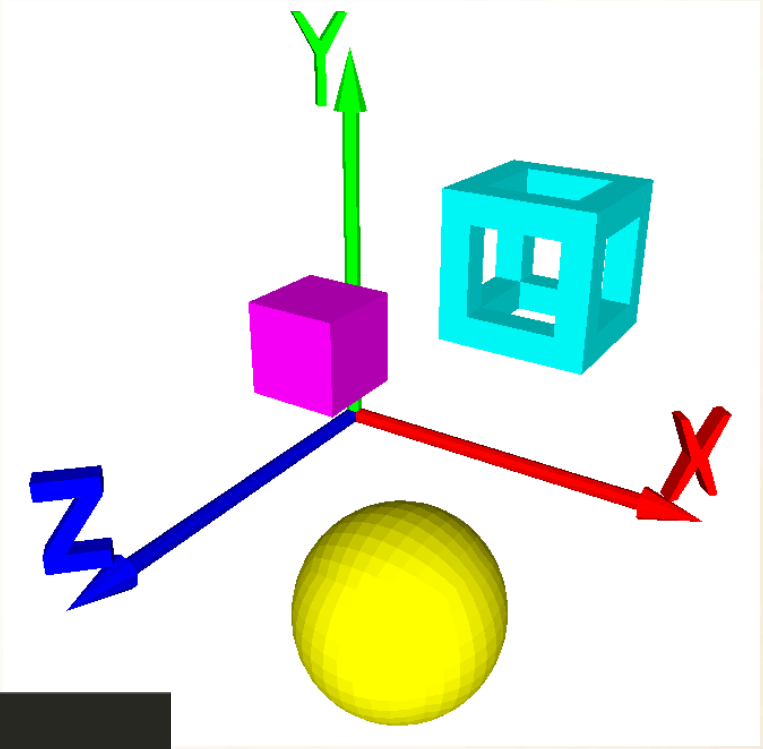


Sluoksniuose galimi 3D modelio sukimai, apjungimai, perstūmimai, mastelio keitimas, spalvų keitimas ir t. t.

def axes(C):

C – koordinačių centras,

axes(C) prideda pagalbines 3D koordinates.




```
import add
add.axes([0,0,0])
add.cube([1,2,2],1,[255,0,255])
add.cube2([3,3,1],1.5,0.3,[0,255,255])
add.sphere([2,-1,2],1.2,10,[255,255,0])
add.off("output4.off")
```

1.2 modulio off.py funkcijos, grąžinančios rezultatą sluoksniuose (1)

def layer(): # grąžinamas pagrindinio sluoksnio 3D modelis ir ištrinamas iš pagrindinio sluoksnio.

def center(M): # grąžinamos pasirinkto 3D modelio M centro koordinatės.

def move(M,V): # grąžinamas pastumtas 3D modelis M vektoriumi V.

def zoom(M,s): # grąžinamas pakeisto dydžio 3D modelis M masteliu s.

def merge([M[0],M[1], …]): # sujungiami keli 3D modeliai į vieną.

def load(mesh): # grąžinamas 3D modelis iš OFF failo.

def color(M,RGB): # 3D modelis M perdažomas spalva RGB=[R,G,B].

Išimtis:

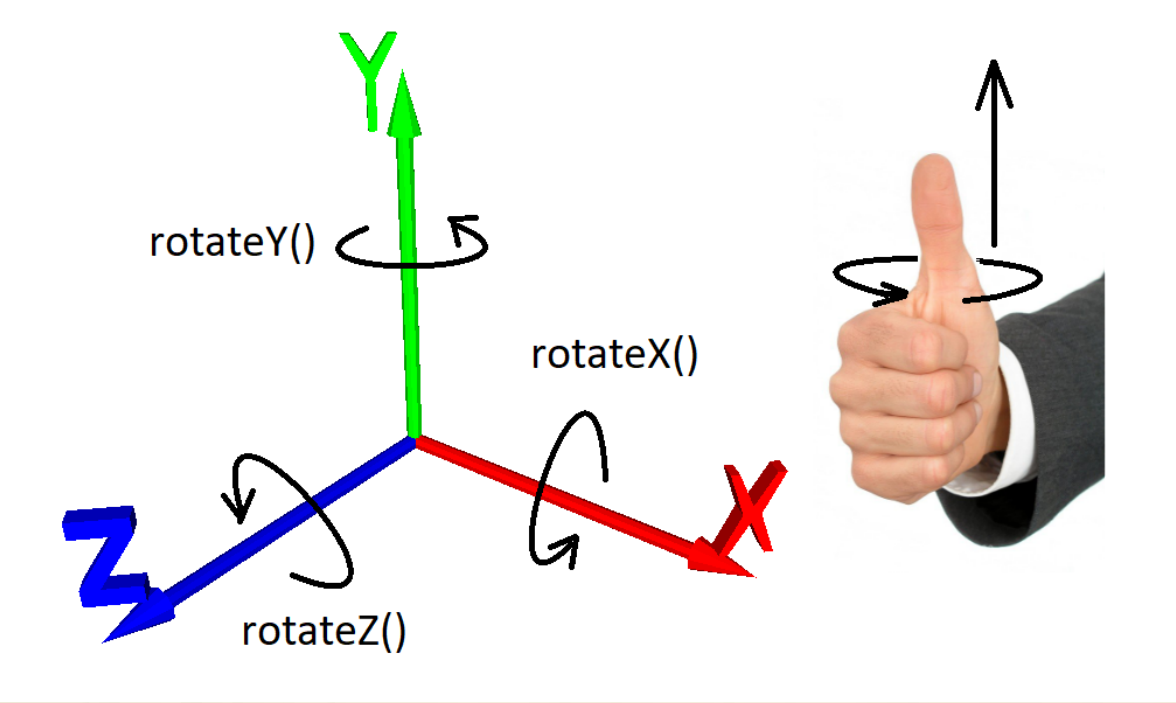
def mesh(M): # 3D modelis M pridedamas prie pagrindinio sluoksnio.

1.2 modulio off.py funkcijos, grąžinančios rezultatą sluoksniuose (2)

def rotateX(M,angle,P): # grąžinamas pasuktas 3D modelis M apie X ašį, angle – posūkio kampas, P – posūkio atskaitos taškas

def rotateY(M,angle,P): # analogiškas pasukimas apie Y ašį

def rotateZ(M,angle,P): # analogiškas pasukimas apie Z ašį



1.2a modulio off.py funkcijos

def stretch(M,s): # funkcija labai panaši į zoom(M,s), tik čia s parametras turi 3 reikšmes s=[a,b,c], pagal kurias atitinkamai keičiamas mastelis X, Y ir Z ašių kryptimis.

