5. MODELIRANJE I REPREZENTACIJA OBJEKATA

Modeliranje - postupak izrade 3D objekata

- različiti postupci modeliranja objekata
- različiti zapisi podataka koji čine objekt (različiti postupci prikaza engl. rendering)

Postupci modeliranja objekata:

- pomoću programskih alata (CAD)
- na osnovi uzorkovanih podataka (medicinski podaci, strojarstvo, stero slike)
- proceduralno modeliranje objekata (drveće, planine, oblaci, vatra) https://demo.marpi.pl/spiders/
- fizikalno temeljeno modeliranje (tkanina, kosa, tekućine, vatra)

Gotovi programski paketi:

- za crtanje - CAD, animacije, uređivanje scene
- za obradu i prikaz podataka, Matlab
- postupci uzorkovanja objekata, pripadni programski paketi

5.1. MODELIRANJE OBJEKATA I SCENE

Modelirani objekti:

- modeliranje **površine**
  - definirana je vanjska ljuska objekta (plašt -"koža")
    - *poligonima* (*trokuti*), *prednja i stražnja strana,*
    - *parametarska površina*
    - *elementima površine* (engl. surfel) – PBG (point base graphics)
- modeliranje **volumena** tijela
  - definirana je unutrašnjost objekta (*implicitno* definirano)
    - *implicitnim funkcijama*
    - *elementima volumena* (voxel) http://mattatz.github.io/THREE.Fire/

možemo iz jednog oblika načiniti drugi (nije uvijek jednostavno)

Zapisivanje scene (strukture više razine):

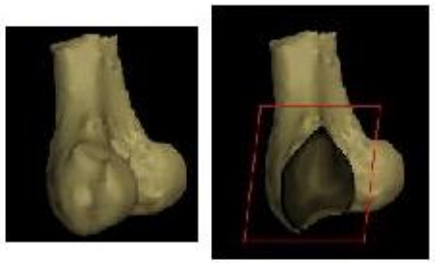
- graf scene – podaci o izvorima, promatračima, animaciji,
- specifični podaci ovisni o aplikaciji – fizikalni elementi

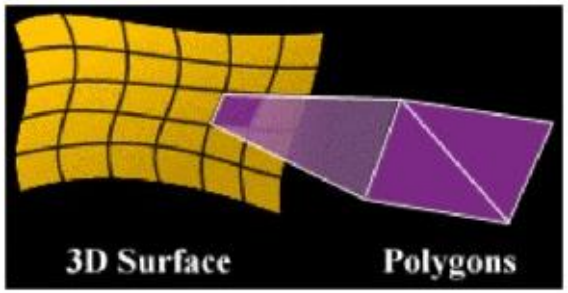
površina i volumen

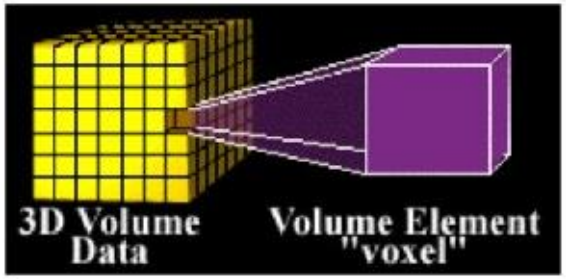
http://mikolalysenko.github.io/voxel-mipmap-demo/

„D:\Program Files (x86)\NVIDIA Corporation\NVIDIA

(Wireframe Rasterize box)







bitna razlika eksplicitnog i implicitnog oblika

- eksplicitni (trokuti, parametarski, surfeli)
  - određivanje točaka površine, tangentnih ravnina
    https://stemkoski.github.io/MathBox/graph3d-deriv.html
- implicitni (voxel, CSG- Constructive solid geometry)
  - jednostavno možemo odrediti

  pripada li neka točka površini je li “iznad” ili “ispod”,

  udaljenost od površine

  booleove operacije nad tijelima, detekcija sudara

Usporedba eksplicitnog i implicitnog načina na jednadžbi ravnine poligon (trokut) definira jednadžbu ravnine

- eksplicitni - parametarski oblik jednadžbe ravnine (površina, tangenta)

$$\mathbf{V} = \begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} a_1 & a_2 & a_3 & a_4 \\ b_1 & b_2 & b_3 & b_4 \\ c_1 & c_2 & c_3 & c_4 \end{bmatrix}$$

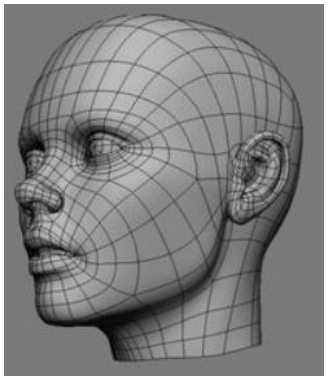
- implicitni oblik jednadžbe ravnine (udaljenost od površine, Boolove operacije, sudari)

$$ax + by + cz + d = 0$$

- eksplicitni način - površina objekta zadana poligonima

  - poligonalni model BREP (*boundary representation*)
  - žična forma objekta
    - geometrijski podaci (položaj vrhova)
    - topološki podaci (povezanost vrhova –poligoni)

https://threejs.org/examples/webgl_materials_wireframe.html



- eksplicitni način - površina zadana patametarski

  modeliranje površine (engl. surface modelling)
  - površina je **glatka** i kontinuirano se može kontrolirati (obrada plohe)

  definirana je površinska ljuska tijela (može biti zatvorena)
  - slobodno oblikovane površine (engl. Freeform surfaces**)**
    - Bezierove površine, NURBS (B-površine),

    - modeliranje plohe objekta http://mattatz.github.io/unity/teddy/
    - rotacione plohe https://mathinsight.org/applet/surface_revolution

      http://math.hws.edu/graphicsbook/source/threejs/curves-and-surfaces.html (lathe) http://inear.se/lathe/

      https://webglfundamentals.org/webgl/webgl-3d-lathe-step-01.html

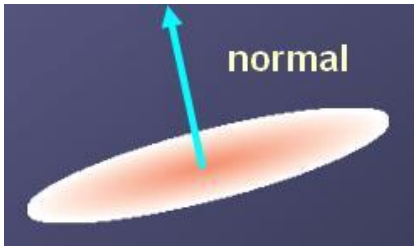
      https://webglfundamentals.org/webgl/webgl-3d-lathe-step-03.html
    - parametarski zadana površina

      https://stemkoski.github.io/Three.js/Graphulus-Surface.html Parameters

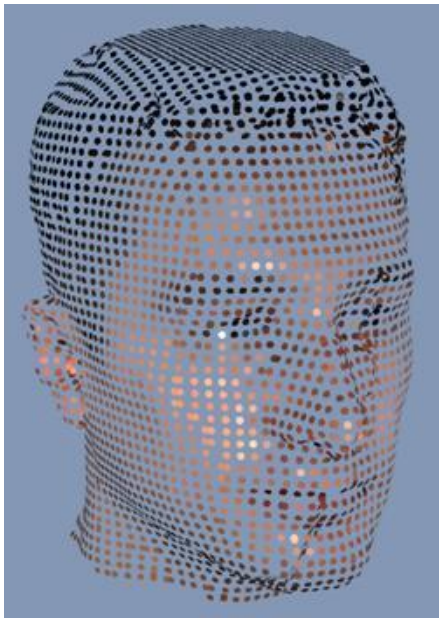
- eksplicitni način - površina zadana točkama (surfeli)

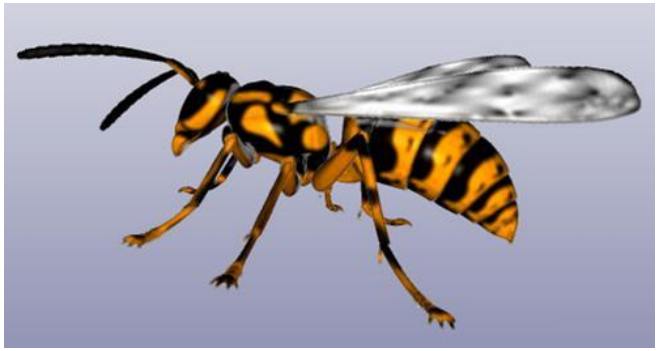
modeliranje elementima površine PBG (engl. point base graphics), fotogrametrija

  - ‘+’ sklopovska podrška za brzu izradu prikaza
  - ‘-’ pojava šupljina na rezultatu, alias
  - surfeli se projiciraju na zaslon (splatting) s rekonstrukcijskom jezgrom ovisno o kutu između normale i promatrača
  - http://potree.org/potree/examples/lion_laz.html
  - http://potree.org/potree/examples/showcase/westend_palais.html
  - https://sketchfab.com/3d-models/hermitage-st-christine-of-lena-year-852-52834b3e16124ebb90bd8d83feb9f87c



surfel





273K surfel-a

- implicitni način - implicitnim funkcijama definiramo volumen tijela

modeliranje volumena tijela (engl. volumetric modelling, solid modelling)

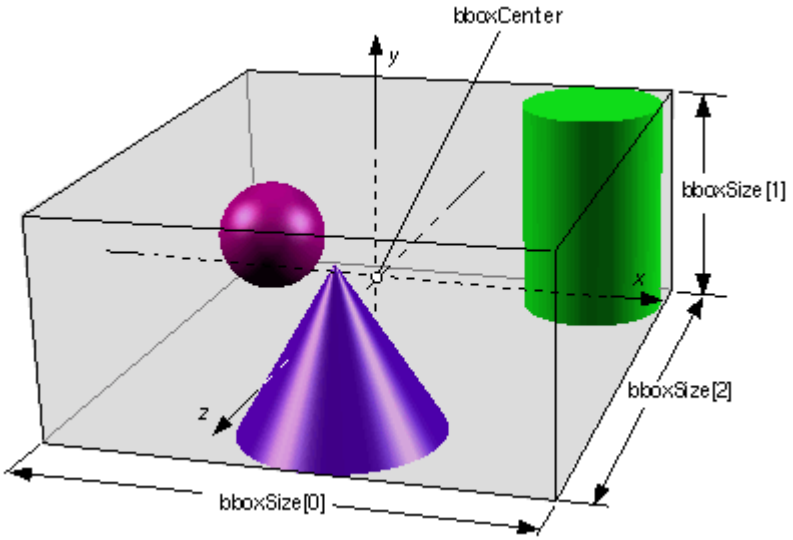
implicitno definirane površine

$$f(x, y, z) = const$$

definirana je unutrašnjost tijela SDF (engl. *Signed distance function*) npr:

unutar tijela $f(x, y, z) \leq const,$

izvan tijela $f(x, y, z) > const.$



$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = R^2$$

- implicitni način - Konstruktivna geometrija tijela

  CSG (engl. Constructive solid geometry)

  - geometrijska tijela (kugla, kocka, valjak, stožac …)

  + Booleove operacije (unija, presjek, razlika)

  - obično se koristi u CAD

- https://threejs.org/examples/webgl_geometry_csg.html
- https://stemkoski.github.io/Three.js/CSG.html
- http://evanw.github.io/csg.js/more.html

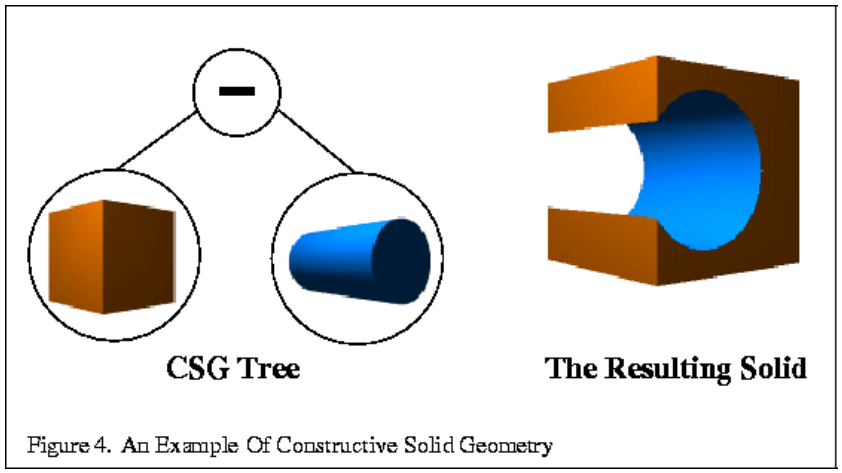
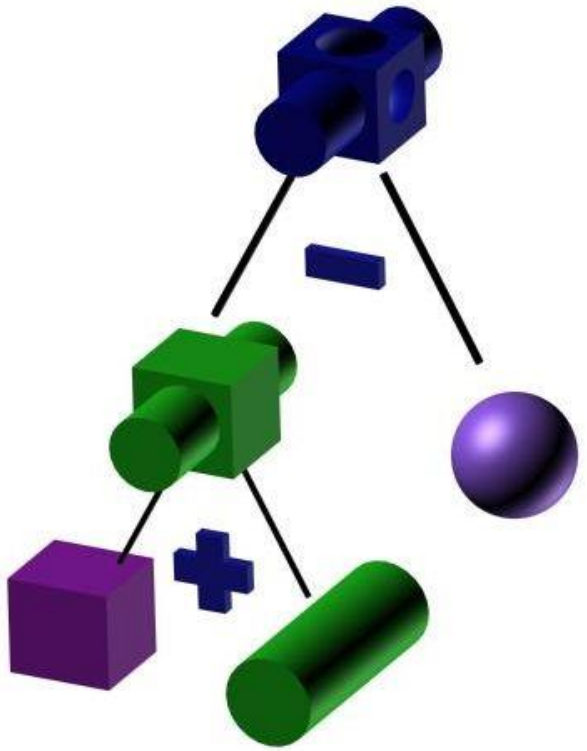


Figure 4. An Example Of Constructive Solid Geometry



– CSG stablo

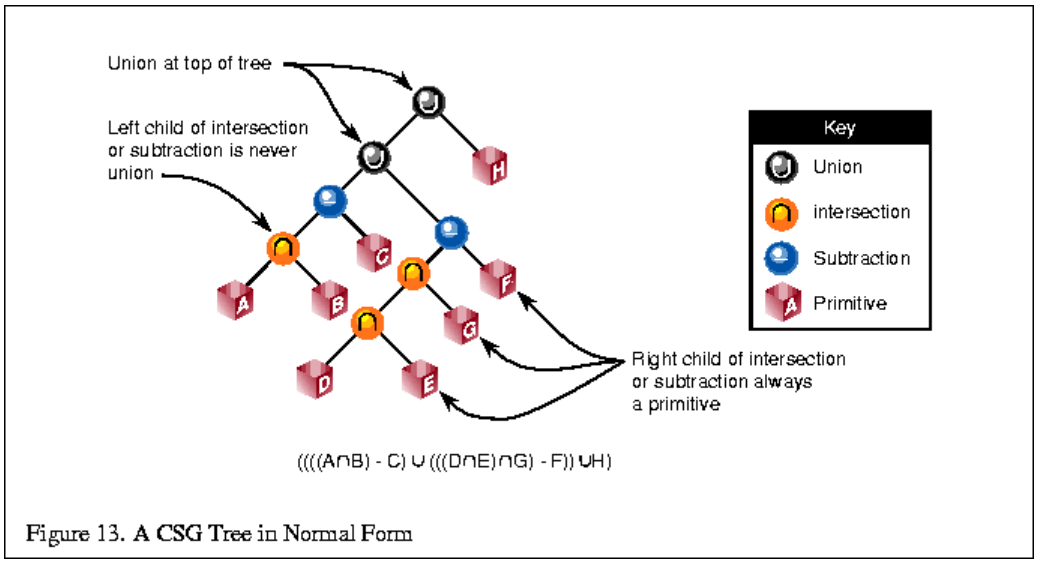
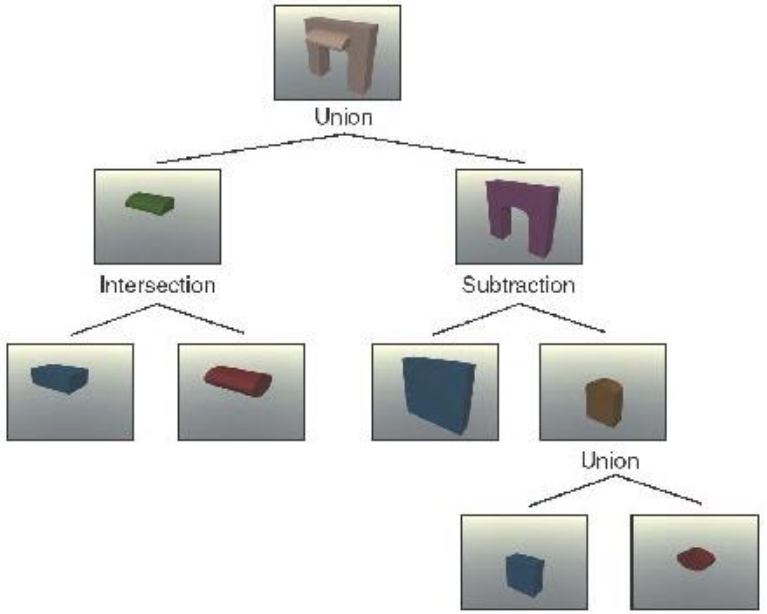


Figure 13. A CSG Tree in Normal Form

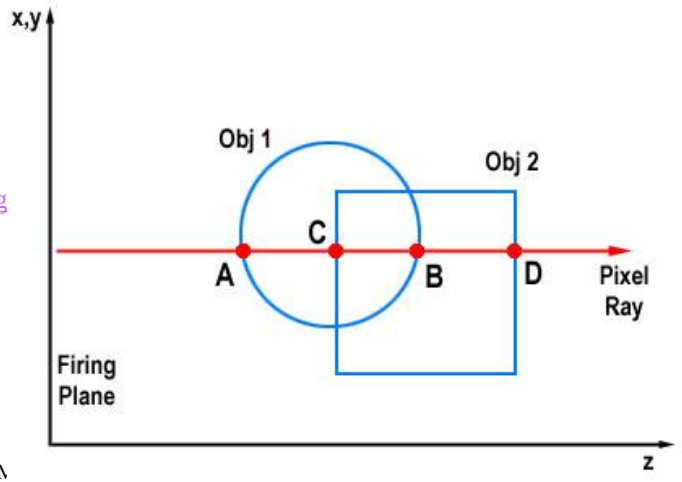


Booleove operacije

http://www.babylonjs.com/Demos/CSG/

https://evanw.github.io/csg.js/

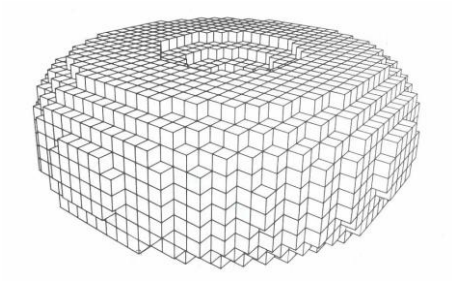
https://gkjohnson.github.io/three-bvh-csg/examples/bundle/g

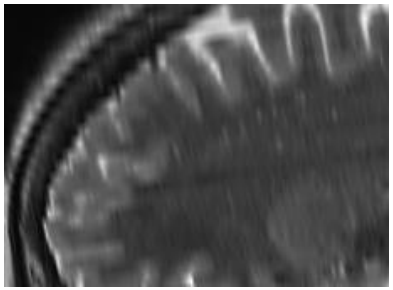


- implicitni način - volumen tijela definiran elementima volumena (engl. voxel)
  - po uzoru na slikovne elemente elementi volumena (voxel) svakoj točki prostora (x, y, z) imaju pridruženu neku vrijednost

https://brainbrowser.cbrain.mcgill.ca/volume-viewer

https://gkjohnson.github.io/three-mesh-bvh/example/bundle/voxelize.html





- podaci su obično ostvareni postupkom uzorkovanja (CT, MR)

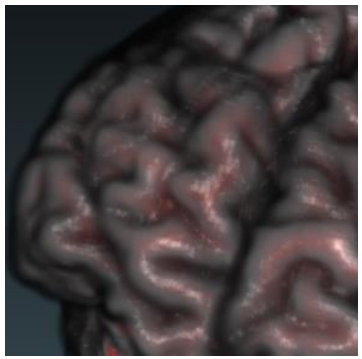
u unutrašnjosti je objekt slojevito predstavljen (kao luk)

izo - površine :