def curve(P,min_t,max_t,grid_t,k,r,RGB,isConnected):

pagrindiniame sluoksnyje funkcija grąžina parametrinę kreivę, kur

P=P(t) – parametrinės kreivės lygtis, t ∈ [min_t, max_t],

grid_t – parametrinės kreivės detalumas t parametro atžvilgiu,

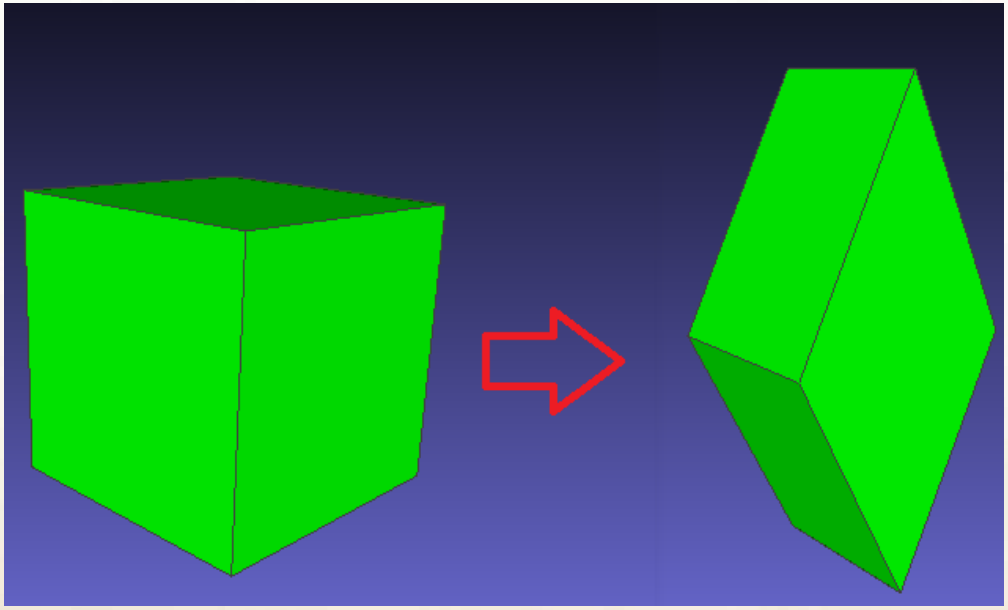
k – parametrinę kreivę sudarančių apskritimų detalumas,

r – parametrinę kreivę sudarančių apskritimų spindulys (gali būti ir kintantis spindulys r=r(t), RGB – parametrinės kreivės spalva, isConnected – parametrui priskiriamos reikšmės true arba false atitinkamai jei kreivės galai sutampa ir jei nesutampa.

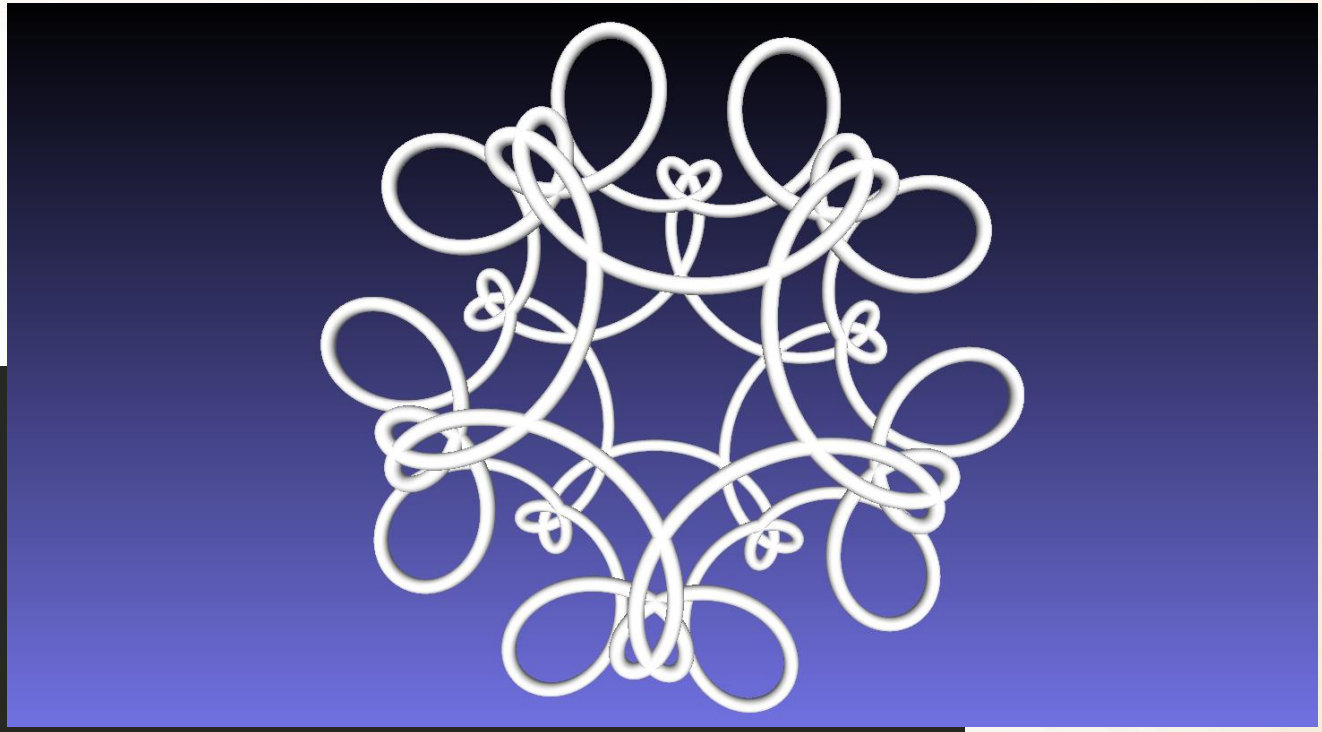
stretch(M,s) funkcija

```
1 import add
2 add.cube([0,0,0],5,[0,255,0])
3 add.off('kubas.off')
```

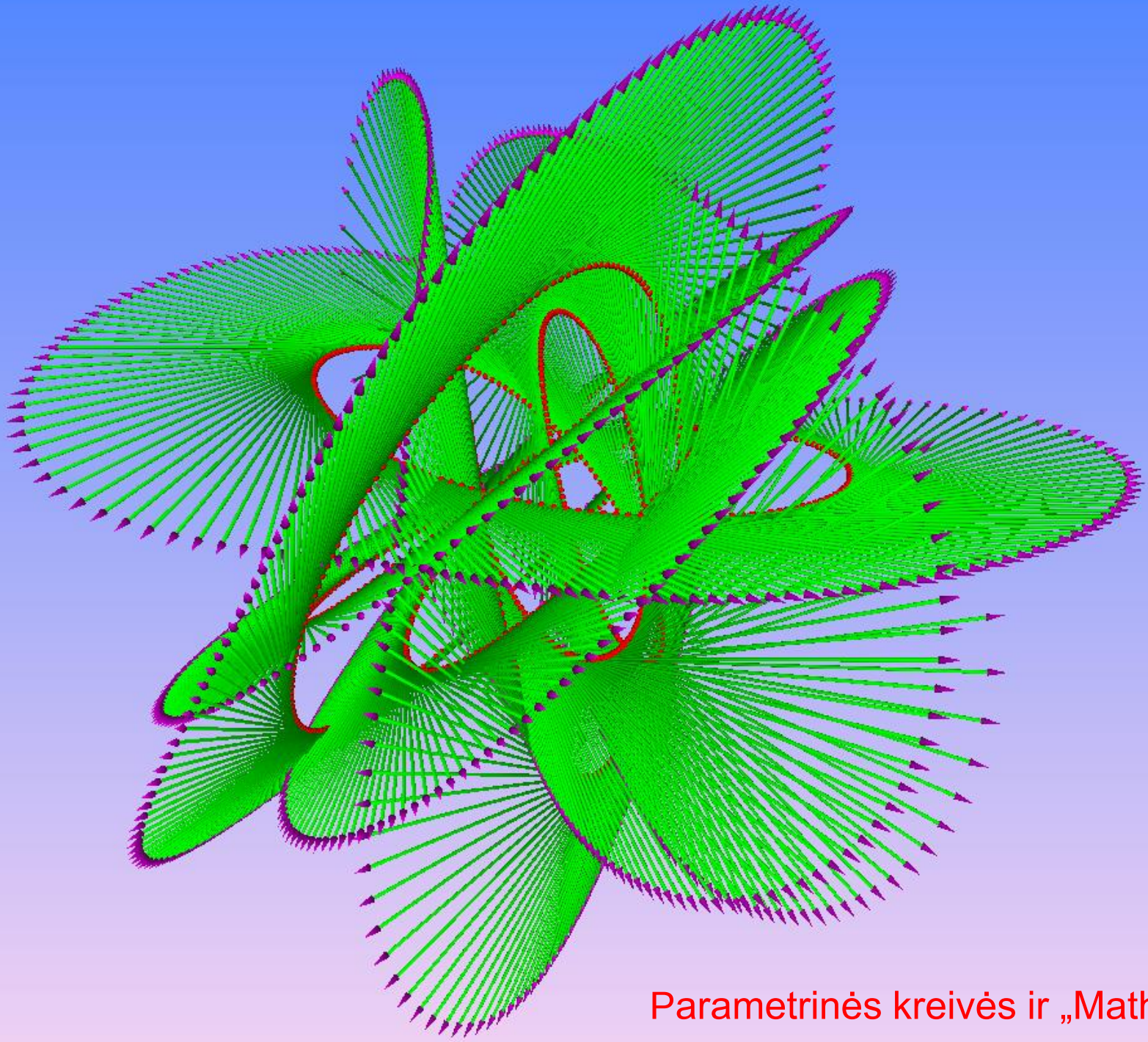
```
1 import add
2 import math
3 add.cube([0,0,0],5,[0,255,0])
4 K = add.layer()
5 K = add.rotateX(K,math.pi/4,[0,0,0])
6 K = add.stretch(K,[1,2,1])
7 add.mesh(K)
8 add.off('kubas.off')
```



curve(P,min_t,max_t,grid_t,k,r,RGB,isConnected) funkcija

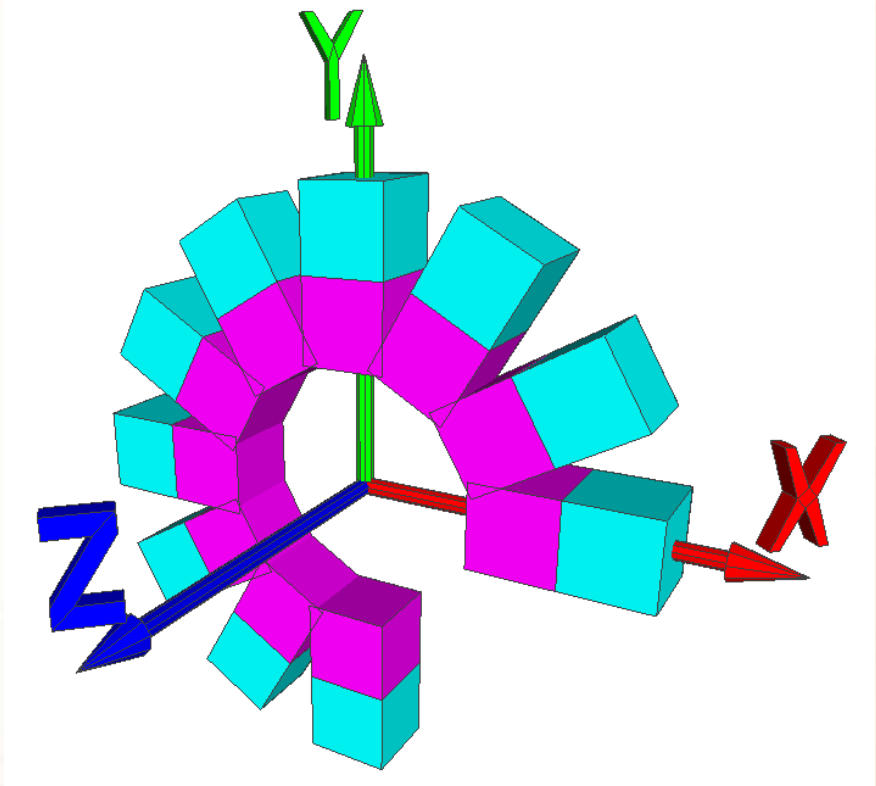


```
import add
import math
a1 = 5
b1 = 10
s1 = 0.4
v1 = -1
a2 = 2.1
b2 = 2.1
s2 = 5.2
v2 = 0.2
def P(t):
  x = a1*math.cos(s1*t)*math.sin(v1*t)+b1*math.sin(s1*t)*math.cos(v1*t)+
      a2*math.cos(s2*t)*math.sin(v2*t)+b2*math.sin(s2*t)*math.cos(v2*t)
  y = 5*math.sin(t)
  z = -a1*math.sin(s1*t)*math.sin(v1*t)+b1*math.cos(s1*t)*math.cos(v1*t)
      -a2*math.sin(s2*t)*math.sin(v2*t)+b2*math.cos(s2*t)*math.cos(v2*t)
  return [x,y,z]
add.curve(P,0,10*math.pi,3000,20,0.25,[255,255,255],True)
add.off('kreive1.off')
```



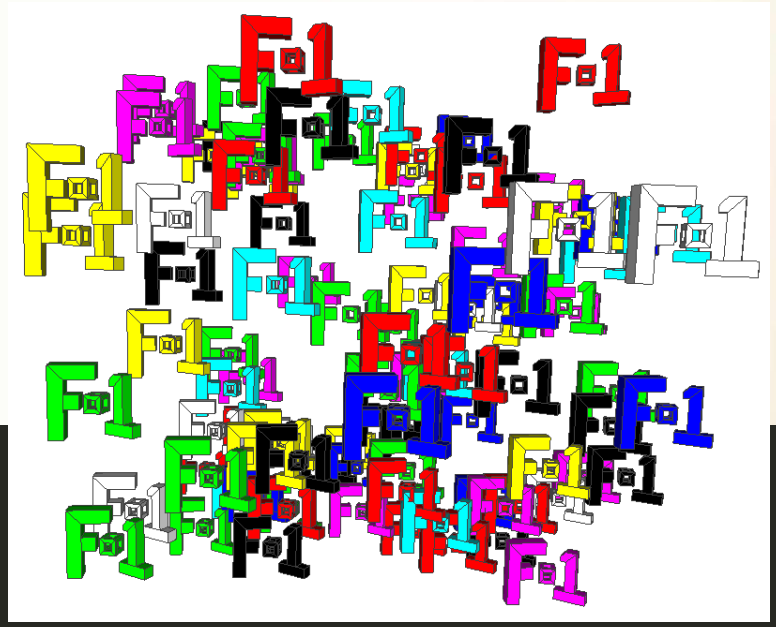
Parametrinės kreivės ir „Math art“

Pavyzdžiai (1)




```
 1 import add
 2 import math
 3 add.cube([2,0,0],1,[255,0,255])
 4 add.cube([3,0,0],1,[0,255,255])
 5 L = add.layer()
 6 add.axes([0,0,0])
 7 for i in range(10):
 8   M = add.rotateZ(L,i*math.pi/6,[0,0,0])
 9   add.mesh(M)
10 add.off('pasukimas.off')
```