$$f(x,y,z) \le con1,$$
$$f(x,y,z) \le con2,$$
$$f(x,y,z) \le con3,\ldots$$

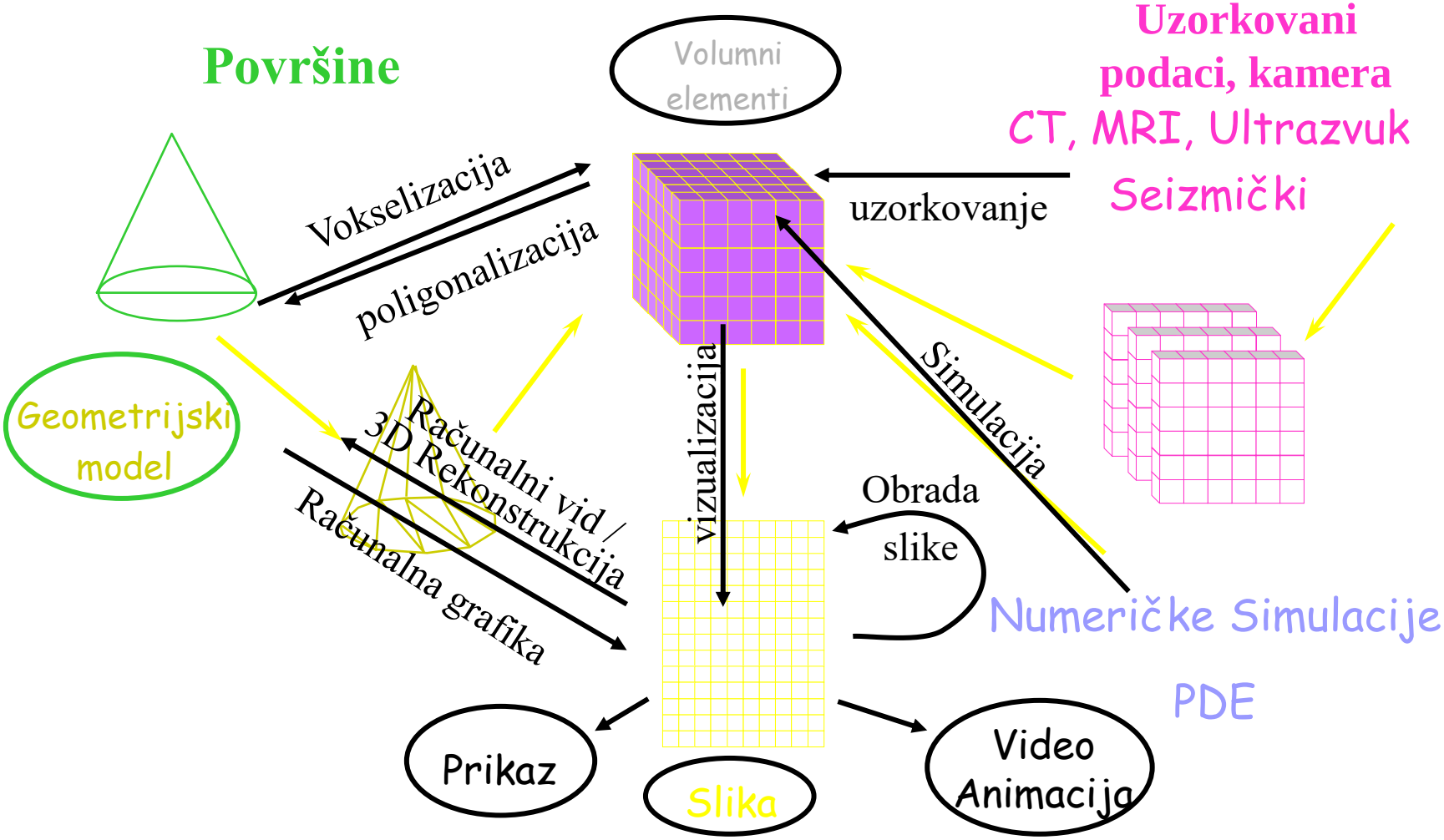
https://brainbrowser.cbrain.mcgill.ca/surface-viewer#ct

https://www.biodigital.com/



Objekti

kombinirani https://zagreb.gdi.net/zg3d/

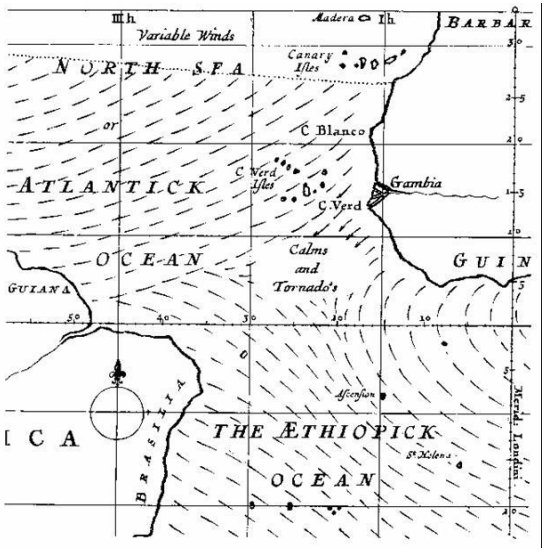


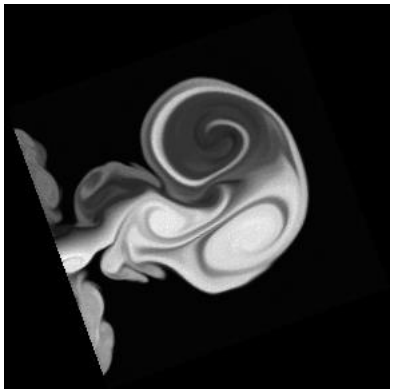
Ostvarivanje prikaza (*rendering*)

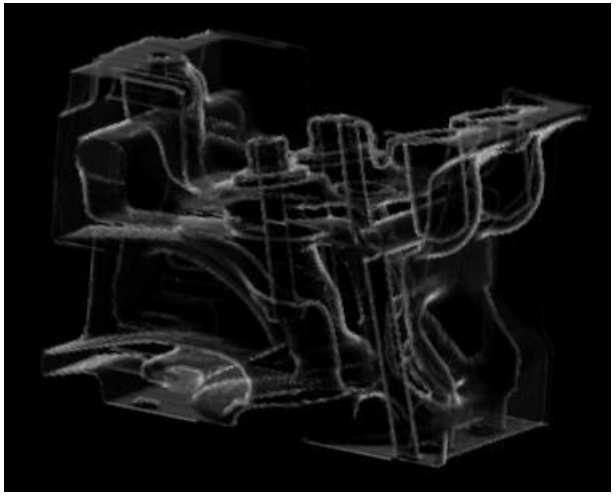
prikaz poligona – klasičan način – fotorealističan prikaz

NPR - ne fotorealističan prikaz (engl. Non-Photorealistic Rendering)

- ne želimo biti ograničeni samo na foto realističan prikaz
- skica objekta https://threejs.org/examples/?q=loader#webgl_loader_mmd
- tehnika prikaza primjenjiva na objekte definirane volumno i površinom
  http://www.clicktorelease.com/code/cross-hatching/





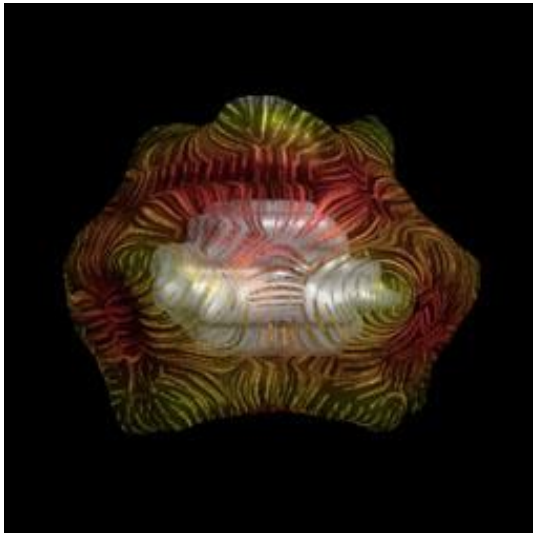


prijenosna funkcija (engl. transfer function)

- određuje što će biti i na koji način preslikano u optičke parametre

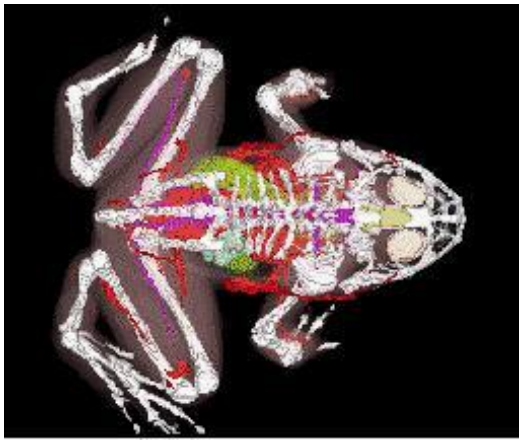
to može biti boja objekta izravno, no može biti i neka druga informacija

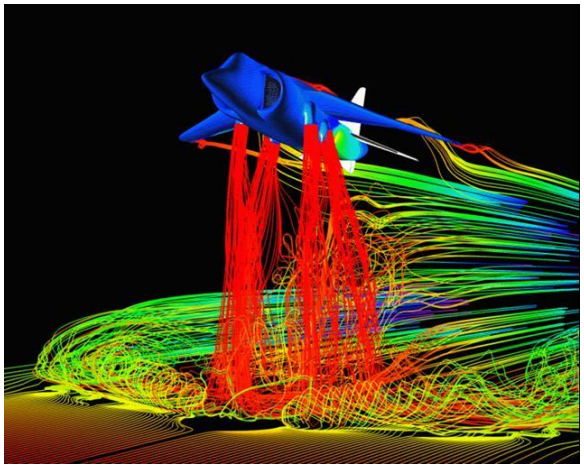
kod NPR tehnike to je mjesto gdje je normala na površinu okomita prema vektoru prema promatraču

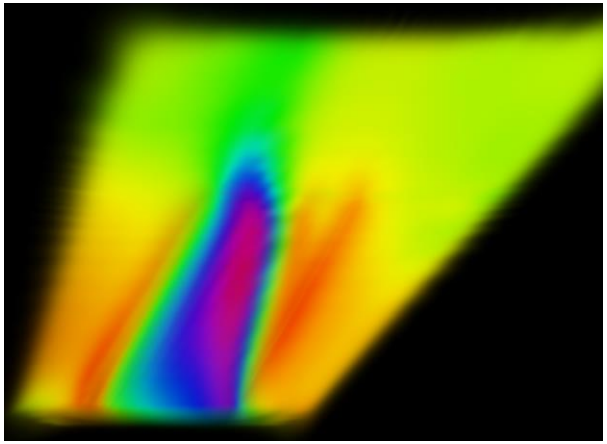


upotreba boje za prikaz dodatne informacije

- razlikovanje dijelova objekta – odjeljivanje cjelina
- razna svojstva objekta u pojedinoj točki (temperatura, brzina)







Složeni zapis objekta

- isti objekt nam je često potreban u različitim razinama složenosti (LOD - Level of detail)
  - prikaz detaljnosti objekta ovisno o udaljenosti i veličini prikaza
  - proračun sudara (kolizije)

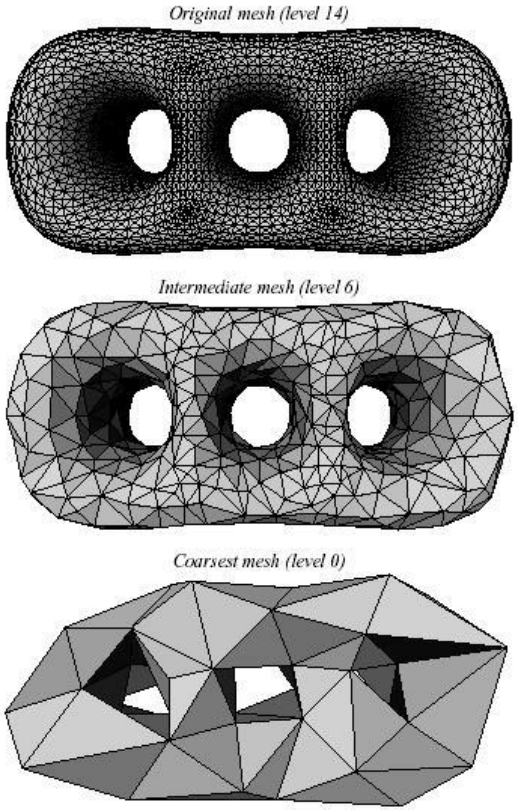
  # kompromis potrebna memorija / vrijeme izračuna
  možemo unaprijed odrediti ili dinamički izračunavati

- ugrubljivanje
  - krećemo od najsitnije podjele

instanced_tessellation.exe - Shortcut.lnk
http://www.realtimerendering.com/erich/udacity/exercises/unit3_spec_tessellation_demo.html

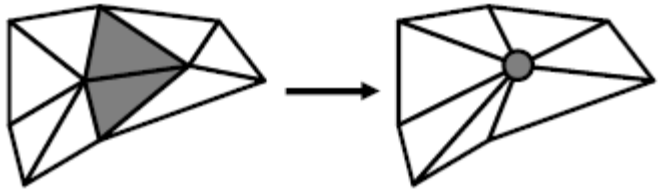
http://threejs.org/examples/#webgl_lod https://examples.x3dom.org/pop-pg13/happy-instance

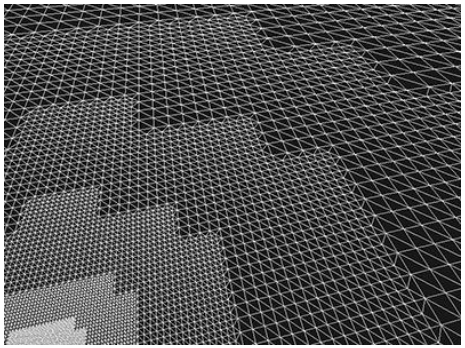
broj vrhova 50 500 2 000



- ugrubljivenje poligonalne mreže
  - spajamo poligone u veće tako da važna obilježja objekta budu sačuvana
  - stapamo vrhove – **kontinuirana** promjena

http://felixpalmer.github.io/lod-terrain/ https://examples.x3dom.org/pop-pg13/happy_color-lod.html



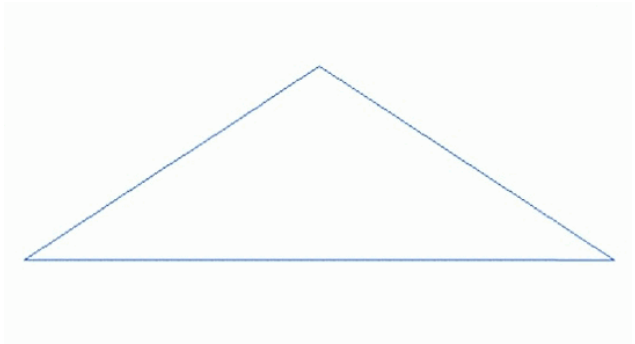


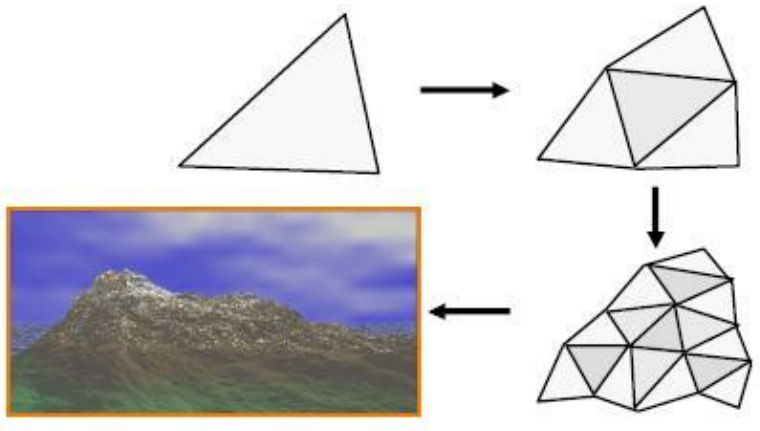
- usitnjavanje poligonalne mreže (engl. *subdivision*)
  - dijelimo poligone najgrublje razine i novonastale vrhove pomičemo (tako da novi objekt bude gladak, npr. algoritam *Catmull-Clark*)
    http://cjcliffe.github.io/CubicVR.js/cubicvr/samples/subdivision/catmull-clark.html

http://stephaneginier.com/sculptgl/ (Topology-Subdivide)

https://threejs.org/examples/#webgl_modifier_subdivision

https://lowpoly3d.xyz/





fraktalna podjela pri izradi planine

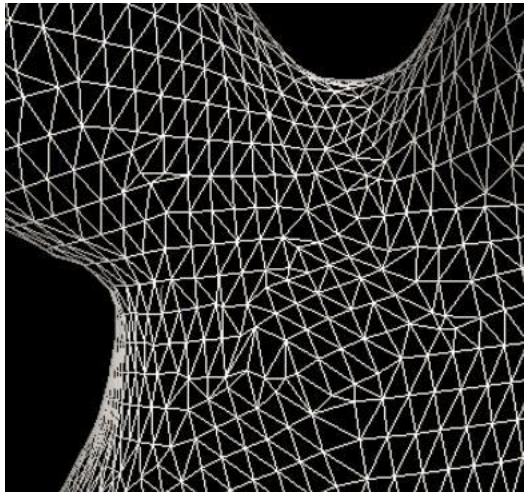
5.2 REPREZENTACIJA OBJEKATA

površina objekta – zapis poligonima odnosno trokutima

- objekte najčešće predstavljamo mrežom poligona

(samo površina objekta)

- trokuti – planarni su,
  - sklopovlje GPU podržava trokute
- kako načiniti strukture podataka
- osnovni elementi (poligonalna mreža)
  - **geometrijski podaci** – vrh (točka u prostoru)
  - **atributi** – boja, normala, koordinate teksture
  - **topološki podaci** – brid (povezuje 2 vrha)
    - poligon (povezuje više vrhova)



http://stemkoski.github.io/Three.js/Topology-Data.html

- objekti
  - geometrijski podaci, atributi, topološki podaci
  - LOD – jedan objekt može imati više poligonalnih mreža

različite složenosti – ovisno o udaljenosti prikazuju se različite mreže

5.3 STRUKTURE PODATAKA ZA ZAPIS POLIGONALNIH OBJEKATA

Objekti zadani poligonima

- ovisno o tome što je potrebno načiniti s objektima potrebno je formirati strukture podataka
  - samo prikaz i osnovne transformacije
  - modificiranje objekta (npr. izobličavanje, promjena broja poligona stapanjem vrhova)
  - ispitivanje kolizije (sudara) objekata
  - eksplozija objekta

Zapis površine objekta B-rep BREP (engl. boundary representation)

Strukture podataka

1. Tablica poligona
2. Tablice vrhova i poligona
3. Tablica bridova, vrhova i poligona
4. Liste susjednosti
5. Krilati brid

1. Tablica poligona

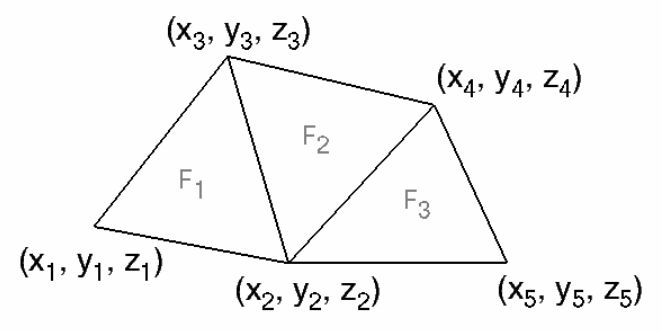
- nužno ako objekt “eksplodira” ili se poligoni rasprše, tada koordinate vrhova moraju biti posebno definirane iako su početno na istom mjestu

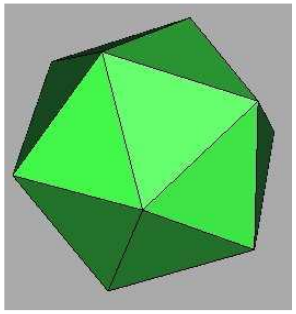
http://mattatz.github.io/DispersingTriangles/ http://edankwan.com/experiments/smashing-mega-scene/
http://oosmoxiecode.com/archive/js_webgl/stanford_bunny/

- nije efikasno ako objekt čini cjelinu, repliciramo podatke
- ako su vrhovi višestruko definirani može doći do pojave pukotina na spojevima poligona (numerička pogreška kod transformacija)
- http://pages.cs.wisc.edu/~lizy/mrdoob-three.js-ef5f05d/examples/webgl_geometry_tessellation.html (Zoom)

- redoslijed vrhova je u primjeru suprotno smjeru kazaljke na satu gledano izvan tijela CCW (određuje redoslijed bridova, određuje normalu po pravilu desne ruke)

Tablica poligona	
F_1	(x_1, y_1, z_1) (x_2, y_2, z_2) (x_3, y_3, z_3)
F_2	(x_2, y_2, z_2) (x_4, y_4, z_4) (x_3, y_3, z_3)
F_3	(x_2, y_2, z_2) (x_5, y_5, z_5) (x_4, y_4, z_4)



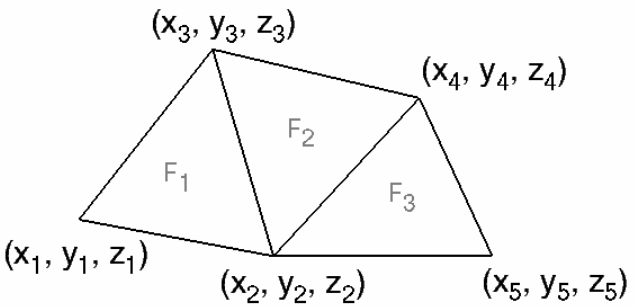


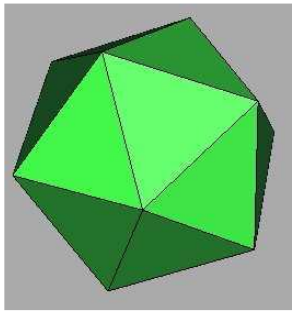
2. Tablice vrhova i poligona

- vrhovi su dijeljeni, zajednički za različite poligone, nisu replicirani,
- u tablici poligona su indeksi http://stemkoski.github.io/Three.js/Labeled-Geometry.html
- pomicanje jednog vrha izobličiti će sve poligone koji ga dijele
  http://www.blurspline.com/labs/3d/geometry_editor3.html
- razdvojena je informacija o geometriji (vrhovi) i topologiji (povezanosti - poligoni)
- nemamo informaciju o susjednosti
  - ako nas zanima za vrh V_2 koji poligoni sadrže taj vrh morati ćemo pretražiti sve poligone, ili koji poligoni čine brid B_{24}
- pogodno je što su istovrsni podaci (vrhovi) zajedno
- redoslijed vrhova u tablici poligona može biti upotrijebljen za određivanje normale
- https://oosmoxiecode.com/archive/js_webgl/sphereblob/