Pavyzdžiai (2)

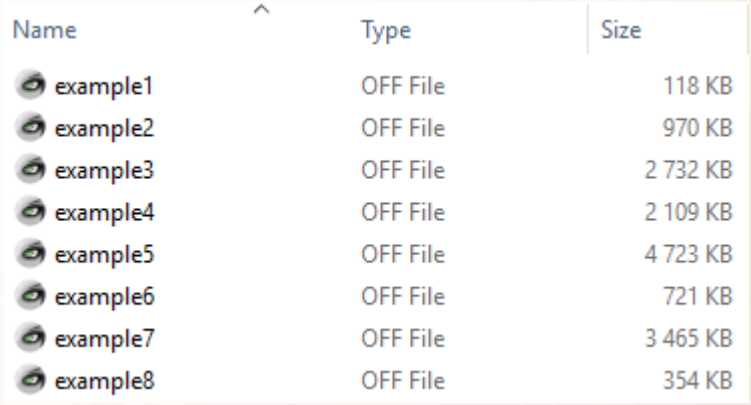


```
 1 import add
 2 import math
 3 import random
 4 add.cube2([3.2,2.5,0.5],1.2,0.3,[255,0,0])
 5 M1 = add.layer()
 6 M2 = add.load("letters/F.off")
 7 M2 = add.rotateY(M2,math.pi/10,add.center(M2))
 8 M3 = add.load("numbers/1.off")
 9 M3 = add.rotateY(M3,-math.pi/10,add.center(M3))
10 M3 = add.move(M3,[3.6,0,0])
11 M3 = add.zoom(M3,0.8)
12 M = add.merge([M1,M2,M3])
13 for i in range(100):
14   M = add.color(M,[random.randint(0,1)*255,random.randint(0,1)*255,random.randint(0,1)*255])
15   add.mesh(add.move(M,[40*random.random(),40*random.random(),40*random.random()]))
16 add.off('failas0.off')
```

*Modulio **add.py*** 3D modelių pavyzdžiai

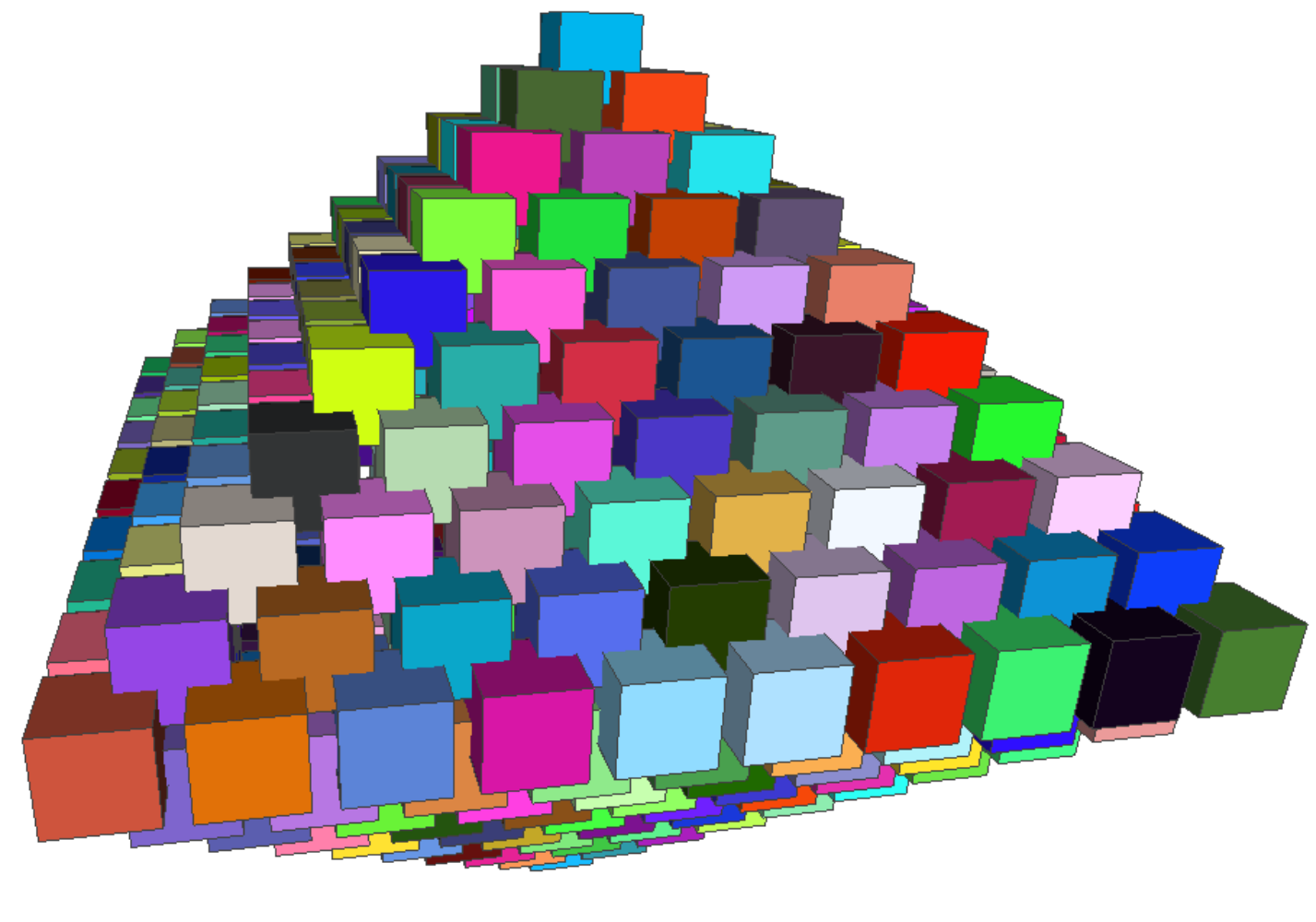
examples.py failas:

```
1  import add
2  add.example1()
3  add.example2()
4  add.example3()
5  add.example4()
6  add.example5()
7  add.example6()
8  add.example7()
9  add.example8()
```