Tablica vrhova			
V_1	X_1	Y_1	Z_1
V_2	X_2	Y_2	Z_2
V_3	X_3	Y_3	Z_3
V_4	X_4	Y_4	Z_4
V_5	X_5	Y_5	Z_5

Tablica poligona			
F_1	V_1	V_2	V_3
F_2	V_2	V_4	V_3
F_3	V_2	V_5	V_4





3. Tablice bridova, vrhova i poligona

- tablica poligona sadrži pokazivače na bridove,
- tablica bridova sadrži pokazivače na vrhove
- redoslijed bridova određuje orijentaciju poligona, redoslijed vrhova određuje orijentaciju bridova
- http://stemkoski.github.io/Three.js/Topology-Data.html

npr: brid2 ide od V1 do V0

ako nam trebaju poligoni koji čine taj brid moramo pretražiti indekse vrhova u listi poligona

KNOT-LIST

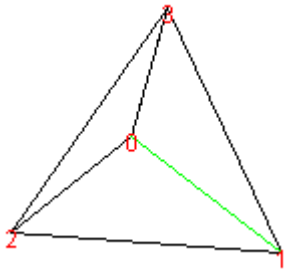
	X	Y	Z
0	0.453608	0.89043	0.037037
1	0.544331	-0.62854	-0.555555
2	-0.090722	-0.366647	0.925925
3	-0.907218	0.104757	-0.407407

EDGE-LIST

	P0	P1
0	0.0	2.0
1	2.0	1.0
2	1.0	0.0
3	1.0	3.0
4	3.0	0.0
5	3.0	2.0

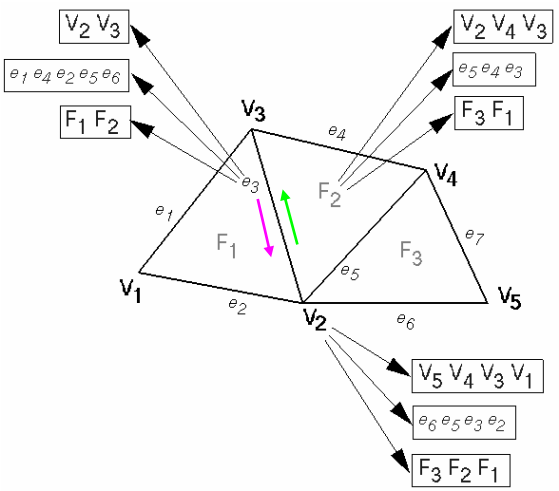
FACE-LIST

	K0	K1	K2
0	0.0	1.0	2.0
1	2.0	3.0	4.0
2	4.0	5.0	0.0
3	1.0	5.0	3.0

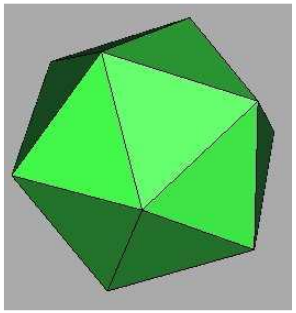


4. Liste susjednosti

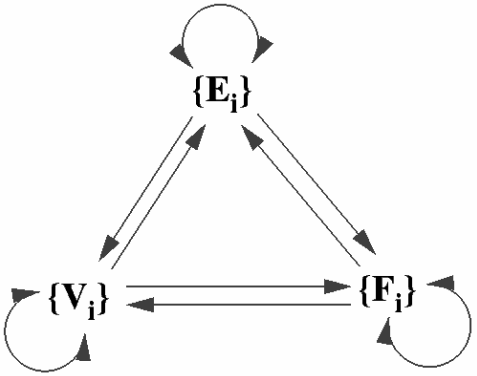
- tablica bridova
  - koji vrhovi čine brid (orijentacija brida V_2V_3, V_3V_2)
  - koji bridovi su susjedni (diraju prvi ili drugi vrh)
  - koji poligoni čine brid



- tablica vrhova
  - susjedni vrhovi
  - incidentni bridovi
  - incidentni poligoni



- tablica poligona
  - vrhovi
  - bridovi koji ga čine
  - susjedni poligoni



- želimo imati informaciju o susljednosti ali

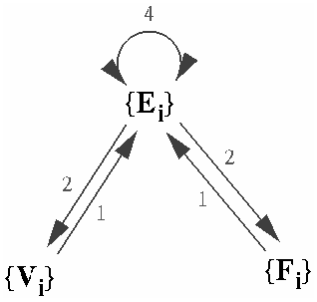
želimo pohranjivati manje podataka

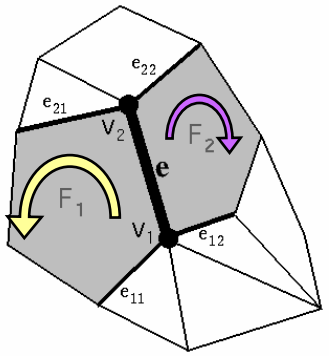
kompromis: potrebna memorija ↔ vrijeme potrebno za određivanje susjednosti

- 9 relacija susjednosti

5. Krilati brid (engl. winged edge)

- tablica bridova
  - koji vrhovi čine brid (početni, završni) V_1 V_2
  - koji poligoni čine brid (lijevi, desni) F_1 F_2
  - bridovi lijevog poligona (brid koji prethodi, brid koji slijedi) e_{11} e_{21}
  - bridovi desnog (brid koji prethodi, brid koji slijedi) e_{12} e_{22}
- tablica vrhova
  - jedan brid (bilo koji)
- tablica poligona
  - jedan brid (bilo koji)





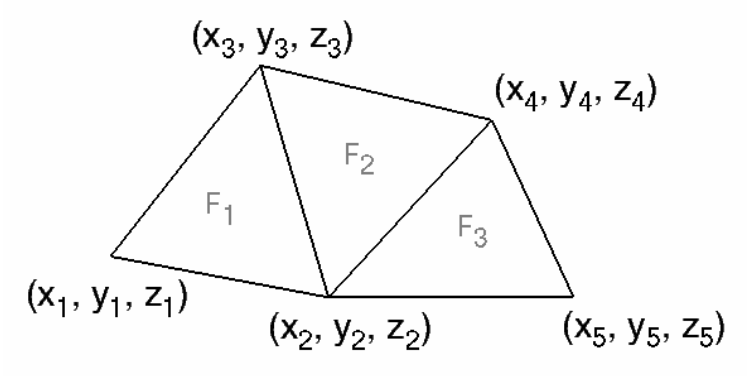
- krila brida **e** su e_{11} e_{21} e_{12} e_{22}

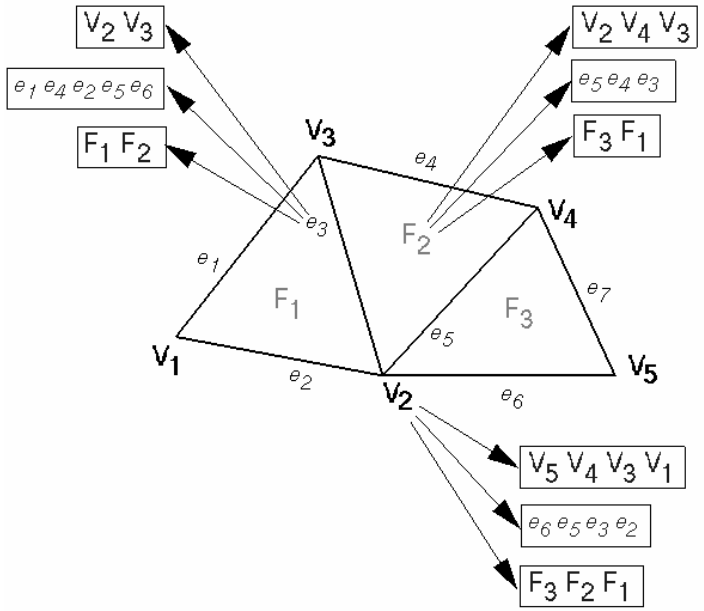
Različite varijante zapisivanja bridova

- orijentacija poligona može biti CW, CCW, određena bridom **e,**
- zapis samo 2 krila

- ispitivanje **relacije susjednosti**:
  - da li je vrh V_1 susjedan poligonu F_3?
  - da li su poligoni F_1 i F_3 susjedni?
- proizvoljni poligoni (nisu nužno trokuti)

primjer: krilati brid (engl. winged edge)





- redoslijed krila brida određen je bridom **e**

Tablica vrhova

V_1	X_1	Y_1	Z_1	e_1
V_2	X_2	Y_2	Z_2	e_6
V_3	X_3	Y_3	Z_3	e_3
V_4	X_4	Y_4	Z_4	e_5
V_5	X_5	Y_5	Z_5	e_6

Tablica bridova

					11	12	21	22
e_1	V_1	V_3		F_1	e_2	e_2	e_4	e_3
e_2	V_1	V_2	F_1		e_1	e_1	e_3	e_6
e_3	V_2	V_3	F_1	F_2	e_2	e_5	e_1	e_4
e_4	$V3$	V_4		F_2	e_1	e_3	e_7	e_5
e_5	V_2	V_4	F_2	F_3	e_3	e_6	e_4	e_7
e_6	V_2	V_5	F_3		e_5	e_2	e_7	e_7
e_7	V_4	V_5		F_3	e_4	e_5	e_6	e_6

Tablica poligona

F_1	e_1
F_2	e_3
F_3	e_5

- primjena CSG, smanjivanje broja poligona

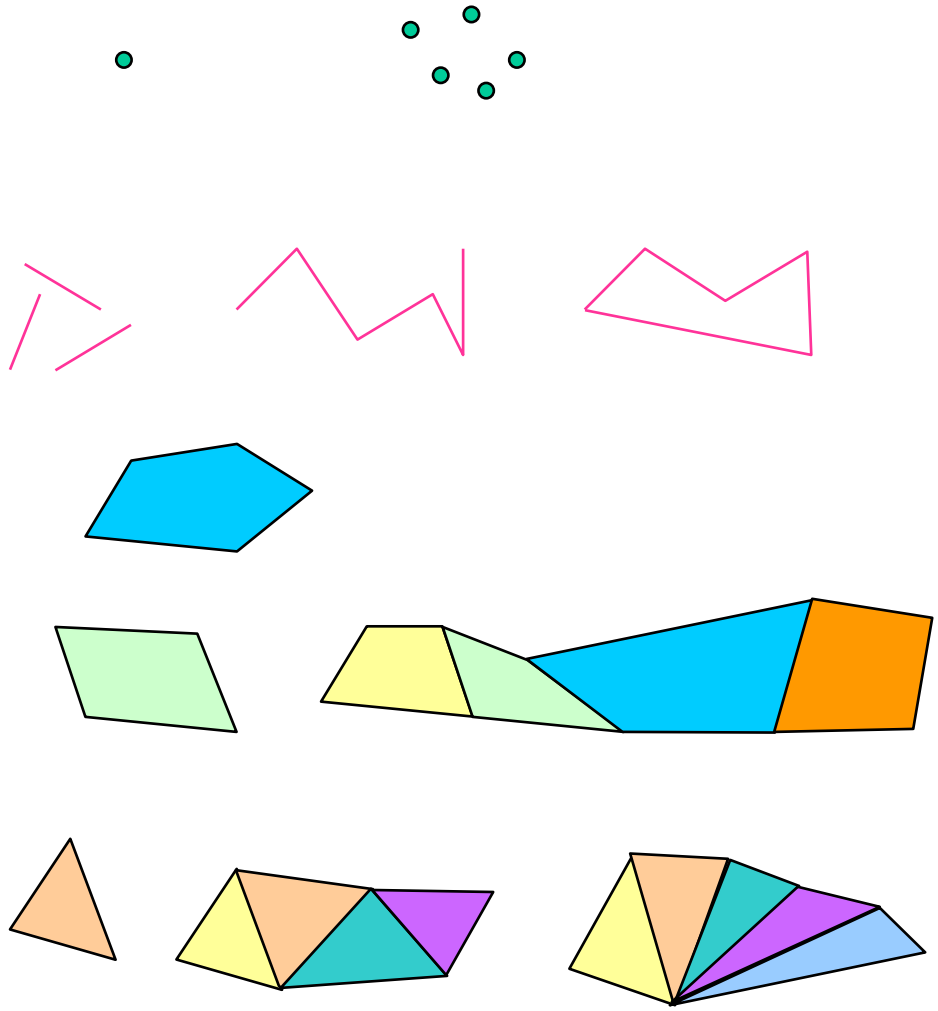
https://3dless.com/

Mutant

https://rigmodels.com/3d_LOD.php?view=5OOOEWDBCBR630L4R438TGJCD

Primitive u OpenGL-u:

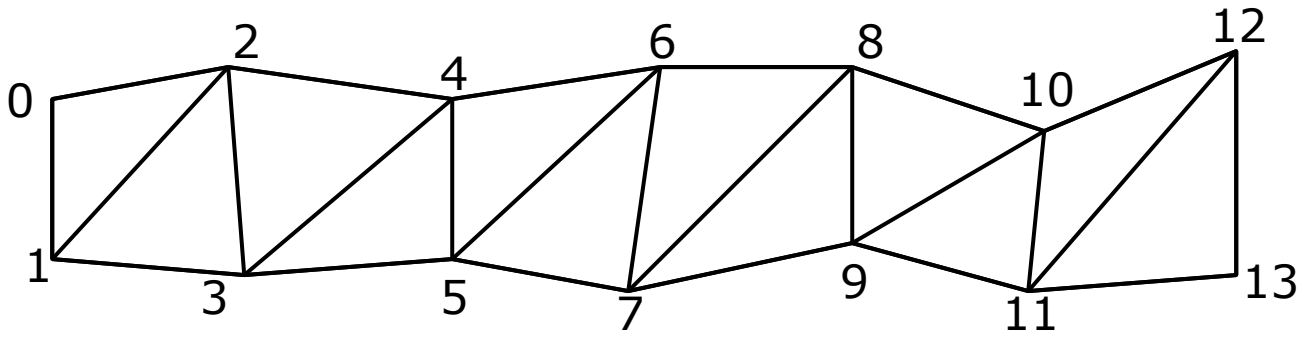
- točke
  - GL_POINTS
- dužina, niz dužina,
  - GL_LINES,
  - GL_LINE_STRIP,
  - GL_LINE_LOOP
- poligon
  - GL_POLYGON,
- četverokut, traka četverokuta
  - GL_QUADS,
  - GL_QUAD_STRIP,
- trokut, traka trokuta
  - GL_TRIANGLES
  - GL_TRIANGLE_STRIP
  - GL_TRIANGLE_FAN



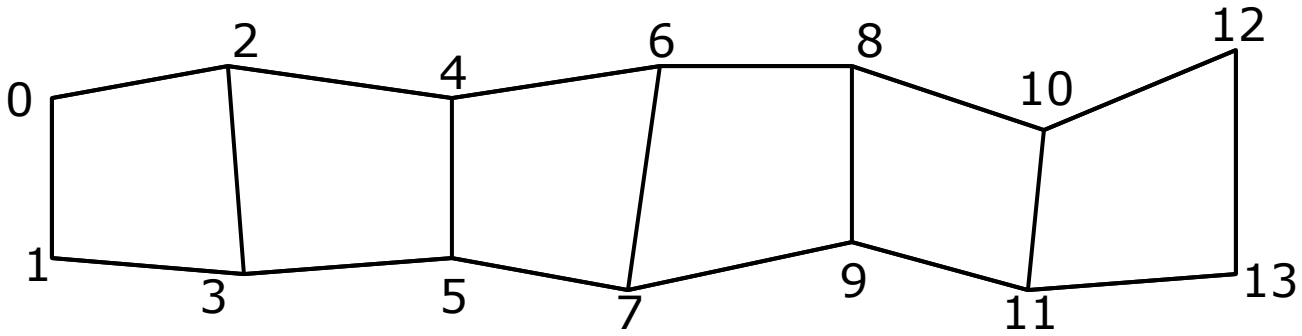
Traka trokuta (triangle strip):

- neki vrhovi zajednički trokutima (značajna ušteda)
- za n trokuta imamo $n + 2$ vrha umjesto $3 \times n$

```
glDrawElements(GL_TRIANGLE_STRIP, indices.size(), GL_UNSIGNED_SHORT, &indices[0]);
```



GL_QUAD_STRIP za n četverokuta imamo $2n + 2$ vrha



Primitive –

GL_TRIANGLE_STRIP

Promjena stanja –

glPointSize(size);
glLineStipple(repeat, pattern);
glShadeModel(GL_SMOOTH);

Aktiviranje mogućnosti –

glEnable(GL_LIGHTING);
glDisable(GL_TEXTURE_2D);

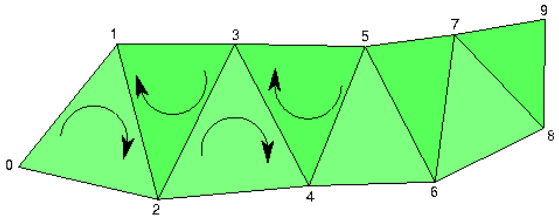


Figure 8. Triangle Strip Winding

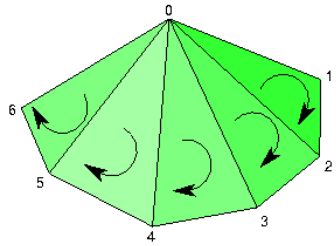


Figure 9. Triangle Fan Winding

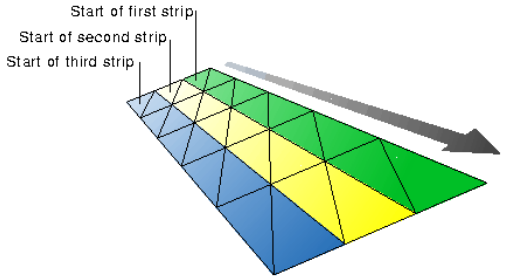
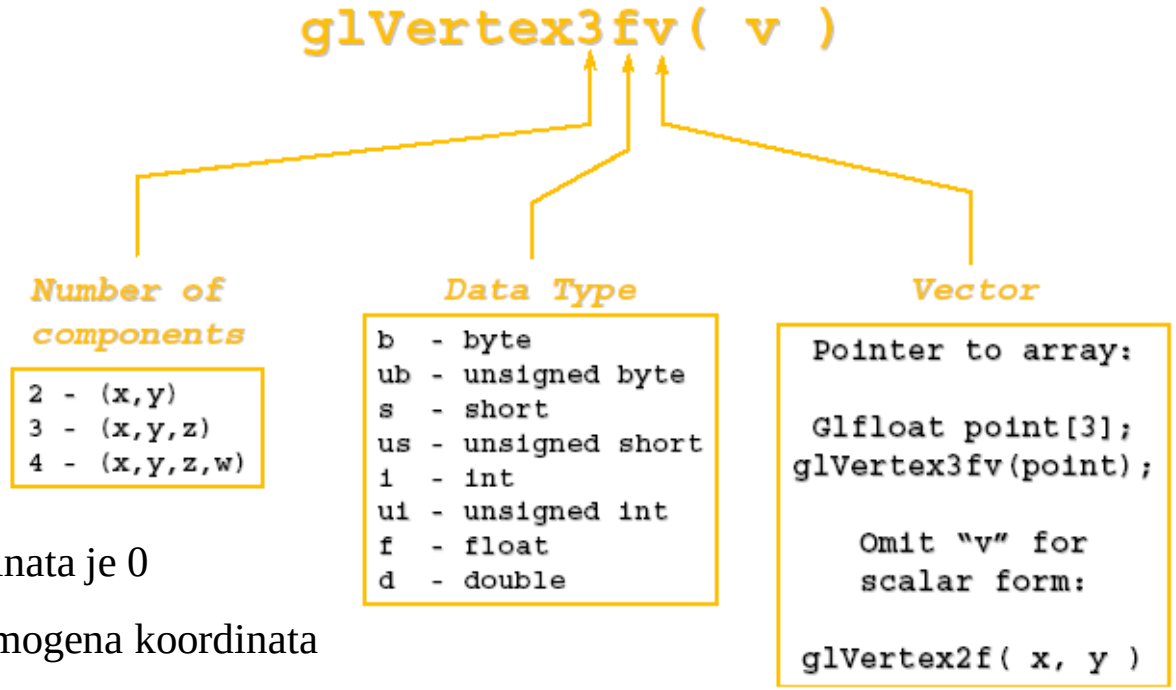


Figure 10. A Mesh Made up of Multiple Triangle Strips

- primjer naredbe u OpenGL-u primjer u Tutor – shapes.exe

```
GLfloat x1, y1;    // zbog različitih realizacija u OpenGL-u je definiran tip GLfloat, GLint …
                                        // ekvivalentan kao u C-u    - prenosivost
glBegin (Primitiva)   // Primitiva je npr. GL_LINES
        glVertex2f (x1, y1);
        glVertex2f (3,27, 5,226);
glEnd;
```

Primitiva – određuje na koji način će točke koje slijede biti kombinirane



2 - z koordinata je 0

4 - w je homogena koordinata

Izravan način:

- **individualan** prijenos podataka - ne uzimamo u obzir da su neki vrhovi **zajednički**

```
glBegin(GL_TRIANGLES);       //slijede podaci - trokuti
      glVertex3f (1f, 2.2f, 3.4f);
      glVertex3f (x1, y1, z1);
      glVertex3f (x2, y2, z2);
glEnd();
```

Umjesto da podacima pristupamo kao pojedinačnim vrhovima, koristimo polja vrhova.