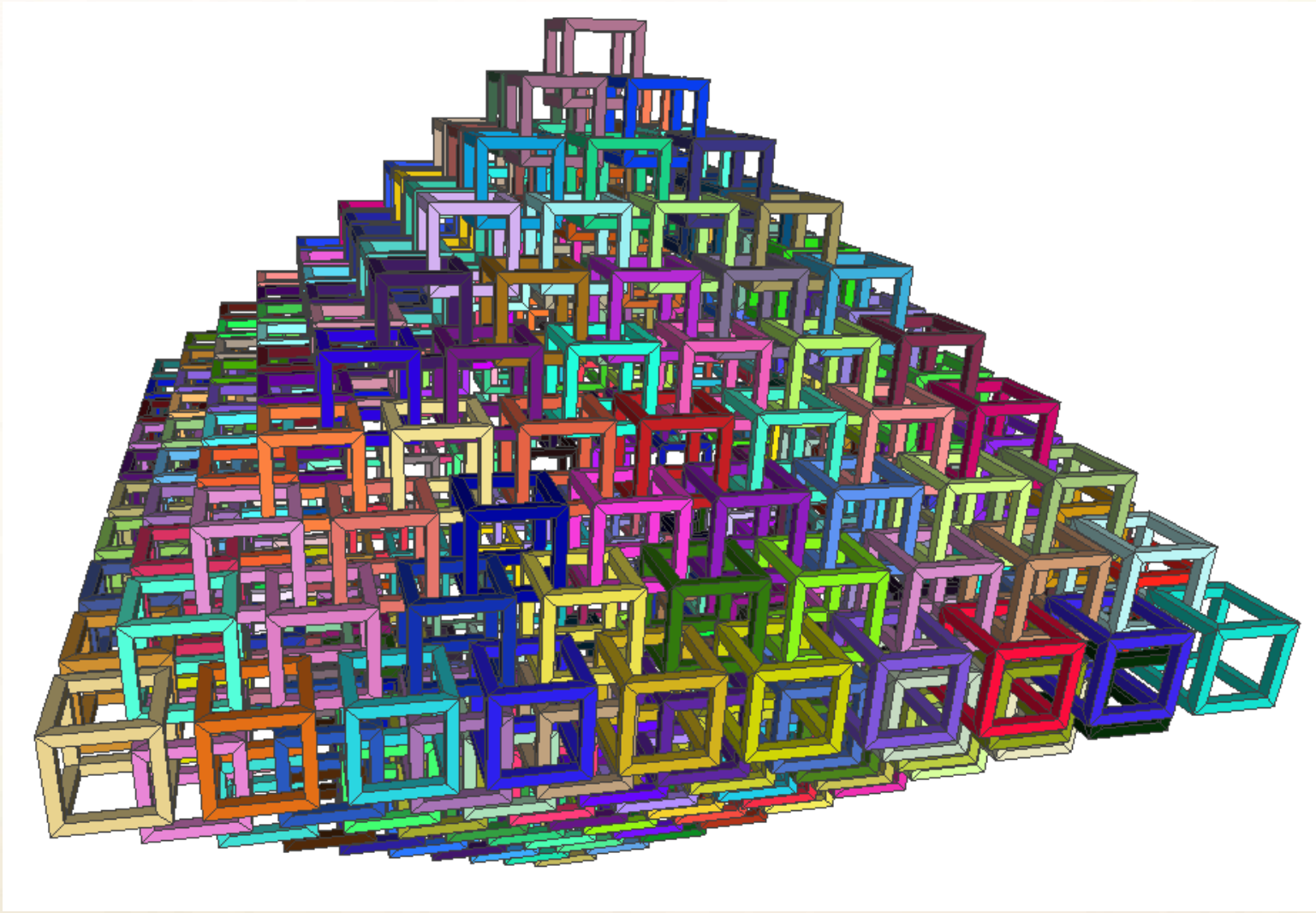


Name	Type	Size
example1	OFF File	118 KB
example2	OFF File	970 KB
example3	OFF File	2 732 KB
example4	OFF File	2 109 KB
example5	OFF File	4 723 KB
example6	OFF File	721 KB
example7	OFF File	3 465 KB
example8	OFF File	354 KB

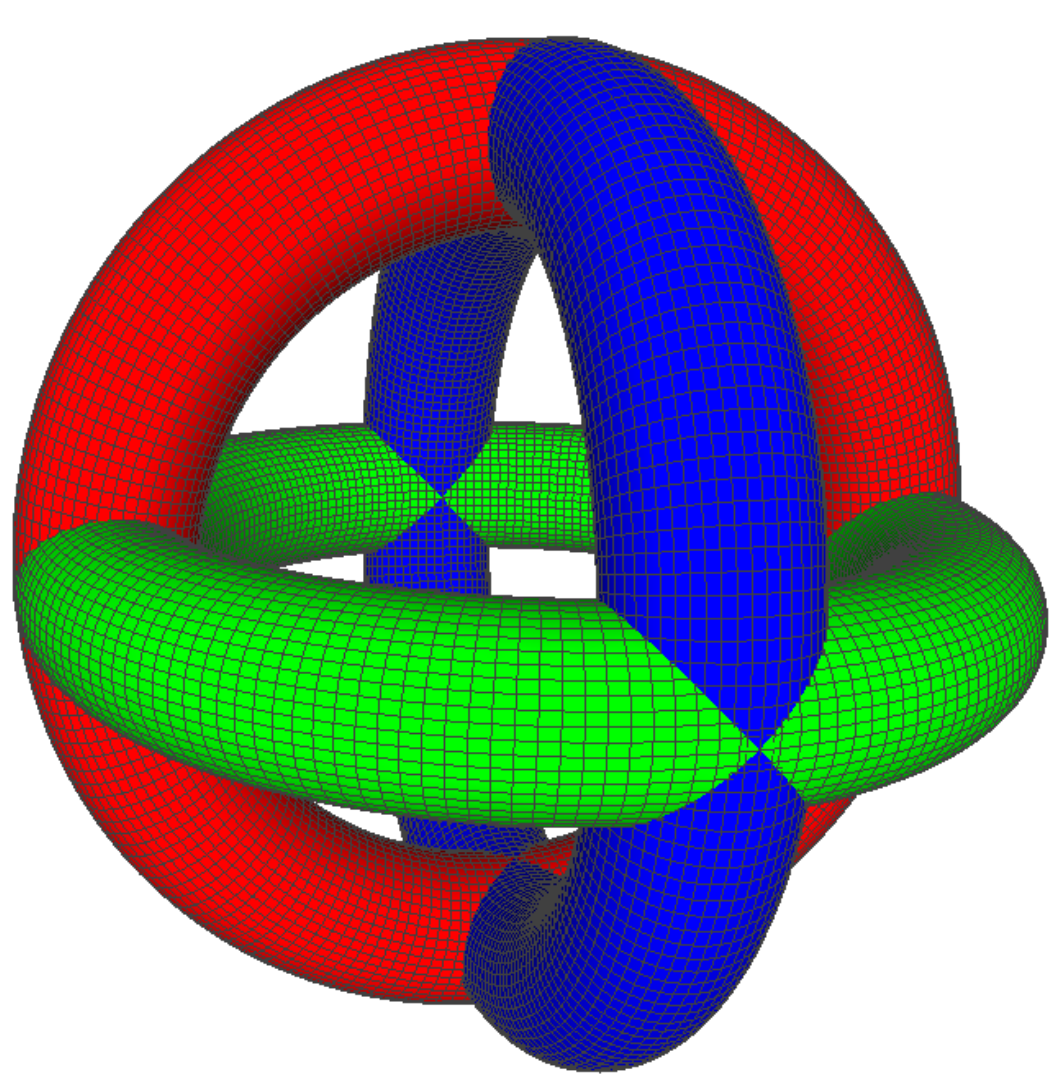
example1.off



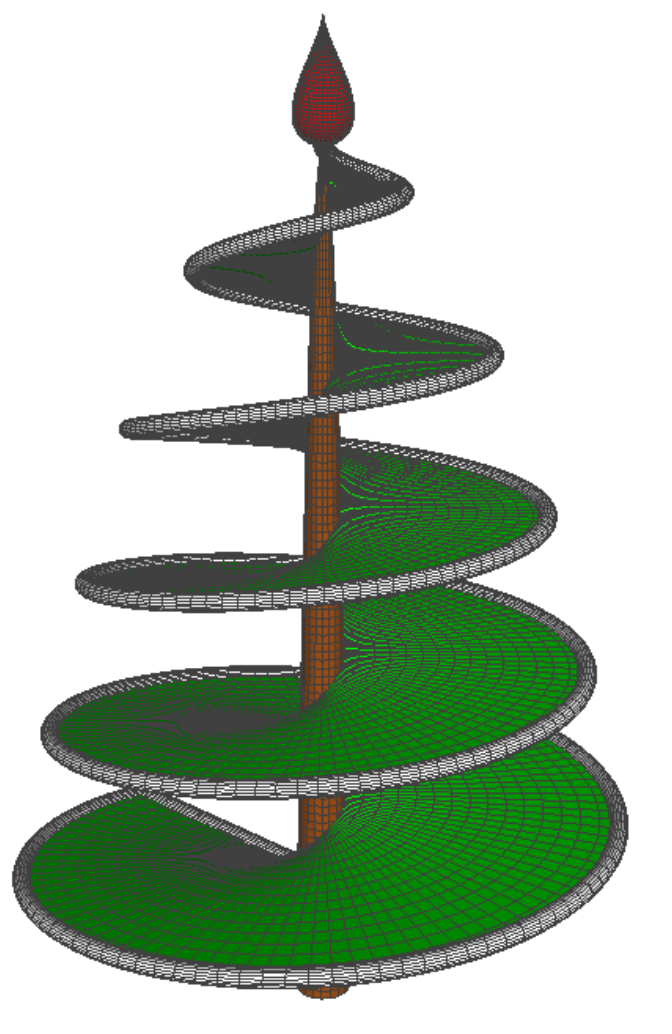
example2.off



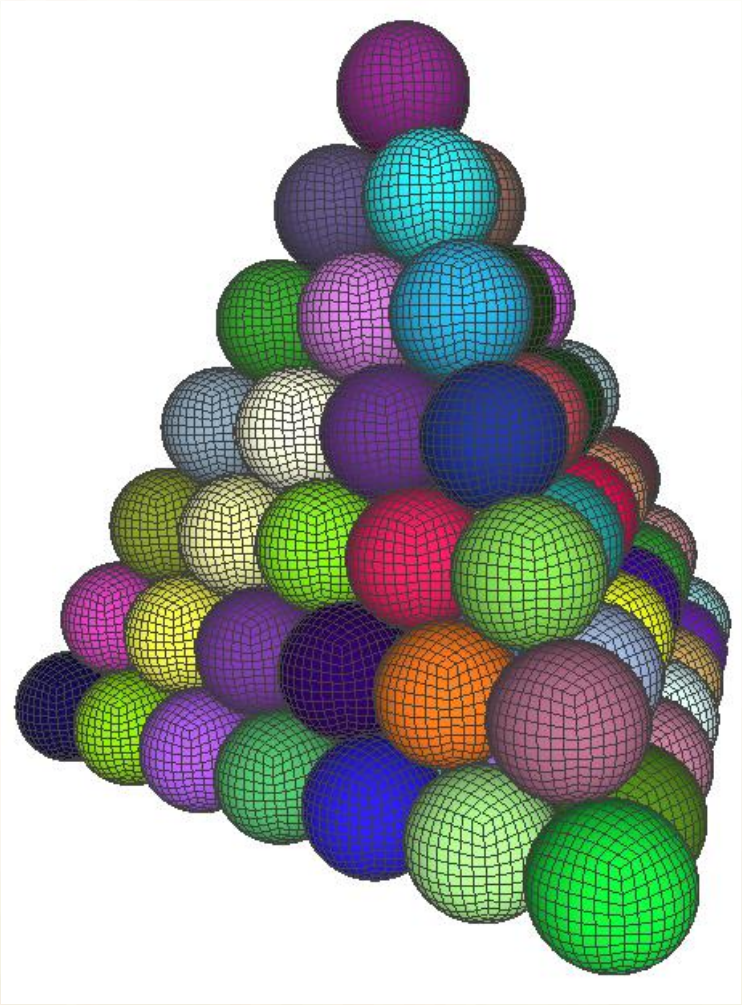
example3.off



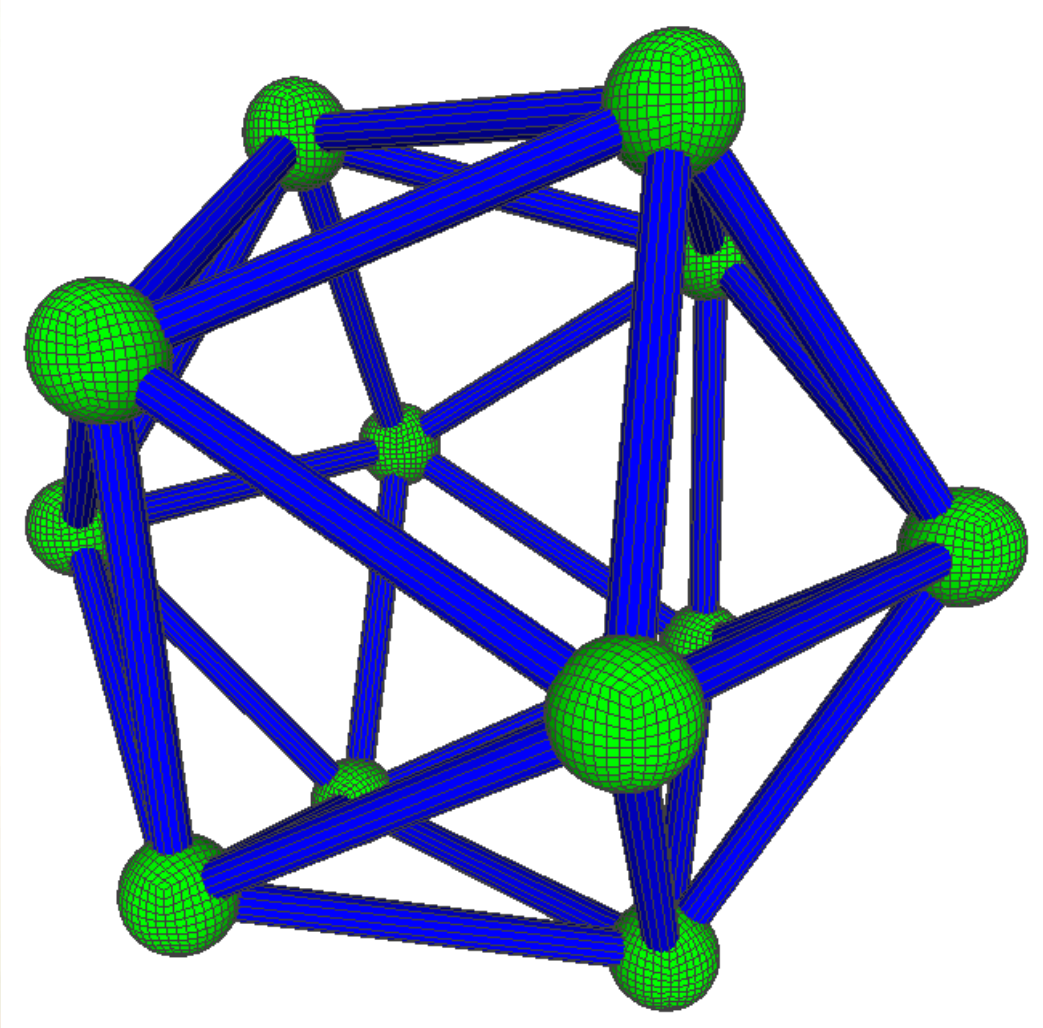
example4.off



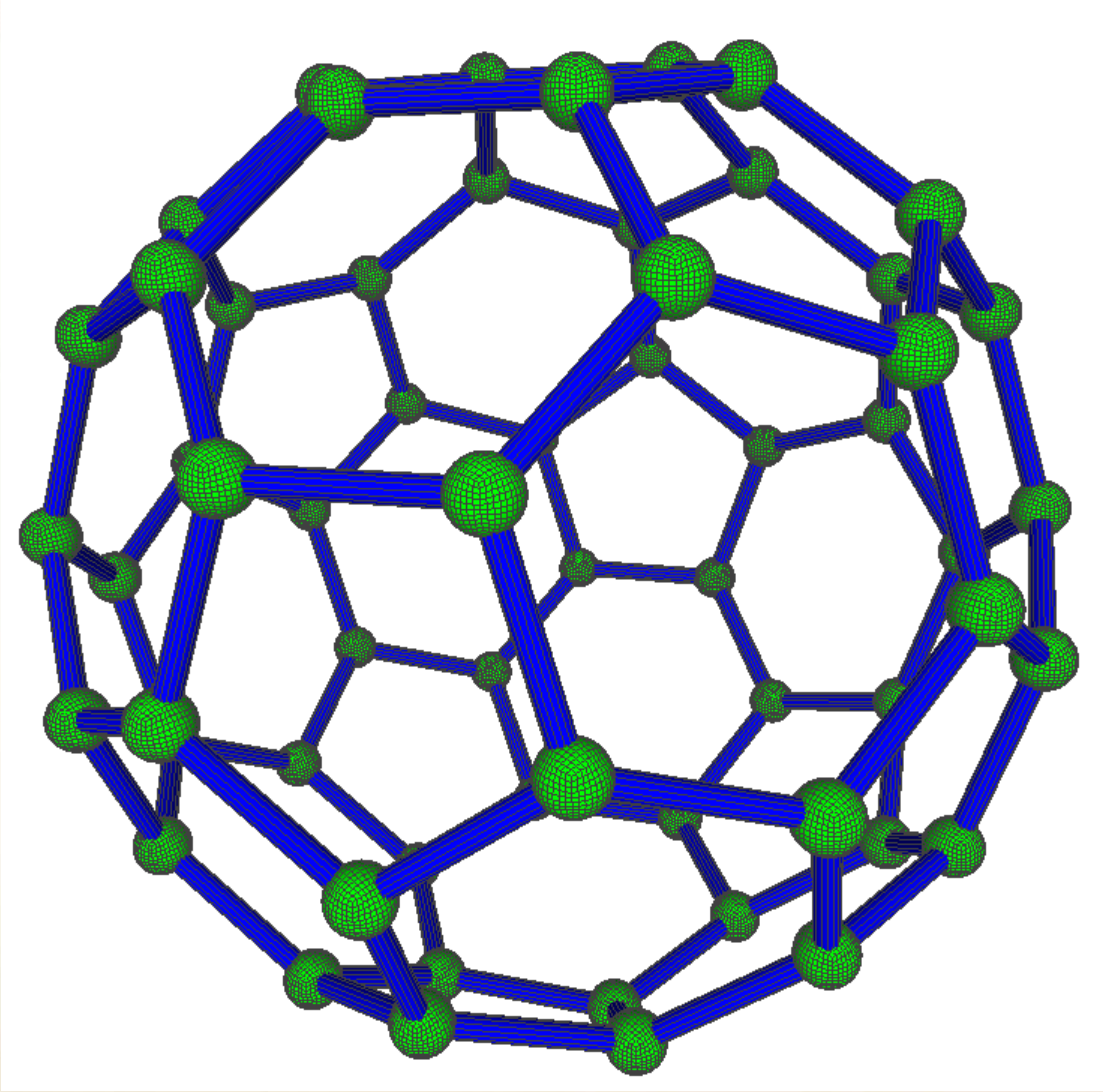
example5.off



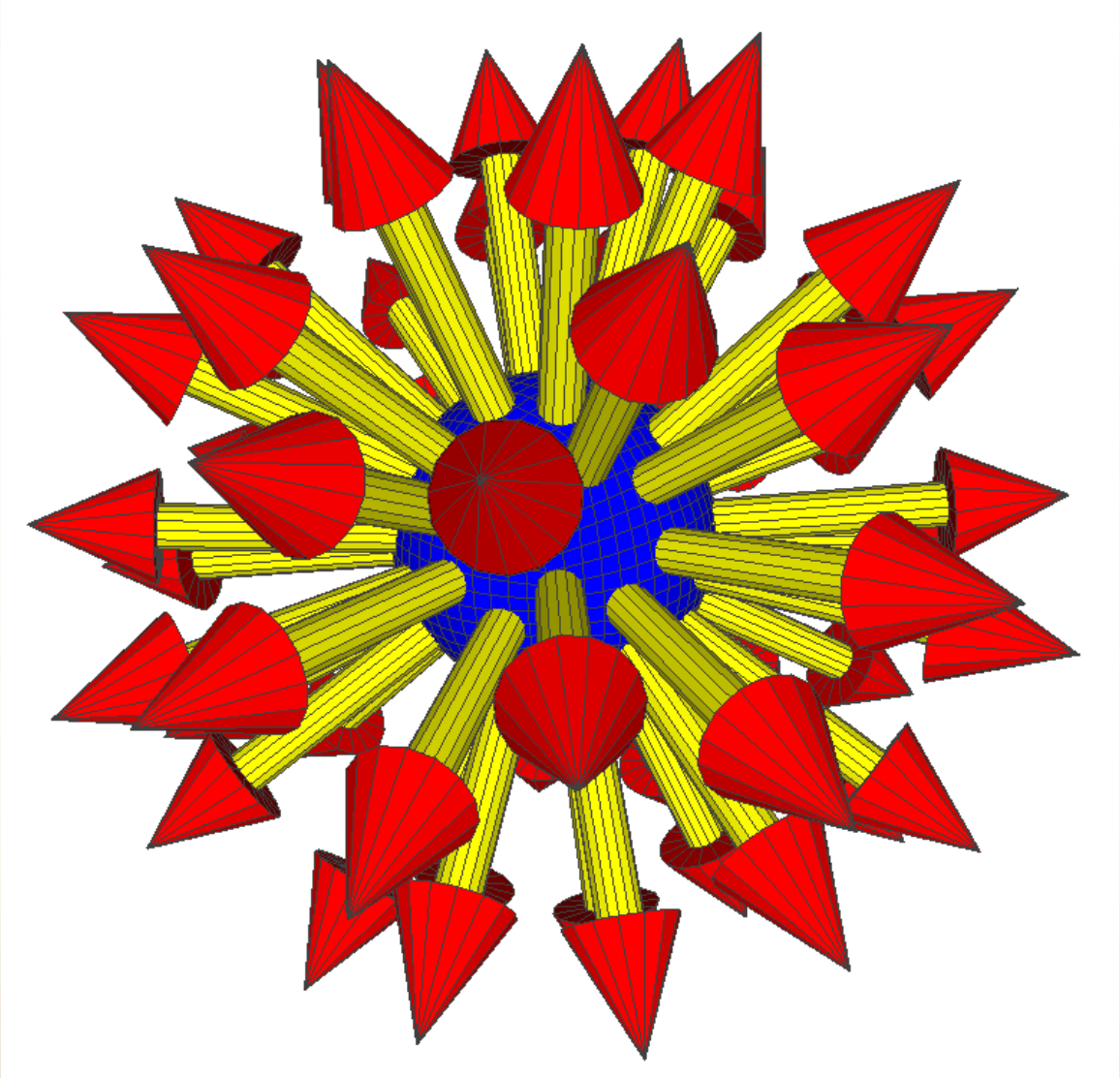
example6.off