- **polje podataka vrhova** (indeksiranih)

```
vrh[0] = {x0,y0,z0};
vrh[1] = {x1,y1,z1};
vrh[2] = {x2,y2,z2};

glBegin(GL_TRIANGLES);              // indeksirani vrhovi
   for (i = 0; i < 3; i++) {
       glNormal3fv (normala[i]);    // atributi za vrh i
       glVertex3fv (vrh[i]);
  }
glEnd();
```

- nije efikasno, za svaki vrh se poziva glVertex(),
  puno **funkcijskih poziva** i prijenosa - **vrlo sporo,**
  podatke želimo **u memoriji grafičke kartice**

Želimo smanjiti broj funkcijskih poziva i želimo pohraniti podatke u memoriji na grafičkoj kartici:

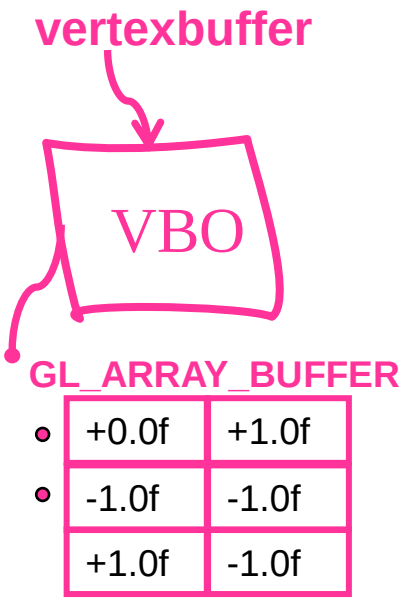
VBO – VertexBufferObject to omogućuje

Uzmimo primjer jednog trokuta s x, y koordinatama:

```
GLfloat verts[] = { +0.0f, +1.0f,    -1.0f, -1.0f,     +1.0f, -1.0f};
                                          // Želimo prenijeti na GKarticu -> VBO
```

```
glGenBuffers          // stvaranje spremnika
glBindBuffer          // aktiviranje spremnika na GKartici
glBufferData          // punjenje aktivnog spremnika podacima

glEnableVertexAttribArray
glVertexAttribPointer
```

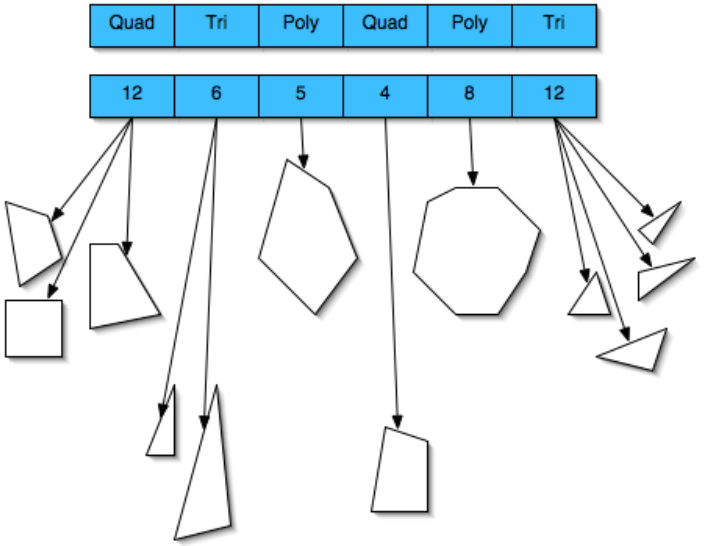
https://threejs.org/examples/#webgl_instancing_performance
https://alteredqualia.com/three/examples/webgl_buffergeometry_perf2.html

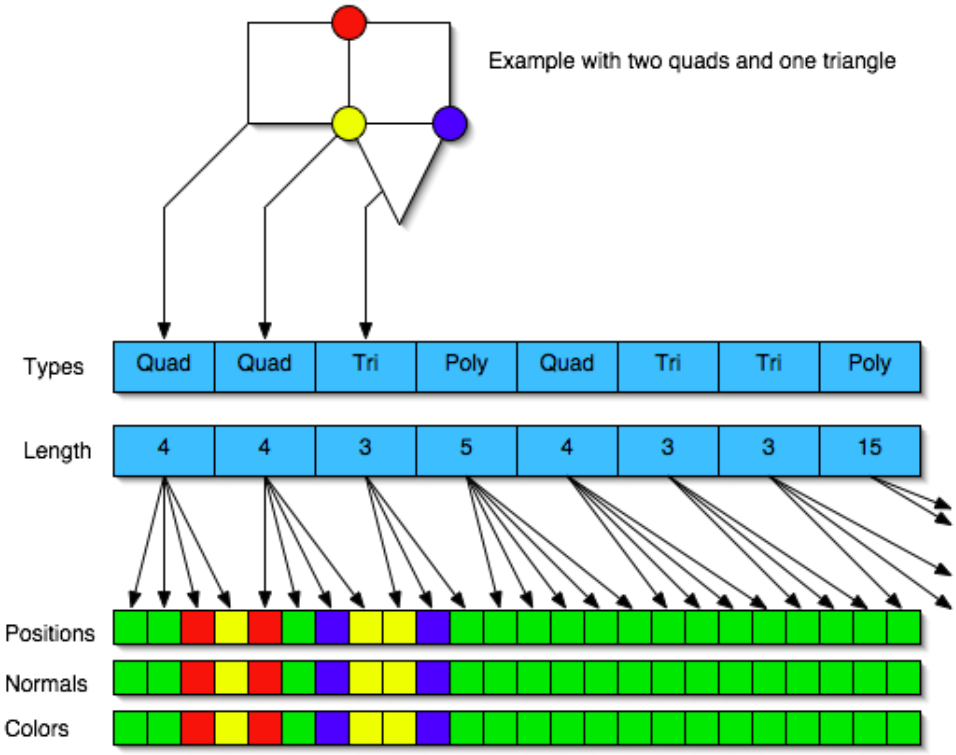



```
// VBO - VertexBufferObject

GLuint vertexbuffer;  // Logička adresa (pokazivač)
glGenBuffers(kolikoSpremnikaTrebaStvoriti, &identifikatorSpremnika)
void glBindBuffer(GLenum target, GLuint buffer);
// target - vrsta spremnika (Binding Point): GL_ARRAY_BUFFER vs.
GL_ELEMENT_ARRAY_BUFFER – indeksirani pristup
glBufferData(GL_ARRAY_BUFFER, sizeof(g_vertex_buffer_data),
g_vertex_buffer_data, GL_STATIC_DRAW);

// Moramo opisati što su podaci koje šaljemo i poslati ih u protočni sustav
void glVertexAttribPointer(GLuint index, GLint size, GLenum type,
GLboolean normalized, GLsizei stride, const GLvoid * pointer);
```

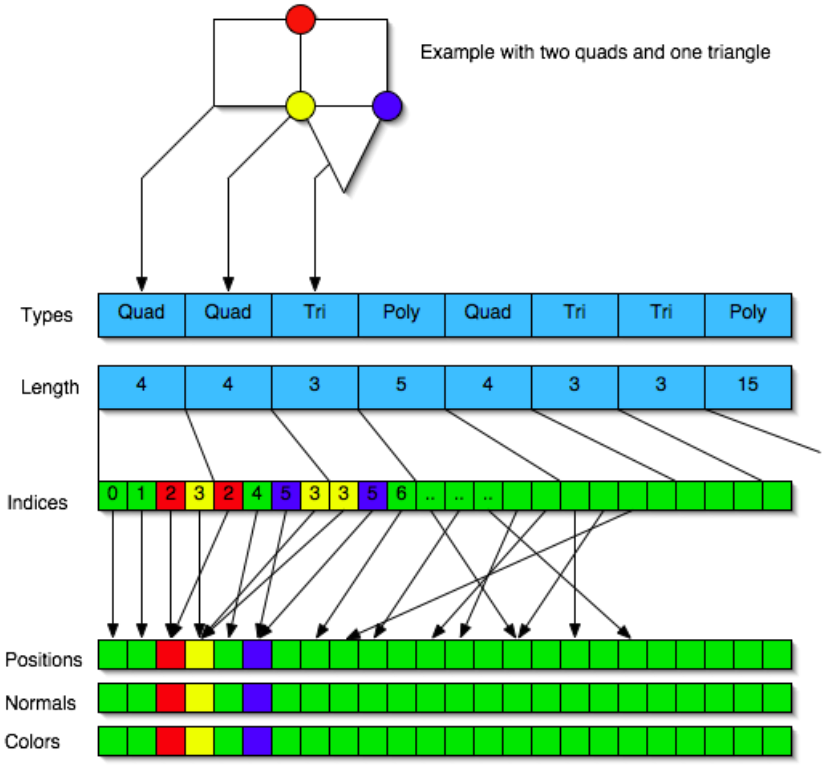
```
// VBO – VertexBufferObject

GLfloat verts[] = { +0.0f, +1.0f,    -1.0f, -1.0f,     +1.0f, -1.0f}; // Želimo prenijeti na GKarticu -> VBO
GLuint vertexbuffer;
glGenBuffers(1, &vertexbuffer);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(verts), verts, GL_STATIC_DRAW);

glEnableVertexAttribArray(0);
glVertexAttribPointer(0,  // indeks atributa, koji mora odgovarati indeksu u sjenčaru
                      2,  // broj koordinata u vrhu – mora biti 1, 2, 3 ili 4
                      GL_FLOAT, // tip koordinate
                      GL_FALSE, // automatska normalizacija na raspon [-1, 1]
                      0,       // razmak u oktetima između vrhova; 0: slijedni vrhovi
                      (void*)0 );  // indeks okteta u nizu na kojem počinju vrhovi
```

- Primjeri organizacije podataka:
  - bez indeksiranja
    - višestruki podaci
    - npr. za eksploziju poligona

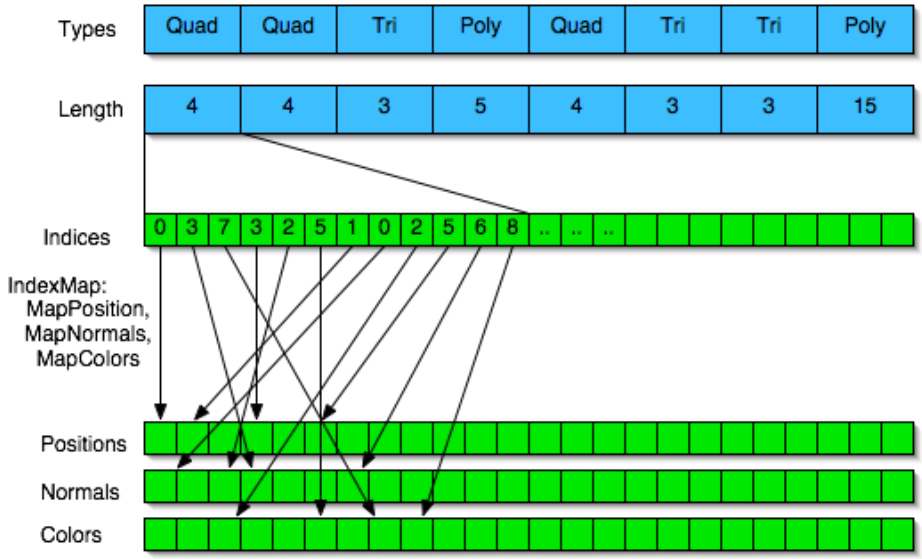


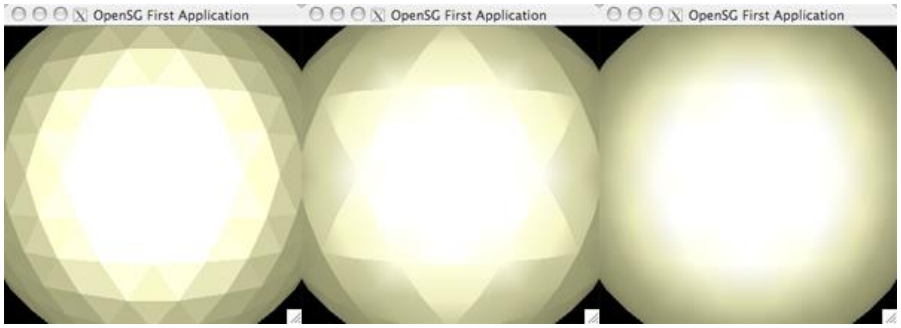


- Primjeri organizacije podataka:
  - indeksirani pristup
    - zajednički podaci za pojedini vrh
    - npr. sjenčanje Gouraud

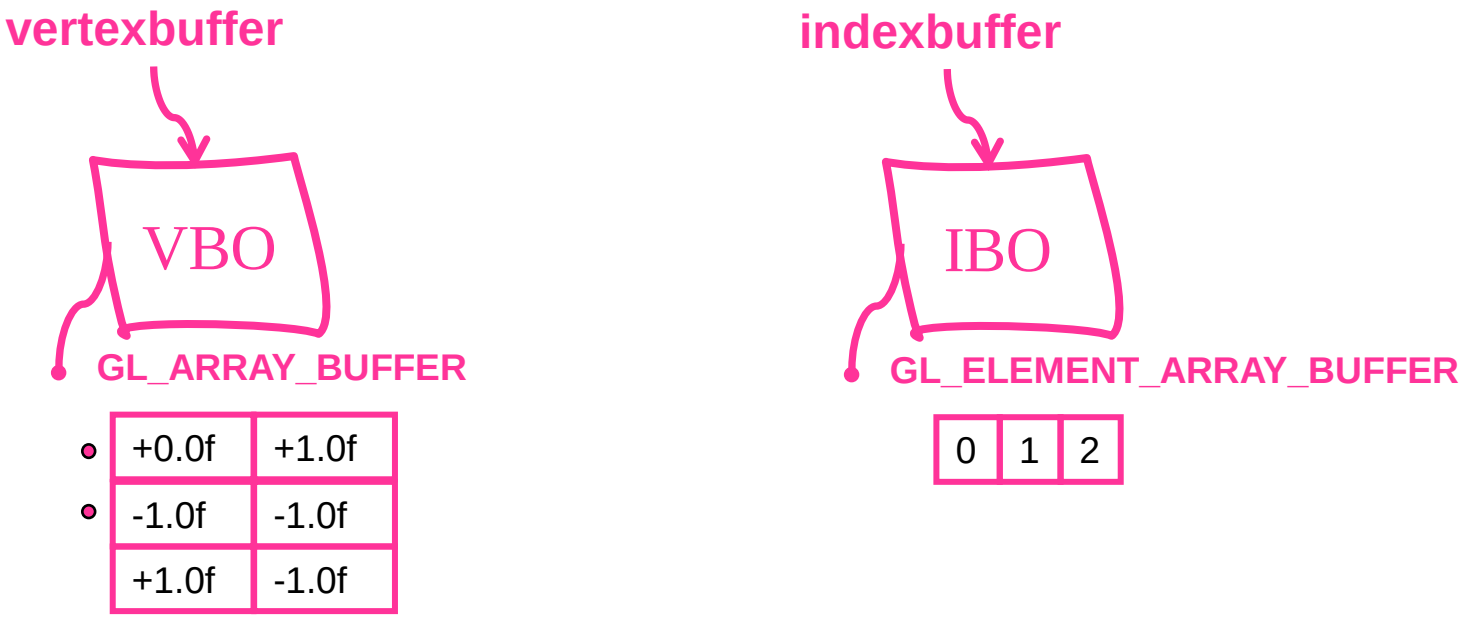


  - višestruko indeksirani pristup
    - poseban pristup normalama, boji i sl.
    - povećan broj indeksa (×3)
    - npr. normale poligona – konstantno





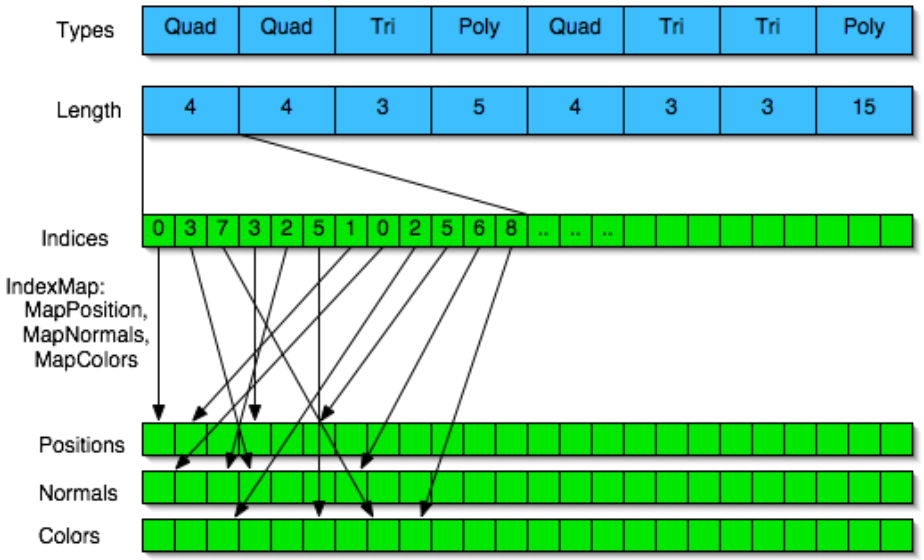
```
// Indeksirani vrhovi – GL_ELEMENT_ARRAY_BUFFER

GLushort vindices[] = {0, 1, 2};
GLuint indexbuffer;
glGenBuffers(1, &indexbuffer);
glBindBuffer(GL_ELEMENT_ARRAY_BUFFER, indexbuffer);
glBufferData(GL_ELEMENT_ARRAY_BUFFER, sizeof(vindices), vindices, GL_STATIC_DRAW);
…

// glDrawArrays(GL_TRIANGLES, 0, 3); // ako je neindeksirano, 0 - je početni vrh, 3 - broj vrhova
glDrawElements(GL_TRIANGLES, 3, GL_UNSIGNED_SHORT, 0);
```

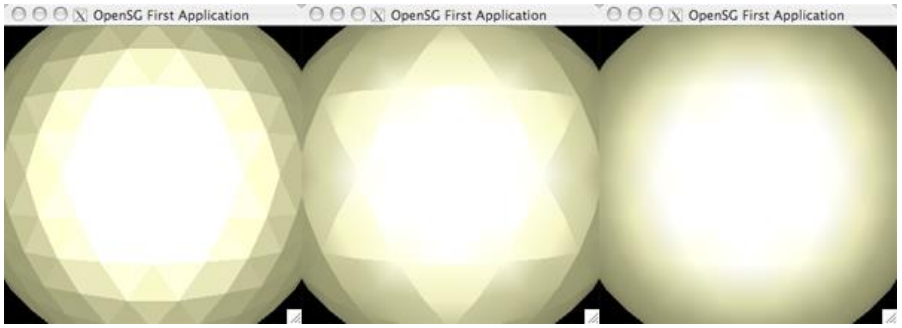


```
// Dodavanje Atributa Boje

GLfloat verts[] = {  +0.0f, +1.0f,     +1.0f,  +0.0f, +0.0f,
                     -1.0f, -1.0f,     +0.0f,  +1.0f, +0.0f,
                     +1.0f, -1.0f ,    +0.0f,  +0.0f, +1.0f};    // Dodali smo boju

GLuint vertexbuffer;
glGenBuffers(1, &vertexbuffer);
glBindBuffer(GL_ARRAY_BUFFER, vertexbuffer);
glBufferData(GL_ARRAY_BUFFER, sizeof(verts), verts, GL_STATIC_DRAW);

glEnableVertexAttribArray(0);
glVertexAttribPointer(0,  2,  GL_FLOAT,  GL_FALSE,
                      sizeof(float) * 5,       // razmak u oktetima između vrhova,  0 -  slijedni vrhovi
                      (void*)0  // indeks okteta u nizu na kojem počinju vrhovi);

glEnableVertexAttribArray(1);   // Novi atribut je boja
glVertexAttribPointer(1 // Koristi polje boje,  3 // Boja ima 3 * float ,  GL_FLOAT,  GL_FALSE,
                      sizeof(float) * 5,       // razmak u oktetima između vrhova , 0 -  slijedni vrhovi
                      (char*)(sizeof(float)  * 2));  // pomak do početka podataka o boji

// Potrebno je napisati sjenčare – minimalno vertex koji će proslijediti boju u fragment da bi
trokut bio obojan

glDrawElements(GL_TRIANGLES, 3, GL_UNSIGNED_SHORT, 0);
glDisableVertexAttribArray(0);  // tako da ostavimo OpenGL stanje očišćeno, pa očistimo i ostale
```

```
// VAO Vertex Array Objects – obično imamo više sjenčara svaki sa svojim atributima,
VAO omogućuje postavljanje aktivnog sjenčara s pripadnim atributima
```

– višestruko indeksirani pristup
  - poseban pristup normalama, boji i sl.
  - povećan broj indeksa (×3)
  - npr. normale poligona – konstantno vbo.exe (Drawingtine/Space)

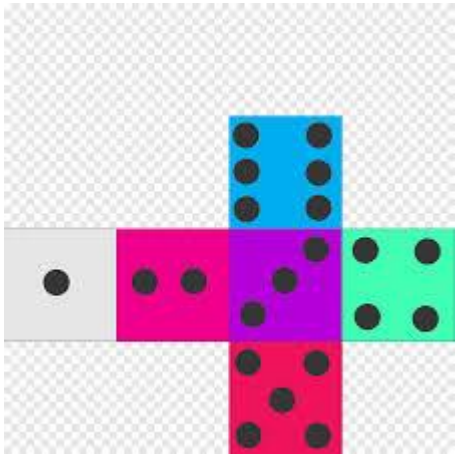




– npr. ušteda kod indeksiranog pristupa za kocku (8 vrhova, 12 trokuta):

Podaci o vrhu:

• pozicija	12 bajta
• koordinate teksture	8 bajta
• normala	12 bajta
• boja	12 bajta
=	44 bajta



Neindeksirano: 12 trokuta × 3 vrha × 44 bajta = **1584** bajta

Indeksirano: 8 vrhova × 44 bajta + 12 trokuta × 3 vrha × 4 bajta (indeks) = **466** bajta