example7.off

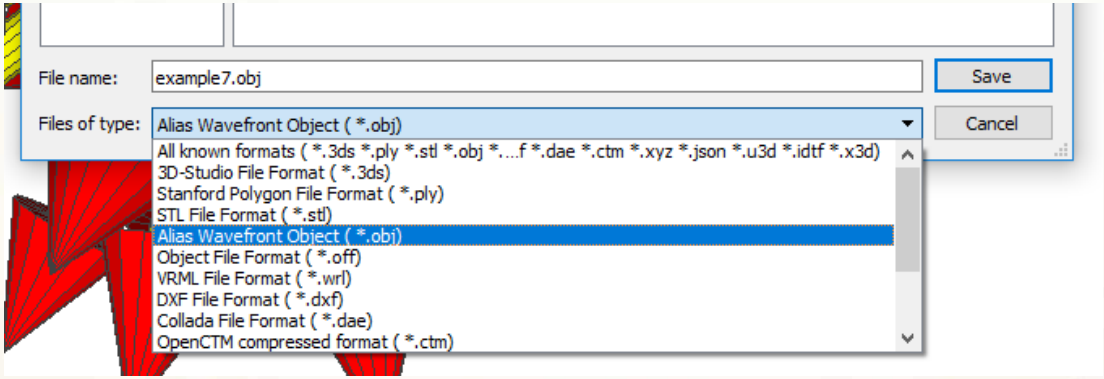


example8.off



3D modelio viešinimas https://sketchfab.com svetainėje

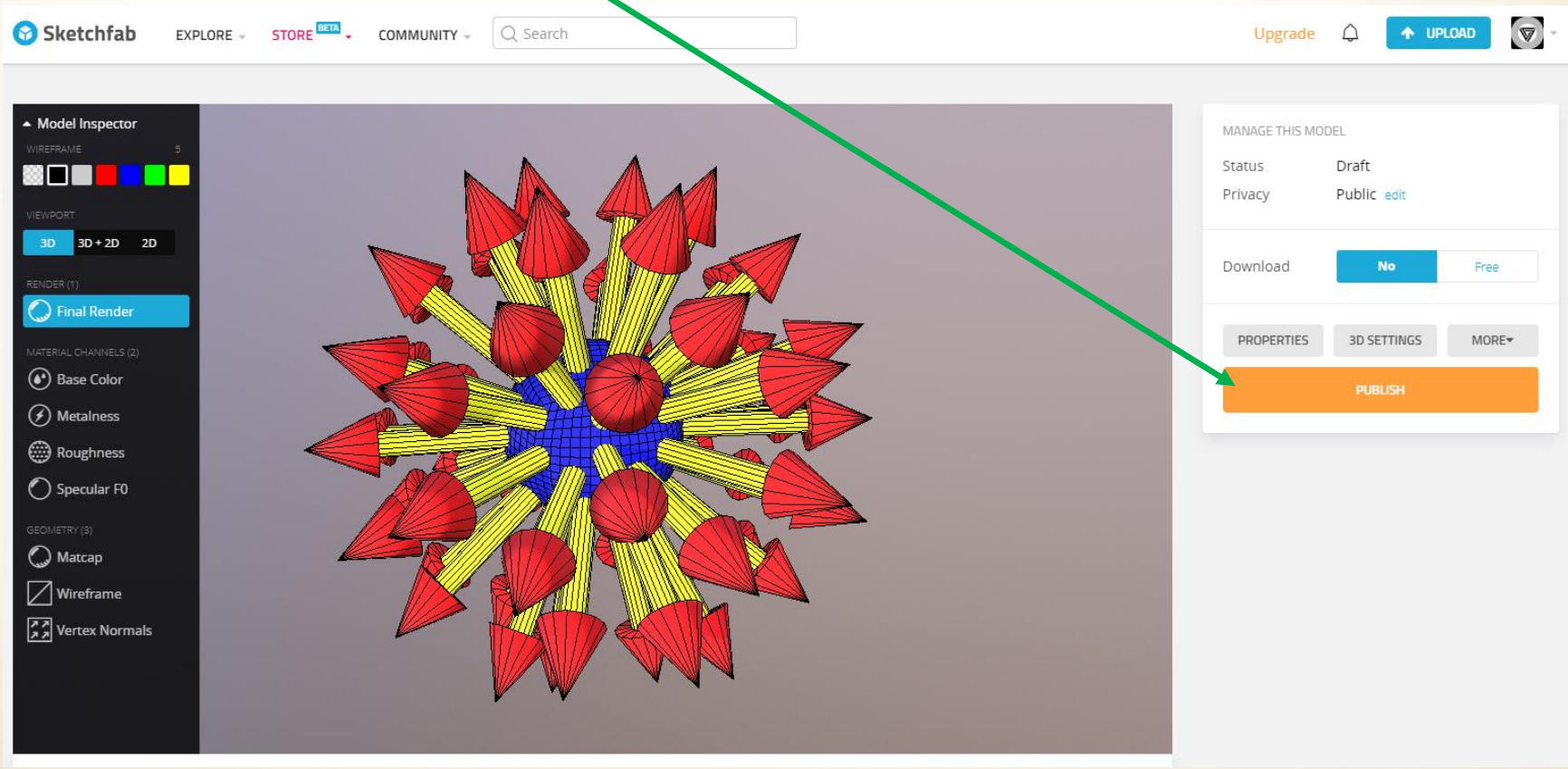
- Naudojant *MeshLab* programą 3D modelį reikia konvertuoti iš OFF formato į OBJ formatą:
  1) File → Export Mesh As...
  2) Pasirinkti *.obj formatą:



  3) Išsaugojus bus gauti 2 failai: *.obj ir *.mtl.
  4) Abu šiuos failus reikia įtraukti į 7z archyvą.
  5) Archyvą įkelti į sketchfab sistemą, susikūrus nemokamą paskyrą.

3D modelio viešinimas https://sketchfab.com svetainėje

Nustačius režimą „Public“, 3D modelis tampa viešai prieinamas, juo galima dalintis nuoroda.



Ačiū už dėmesį.